

# Speechresearch/gmisclib

## API Documentation

September 22, 2011

## Contents

<b>Contents</b>	<b>1</b>
<b>1 Package gmisclib</b>	<b>27</b>
1.1 Modules . . . . .	27
1.2 Variables . . . . .	31
<b>2 Module gmisclib.HList</b>	<b>32</b>
2.1 Functions . . . . .	32
2.2 Variables . . . . .	32
<b>3 Module gmisclib.HTK_HMM_io</b>	<b>33</b>
3.1 Functions . . . . .	33
3.2 Variables . . . . .	33
3.3 Class BadFormatError . . . . .	34
3.3.1 Methods . . . . .	34
3.3.2 Properties . . . . .	34
3.4 Class vec . . . . .	35
3.4.1 Methods . . . . .	35
3.4.2 Properties . . . . .	35
3.4.3 Class Variables . . . . .	35
3.5 Class mixture . . . . .	36
3.5.1 Methods . . . . .	36
3.5.2 Properties . . . . .	37
3.5.3 Class Variables . . . . .	37
3.6 Class ref . . . . .	37
3.6.1 Methods . . . . .	37
3.6.2 Properties . . . . .	37
3.7 Class state . . . . .	38
3.7.1 Methods . . . . .	38
3.7.2 Properties . . . . .	38
3.8 Class hmm . . . . .	38
3.8.1 Methods . . . . .	39
3.8.2 Properties . . . . .	39
3.9 Class hmmset . . . . .	39
3.9.1 Methods . . . . .	39
3.9.2 Properties . . . . .	40
3.10 Class tokstream . . . . .	40

3.10.1	Methods . . . . .	40
3.10.2	Properties . . . . .	41
3.10.3	Class Variables . . . . .	41
3.11	Class NoPhonemeToFormat . . . . .	42
3.11.1	Methods . . . . .	42
3.11.2	Properties . . . . .	42
<b>4</b>	<b>Module gmisclib.HTK_MLF_io</b>	<b>43</b>
4.1	Functions . . . . .	43
4.2	Variables . . . . .	45
4.3	Class ReferencedFileNotFound . . . . .	45
4.3.1	Methods . . . . .	45
4.3.2	Properties . . . . .	46
4.4	Class writer . . . . .	46
4.4.1	Methods . . . . .	46
4.4.2	Properties . . . . .	47
<b>5</b>	<b>Module gmisclib.MFCCFile</b>	<b>48</b>
5.1	Variables . . . . .	48
5.2	Class MFCCFile . . . . .	48
5.2.1	Methods . . . . .	48
<b>6</b>	<b>Module gmisclib.MLF_file</b>	<b>49</b>
6.1	Variables . . . . .	49
6.2	Class BadFormatError . . . . .	49
6.2.1	Methods . . . . .	49
6.2.2	Properties . . . . .	49
6.3	Class NotInMLFFile . . . . .	50
6.3.1	Methods . . . . .	50
6.3.2	Properties . . . . .	50
6.4	Class block_MLF_file . . . . .	51
6.4.1	Methods . . . . .	51
6.4.2	Properties . . . . .	52
6.4.3	Class Variables . . . . .	52
6.4.4	Instance Variables . . . . .	52
<b>7</b>	<b>Module gmisclib.Num</b>	<b>53</b>
7.1	Variables . . . . .	53
<b>8</b>	<b>Module gmisclib.Numeric_gpk</b>	<b>58</b>
8.1	Functions . . . . .	58
8.2	Variables . . . . .	62
8.3	Class EdgesTooWide . . . . .	63
8.3.1	Methods . . . . .	63
8.3.2	Properties . . . . .	63
<b>9</b>	<b>Module gmisclib.accent_spec</b>	<b>64</b>
9.1	Functions . . . . .	64
9.2	Variables . . . . .	65
9.3	Class BadMatchError . . . . .	65
9.3.1	Methods . . . . .	65
9.3.2	Properties . . . . .	66

<b>10 Module gmisclib.array_window</b>	<b>67</b>
10.1 Functions . . . . .	67
10.2 Variables . . . . .	67
10.3 Class array_window . . . . .	67
10.3.1 Methods . . . . .	67
10.3.2 Class Variables . . . . .	68
<b>11 Module gmisclib.avio</b>	<b>69</b>
11.1 Functions . . . . .	69
11.2 Variables . . . . .	70
11.3 Class BadFormatError . . . . .	70
11.3.1 Methods . . . . .	70
11.3.2 Properties . . . . .	71
11.4 Class writer . . . . .	71
11.4.1 Methods . . . . .	71
11.4.2 Properties . . . . .	72
11.4.3 Class Variables . . . . .	72
<b>12 Module gmisclib.bark_scale</b>	<b>73</b>
12.1 Functions . . . . .	73
12.2 Variables . . . . .	73
<b>13 Module gmisclib.beamsearch</b>	<b>74</b>
13.1 Functions . . . . .	74
13.2 Variables . . . . .	74
<b>14 Module gmisclib.blue_data_attributes</b>	<b>75</b>
14.1 Functions . . . . .	75
14.2 Variables . . . . .	75
14.3 Class blue_attributes . . . . .	75
14.3.1 Methods . . . . .	75
14.3.2 Properties . . . . .	76
<b>15 Module gmisclib.blue_data_selector</b>	<b>77</b>
15.1 Functions . . . . .	77
15.2 Variables . . . . .	77
15.3 Class bluedata . . . . .	77
15.3.1 Methods . . . . .	78
15.3.2 Properties . . . . .	80
<b>16 Module gmisclib.cache</b>	<b>81</b>
16.1 Functions . . . . .	81
16.2 Variables . . . . .	82
16.3 Class CannotGetStat . . . . .	82
16.3.1 Methods . . . . .	82
16.3.2 Properties . . . . .	83
16.4 Class BadFileFormat . . . . .	83
16.4.1 Methods . . . . .	83
16.4.2 Properties . . . . .	84
16.5 Class cache_info . . . . .	84
16.5.1 Methods . . . . .	87
16.5.2 Properties . . . . .	89

16.5.3 Class Variables . . . . .	89
<b>17 Module gmisclib.chunkio</b>	<b>90</b>
17.1 Functions . . . . .	90
17.2 Variables . . . . .	90
17.3 Class BadFileFormat . . . . .	90
17.3.1 Methods . . . . .	91
17.3.2 Properties . . . . .	91
17.4 Class chunk . . . . .	91
17.4.1 Methods . . . . .	91
17.4.2 Class Variables . . . . .	92
17.5 Class datachunk . . . . .	93
17.5.1 Methods . . . . .	93
17.5.2 Class Variables . . . . .	94
17.6 Class stringchunk . . . . .	94
17.6.1 Methods . . . . .	94
17.6.2 Class Variables . . . . .	96
17.7 Class chunk_w . . . . .	96
17.7.1 Methods . . . . .	96
17.8 Class datachunk_w . . . . .	97
17.8.1 Methods . . . . .	97
17.8.2 Class Variables . . . . .	98
17.9 Class chunkstring_w . . . . .	99
17.9.1 Methods . . . . .	99
17.9.2 Class Variables . . . . .	100
<b>18 Module gmisclib.convex_hull2d</b>	<b>101</b>
18.1 Functions . . . . .	101
18.2 Variables . . . . .	101
18.3 Class DuplicatePoints . . . . .	102
18.3.1 Methods . . . . .	102
18.3.2 Properties . . . . .	102
<b>19 Module gmisclib.dict_vector</b>	<b>103</b>
19.1 Functions . . . . .	103
19.2 Variables . . . . .	104
19.3 Class dict_vector . . . . .	104
19.3.1 Methods . . . . .	105
19.3.2 Properties . . . . .	107
19.3.3 Class Variables . . . . .	107
<b>20 Module gmisclib.dictops</b>	<b>109</b>
20.1 Functions . . . . .	109
20.2 Variables . . . . .	111
20.3 Class BadFileFormatError . . . . .	112
20.3.1 Methods . . . . .	112
20.3.2 Properties . . . . .	112
20.4 Class dict_of_lists . . . . .	113
20.4.1 Methods . . . . .	113
20.4.2 Properties . . . . .	114
20.4.3 Class Variables . . . . .	114
20.5 Class dict_of_sets . . . . .	114

20.5.1	Methods . . . . .	114
20.5.2	Properties . . . . .	115
20.5.3	Class Variables . . . . .	115
20.6	Class dict_of_accums . . . . .	115
20.6.1	Methods . . . . .	115
20.6.2	Properties . . . . .	116
20.6.3	Class Variables . . . . .	116
20.7	Class dict_of_averages . . . . .	116
20.7.1	Methods . . . . .	117
20.7.2	Properties . . . . .	117
20.7.3	Class Variables . . . . .	117
20.8	Class dict_of_maxes . . . . .	118
20.8.1	Methods . . . . .	118
20.8.2	Properties . . . . .	118
20.8.3	Class Variables . . . . .	118
20.9	Class dict_of_X . . . . .	119
20.9.1	Methods . . . . .	119
20.9.2	Properties . . . . .	119
20.9.3	Class Variables . . . . .	120
20.10	Class list_of_dicts . . . . .	120
20.10.1	Methods . . . . .	120
20.10.2	Properties . . . . .	121
20.10.3	Class Variables . . . . .	121
<b>21</b>	<b>Module gmisclib.die</b>	<b>122</b>
21.1	Functions . . . . .	122
21.2	Variables . . . . .	124
21.3	Class counter . . . . .	124
21.3.1	Methods . . . . .	124
21.3.2	Class Variables . . . . .	125
<b>22</b>	<b>Module gmisclib.die2</b>	<b>126</b>
22.1	Functions . . . . .	126
22.2	Variables . . . . .	126
<b>23</b>	<b>Module gmisclib.do_exec</b>	<b>128</b>
23.1	Functions . . . . .	128
23.2	Variables . . . . .	128
<b>24</b>	<b>Module gmisclib.ds9_region</b>	<b>129</b>
24.1	Variables . . . . .	129
24.2	Class writer . . . . .	129
24.2.1	Methods . . . . .	129
<b>25</b>	<b>Module gmisclib.dtw4</b>	<b>130</b>
25.1	Functions . . . . .	130
25.2	Variables . . . . .	133
25.3	Class NoRoute . . . . .	133
25.3.1	Methods . . . . .	133
25.3.2	Properties . . . . .	133
25.4	Class slice_c . . . . .	134
25.4.1	Methods . . . . .	134

25.4.2	Properties . . . . .	134
25.5	Class state_c . . . . .	135
25.5.1	Methods . . . . .	135
25.5.2	Properties . . . . .	135
25.6	Class state_jump_c . . . . .	135
25.6.1	Methods . . . . .	136
25.6.2	Properties . . . . .	136
<b>26</b>	<b>Module gmisclib.dyn_prog</b>	<b>137</b>
26.1	Functions . . . . .	137
26.2	Variables . . . . .	137
<b>27</b>	<b>Module gmisclib.edit_distance</b>	<b>138</b>
27.1	Functions . . . . .	138
27.2	Variables . . . . .	140
27.3	Class text_cost . . . . .	140
27.3.1	Methods . . . . .	141
27.3.2	Properties . . . . .	142
<b>28</b>	<b>Module gmisclib.entropy</b>	<b>143</b>
28.1	Functions . . . . .	143
28.2	Variables . . . . .	143
<b>29</b>	<b>Module gmisclib.erb_scale</b>	<b>144</b>
29.1	Functions . . . . .	144
29.2	Variables . . . . .	144
<b>30</b>	<b>Module gmisclib.evolution</b>	<b>145</b>
30.1	Variables . . . . .	145
<b>31</b>	<b>Module gmisclib.fake_file</b>	<b>146</b>
31.1	Functions . . . . .	146
31.2	Variables . . . . .	146
31.3	Class file . . . . .	146
31.3.1	Methods . . . . .	146
<b>32</b>	<b>Module gmisclib.fiat_merge</b>	<b>147</b>
32.1	Functions . . . . .	147
32.2	Variables . . . . .	147
<b>33</b>	<b>Module gmisclib.fiatio</b>	<b>148</b>
33.1	Functions . . . . .	149
33.2	Variables . . . . .	154
33.3	Class FiatioWarning . . . . .	154
33.3.1	Methods . . . . .	154
33.3.2	Properties . . . . .	155
33.4	Class writer . . . . .	155
33.4.1	Methods . . . . .	155
33.4.2	Properties . . . . .	157
33.5	Class merged_writer . . . . .	157
33.5.1	Methods . . . . .	158
33.5.2	Properties . . . . .	159
33.6	Class ConflictingColumnSpecification . . . . .	160

33.6.1	Methods . . . . .	160
33.6.2	Properties . . . . .	160
<b>34</b>	<b>Module gmisclib.find_home</b>	<b>161</b>
34.1	Functions . . . . .	161
34.2	Variables . . . . .	161
<b>35</b>	<b>Module gmisclib.find_ngram</b>	<b>162</b>
35.1	Functions . . . . .	162
35.2	Variables . . . . .	163
<b>36</b>	<b>Module gmisclib.findleak</b>	<b>164</b>
36.1	Functions . . . . .	164
36.2	Variables . . . . .	164
<b>37</b>	<b>Module gmisclib.ftest</b>	<b>165</b>
37.1	Functions . . . . .	165
37.2	Variables . . . . .	166
<b>38</b>	<b>Module gmisclib.fuzzygraph</b>	<b>167</b>
38.1	Functions . . . . .	167
38.2	Variables . . . . .	167
<b>39</b>	<b>Module gmisclib.g2_select</b>	<b>168</b>
39.1	Functions . . . . .	168
39.2	Variables . . . . .	169
39.3	Class selector_c . . . . .	169
39.3.1	Methods . . . . .	169
39.3.2	Properties . . . . .	170
<b>40</b>	<b>Module gmisclib.g_closure</b>	<b>171</b>
40.1	Variables . . . . .	171
40.2	Class NotYet . . . . .	171
40.2.1	Methods . . . . .	171
40.3	Class ArgUnspecifiedError . . . . .	171
40.3.1	Methods . . . . .	172
40.3.2	Properties . . . . .	172
40.4	Class Closure . . . . .	172
40.4.1	Methods . . . . .	172
<b>41</b>	<b>Module gmisclib.g_datetime</b>	<b>174</b>
41.1	Functions . . . . .	174
41.2	Variables . . . . .	174
<b>42</b>	<b>Module gmisclib.g_ds9</b>	<b>175</b>
42.1	Functions . . . . .	175
42.2	Variables . . . . .	176
42.3	Class UnknownProperty . . . . .	176
42.3.1	Methods . . . . .	177
42.3.2	Properties . . . . .	177
42.4	Class execwait . . . . .	177
42.4.1	Methods . . . . .	177
42.4.2	Properties . . . . .	178

42.5	Class ds9_io . . . . .	178
42.5.1	Methods . . . . .	178
42.5.2	Properties . . . . .	179
42.6	Class BadTagError . . . . .	179
42.6.1	Methods . . . . .	179
42.6.2	Properties . . . . .	180
42.7	Class Region . . . . .	180
42.7.1	Methods . . . . .	180
42.7.2	Properties . . . . .	181
42.8	Class line . . . . .	181
42.8.1	Methods . . . . .	181
42.8.2	Properties . . . . .	182
42.9	Class circle . . . . .	182
42.9.1	Methods . . . . .	182
42.9.2	Properties . . . . .	183
42.10	Class point . . . . .	183
42.10.1	Methods . . . . .	184
42.10.2	Properties . . . . .	184
42.11	Class text . . . . .	185
42.11.1	Methods . . . . .	185
42.11.2	Properties . . . . .	185
42.12	Class ds9 . . . . .	186
42.12.1	Methods . . . . .	186
42.12.2	Properties . . . . .	188
42.12.3	Class Variables . . . . .	188
42.13	Class ds9_file . . . . .	189
42.13.1	Methods . . . . .	189
42.13.2	Properties . . . . .	189
<b>43</b>	<b>Module gmisclib.g_encode . . . . .</b>	<b>190</b>
43.1	Functions . . . . .	190
43.2	Variables . . . . .	190
43.3	Class BadFormatError . . . . .	190
43.3.1	Methods . . . . .	190
43.3.2	Properties . . . . .	191
43.4	Class encoder . . . . .	191
43.4.1	Methods . . . . .	191
<b>44</b>	<b>Module gmisclib.g_entropy . . . . .</b>	<b>192</b>
44.1	Functions . . . . .	192
44.2	Variables . . . . .	192
<b>45</b>	<b>Module gmisclib.g_exec . . . . .</b>	<b>193</b>
45.1	Functions . . . . .	194
45.2	Variables . . . . .	197
45.3	Class ExecError . . . . .	197
45.3.1	Methods . . . . .	198
45.3.2	Properties . . . . .	198
<b>46</b>	<b>Module gmisclib.g_implements . . . . .</b>	<b>199</b>
46.1	Functions . . . . .	199
46.2	Variables . . . . .	201



46.3 Class <code>GTypeError</code> . . . . .	201
46.3.1 Methods . . . . .	201
46.3.2 Properties . . . . .	202
<b>47 Module <code>gmisclib.g_keyed_accum</code></b>	<b>203</b>
47.1 Functions . . . . .	203
47.2 Variables . . . . .	203
47.3 Class <code>keyed_avg</code> . . . . .	203
47.3.1 Methods . . . . .	203
<b>48 Module <code>gmisclib.g_lin_fit</code></b>	<b>204</b>
48.1 Functions . . . . .	204
48.2 Variables . . . . .	205
<b>49 Module <code>gmisclib.g_localfit</code></b>	<b>206</b>
49.1 Functions . . . . .	206
49.2 Variables . . . . .	211
<b>50 Module <code>gmisclib.g_pipe</code></b>	<b>212</b>
50.1 Functions . . . . .	212
50.2 Variables . . . . .	212
50.3 Class <code>pfd</code> . . . . .	212
50.3.1 Methods . . . . .	212
50.3.2 Properties . . . . .	213
50.3.3 Class Variables . . . . .	213
<b>51 Module <code>gmisclib.g_pipe_old</code></b>	<b>214</b>
51.1 Functions . . . . .	214
51.2 Variables . . . . .	214
51.3 Class <code>pfd</code> . . . . .	214
51.3.1 Methods . . . . .	214
51.3.2 Properties . . . . .	215
51.3.3 Class Variables . . . . .	215
<b>52 Module <code>gmisclib.g_place_label</code></b>	<b>216</b>
52.1 Functions . . . . .	216
52.2 Variables . . . . .	216
52.3 Class <code>boxc</code> . . . . .	216
52.3.1 Methods . . . . .	216
52.3.2 Properties . . . . .	217
52.4 Class <code>text.template</code> . . . . .	217
52.4.1 Methods . . . . .	217
52.4.2 Properties . . . . .	218
52.5 Class <code>viewport</code> . . . . .	218
52.5.1 Methods . . . . .	218
52.5.2 Properties . . . . .	219
<b>53 Module <code>gmisclib.g_pylab</code></b>	<b>220</b>
<b>54 Module <code>gmisclib.g_selector</code></b>	<b>221</b>
54.1 Functions . . . . .	221
54.2 Variables . . . . .	221
54.3 Class <code>selector</code> . . . . .	221

54.3.1	Methods . . . . .	221
54.4	Class true . . . . .	222
54.4.1	Methods . . . . .	222
54.5	Class false . . . . .	222
54.5.1	Methods . . . . .	223
54.6	Class selector_op . . . . .	223
54.6.1	Methods . . . . .	223
54.7	Class one_selector . . . . .	224
54.7.1	Methods . . . . .	224
<b>55</b>	<b>Module gmisclib.g_ubyte</b>	<b>226</b>
55.1	Functions . . . . .	226
55.2	Variables . . . . .	226
<b>56</b>	<b>Module gmisclib.gpk_getopt</b>	<b>227</b>
56.1	Functions . . . . .	227
56.2	Variables . . . . .	227
<b>57</b>	<b>Module gmisclib.gpk_hdr</b>	<b>228</b>
57.1	Functions . . . . .	228
57.2	Variables . . . . .	228
57.3	Class hdr . . . . .	228
57.3.1	Methods . . . . .	228
<b>58</b>	<b>Module gmisclib.gpk_lapack</b>	<b>229</b>
58.1	Functions . . . . .	229
58.2	Class NoSolutionError . . . . .	229
58.2.1	Methods . . . . .	229
58.2.2	Properties . . . . .	230
<b>59</b>	<b>Module gmisclib.gpk_lsqr</b>	<b>231</b>
59.1	Functions . . . . .	231
59.2	Variables . . . . .	231
59.3	Class lsqr_base . . . . .	231
59.3.1	Methods . . . . .	232
59.3.2	Properties . . . . .	233
59.4	Class linear_least_squares . . . . .	233
59.4.1	Methods . . . . .	234
59.4.2	Properties . . . . .	236
59.5	Class reg_linear_least_squares . . . . .	236
59.5.1	Methods . . . . .	237
59.5.2	Properties . . . . .	239
<b>60</b>	<b>Module gmisclib.gpk_rlsqr</b>	<b>240</b>
60.1	Functions . . . . .	240
60.2	Variables . . . . .	240
60.3	Class NotEnoughData . . . . .	240
60.3.1	Methods . . . . .	240
60.3.2	Properties . . . . .	241
60.4	Class w_linear_least_squares . . . . .	241
60.4.1	Methods . . . . .	242
60.4.2	Properties . . . . .	244

60.5 Class <code>robust_linear_fit</code> . . . . .	244
60.5.1 Methods . . . . .	245
60.5.2 Properties . . . . .	247
60.5.3 Class Variables . . . . .	247
<b>61 Module <code>gmisclib.gpk_writer</code></b>	<b>248</b>
61.1 Variables . . . . .	248
61.2 Class <code>writer</code> . . . . .	248
61.2.1 Methods . . . . .	248
61.2.2 Properties . . . . .	249
61.3 Class <code>null_writer</code> . . . . .	249
61.3.1 Methods . . . . .	249
61.3.2 Properties . . . . .	250
<b>62 Module <code>gmisclib.gpkmisc</code></b>	<b>251</b>
62.1 Functions . . . . .	251
62.2 Variables . . . . .	257
62.3 Class <code>threaded_readable_file</code> . . . . .	258
62.3.1 Methods . . . . .	258
62.3.2 Properties . . . . .	258
62.3.3 Class Variables . . . . .	258
62.4 Class <code>dir_lock</code> . . . . .	259
62.4.1 Methods . . . . .	259
62.4.2 Properties . . . . .	259
62.5 Class <code>PrereqError</code> . . . . .	259
62.5.1 Methods . . . . .	260
62.5.2 Properties . . . . .	260
<b>63 Module <code>gmisclib.hilbert_xform</code></b>	<b>261</b>
63.1 Functions . . . . .	261
63.2 Variables . . . . .	261
<b>64 Module <code>gmisclib.kl_dist</code></b>	<b>262</b>
64.1 Functions . . . . .	262
64.2 Variables . . . . .	263
64.3 Class <code>NoConvergenceError</code> . . . . .	264
64.3.1 Methods . . . . .	264
64.3.2 Properties . . . . .	264
64.4 Class <code>NotMarkovError</code> . . . . .	265
64.4.1 Methods . . . . .	265
64.4.2 Properties . . . . .	265
<b>65 Module <code>gmisclib.load_mod</code></b>	<b>266</b>
65.1 Functions . . . . .	266
65.2 Variables . . . . .	268
<b>66 Module <code>gmisclib.lreg_fill</code></b>	<b>269</b>
66.1 Functions . . . . .	269
66.2 Variables . . . . .	269
<b>67 Module <code>gmisclib.makemake</code></b>	<b>270</b>
67.1 Functions . . . . .	270
67.2 Variables . . . . .	271

67.3 Class FileNotFoundError . . . . .	271
67.3.1 Methods . . . . .	271
67.3.2 Properties . . . . .	272
<b>68 Module gmsclib.matrix_arrange</b>	<b>273</b>
68.1 Functions . . . . .	273
68.2 Variables . . . . .	273
<b>69 Module gmsclib.matrix_arrange_entropy</b>	<b>274</b>
69.1 Functions . . . . .	274
69.2 Variables . . . . .	274
<b>70 Module gmsclib.matrix_rearrange_labels</b>	<b>275</b>
70.1 Functions . . . . .	275
70.2 Variables . . . . .	280
70.3 Class diagfom . . . . .	280
70.3.1 Methods . . . . .	280
70.3.2 Properties . . . . .	281
70.4 Class diagfom2 . . . . .	281
70.4.1 Methods . . . . .	281
70.4.2 Properties . . . . .	282
70.5 Class blockfom . . . . .	282
70.5.1 Methods . . . . .	282
70.5.2 Properties . . . . .	282
<b>71 Module gmsclib.mcmc</b>	<b>283</b>
71.1 Functions . . . . .	284
71.2 Variables . . . . .	284
71.3 Class problem_definition . . . . .	285
71.3.1 Methods . . . . .	285
71.3.2 Properties . . . . .	287
71.4 Class problem_definition_F . . . . .	287
71.4.1 Methods . . . . .	287
71.4.2 Properties . . . . .	290
71.5 Class position_base . . . . .	290
71.5.1 Methods . . . . .	290
71.5.2 Properties . . . . .	292
71.6 Class position_repeatable . . . . .	292
71.6.1 Methods . . . . .	292
71.6.2 Properties . . . . .	294
71.6.3 Class Variables . . . . .	294
71.7 Class position_nonrepeatable . . . . .	295
71.7.1 Methods . . . . .	295
71.7.2 Properties . . . . .	296
71.7.3 Class Variables . . . . .	297
71.8 Class acceptor_base . . . . .	297
71.8.1 Methods . . . . .	297
71.8.2 Properties . . . . .	298
71.8.3 Instance Variables . . . . .	299
71.9 Class T_acceptor . . . . .	299
71.9.1 Methods . . . . .	299
71.9.2 Properties . . . . .	300

71.9.3	Instance Variables	301
71.10	Class rough_acceptor_base	301
71.10.1	Methods	301
71.10.2	Properties	303
71.10.3	Instance Variables	303
71.11	Class rough_T_acceptor	303
71.11.1	Methods	304
71.11.2	Properties	305
71.11.3	Instance Variables	306
71.12	Class stepper	306
71.12.1	Methods	306
71.12.2	Properties	307
71.12.3	Instance Variables	307
71.13	Class adjuster	308
71.13.1	Methods	308
71.13.2	Properties	309
71.13.3	Class Variables	309
71.13.4	Instance Variables	309
71.14	Class NoBoot	310
71.14.1	Methods	310
71.14.2	Properties	310
71.15	Class NotGoodPosition	311
71.15.1	Methods	311
71.15.2	Properties	312
71.16	Class hashcounter_c	312
71.16.1	Methods	312
71.16.2	Properties	313
71.16.3	Class Variables	313
71.17	Class Archive	313
71.17.1	Methods	313
71.17.2	Properties	315
71.17.3	Class Variables	315
71.17.4	Instance Variables	315
71.18	Class ContPrmArchive	315
71.18.1	Methods	316
71.18.2	Properties	317
71.18.3	Class Variables	317
71.18.4	Instance Variables	318
71.19	Class BootStepper	318
71.19.1	Methods	319
71.19.2	Properties	321
71.19.3	Class Variables	321
71.19.4	Instance Variables	322
<b>72</b>	<b>Module gmisclib.mcmcS2</b>	<b>323</b>
72.1	Functions	323
<b>73</b>	<b>Module gmisclib.mcmc_big</b>	<b>324</b>
73.1	Functions	324
73.2	Variables	324
73.3	Class BootStepper	324
73.3.1	Methods	325

73.3.2	Properties . . . . .	327
73.3.3	Class Variables . . . . .	327
73.3.4	Instance Variables . . . . .	328
<b>74</b>	<b>Module gmisclib.mcmc_cooperate</b>	<b>329</b>
74.1	Functions . . . . .	329
74.2	Variables . . . . .	329
74.3	Class Barrier . . . . .	329
74.3.1	Methods . . . . .	330
74.3.2	Properties . . . . .	330
74.4	Class Oops . . . . .	330
74.4.1	Methods . . . . .	331
74.4.2	Properties . . . . .	331
74.5	Class LateToBarrier . . . . .	331
74.5.1	Methods . . . . .	332
74.5.2	Properties . . . . .	332
74.6	Class connection . . . . .	332
74.6.1	Methods . . . . .	332
74.6.2	Properties . . . . .	333
74.6.3	Class Variables . . . . .	334
<b>75</b>	<b>Module gmisclib.mcmc_helper</b>	<b>335</b>
75.1	Functions . . . . .	335
75.2	Variables . . . . .	336
75.3	Class TooManyLoops . . . . .	336
75.3.1	Methods . . . . .	336
75.3.2	Properties . . . . .	336
75.4	Class warnevery . . . . .	337
75.4.1	Methods . . . . .	337
75.4.2	Properties . . . . .	337
75.5	Class logger_template . . . . .	337
75.5.1	Methods . . . . .	338
75.5.2	Properties . . . . .	338
75.6	Class stepper . . . . .	338
75.6.1	Methods . . . . .	338
75.6.2	Properties . . . . .	341
75.7	Class step_acceptor . . . . .	342
75.7.1	Methods . . . . .	343
75.7.2	Properties . . . . .	345
75.7.3	Instance Variables . . . . .	345
<b>76</b>	<b>Module gmisclib.mcmc_idxr</b>	<b>347</b>
76.1	Functions . . . . .	347
76.2	Variables . . . . .	347
76.3	Class probdist_c . . . . .	347
76.3.1	Methods . . . . .	347
76.3.2	Properties . . . . .	348
76.4	Class Fixed . . . . .	348
76.4.1	Methods . . . . .	348
76.4.2	Properties . . . . .	349
76.4.3	Class Variables . . . . .	349
76.5	Class Weibull . . . . .	349

76.5.1	Methods . . . . .	350
76.5.2	Properties . . . . .	350
76.6	Class Expo . . . . .	351
76.6.1	Methods . . . . .	351
76.6.2	Properties . . . . .	352
76.7	Class Normal . . . . .	352
76.7.1	Methods . . . . .	353
76.7.2	Properties . . . . .	353
76.8	Class Uniform . . . . .	354
76.8.1	Methods . . . . .	354
76.8.2	Properties . . . . .	355
76.9	Class LogNormal . . . . .	355
76.9.1	Methods . . . . .	356
76.9.2	Properties . . . . .	357
76.10	Class problem_definition . . . . .	357
76.10.1	Methods . . . . .	357
76.10.2	Properties . . . . .	360
76.10.3	Class Variables . . . . .	360
<b>77</b>	<b>Module gmisclib.mcmc_indexclass</b>	<b>361</b>
77.1	Functions . . . . .	361
77.2	Variables . . . . .	361
77.3	Class PriorProbDist . . . . .	362
77.3.1	Methods . . . . .	362
77.3.2	Properties . . . . .	363
77.4	Class IndexKeyError . . . . .	363
77.4.1	Methods . . . . .	363
77.4.2	Properties . . . . .	364
77.5	Class index_base . . . . .	364
77.5.1	Methods . . . . .	364
77.5.2	Properties . . . . .	366
77.6	Class MissingParameterError . . . . .	366
77.6.1	Methods . . . . .	366
77.6.2	Properties . . . . .	367
77.7	Class sampler . . . . .	367
77.7.1	Methods . . . . .	367
77.7.2	Properties . . . . .	368
77.8	Class guess . . . . .	368
77.8.1	Methods . . . . .	369
77.8.2	Properties . . . . .	372
77.9	Class index . . . . .	372
77.9.1	Methods . . . . .	373
77.9.2	Properties . . . . .	376
77.9.3	Class Variables . . . . .	376
77.10	Class index_counted . . . . .	377
77.10.1	Methods . . . . .	377
77.10.2	Properties . . . . .	381
77.10.3	Class Variables . . . . .	381
<b>78</b>	<b>Module gmisclib.mcmc_logger</b>	<b>382</b>
78.1	Functions . . . . .	382
78.2	Variables . . . . .	383

78.3	Class WildNumber . . . . .	383
78.3.1	Methods . . . . .	383
78.3.2	Properties . . . . .	383
78.4	Class logger_c . . . . .	384
78.4.1	Methods . . . . .	384
78.4.2	Properties . . . . .	385
78.5	Class logger_A . . . . .	385
78.5.1	Methods . . . . .	385
78.5.2	Properties . . . . .	386
78.6	Class logger_N . . . . .	387
78.6.1	Methods . . . . .	387
78.6.2	Properties . . . . .	388
78.7	Class lti_c . . . . .	388
78.7.1	Methods . . . . .	388
78.7.2	Properties . . . . .	389
78.8	Class NoDataError . . . . .	389
78.8.1	Methods . . . . .	389
78.8.2	Properties . . . . .	389
<b>79</b>	<b>Module gmisclib.mcmc_logtools</b>	<b>391</b>
79.1	Functions . . . . .	391
79.2	Variables . . . . .	394
79.3	Class onelog . . . . .	394
79.3.1	Methods . . . . .	394
<b>80</b>	<b>Module gmisclib.mcmc_m4p</b>	<b>395</b>
80.1	Functions . . . . .	395
80.2	Variables . . . . .	395
80.3	Class stepper . . . . .	396
80.3.1	Methods . . . . .	396
80.3.2	Properties . . . . .	399
80.3.3	Class Variables . . . . .	399
<b>81</b>	<b>Module gmisclib.mcmc_mpi</b>	<b>400</b>
81.1	Functions . . . . .	400
81.2	Variables . . . . .	400
81.3	Class stepper . . . . .	400
81.3.1	Methods . . . . .	400
81.3.2	Properties . . . . .	404
81.3.3	Class Variables . . . . .	404
<b>82</b>	<b>Module gmisclib.mcmc_newlogger</b>	<b>405</b>
82.1	Functions . . . . .	405
82.2	Variables . . . . .	408
82.3	Class WildNumber . . . . .	409
82.3.1	Methods . . . . .	409
82.3.2	Properties . . . . .	409
82.4	Class logger_base . . . . .	410
82.4.1	Methods . . . . .	410
82.4.2	Properties . . . . .	410
82.5	Class DBGlogger . . . . .	411
82.5.1	Methods . . . . .	411



82.5.2 Properties . . . . .	411
82.6 Class logger . . . . .	412
82.6.1 Methods . . . . .	412
82.6.2 Properties . . . . .	413
82.7 Class NoDataError . . . . .	414
82.7.1 Methods . . . . .	414
82.7.2 Properties . . . . .	414
82.8 Class logline . . . . .	415
82.8.1 Methods . . . . .	415
82.8.2 Properties . . . . .	415
<b>83 Module gmisclib.mcmc_pypar</b>	<b>417</b>
83.1 Functions . . . . .	417
83.2 Class stepper . . . . .	417
83.2.1 Methods . . . . .	417
83.2.2 Properties . . . . .	420
83.2.3 Class Variables . . . . .	420
<b>84 Module gmisclib.mcmc_restart</b>	<b>421</b>
84.1 Functions . . . . .	421
84.2 Variables . . . . .	421
<b>85 Module gmisclib.mcmc_socket</b>	<b>422</b>
85.1 Functions . . . . .	422
85.2 Variables . . . . .	422
85.3 Class stepper . . . . .	422
85.3.1 Methods . . . . .	422
85.3.2 Properties . . . . .	426
<b>86 Module gmisclib.multivariance_classes</b>	<b>427</b>
86.1 Functions . . . . .	427
86.2 Variables . . . . .	427
86.3 Class QuadraticNotNormalizable . . . . .	427
86.3.1 Methods . . . . .	427
86.3.2 Properties . . . . .	428
86.4 Class modeldesc . . . . .	428
86.4.1 Methods . . . . .	428
86.4.2 Class Variables . . . . .	429
86.5 Class model_with_numbers . . . . .	429
86.5.1 Methods . . . . .	429
86.5.2 Class Variables . . . . .	430
<b>87 Module gmisclib.multivariance_mm</b>	<b>431</b>
87.1 Functions . . . . .	431
87.2 Variables . . . . .	431
87.3 Class datum_c . . . . .	431
87.3.1 Methods . . . . .	431
87.4 Class multi_mu . . . . .	431
87.4.1 Methods . . . . .	432
87.4.2 Class Variables . . . . .	432
87.5 Class multi_mu_with_numbers . . . . .	433
87.5.1 Methods . . . . .	433

87.5.2 Class Variables . . . . .	434
87.6 Class multi_mu_diag . . . . .	434
87.6.1 Methods . . . . .	434
87.6.2 Class Variables . . . . .	435
87.7 Class multi_mu_diag_with_numbers . . . . .	435
87.7.1 Methods . . . . .	435
87.7.2 Class Variables . . . . .	436
<b>88 Module gmsclib.multivariate_q</b>	<b>437</b>
88.1 Variables . . . . .	437
88.2 Class quadratic . . . . .	437
88.2.1 Methods . . . . .	437
88.2.2 Class Variables . . . . .	438
88.3 Class quadratic_with_numbers . . . . .	438
88.3.1 Methods . . . . .	438
88.3.2 Class Variables . . . . .	439
88.4 Class diag_quadratic . . . . .	439
88.4.1 Methods . . . . .	439
88.4.2 Class Variables . . . . .	440
88.5 Class diag_quadratic_with_numbers . . . . .	441
88.5.1 Methods . . . . .	441
88.5.2 Class Variables . . . . .	442
<b>89 Module gmsclib.multivariate_normal</b>	<b>443</b>
89.1 Functions . . . . .	443
89.2 Variables . . . . .	443
89.3 Class multivariate_normal . . . . .	443
89.3.1 Methods . . . . .	443
89.3.2 Class Variables . . . . .	443
<b>90 Module gmsclib.named_block_file</b>	<b>444</b>
90.1 Functions . . . . .	444
90.2 Variables . . . . .	444
<b>91 Module gmsclib.nbest</b>	<b>445</b>
91.1 Functions . . . . .	445
91.2 Variables . . . . .	445
91.3 Class node . . . . .	445
91.3.1 Methods . . . . .	445
91.3.2 Class Variables . . . . .	445
<b>92 Module gmsclib.nice_hash</b>	<b>447</b>
92.1 Functions . . . . .	447
92.2 Variables . . . . .	447
92.3 Class DontHashThis . . . . .	447
92.3.1 Methods . . . . .	447
92.3.2 Properties . . . . .	448
92.4 Class NotInHash . . . . .	448
92.4.1 Methods . . . . .	448
92.4.2 Properties . . . . .	449
92.5 Class simple . . . . .	449
92.5.1 Methods . . . . .	449

92.5.2	Properties . . . . .	450
92.5.3	Class Variables . . . . .	450
92.6	Class hash . . . . .	451
92.6.1	Methods . . . . .	451
92.6.2	Properties . . . . .	452
92.6.3	Class Variables . . . . .	453
92.7	Class hash . . . . .	453
92.7.1	Methods . . . . .	453
92.7.2	Properties . . . . .	455
92.7.3	Class Variables . . . . .	455
<b>93</b>	<b>Module gmisclib.nicknames</b>	<b>456</b>
93.1	Functions . . . . .	456
93.2	Variables . . . . .	456
<b>94</b>	<b>Module gmisclib.nmf</b>	<b>457</b>
94.1	Functions . . . . .	457
94.2	Variables . . . . .	457
<b>95</b>	<b>Module gmisclib.opt</b>	<b>458</b>
95.1	Functions . . . . .	458
95.2	Variables . . . . .	461
95.3	Class OptError . . . . .	461
95.3.1	Methods . . . . .	461
95.3.2	Properties . . . . .	462
95.4	Class NoDownhill . . . . .	462
95.4.1	Methods . . . . .	462
95.4.2	Properties . . . . .	463
95.5	Class NoDerivative . . . . .	463
95.5.1	Methods . . . . .	463
95.5.2	Properties . . . . .	464
95.6	Class BadParamError . . . . .	464
95.6.1	Methods . . . . .	464
95.6.2	Properties . . . . .	465
95.6.3	Class Variables . . . . .	465
95.7	Class BadResult . . . . .	465
95.7.1	Methods . . . . .	465
95.7.2	Properties . . . . .	466
95.7.3	Class Variables . . . . .	466
95.8	Class mysem . . . . .	466
95.8.1	Methods . . . . .	466
95.8.2	Class Variables . . . . .	466
95.9	Class semclass . . . . .	467
95.9.1	Methods . . . . .	467
95.9.2	Class Variables . . . . .	467
95.10	Class prms . . . . .	467
95.10.1	Methods . . . . .	467
95.10.2	Class Variables . . . . .	468
95.11	Class LockedList . . . . .	468
95.11.1	Methods . . . . .	468
95.12	Class opt . . . . .	468
95.12.1	Methods . . . . .	469

95.12.2 Class Variables . . . . .	471
95.12.3 Instance Variables . . . . .	471
<b>96 Module gmisclib.ortho_poly</b>	<b>472</b>
96.1 Functions . . . . .	472
96.2 Variables . . . . .	472
96.3 Class ortho . . . . .	472
96.3.1 Methods . . . . .	472
96.3.2 Class Variables . . . . .	473
96.4 Class ortho_poly . . . . .	473
96.4.1 Methods . . . . .	474
96.4.2 Class Variables . . . . .	474
96.5 Class Legendre . . . . .	475
96.5.1 Methods . . . . .	475
96.5.2 Class Variables . . . . .	476
96.6 Class Chebyshev . . . . .	476
96.6.1 Methods . . . . .	477
96.6.2 Class Variables . . . . .	477
96.7 Class Chebyshev2 . . . . .	478
96.7.1 Methods . . . . .	478
96.7.2 Class Variables . . . . .	479
96.8 Class SinCos . . . . .	479
96.8.1 Methods . . . . .	480
96.8.2 Class Variables . . . . .	480
96.9 Class SLTB . . . . .	481
96.9.1 Methods . . . . .	481
96.9.2 Class Variables . . . . .	482
<b>97 Module gmisclib.parse_tree_number</b>	<b>483</b>
97.1 Functions . . . . .	483
97.2 Variables . . . . .	483
97.3 Class Array . . . . .	483
97.3.1 Methods . . . . .	484
97.3.2 Properties . . . . .	484
97.4 Class output_mixin . . . . .	485
97.4.1 Methods . . . . .	485
97.4.2 Properties . . . . .	485
97.5 Class abstract_number . . . . .	486
97.5.1 Methods . . . . .	486
97.5.2 Properties . . . . .	487
97.6 Class Expression . . . . .	487
97.6.1 Methods . . . . .	487
97.6.2 Properties . . . . .	489
97.7 Class Name . . . . .	490
97.7.1 Methods . . . . .	490
97.7.2 Properties . . . . .	492
97.8 Class Float . . . . .	492
97.8.1 Methods . . . . .	492
97.8.2 Properties . . . . .	494
97.9 Class operator1 . . . . .	494
97.9.1 Methods . . . . .	495
97.9.2 Properties . . . . .	496

97.9.3 Class Variables . . . . .	497
97.10 Class elementof . . . . .	497
97.10.1 Methods . . . . .	497
97.10.2 Properties . . . . .	499
97.10.3 Class Variables . . . . .	499
97.11 Class operatorNg1 . . . . .	500
97.11.1 Methods . . . . .	500
97.11.2 Properties . . . . .	502
97.11.3 Class Variables . . . . .	502
<b>98 Module gmisclib.permute</b>	<b>503</b>
98.1 Functions . . . . .	503
98.2 Variables . . . . .	503
<b>99 Module gmisclib.pylab_oneaxis</b>	<b>504</b>
99.1 Functions . . . . .	504
99.2 Variables . . . . .	504
<b>100 Module gmisclib.pylab_starplot</b>	<b>505</b>
100.1 Functions . . . . .	505
100.2 Variables . . . . .	505
100.3 Class errorbar_maker . . . . .	505
100.3.1 Methods . . . . .	506
100.3.2 Properties . . . . .	506
100.4 Class errorbar . . . . .	506
100.4.1 Methods . . . . .	506
100.4.2 Properties . . . . .	507
<b>101 Module gmisclib.read_dicom</b>	<b>508</b>
101.1 Functions . . . . .	508
101.2 Variables . . . . .	508
101.3 Class img_with_mx . . . . .	508
101.3.1 Methods . . . . .	508
101.3.2 Properties . . . . .	509
<b>102 Module gmisclib.robust_multivariate</b>	<b>510</b>
102.1 Functions . . . . .	510
102.2 Variables . . . . .	510
<b>103 Module gmisclib.root</b>	<b>511</b>
103.1 Functions . . . . .	511
103.2 Variables . . . . .	511
<b>104 Module gmisclib.rubber_array</b>	<b>512</b>
104.1 Variables . . . . .	512
104.2 Class ext_block . . . . .	512
104.2.1 Methods . . . . .	512
104.2.2 Properties . . . . .	512
104.3 Class extensible_array . . . . .	513
104.3.1 Methods . . . . .	513
104.3.2 Properties . . . . .	513
<b>105 Module gmisclib.s_lin_fit</b>	<b>514</b>

105.1	Functions	514
105.2	Variables	514
105.3	Class Error	514
105.3.1	Methods	515
105.3.2	Properties	515
105.4	Class NoDataError	515
105.4.1	Methods	516
105.4.2	Properties	516
<b>106</b>	<b>Module gmisclib.sbd_array</b>	<b>517</b>
106.1	Functions	517
106.2	Variables	517
106.3	Class sbd	518
106.3.1	Methods	518
106.4	Class NoSolutionError	519
106.4.1	Methods	519
106.4.2	Properties	519
<b>107</b>	<b>Module gmisclib.segmentfile</b>	<b>520</b>
107.1	Functions	520
107.2	Variables	520
107.3	Class segment	520
107.3.1	Methods	520
107.4	Class group	521
107.4.1	Methods	521
<b>108</b>	<b>Module gmisclib.sharp_energy</b>	<b>522</b>
108.1	Functions	522
<b>109</b>	<b>Module gmisclib.solve_sum_abs</b>	<b>523</b>
109.1	Functions	523
109.2	Variables	524
<b>110</b>	<b>Module gmisclib.spread_jobs</b>	<b>525</b>
110.1	Functions	527
110.2	Variables	528
110.3	Class notComputed	528
110.3.1	Methods	528
110.3.2	Properties	528
110.4	Class NoResponse	529
110.4.1	Methods	529
110.4.2	Properties	529
110.5	Class RemoteException	530
110.5.1	Methods	530
110.5.2	Properties	530
110.6	Class TooBusy	531
110.6.1	Methods	531
110.6.2	Properties	531
110.7	Class PastPerformance	531
110.7.1	Methods	532
110.7.2	Properties	533
110.7.3	Class Variables	533

110.8	Class CannotCreateConnection . . . . .	533
110.8.1	Methods . . . . .	533
110.8.2	Properties . . . . .	534
110.9	Class Connection . . . . .	534
110.9.1	Methods . . . . .	534
110.9.2	Properties . . . . .	535
110.9.3	Class Variables . . . . .	535
110.9.4	Instance Variables . . . . .	536
110.10	Class Connection_subprocess . . . . .	536
110.10.1	Methods . . . . .	536
110.10.2	Properties . . . . .	537
110.10.3	Class Variables . . . . .	538
110.10.4	Instance Variables . . . . .	538
110.11	Class workers_c . . . . .	538
110.11.1	Methods . . . . .	538
110.11.2	Properties . . . . .	539
110.12	Class unpickled_pseudofile . . . . .	539
110.12.1	Methods . . . . .	539
<b>111</b>	<b>Module gmisclib.sqlbase . . . . .</b>	<b>540</b>
111.1	Functions . . . . .	540
111.2	Variables . . . . .	542
111.3	Class DBMetaClass . . . . .	542
111.3.1	Methods . . . . .	542
111.3.2	Properties . . . . .	542
111.4	Class SQLError . . . . .	543
111.4.1	Methods . . . . .	543
111.4.2	Properties . . . . .	543
111.5	Class NoSuchTable . . . . .	544
111.5.1	Methods . . . . .	544
111.5.2	Properties . . . . .	544
111.6	Class ColumnMismatchError . . . . .	545
111.6.1	Methods . . . . .	545
111.6.2	Properties . . . . .	545
111.7	Class DB . . . . .	546
111.7.1	Methods . . . . .	546
111.7.2	Properties . . . . .	549
111.7.3	Class Variables . . . . .	549
111.8	Class DBx . . . . .	549
111.8.1	Methods . . . . .	550
111.8.2	Properties . . . . .	552
111.8.3	Class Variables . . . . .	552
111.8.4	Instance Variables . . . . .	553
<b>112</b>	<b>Module gmisclib.stats . . . . .</b>	<b>554</b>
112.1	Functions . . . . .	554
112.2	Variables . . . . .	556
<b>113</b>	<b>Module gmisclib.system_load . . . . .</b>	<b>557</b>
113.1	Functions . . . . .	557
113.2	Variables . . . . .	557
113.3	Class pstat . . . . .	558

113.3.1 Methods . . . . .	558
113.3.2 Properties . . . . .	558
<b>114Module gmisclib.threaded_io</b>	<b>559</b>
114.1Functions . . . . .	559
114.2Variables . . . . .	559
114.3Class CaughtException . . . . .	559
114.3.1 Methods . . . . .	560
114.3.2 Properties . . . . .	560
114.4Class threading_with_result . . . . .	560
114.4.1 Methods . . . . .	561
114.4.2 Properties . . . . .	561
114.5Class Thread_pool . . . . .	561
114.5.1 Methods . . . . .	562
114.5.2 Properties . . . . .	562
114.6Class Thread_poolW . . . . .	562
114.6.1 Methods . . . . .	563
114.6.2 Properties . . . . .	563
114.7Class Thread_poolW . . . . .	564
114.7.1 Methods . . . . .	564
114.7.2 Properties . . . . .	564
114.8Class Thread_poolR . . . . .	565
114.8.1 Methods . . . . .	565
114.8.2 Properties . . . . .	566
<b>115Module gmisclib.tsops</b>	<b>567</b>
115.1Functions . . . . .	567
115.2Variables . . . . .	569
115.3Class axis . . . . .	569
115.3.1 Methods . . . . .	569
<b>116Module gmisclib.wavesurfer_lab</b>	<b>571</b>
116.1Functions . . . . .	571
116.2Variables . . . . .	571
<b>117Module gmisclib.wavio</b>	<b>573</b>
117.1Functions . . . . .	573
117.2Variables . . . . .	574
117.3Class Overflow . . . . .	575
117.3.1 Methods . . . . .	575
117.3.2 Properties . . . . .	575
117.4Class BadFileFormatError . . . . .	576
117.4.1 Methods . . . . .	576
117.4.2 Properties . . . . .	576
<b>118Module gmisclib.weighted_percentile</b>	<b>577</b>
118.1Functions . . . . .	577
118.2Variables . . . . .	578
<b>119Module gmisclib.xmlmisc</b>	<b>580</b>
119.1Functions . . . . .	580
119.2Variables . . . . .	580



119.3	Class <code>loc_finder</code> . . . . .	580
119.3.1	Methods . . . . .	581
119.3.2	Properties . . . . .	582
<b>120</b>	<b>Module <code>gmisclib.xwaves_errs</code></b>	<b>583</b>
120.1	Variables . . . . .	583
120.2	Class <code>Error</code> . . . . .	583
120.2.1	Methods . . . . .	583
120.2.2	Properties . . . . .	584
120.3	Class <code>NoSuchFileError</code> . . . . .	584
120.3.1	Methods . . . . .	584
120.3.2	Properties . . . . .	585
120.4	Class <code>BadFileFormatError</code> . . . . .	585
120.4.1	Methods . . . . .	585
120.4.2	Properties . . . . .	586
120.5	Class <code>DataError</code> . . . . .	586
120.5.1	Methods . . . . .	586
120.5.2	Properties . . . . .	587
120.6	Class <code>DataOutOfOrderError</code> . . . . .	587
120.6.1	Methods . . . . .	587
120.6.2	Properties . . . . .	588
<b>121</b>	<b>Module <code>gmisclib.xwaves_lab</code></b>	<b>589</b>
121.1	Functions . . . . .	589
121.2	Variables . . . . .	589
<b>122</b>	<b>Module <code>gmisclib.xwaves_mark</code></b>	<b>590</b>
122.1	Functions . . . . .	590
122.2	Variables . . . . .	590
<b>123</b>	<b>Module <code>gpk_wavio</code></b>	<b>591</b>
123.1	Variables . . . . .	591
<b>124</b>	<b>Module <code>mcmc</code></b>	<b>592</b>
124.1	Functions . . . . .	594
124.2	Variables . . . . .	594
124.3	Class <code>def_logger</code> . . . . .	594
124.3.1	Methods . . . . .	594
<b>125</b>	<b>Module <code>q3html</code></b>	<b>595</b>
125.1	Functions . . . . .	596
125.2	Variables . . . . .	597
125.3	Class <code>lineC</code> . . . . .	598
125.3.1	Methods . . . . .	598
<b>126</b>	<b>Module <code>run_several</code></b>	<b>599</b>
126.1	Functions . . . . .	600
126.2	Variables . . . . .	600
<b>127</b>	<b>Module <code>select_fiat_entries</code></b>	<b>601</b>
127.1	Functions . . . . .	601
127.2	Variables . . . . .	601
127.3	Class <code>writer</code> . . . . .	602

---

127.3.1 Methods . . . . .	602
127.3.2 Properties . . . . .	603
<b>Index</b>	<b>604</b>

# 1 Package gmisclib

## 1.1 Modules

- **HList**: This reads HTK feature vectors via the HList program.  
(Section 2, p. 32)
- **HTK\_HMM\_io**: This reads in HMM files produced by HTK.  
(Section 3, p. 33)
- **HTK\_MLF\_io**: This reads MLF (Master Label Files) for/from the HTK speech recognition toolkit.  
(Section 4, p. 43)
- **MFCCFile** (Section 5, p. 48)
- **MLF\_file** (Section 6, p. 49)
- **Num**: This is a compatibility module, allowing python to work with either scipy, Numeric, or numarray.  
(Section 7, p. 53)
- **Numeric\_gpk** (Section 8, p. 58)
- **accent\_spec**: This module provides a way of safely specifying accent positions in running text.  
(Section 9, p. 64)
- **array\_window**: Defines array\_window class.  
(Section 10, p. 67)
- **avio**: Module to parse and create lines in "a=v;" format.  
(Section 11, p. 69)
- **bark\_scale** (Section 12, p. 73)
- **beamsearch** (Section 13, p. 74)
- **blue\_data\_attributes**: This chooses samples such that the specified attributes are broadly distributed.  
(Section 14, p. 75)
- **blue\_data\_selector**: This takes anticorrelated samples from a sequence of data.  
(Section 15, p. 77)
- **cache** (Section 16, p. 81)
- **chunkio**: These are I/O routines to allow you to write stuff like arrays and dictionaries (and arrays of dictionaries) to a human-readable file.  
(Section 17, p. 90)
- **convex\_hull2d**: convexhull.py  
(Section 18, p. 101)
- **dict\_vector**: Vectors of numbers, but indexed as a dictionary.  
(Section 19, p. 103)
- **dictops**: Operations on dictionaries.  
(Section 20, p. 109)
- **die** (Section 21, p. 122)
- **die2** (Section 22, p. 126)
- **do\_exec**: Execute a \*nix command and capture the output.  
(Section 23, p. 128)
- **ds9\_region** (Section 24, p. 129)
- **dtw4** (Section 25, p. 130)
- **dyn\_prog**: Viterbi search  
(Section 26, p. 137)
- **edit\_distance**: Levenshtein (edit) distance between two strings of symbols.  
(Section 27, p. 138)
- **entropy** (Section 28, p. 143)
- **erb\_scale**: ERB Perceptual frequency scale.  
(Section 29, p. 144)
- **evolution** (Section 30, p. 145)

- **fake\_file**: A class that implements a simple file in memory.  
(Section 31, p. 146)
- **fiat\_merge** (Section 32, p. 147)
- **fiatio**: Fiatio reads and writes an extension of the FIAT file format originally defined by David Wittman (UC Davis).  
(Section 33, p. 148)
- **find\_home**: This lets you find files with just a path name relative to where the program's executable code script sits.  
(Section 34, p. 161)
- **find\_ngram**: This module lets you search through label files to find particular ngrams.  
(Section 35, p. 162)
- **findleak** (Section 36, p. 164)
- **ftest** (Section 37, p. 165)
- **fuzzygraph** (Section 38, p. 167)
- **g2\_select** (Section 39, p. 168)
- **g\_closure**: This module defines closures.  
(Section 40, p. 171)
- **g\_datetime** (Section 41, p. 174)
- **g\_ds9**: Handles interactions with ds9 (<http://hea-www.harvard.edu/RD/ds9/>).  
(Section 42, p. 175)
- **g\_encode**: This module allows strings to be encoded into a reduced subset.  
(Section 43, p. 190)
- **g\_entropy**: This returns the entropy of a probability distribution that produced a given sample.  
(Section 44, p. 192)
- **g\_exec**: Run a Linux command and capture the result.  
(Section 45, p. 193)
- **g\_implements**: This module tells you if an object's signature matches a class.  
(Section 46, p. 199)
- **g\_keyed\_accum** (Section 47, p. 203)
- **g\_lin\_fit**: Generic linear best fits.  
(Section 48, p. 204)
- **g\_localfit**: Fit a linear transform to a bunch of tt input/output vectors.  
(Section 49, p. 206)
- **g\_pipe**: A multithreaded version of os.popen2().  
(Section 50, p. 212)
- **g\_pipe\_old**: A multithreaded version of os.popen2().  
(Section 51, p. 214)
- **g\_place\_label** (Section 52, p. 216)
- **g\_pylab**: This class works in concert with bin/pylab\_server.py to allow you to display simple pylab/matplotlib graphics on another machine.  
(Section 53, p. 220)
- **g\_selector** (Section 54, p. 221)
- **g\_unicode**: Functions to make unicode handling easier for Python 2.X.  
(Section 55, p. 226)
- **gpk\_getopt** (Section 56, p. 227)
- **gpk\_hdr** (Section 57, p. 228)
- **gpk\_lapack** (Section 58, p. 229)
- **gpk\_lsq** (Section 59, p. 231)
- **gpk\_rlsq** (Section 60, p. 240)
- **gpk\_writer** (Section 61, p. 248)
- **gpkmisc** (Section 62, p. 251)

- **hilbert\_xform** (Section 63, p. 261)
- **kl\_dist**: Suppose there is a random variable with true distribution  $p$ .  
(Section 64, p. 262)
- **load\_mod**: This module has functions that help you dynamically import modules.  
(Section 65, p. 266)
- **lreg\_fill** (Section 66, p. 269)
- **makemake**: This module is designed to build makefiles.  
(Section 67, p. 270)
- **matrix\_arrange**: Take a covariance-like matrix, and re-order it to be close to a diagonal matrix.  
(Section 68, p. 273)
- **matrix\_arrange\_entropy**: This rotates a matrix by multiplying with a unitary matrix so that the resulting elements are either nearly zero or relatively large.  
(Section 69, p. 274)
- **matrix\_rearrange\_labels**: This module helps you plot confusion matrices or similar images where the axis labels do not have a natural order.  
(Section 70, p. 275)
- **mcmc**: Bootstrap Markov-Chain Monte-Carlo algorithm.  
(Section 71, p. 283)
- **mcmcS2**: Markov-Chain Monte-Carlo algorithms.  
(Section 72, p. 323)
- **mcmc\_big**: An extension of mcmc that includes new stepping algorithms.  
(Section 73, p. 324)
- **mcmc\_cooperate** (Section 74, p. 329)
- **mcmc\_helper**: This is a helper module to make use of mcmc.py and mcmc\_big.py.  
(Section 75, p. 335)
- **mcmc\_idxr** (Section 76, p. 347)
- **mcmc\_indexclass**: This module provides several classes to manage the parameters of an algorithm, particularly so that you can run the mcmc.py optimizer on it.  
(Section 77, p. 361)
- **mcmc\_logger** (Section 78, p. 382)
- **mcmc\_logtools** (Section 79, p. 391)
- **mcmc\_m4p**: This is a helper module to make use of mcmc.py and mcmc\_big.py.  
(Section 80, p. 395)
- **mcmc\_mpi**: This is a helper module to make use of mcmc.py and mcmc\_big.py.  
(Section 81, p. 400)
- **mcmc\_newlogger** (Section 82, p. 405)
- **mcmc\_pypar**: This is a helper module to make use of mcmc.py and mcmc\_big.py.  
(Section 83, p. 417)
- **mcmc\_restart** (Section 84, p. 421)
- **mcmc\_socket**: This is a helper module to make use of mcmc.py and mcmc\_big.py.  
(Section 85, p. 422)
- **multivariance\_classes**: Support module for multivariance.py  
(Section 86, p. 427)
- **multivariance\_mm**: This a helper module for multivariance.py  
(Section 87, p. 431)
- **multivariance\_q**: This a helper module for multivariance.py  
(Section 88, p. 437)
- **multivariate\_normal** (Section 89, p. 443)
- **named\_block\_file**: Read and write files in the form [label] text text text [label2] text text ...  
(Section 90, p. 444)
- **nbest**: Beam search through a graph.

- (Section 91, p. 445)
- **nice\_hash** (Section 92, p. 447)
- **nicknames** (Section 93, p. 456)
- **nmf**:  $V=WH$ , where  $W$  and  $H$  are non-negative.  
(Section 94, p. 457)
- **opt**: This module is a Levenberg-Marquardt optimizer with stem-size control for numeric differentiation.  
(Section 95, p. 458)
- **ortho\_poly** (Section 96, p. 472)
- **parse\_tree\_number** (Section 97, p. 483)
- **permute**: Find different permutations of an array.  
(Section 98, p. 503)
- **pylab\_oneaxis**: You can control the axis tick and grid properties  
(Section 99, p. 504)
- **pylab\_starplot**: This makes a little 'star' of error bars, using pylab/matplotlib.  
(Section 100, p. 505)
- **read\_dicom** (Section 101, p. 508)
- **robust\_multivariate**: Does a robust estimate of covariance.  
(Section 102, p. 510)
- **root**: Solve a 1-dimensional equation to find the roots.  
(Section 103, p. 511)
- **rubber\_array**: This acts like a dictionary, but indexed by integers and stored rather more efficiently, at least in terms of memory.  
(Section 104, p. 512)
- **s\_lin\_fit**: Fit a plane to some data.  
(Section 105, p. 514)
- **sbd\_array** (Section 106, p. 517)
- **segmentfile**: This parses a '.in' file from xwaves.  
(Section 107, p. 520)
- **sharp\_energy** (Section 108, p. 522)
- **solve\_sum\_abs**: Solves various equations involving minimizing the sum of absolute values of things.  
(Section 109, p. 523)
- **spread\_jobs**: A module that starts a bunch of subprocesses and distributes work amongst them, then collects the results.  
(Section 110, p. 525)
- **sqlbase**: This is a module for using a SQLite database as a collection of python objects.  
(Section 111, p. 540)
- **stats**: Some of these functions, specifically `f.value()`, `fprob()`, `betai()`, and `betacf()`, are taken from `stats.py` "A collection of basic statistical functions for python." by Gary Strangman.  
(Section 112, p. 554)
- **system\_load** (Section 113, p. 557)
- **threaded\_io**: This is designed to let you do asynchronous I/O conveniently.  
(Section 114, p. 559)
- **tsops**: Do mathematical operations on time series, where the two operands don't necessarily have the same sampling times.  
(Section 115, p. 567)
- **wavesurfer\_lab**: When used as a script, this reads label files produced by wavesurfer and prints the result.  
(Section 116, p. 571)
- **wavio**: When run as a script: `python ~/lib/wavio.py [-g gain] -wavin|-wavout infile outfile` Reads or writes .wav files from any format supported by `gpkimgclass.py`.

(Section 117, p. 573)

- **weighted\_percentile**: This computes order statistics on data with weights.

(Section 118, p. 577)

- **xmlmisc**: This contains helper functions and classes for processing XML, based on the ElementTree module.

(Section 119, p. 580)

- **xwaves\_errs**: Errors for reading label (typically speech transcription) files.

(Section 120, p. 583)

- **xwaves\_lab**: Reads label files produced by ESPS xwaves.

(Section 121, p. 589)

- **xwaves\_mark**: Read in .in files produced by ESPS xmark.

(Section 122, p. 590)

## 1.2 Variables

Name	Description
__package__	<b>Value:</b> None

## 2 Module *gmisclib.HList*

This reads HTK feature vectors via the HList program.

### 2.1 Functions

<code>read(<i>fname</i>)</code>
---------------------------------

### 2.2 Variables

Name	Description
<code>--package--</code>	<b>Value:</b> <code>'gmisclib'</code>



### 3 Module `gmisclib.HTK_HMM_io`

This reads in HMM files produced by HTK.

#### 3.1 Functions

<code>deref(<i>x</i>)</code>
------------------------------

<code>read_floats_t(<i>vtype</i>, <i>dim</i>, <i>t</i>, <i>tok</i>, <i>phoneme</i>='???')</code>
--

<code>read_floats_b(<i>vtype</i>, <i>dim</i>, <i>t</i>, <i>tok</i>, <i>phoneme</i>='???' , <i>name</i>=None)</code>
---

<code>read_vec(<i>dim</i>, <i>n</i>, <i>t</i>, <i>name</i>=None, <i>vtype</i>=None)</code>
--

<code>read(<i>fdlist</i>)</code>
----------------------------------

<code>HTKformat(<i>s</i>)</code>
----------------------------------

Format for HTK state name.
----------------------------

<code>dict_format(<i>s</i>)</code>
------------------------------------

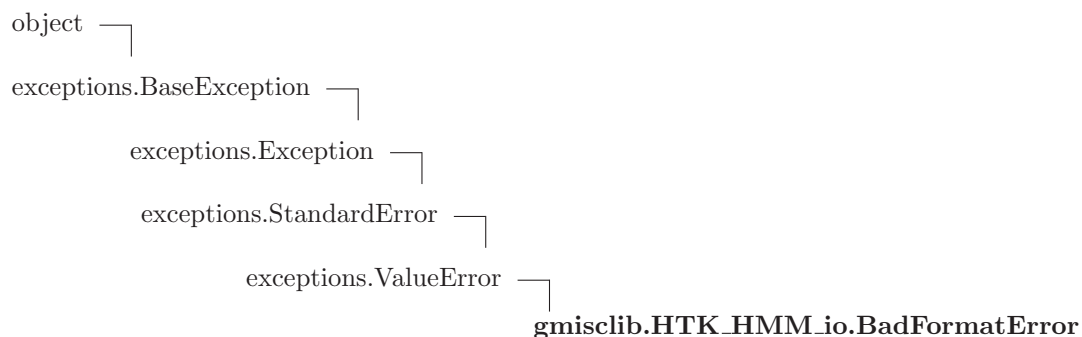
Format for center [bracket] in HTK dictionary.
--

<code>test()</code>
---------------------

#### 3.2 Variables

Name	Description
I	Value: ' '
__package__	Value: 'gmisclib'

### 3.3 Class *BadFormatError*



#### 3.3.1 Methods

`__init__(self, t, s)`  
`x.__init__(...)` initializes `x`; see `help(type(x))` for signature  
 Overrides: `object.__init__` `exitit`(inherited documentation)

*Inherited from `exceptions.ValueError`*

`__new__()`

*Inherited from `exceptions.BaseException`*

`__delattr__()`, `__getattr__()`, `__getitem__()`, `__getslice__()`, `__reduce__()`, `__repr__()`,  
`__setattr__()`, `__setstate__()`, `__str__()`, `__unicode__()`

*Inherited from `object`*

`__format__()`, `__hash__()`, `__reduce_ex__()`, `__sizeof__()`, `__subclasshook__()`

#### 3.3.2 Properties

Name	Description
<i>Inherited from <code>exceptions.BaseException</code></i>	
<code>args</code> , <code>message</code>	
<i>Inherited from <code>object</code></i>	
<code>__class__</code>	

### 3.4 Class *vec*



#### 3.4.1 Methods

**`__init__(self, vtype, v, name=None)`**

`x.__init__(...)` initializes `x`; see `help(type(x))` for signature

**Parameters**

**`vtype`:** (*type=string, one of 'u', 'v', 'i', 't'*)

Overrides: `object.__init__`

**`__eq__(self, other)`**

**`__hash__(self)`**

`hash(x)`

Overrides: `object.__hash__` `exitit`(inherited documentation)

**`__repr__(self)`**

`repr(x)`

Overrides: `object.__repr__` `exitit`(inherited documentation)

**`write(self)`**

#### *Inherited from object*

`__delattr__()`, `__format__()`, `__getattr__()`, `__new__()`, `__reduce__()`, `__reduce_ex__()`,  
`__setattr__()`, `__sizeof__()`, `__str__()`, `__subclasshook__()`

#### 3.4.2 Properties

Name	Description
<i>Inherited from object</i>	
<code>__class__</code>	

#### 3.4.3 Class Variables

Name	Description
cname	<b>Value:</b> {'i': 'InvCov', 't': 'TransP', 'u': 'Mean', 'v': 'Variance'}
dim	<b>Value:</b> {'i': -2, 't': 2, 'u': 1, 'v': 1}

### 3.5 Class mixture

object └─  
gmisclib.HTK\_HMM\_io.mixture

#### 3.5.1 Methods

**\_\_init\_\_**(*self*, *wt*=1.0, *mean*=None, *var*=None, *name*=None)

*x*.\_\_init\_\_(...) initializes *x*; see help(type(*x*)) for signature

Overrides: object.\_\_init\_\_ extit(inherited documentation)

**\_\_hash\_\_**(*self*)

hash(*x*)

Overrides: object.\_\_hash\_\_ extit(inherited documentation)

**\_\_eq\_\_**(*self*, *other*)

**check**(*self*)

**\_\_repr\_\_**(*self*)

repr(*x*)

Overrides: object.\_\_repr\_\_ extit(inherited documentation)

**write**(*self*, *imix*=None)

#### *Inherited from object*

\_\_delattr\_\_(), \_\_format\_\_(), \_\_getattr\_\_(), \_\_new\_\_(), \_\_reduce\_\_(), \_\_reduce\_ex\_\_(),  
\_\_setattr\_\_(), \_\_sizeof\_\_(), \_\_str\_\_(), \_\_subclasshook\_\_()

### 3.5.2 Properties

Name	Description
<i>Inherited from object</i>	
<code>--class--</code>	

### 3.5.3 Class Variables

Name	Description
<code>code</code>	<b>Value:</b> 'm'

## 3.6 Class ref

object └─ **gmisclib.HTK\_HMM\_io.ref**

### 3.6.1 Methods

**`--init--`**(*self*, *thing*)  
*x*.**`--init--`**(...) initializes *x*; see `help(type(x))` for signature  
 Overrides: `object.--init--` `exitit`(inherited documentation)

**`write`**(*self*)

#### *Inherited from object*

`--delattr--`(), `--format--`(), `--getattr--`(), `--hash--`(), `--new--`(), `--reduce--`(), `--reduce_ex--`(),  
`--repr--`(), `--setattr--`(), `--sizeof--`(), `--str--`(), `--subclasshook--`()

### 3.6.2 Properties

Name	Description
<i>Inherited from object</i>	
<code>--class--</code>	

### 3.7 Class state



#### 3.7.1 Methods

```
__init__(self, name=None)
x.__init__(...) initializes x; see help(type(x)) for signature
Overrides: object.__init__ extit(inherited documentation)
```

```
create_mx(self, i)
```

```
check(self, istate=-1, phone='???')
```

```
__repr__(self)
repr(x)
Overrides: object.__repr__ extit(inherited documentation)
```

```
write(self, istate)
```

#### *Inherited from object*

```
__delattr__(), __format__(), __getattr__(), __hash__(), __new__(), __reduce__(), __reduce_ex__(),
__setattr__(), __sizeof__(), __str__(), __subclasshook__()
```

#### 3.7.2 Properties

Name	Description
<i>Inherited from object</i>	
<code>__class__</code>	

### 3.8 Class *hmm*



### 3.8.1 Methods

```
__init__(self, numstates, name=None)
```

*x*.**\_\_init\_\_**(...) initializes *x*; see `help(type(x))` for signature

Overrides: `object.__init__` `extit`(inherited documentation)

```
check(self)
```

```
write(self)
```

#### *Inherited from object*

`__delattr__()`, `__format__()`, `__getattr__()`, `__hash__()`, `__new__()`, `__reduce__()`, `__reduce_ex__()`,  
`__repr__()`, `__setattr__()`, `__sizeof__()`, `__str__()`, `__subclasshook__()`

### 3.8.2 Properties

Name	Description
<i>Inherited from object</i>	
<code>__class__</code>	

## 3.9 Class *hmmset*



### 3.9.1 Methods

```
__init__(self)
```

*x*.**\_\_init\_\_**(...) initializes *x*; see `help(type(x))` for signature

Overrides: `object.__init__` `extit`(inherited documentation)

```
twiddle_o(self)
```

```
write(self)
```

#### *Inherited from object*

`__delattr__()`, `__format__()`, `__getattr__()`, `__hash__()`, `__new__()`, `__reduce__()`, `__reduce_ex__()`,  
`__repr__()`, `__setattr__()`, `__sizeof__()`, `__str__()`, `__subclasshook__()`

### 3.9.2 Properties

Name	Description
<i>Inherited from object</i>	
<code>__class__</code>	

## 3.10 Class tokstream

object └─ **gmisclib.HTK\_HMM\_io.tokstream**

This breaks the HTK HMM file down into tokens.

### 3.10.1 Methods

<b><code>__init__(self, fdlist)</code></b> <code>x.__init__(...)</code> initializes x; see <code>help(type(x))</code> for signature Overrides: <code>object.__init__</code> <code>exitit</code> (inherited documentation)
<b><code>nextline(self)</code></b> Get the next line from the input files(s). Sets <code>self.line</code> . <b>Return Value</b> the line. <i>(type=</i> str <i> or </i> None <i>.)</i>
<b><code>__nonzero__(self)</code></b> <b>Return Value</b> True when there is more work to be done.



**get(*self*)**

Get a token. This also sets `self.fulltok` with the full text of the token.

**Return Value**

The regular expression that matches the next token, and the relevant text.

(*type*=(*compiled regular expression*, *str*))

**Raises**

**BadFormatError** When the remainder of the line doesn't match any legal pattern.

**get\_name(*self*)****get\_bracket(*self*)****get\_int(*self*)****get\_float(*self*)****Inherited from object**

`--delattr--()`, `--format--()`, `--getattr--()`, `--hash--()`, `--new--()`, `--reduce--()`, `--reduce_ex--()`, `--repr--()`, `--setattr--()`, `--sizeof--()`, `--str--()`, `--subclasshook--()`

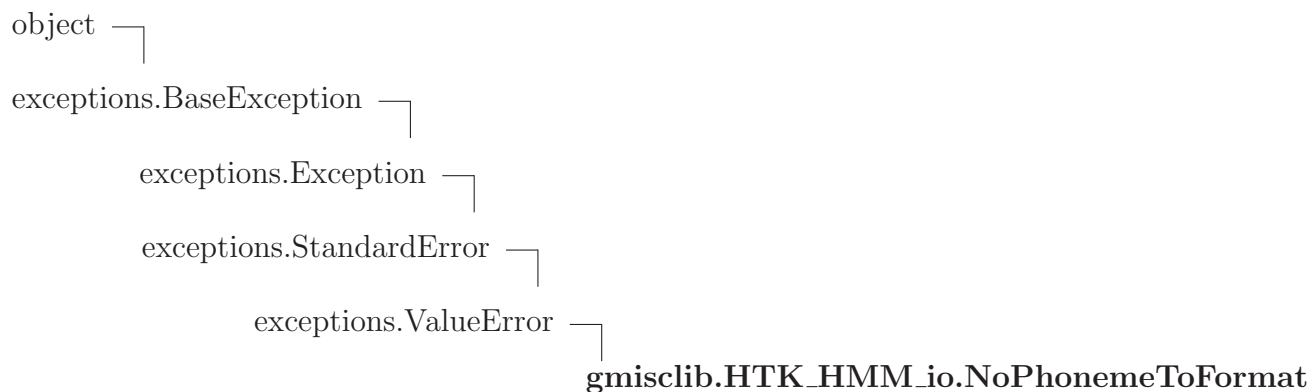
**3.10.2 Properties**

Name	Description
<i>Inherited from object</i>	
<code>--class--</code>	

**3.10.3 Class Variables**

Name	Description
<code>int</code>	<b>Value:</b> <code>re.compile(r'([\+-]?[0-9]+)')</code>
<code>float</code>	<b>Value:</b> <code>re.compile(r'([\+-]?[0-9]*\.[0-9]*[e0-9\+-]*)')</code>
<code>bracket</code>	<b>Value:</b> <code>re.compile(r'&lt;s*(\w+)\s*&gt;')</code>
<code>macro</code>	<b>Value:</b> <code>re.compile(r'~([a-zA-Z])')</code>
<code>name</code>	<b>Value:</b> <code>re.compile(r'"([a-zA-Z_]\w*)"')</code>

### 3.11 Class *NoPhonemeToFormat*



#### 3.11.1 Methods

**`--init--`**(*self*, \**s*)

*x*.**`--init--`**(...) initializes *x*; see `help(type(x))` for signature

Overrides: *object*.**`--init--`** `extit`(inherited documentation)

*Inherited from exceptions.ValueError*

**`--new--`**()

*Inherited from exceptions.BaseException*

**`--delattr--`**(), **`--getattr--`**(), **`--getitem--`**(), **`--getslice--`**(), **`--reduce--`**(), **`--repr--`**(),  
**`--setattr--`**(), **`--setstate--`**(), **`--str--`**(), **`--unicode--`**()

*Inherited from object*

**`--format--`**(), **`--hash--`**(), **`--reduce.ex--`**(), **`--sizeof--`**(), **`--subclasshook--`**()

#### 3.11.2 Properties

Name	Description
<i>Inherited from exceptions.BaseException</i>	
args, message	
<i>Inherited from object</i>	
<b><code>--class--</code></b>	

## 4 Module `gmisclib.HTK_MLF_io`

This reads MLF (Master Label Files) for/from the HTK speech recognition toolkit.

### 4.1 Functions

**`parse_label_line(s, tq)`**

This parses a line from a MLF into a tuple.

**Parameters**

**`tq`**: time quantum (normally 1e-7 seconds)

(*type=float*)

**`s`**: line to be parsed

(*type=str*)

**Raises**

`BadFileFormatError` when parsing is not possible.

**`readone(mlf_efn, postfix='.wav', datapath='.', strict=True, findfile=True, pathedit=None, time_quantum=1e-07, verbose=False)`**

Read a single set of labels from a MLF file. You specify the labels as part of the extended filename, like this: `name_of_MLF_file:name_of_labels'`. The function returns only a single value and raises an exception if the extended filename is ambiguous.

**Parameters**

**`mlf_efn`**: (*type=string in the form "F:S" @rtype dict()*)

**Return Value**

a dictionary that describes the labels as per `readiter`.

```
readiter(mlf_fn, postfix='.wav', datapath='.', strict=True, findfile=True,
pathedit=None, time_quantum=1e-07, verbose=False)
```

Read a HTK Master Label (MLF) file. *Datapath* and *pathedit* are ways to deal with the situation where the MLF file has been moved, or (for other reasons) the filenames in the MLF file don't point to the actual data.

### Parameters

***mlf\_fn*:** filename of the data file.  
*(type=*`str`*)*

***strict*:** If true, raise an exception if an audio file cannot be found.  
*(type=*`bool`*)*

***time\_quantum*:** A factor to convert from the time information in the MLF to real units of time (like seconds). Ideally, *time\_quantum*=1e-7 seconds for MLF files, but that isn't exactly accurate for some sampling rates (like 11025 samples/sec) when the sampling interval is not an integral multiple of 100 nanoseconds.  
*(type=*`float`*)*

### Return Value

sequence of `{'filespec':path, 'd': d, 'f': f, 'symbols': [...]} , ...`. This is an iterator of dictionaries. Each dictionary corresponds to one utterance, or one "label file" in the MLF. Attributes `'d'` and `'f'` are only present if *findfile*==True; `os.path.join(x['d'], x['f'])` is a path to the corresponding audio. `x['filespec']` is the path information in the MLF, `x['i']` is an `int` indexing which utterance this is within the MLF, and `x['symbols']` is the label information for that utterance. It is a list of tuples produced by `parse_label_line`.

*(type=an iterator producing dict(str: various))*

**read**(*mlf\_fn*, *\*\*kw*)

Read a HTK Master Label (MLF) file. Datapath and pathedit are ways to deal with the situation where the MLF file has been moved, or (for other reasons) the filenames in the MLF file don't point to the actual data.

**Parameters**

*mlf\_fn*: filename of the data file.

(*type=**str*)

*kw*: Key-value parameters from *readiter*.

**Return Value**

see *readiter* for details.

(*type=**list of dict*. See *readiter* for details.)

## 4.2 Variables

Name	Description
TIME_QUANTUM	Value: 1e-07
--package--	Value: 'gmisclib'

## 4.3 Class *ReferencedFileNotFound*

object └

exceptions.BaseException └

exceptions.Exception └

gmisclib.xwaves\_errs.Error └

**gmisclib.HTK\_MLF\_io.ReferencedFileNotFound**

### 4.3.1 Methods

**\_\_init\_\_**(*self*, *\*s*)

*x.\_\_init\_\_*(...) initializes *x*; see *help*(*type*(*x*)) for signature

Overrides: *object.\_\_init\_\_* *extit*(inherited documentation)

*Inherited from exceptions.Exception*

`__new__()`

*Inherited from exceptions.BaseException*

`__delattr__()`, `__getattr__()`, `__getitem__()`, `__getslice__()`, `__reduce__()`, `__repr__()`,  
`__setattr__()`, `__setstate__()`, `__str__()`, `__unicode__()`

*Inherited from object*

`__format__()`, `__hash__()`, `__reduce_ex__()`, `__sizeof__()`, `__subclasshook__()`

#### 4.3.2 Properties

Name	Description
<i>Inherited from exceptions.BaseException</i>	
args, message	
<i>Inherited from object</i>	
__class__	

### 4.4 Class writer



#### 4.4.1 Methods

**`__init__(self, mlf_fd, time_quantum=1e-07)`**  
`x.__init__(...)` initializes x; see `help(type(x))` for signature  
 Overrides: `object.__init__` extit(inherited documentation)

**`chunk(self, filespec, data)`**

**`threecol(self, filespec, tcddata)`**

**`close(self)`**

**`__del__(self)`**

*Inherited from object*

`--delattr--()`, `--format--()`, `--getattr--()`, `--hash--()`, `--new--()`, `--reduce--()`, `--reduce_ex--()`,  
`--repr--()`, `--setattr--()`, `--sizeof--()`, `--str--()`, `--subclasshook--()`

#### 4.4.2 Properties

Name	Description
<i>Inherited from object</i>	
<code>--class--</code>	

## 5 Module gmisclib.MFCCFile

### 5.1 Variables

Name	Description
<code>--package--</code>	<b>Value:</b> 'gmisclib'

### 5.2 Class MFCCFile

#### 5.2.1 Methods

<code>--init--(<i>self</i>, <i>filename</i>, <i>DEBUG</i>=0, <i>BYTEORDER</i>='@')</code>
---

<code>Save(<i>self</i>, <i>filename</i>)</code>
---

<code>ReturnSent(<i>self</i>, <i>sentnum</i>)</code>
--

<code>ReturnVector(<i>self</i>, <i>sentnum</i>, <i>vecnum</i>)</code>
---

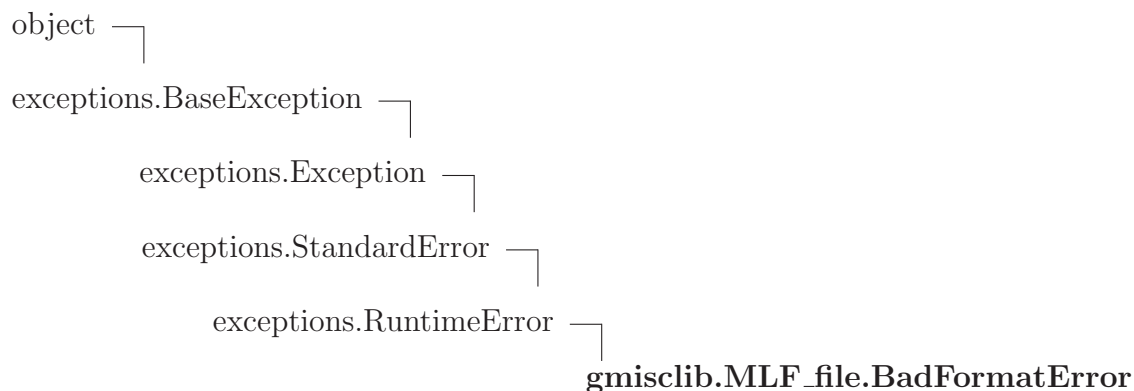


## 6 Module *gmisclib.MLF\_file*

### 6.1 Variables

Name	Description
<code>__package__</code>	<b>Value:</b> None

### 6.2 Class *BadFormatError*



#### 6.2.1 Methods

```

__init__(self, *s)

x.__init__(...) initializes x; see help(type(x)) for signature
Overrides: object.__init__ extit(inherited documentation)

```

*Inherited from exceptions.RuntimeError*

`__new__()`

*Inherited from exceptions.BaseException*

`__delattr__()`, `__getattr__()`, `__getitem__()`, `__getslice__()`, `__reduce__()`, `__repr__()`,  
`__setattr__()`, `__setstate__()`, `__str__()`, `__unicode__()`

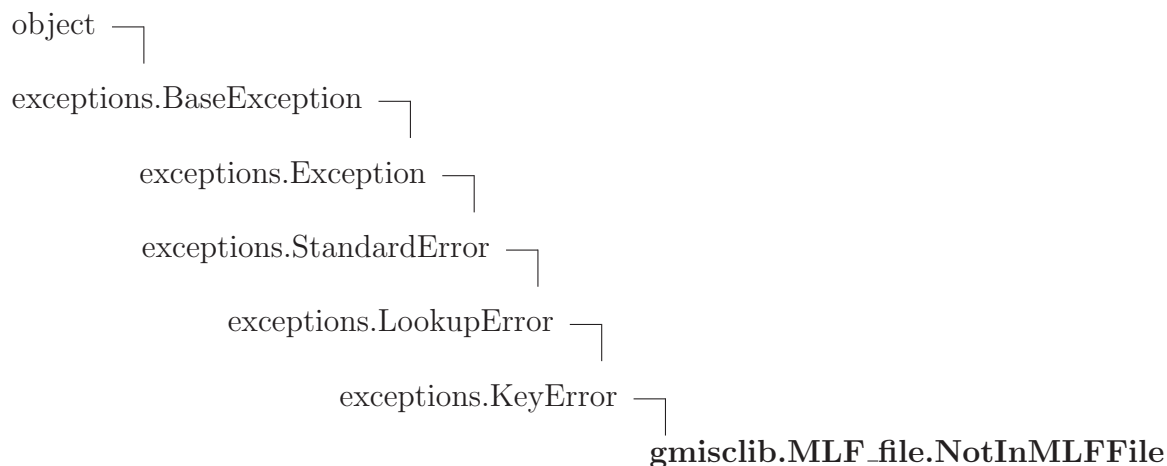
*Inherited from object*

`__format__()`, `__hash__()`, `__reduce_ex__()`, `__sizeof__()`, `__subclasshook__()`

#### 6.2.2 Properties

Name	Description
<i>Inherited from exceptions.BaseException</i>	
args, message	
<i>Inherited from object</i>	
__class__	

### 6.3 Class *NotInMLFFile*



#### 6.3.1 Methods

```

__init__(self, *s)

x.__init__(...) initializes x; see help(type(x)) for signature
Overrides: object.__init__ extit(inherited documentation)

```

*Inherited from exceptions.KeyError*

```
__new__(), __str__()
```

*Inherited from exceptions.BaseException*

```

__delattr__(), __getattr__(), __getitem__(), __getslice__(), __reduce__(), __repr__(),
__setattr__(), __setstate__(), __unicode__()

```

*Inherited from object*

```
__format__(), __hash__(), __reduce_ex__(), __sizeof__(), __subclasshook__()
```

#### 6.3.2 Properties

Name	Description
<i>Inherited from <code>exceptions.BaseException</code></i>	
<code>args</code> , <code>message</code>	
<i>Inherited from <code>object</code></i>	
<code>__class__</code>	

## 6.4 Class `block_MLF_file`



This class reads in and stores a MLF file. It does not interpret the interior data, but rather just breaks it up into blocks, each corresponding to an utterance.

### 6.4.1 Methods

<b><code>__init__(self, fname, preprocessor=None)</code></b>
Read in a MLF file and store the information in <code>self.block</code> .
<b>Parameters</b>
<b><code>fname</code>:</b> Filename to read ( <i>type=</i> <code>str</code> )
<b><code>preprocessor</code>:</b> A function to project the name of each block onto something that you want to use as an index of blocks. Typically, this function cleans up the names, removing asterisks and such. ( <i>type=</i> <code>function str -&gt; str</code> )
Overrides: <code>object.__init__</code>
<b><code>get(self, key)</code></b>
Get a block of text from a MLF file.

**get3**(*self*, *key*, *n*=3)

Get a block of time-aligned labels from a MLF file and interpret it.

**Return Value**

(**start**, **end**, **label**, ...) tuples, with **start** and **end** in seconds, **label** is a string indicating a phoneme or word or whatever. If there is more information on a line, it will be passed along in the tuple.

(*type*=*list(tuple(start, end, label, ...), ...)*)

***Inherited from object***

`__delattr__()`, `__format__()`, `__getattr__()`, `__hash__()`, `__new__()`, `__reduce__()`, `__reduce_ex__()`, `__repr__()`, `__setattr__()`, `__sizeof__()`, `__str__()`, `__subclasshook__()`

**6.4.2 Properties**

Name	Description
<i>Inherited from object</i>	
<code>__class__</code>	

**6.4.3 Class Variables**

Name	Description
Quantum	<b>Value:</b> 1e-07

**6.4.4 Instance Variables**

Name	Description
block	self.block is where all the data is kept. This is a dictionary mapping from a file pattern (name, roughly) to a block of label information. The block is a list of strings, one per line in the MLF file. The file pattern is the output of <code>preprocessor.dict(str: list(str))</code>
fname	The name of the MLF file
blockname	Mapping from a file pattern to the name of the block.

## 7 Module gmisclib.Num

This is a compatibility module, allowing python to work with either scipy, Numeric, or numarray. It tries to import them, in that order, only reporting an error if none are available.

Note that this does not pretend to solve all compatibility problems; it just tries importing all three, so you can only count on the lowest common denominator.

**Version:** 1.5.1

### 7.1 Variables

Name	Description
NewAxis	<b>Value:</b> None
ALLOW_THREADS	<b>Value:</b> 1
BUFSIZE	<b>Value:</b> 10000
CLIP	<b>Value:</b> 0
ERR_CALL	<b>Value:</b> 3
ERR_DEFAULT	<b>Value:</b> 0
ERR_DEFAULT2	<b>Value:</b> 2084
ERR_IGNORE	<b>Value:</b> 0
ERR_LOG	<b>Value:</b> 5
ERR_PRINT	<b>Value:</b> 4
ERR_RAISE	<b>Value:</b> 2
ERR_WARN	<b>Value:</b> 1
FLOATING_POINT_SUPPORT	<b>Value:</b> 1
FPE_DIVIDEBYZERO	<b>Value:</b> 1
FPE_INVALID	<b>Value:</b> 8
FPE_OVERFLOW	<b>Value:</b> 2
FPE_UNDERFLOW	<b>Value:</b> 4
False_	<b>Value:</b> False
Inf	<b>Value:</b> inf
Infinity	<b>Value:</b> inf
MAXDIMS	<b>Value:</b> 32
NAN	<b>Value:</b> nan
NINF	<b>Value:</b> -inf
NZERO	<b>Value:</b> -0.0
NaN	<b>Value:</b> nan
PINF	<b>Value:</b> inf
PZERO	<b>Value:</b> 0.0
RAISE	<b>Value:</b> 2
SHIFT_DIVIDEBYZERO	<b>Value:</b> 0

*continued on next page*

Name	Description
<code>SHIFT_INVALID</code>	Value: 9
<code>SHIFT_OVERFLOW</code>	Value: 3
<code>SHIFT_UNDERFLOW</code>	Value: 6
<code>ScalarType</code>	Value: (<type 'int'>, <type 'float'>, <type 'complex'>, <type '1...
<code>True_</code>	Value: True
<code>UFUNC_BUFSIZE_DEFAULT</code>	Value: 10000
<code>UFUNC_PYVALS_NAME</code>	Value: 'UFUNC_PYVALS'
<code>WRAP</code>	Value: 1
<code>--package--</code>	Value: 'gmisclib'
<code>absolute</code>	Value: <ufunc 'absolute'>
<code>add</code>	Value: <ufunc 'add'>
<code>arccos</code>	Value: <ufunc 'arccos'>
<code>arccosh</code>	Value: <ufunc 'arccosh'>
<code>arcsin</code>	Value: <ufunc 'arcsin'>
<code>arcsinh</code>	Value: <ufunc 'arcsinh'>
<code>arctan</code>	Value: <ufunc 'arctan'>
<code>arctan2</code>	Value: <ufunc 'arctan2'>
<code>arctanh</code>	Value: <ufunc 'arctanh'>
<code>bitwise_and</code>	Value: <ufunc 'bitwise_and'>
<code>bitwise_not</code>	Value: <ufunc 'invert'>
<code>bitwise_or</code>	Value: <ufunc 'bitwise_or'>
<code>bitwise_xor</code>	Value: <ufunc 'bitwise_xor'>
<code>c_</code>	Value: <numpy.lib.index_tricks.CClass object at 0x30a95d0>
<code>cast</code>	Value: {<type 'numpy.complex128'>: <function <lambda> at 0x2f022...
<code>ceil</code>	Value: <ufunc 'ceil'>
<code>conj</code>	Value: <ufunc 'conjugate'>
<code>conjugate</code>	Value: <ufunc 'conjugate'>
<code>copysign</code>	Value: <ufunc 'copysign'>
<code>cos</code>	Value: <ufunc 'cos'>
<code>cosh</code>	Value: <ufunc 'cosh'>
<code>deg2rad</code>	Value: <ufunc 'deg2rad'>
<code>degrees</code>	Value: <ufunc 'degrees'>
<code>divide</code>	Value: <ufunc 'divide'>
<code>e</code>	Value: 2.71828182846
<code>equal</code>	Value: <ufunc 'equal'>
<code>exp</code>	Value: <ufunc 'exp'>
<code>exp2</code>	Value: <ufunc 'exp2'>
<code>expm1</code>	Value: <ufunc 'expm1'>

*continued on next page*

Name	Description
<code>fabs</code>	Value: <code>&lt;ufunc 'fabs'&gt;</code>
<code>floor</code>	Value: <code>&lt;ufunc 'floor'&gt;</code>
<code>floor_divide</code>	Value: <code>&lt;ufunc 'floor_divide'&gt;</code>
<code>fmax</code>	Value: <code>&lt;ufunc 'fmax'&gt;</code>
<code>fmin</code>	Value: <code>&lt;ufunc 'fmin'&gt;</code>
<code>fmod</code>	Value: <code>&lt;ufunc 'fmod'&gt;</code>
<code>frexp</code>	Value: <code>&lt;ufunc 'frexp'&gt;</code>
<code>greater</code>	Value: <code>&lt;ufunc 'greater'&gt;</code>
<code>greater_equal</code>	Value: <code>&lt;ufunc 'greater_equal'&gt;</code>
<code>hypot</code>	Value: <code>&lt;ufunc 'hypot'&gt;</code>
<code>index_exp</code>	Value: <code>&lt;numpy.lib.index_tricks.IndexExpression object at 0x30a9650&gt;</code>
<code>inf</code>	Value: <code>inf</code>
<code>infty</code>	Value: <code>inf</code>
<code>invert</code>	Value: <code>&lt;ufunc 'invert'&gt;</code>
<code>isfinite</code>	Value: <code>&lt;ufunc 'isfinite'&gt;</code>
<code>isinf</code>	Value: <code>&lt;ufunc 'isinf'&gt;</code>
<code>isnan</code>	Value: <code>&lt;ufunc 'isnan'&gt;</code>
<code>ldexp</code>	Value: <code>&lt;ufunc 'ldexp'&gt;</code>
<code>left_shift</code>	Value: <code>&lt;ufunc 'left_shift'&gt;</code>
<code>less</code>	Value: <code>&lt;ufunc 'less'&gt;</code>
<code>less_equal</code>	Value: <code>&lt;ufunc 'less_equal'&gt;</code>
<code>little_endian</code>	Value: <code>True</code>
<code>log</code>	Value: <code>&lt;ufunc 'log'&gt;</code>
<code>log10</code>	Value: <code>&lt;ufunc 'log10'&gt;</code>
<code>log1p</code>	Value: <code>&lt;ufunc 'log1p'&gt;</code>
<code>log2</code>	Value: <code>&lt;ufunc 'log2'&gt;</code>
<code>logaddexp</code>	Value: <code>&lt;ufunc 'logaddexp'&gt;</code>
<code>logaddexp2</code>	Value: <code>&lt;ufunc 'logaddexp2'&gt;</code>
<code>logical_and</code>	Value: <code>&lt;ufunc 'logical_and'&gt;</code>
<code>logical_not</code>	Value: <code>&lt;ufunc 'logical_not'&gt;</code>
<code>logical_or</code>	Value: <code>&lt;ufunc 'logical_or'&gt;</code>
<code>logical_xor</code>	Value: <code>&lt;ufunc 'logical_xor'&gt;</code>
<code>maximum</code>	Value: <code>&lt;ufunc 'maximum'&gt;</code>
<code>mgrid</code>	Value: <code>&lt;numpy.lib.index_tricks.nd_grid object at 0x30a9510&gt;</code>
<code>minimum</code>	Value: <code>&lt;ufunc 'minimum'&gt;</code>
<code>mod</code>	Value: <code>&lt;ufunc 'remainder'&gt;</code>
<code>modf</code>	Value: <code>&lt;ufunc 'modf'&gt;</code>
<code>multiply</code>	Value: <code>&lt;ufunc 'multiply'&gt;</code>
<code>nan</code>	Value: <code>nan</code>

*continued on next page*

Name	Description
<code>nbytes</code>	Value: {<type 'numpy.complex128': 16, <type 'numpy.void': 0, <...
<code>negative</code>	Value: <ufunc 'negative'>
<code>newaxis</code>	Value: None
<code>nextafter</code>	Value: <ufunc 'nextafter'>
<code>not_equal</code>	Value: <ufunc 'not_equal'>
<code>ogrid</code>	Value: <numpy.lib.index_tricks.nd_grid object at 0x30a9550>
<code>ones_like</code>	Value: <ufunc 'ones_like'>
<code>pi</code>	Value: 3.14159265359
<code>power</code>	Value: <ufunc 'power'>
<code>r_</code>	Value: <numpy.lib.index_tricks.RClass object at 0x30a9590>
<code>rad2deg</code>	Value: <ufunc 'rad2deg'>
<code>radians</code>	Value: <ufunc 'radians'>
<code>reciprocal</code>	Value: <ufunc 'reciprocal'>
<code>remainder</code>	Value: <ufunc 'remainder'>
<code>right_shift</code>	Value: <ufunc 'right_shift'>
<code>rint</code>	Value: <ufunc 'rint'>
<code>s_</code>	Value: <numpy.lib.index_tricks.IndexExpression object at 0x30a96d0>
<code>sctypeDict</code>	Value: {0: <type 'numpy.bool_'>, 1: <type 'numpy.int8'>, 2: <typ...
<code>sctypeNA</code>	Value: {'?': 'Bool', 'B': 'UInt8', 'Bool': <type 'numpy.bool_'>,...
<code>sctypes</code>	Value: {'complex': [<type 'numpy.complex64'>, <type 'numpy.compl...
<code>sign</code>	Value: <ufunc 'sign'>
<code>signbit</code>	Value: <ufunc 'signbit'>
<code>sin</code>	Value: <ufunc 'sin'>
<code>sinh</code>	Value: <ufunc 'sinh'>
<code>spacing</code>	Value: <ufunc 'spacing'>
<code>sqrt</code>	Value: <ufunc 'sqrt'>
<code>square</code>	Value: <ufunc 'square'>
<code>subtract</code>	Value: <ufunc 'subtract'>
<code>tan</code>	Value: <ufunc 'tan'>
<code>tanh</code>	Value: <ufunc 'tanh'>
<code>true_divide</code>	Value: <ufunc 'true_divide'>
<code>trunc</code>	Value: <ufunc 'trunc'>
<code>typeDict</code>	Value: {0: <type 'numpy.bool_'>, 1: <type 'numpy.int8'>, 2: <typ...

*continued on next page*



Name	Description
typeNA	<b>Value:</b> {'?': 'Bool', 'B': 'UInt8', 'Bool': <type 'numpy.bool_'>, ...
typecodes	<b>Value:</b> {'All': '?bhilqpBHILQPfdgFDGSUV0', 'AllFloat': 'fdgFDG', ...

## 8 Module *gmisclib.Numeric\_gpk*

### 8.1 Functions

**add\_overlap**(*a*, *astart*, *b*, *bstart*)

Add arrays *a* and *b* in the overlap region. Return (*data*, *start*). If *a*, *b* are time series, they are assumed to have the same sampling rate. *Astart* and *Bstart* apply to the zeroth index. All other indices are assumed to match start and length.

**zero\_pad\_end**(*d*, *padfactor*=1)

**Poisson**(*nbar*)

Return a Poisson random integer, whose distribution has a mean = *nbar*.

**bevel\_concat**(*a*, *b*, *bevel*=0, *bevel\_overlap*=1.0, *delay*=0, *ta*=None, *tb*=None)

Concatenate two time series. Bevel the edges, and overlap them slightly.

*Bevel\_overlap* controls the fractional overlap of the two bevels, and *delay* specifies an extra delay for *b*.

If *ta* and/or *tb* are specified, return a tuple of (*concatenated\_time\_series*, *tma*, *tmb*) where *tma* and *tmb* are the locations corresponding to *ta* and *tb* in the corresponding input arrays.

**argmax**(*a*)

**N\_maximum**(*a*)

**N\_minimum**(*a*)

**N\_frac\_rank**(*a*, *fr*)

**N\_mean\_ad**(*a*)

Mean absolute deviation. For a multi-dimensional array, it takes the MAD along the first axis, so `N_mean_ad(x)[0]==N_mean_ad(x[:,0])`.

**median\_across**(*list\_of\_vec*)

Returns an element-by-element median of a list of Numeric vectors.

**N\_median**(*a*, *axis*=0)

Returns an element-by-element median of a list of Numeric vectors.

**N\_median\_across**(*a*)

**set\_diag**(*x*, *a*)

Set the diagonal of a matrix *x* to be the vector *a*. If *a* is shorter than the diagonal of *x*, just set the beginning.

**limit**(*low*, *x*, *high*)

**trimmed\_mean\_sigma\_across**(*list\_of\_vec*, *weights*, *clip*)

**trimmed\_mean\_across**(*list\_of\_vec*, *weights*, *clip*)

**trimmed\_stdev\_across**(*list\_of\_vec*, *weights*, *clip*)

**vec\_variance**(*x*)

Take a component-by-component variance of a list of vectors.

**qform**(*vec*, *mat*)

A quadratic form: *vec*\**mat*\**vec*, or *vecs*\**mat*\**transpose*(*vecs*)

**KolmogorovSmirnov**(*d1*, *d2*, *w1*=None, *w2*=None)

**interpN**(*a*, *t*)

Interpolate array *a* to floating point indices, *t*, via nearest-neighbor interpolation. Returns a Numeric array.

**interp**(*a*, *t*)

Interpolate to a specified time axis. This does a linear interpolation. *A* is a Numpy array, and *t* is an array of times.

**Return Value**

interpolated values

(*type*=*numpy array*.)

---

**split\_into\_clumps**(*x*, *threshold*, *minsize*=1)

---

This reports when the signal is above the threshold.

**Parameters**

**x**: a signal  
(*type*=*numpy.ndarray*, *one-dimensional*.)

**threshold**: a threshold.  
(*type*=*float*)

**Return Value**

[(start, stop), ...] for each region ("clump") where **x**>**threshold**.

---

**block\_stdev**(*x*)

---

This is just a alternative implementation of the standard deviation of each channel, but it is designed in a block-wise fashion so the total memory usage is not large.

---

**convolve**(*x*, *kernel*)

---

This is basically like `numpy.convolve(x, kernel, 1)` except that it properly handles the case where the kernel is longer than the data.

```
edge_window(n, eleft, eright, typeleft='linear', typeright='linear',  
norm=None)
```

Creates a window which is basically flat, but tapers off on the left and right edges. The widths of the tapers can be controlled, as can the shapes.

#### Parameters

- n:** the width of the window  
(*type*=*int*)
- eleft:** the width of the left taper (i.e. at zero index, in samples).  
(*type*=*int*)
- eright:** the width of the right taper (i.e. at index near *n*, in samples).  
(*type*=*int*)
- typeleft:** what kind of taper on the left? (Defaults to "linear").  
(*type*='linear' or 'cos')
- typeright:** what kind of taper on the right? (Defaults to "linear").  
(*type*='linear' or 'cos')
- norm:** How to normalize? The default is *None*, which means no normalization. Providing a number *x* will normalize the window so that the average of the *window\*\*x*==1.  
(*type*=*None* or *float* != 0.)

```
edge_window_t(t, eleft, eright, typeleft='linear', typeright='linear',
norm=None)
```

Computes a window which is basically flat, but tapers off on the left and right edges. The widths of the tapers can be controlled, as can the shapes.

#### Parameters

**t:** an array of time values. They are required to be monotonically increasing, and assumed to be linearly spaced.  
(*type=*`numpy.ndarray`)

**eleft:** the width of the left taper (i.e. at zero index, in time units).  
(*type=*`float`)

**eright:** the width of the right taper (i.e. at index near *n*, in time units).  
(*type=*`float`)

**typeleft:** what kind of taper on the left? (Defaults to "linear").  
(*type=*`'linear'` or `'cos'`)

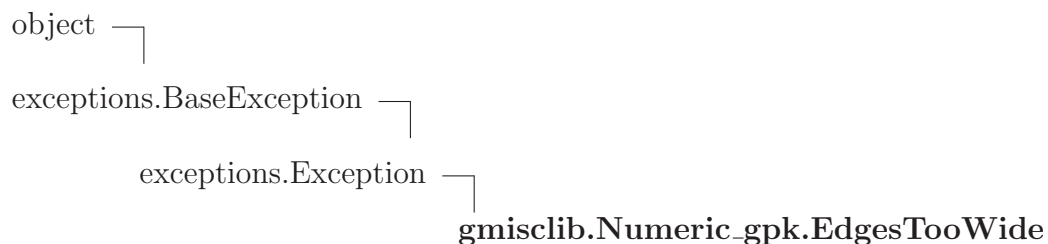
**typeright:** what kind of taper on the right? (Defaults to "linear").  
(*type=*`'linear'` or `'cos'`)

**norm:** How to normalize? The default is `None`, which means no normalization. Providing a number *x* will normalize the window so that the average of the `window**x==1`.  
(*type=*`None` or `float != 0`.)

## 8.2 Variables

Name	Description
<code>pylab</code>	<b>Value:</b> <code>None</code>
<code>asinh</code>	<b>Value:</b> <code>&lt;ufunc 'arcsinh'&gt;</code>
<code>acosh</code>	<b>Value:</b> <code>&lt;ufunc 'arccosh'&gt;</code>
<code>BLOCK</code>	<b>Value:</b> <code>8192</code>
<code>--package--</code>	<b>Value:</b> <code>'gmisclib'</code>

### 8.3 Class *EdgesTooWide*



#### 8.3.1 Methods

**`--init--`**(*self*, \**s*)

*x*.**`--init--`**(...) initializes *x*; see `help(type(x))` for signature

Overrides: `object.__init__` `extit`(inherited documentation)

#### *Inherited from exceptions.Exception*

`--new--`()

#### *Inherited from exceptions.BaseException*

`--delattr--`(), `--getattr--`(), `--getitem--`(), `--getslice--`(), `--reduce--`(), `--repr--`(),  
`--setattr--`(), `--setstate--`(), `--str--`(), `--unicode--`()

#### *Inherited from object*

`--format--`(), `--hash--`(), `--reduce_ex--`(), `--sizeof--`(), `--subclasshook--`()

#### 8.3.2 Properties

Name	Description
<i>Inherited from exceptions.BaseException</i>	
<code>args</code> , <code>message</code>	
<i>Inherited from object</i>	
<code>--class--</code>	

## 9 Module `gmisclib.accent_spec`

This module provides a way of safely specifying accent positions in running text.

If you have a transcription "I did not eat the orange ball.", you can attach a "+" symbol to the word "eat" like this: "+eat".

You enter an accent specification, which consists of words with a prefix (the prefix is anything that ends in a punctuation mark). The program matches up the words in the accent specification to the words in the transcription. You can have many words in the accent specification, if necessary. You can also disambiguate things by putting context words into the accent spec (without a prefix). All matching is done left-to-right.

**Version:** \$Revision: 1.4 \$

### 9.1 Functions

```
prefix(text_array, accent_spec, map_fcn=<function <lambda> at
0x3456aa0>)
```

This function takes an array of words and an accent spec. It matches the accent spec to the words, and outputs an array of tuples which tells you where the accents are, and what kind. The optional `map_fcn` can be used to map other kinds of objects into a array of strings.

More specifically, an `accent_spec` is a whitespace-separated list of strings. Each string is a word from the `text_array`, with an optional prefix. The strings are matched in order to the words, and the output array is a list of `(index_in_text_array, prefix_text)` tuples.

So, if you have `text_array = ['my', 'cat', 'is', 'my', 'cat']` and `accent_spec="is +my"`, then `align()` will match "is" to `text_array[2]`, and "+my" to `text_array[3]`, and it will return `[(3, "+")]`. Note that "+is +my" does not imply that 'is' and 'my' are adjacent, and "+is +cat" simply returns `[(2, "+"), (4, "+")]`.

If the `accent_spec` were "+my", then it would match `text_array[0]`, and return `[(0, "+")]`.

Prefixes can be multiple characters, but they cannot end in letters, digits, or underscore.

```
suffix(text_array, accent_spec, map_fcn=<function <lambda> at
0x3456c08>)
```

See `prefix`, but with the obvious changes.



```
preshow(text_array, alignment, map_fcn=<function <lambda> at 0x3456cf8>)
```

Shows an alignment in a printable form. It puts the prefixes in the appropriate places.

```
sufshow(text_array, alignment, map_fcn=<function <lambda> at 0x3456de8>)
```

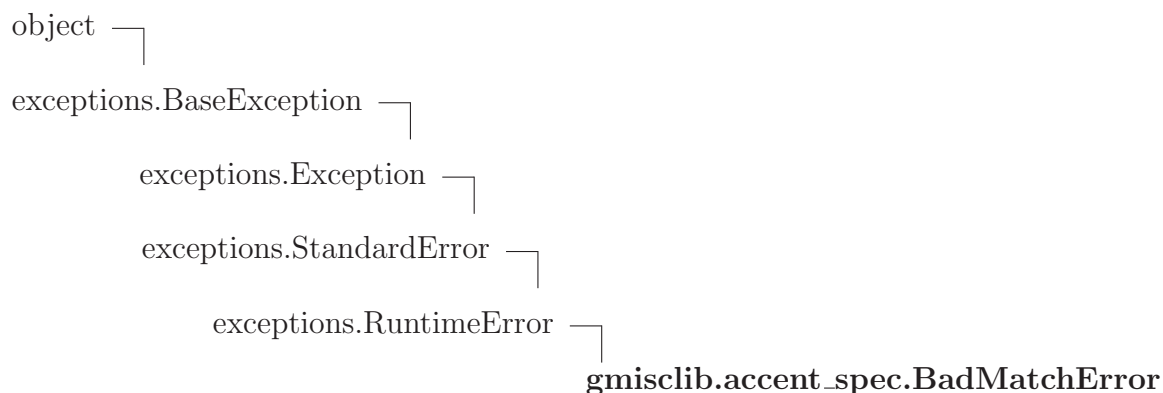
Shows an alignment in a printable form. It puts the suffixes in the appropriate places.

```
test()
```

## 9.2 Variables

Name	Description
<code>__package__</code>	<b>Value:</b> 'gmisclib'

## 9.3 Class **BadMatchError**



### 9.3.1 Methods

```
__init__(self, x)
```

`x.__init__(...)` initializes `x`; see `help(type(x))` for signature

Overrides: `object.__init__` `extit`(inherited documentation)

*Inherited from **exceptions.RuntimeError***

`__new__()`

***Inherited from exceptions.BaseException***

`__delattr__()`, `__getattr__()`, `__getitem__()`, `__getslice__()`, `__reduce__()`, `__repr__()`,  
`__setattr__()`, `__setstate__()`, `__str__()`, `__unicode__()`

***Inherited from object***

`__format__()`, `__hash__()`, `__reduce_ex__()`, `__sizeof__()`, `__subclasshook__()`

### 9.3.2 Properties

Name	Description
<i>Inherited from exceptions.BaseException</i>	
args, message	
<i>Inherited from object</i>	
<code>__class__</code>	

## 10 Module `gmisclib.array_window`

Defines `array_window` class.

**Version:** \$Revision: 1.4 \$

### 10.1 Functions

<code>test()</code>
---------------------

### 10.2 Variables

Name	Description
<code>--package--</code>	<b>Value:</b> None

### 10.3 Class `array_window`

Lets you reference 1-d array items relative to an arbitrary zero.

#### 10.3.1 Methods

<code>--init--(<i>self</i>, <i>array</i>, <i>zero</i>)</code>
---

<code>--setattr--(<i>self</i>, <i>name</i>, <i>value</i>)</code>
--

<code>--delattr--(<i>self</i>, <i>name</i>)</code>
--

<code>--getitem--(<i>self</i>, <i>index</i>)</code>
---

<code>--setitem--(<i>self</i>, <i>index</i>, <i>value</i>)</code>
---

<code>--getslice--(<i>self</i>, <i>i</i>, <i>j</i>)</code>
--

<code>--setslice--(<i>self</i>, <i>i</i>, <i>j</i>, <i>sequence</i>)</code>
---

<code>--str--(<i>self</i>)</code>
-----------------------------------

<code>--repr--(self)</code>
-----------------------------

<code>--len--(self)</code>
----------------------------

Of course, one can't assume that one can say <code>x[i]</code> for <code>i</code> in <code>range(len(x))</code> .
---

### 10.3.2 Class Variables

Name	Description
<code>--doc--</code>	<b>Value:</b> <code>"""Lets you reference 1-d array items relative to an arbi...</code>

## 11 Module *gmisclib.avio*

Module to parse and create lines in "a=v;" format. Normally, the entry points are `parse()` and `concoct()`.

**Version:** \$Revision: 1.21 \$

### 11.1 Functions

**parse**(*s*, *sep*=';')

Parses a line in a=v; a=v form into a dictionary. The line may be terminated by a '#' followed by an arbitrary comments. Both attributes and values are assumed to be encoded with the default `g_encode` encoder. (However, the comment is not assumed to be encoded.)

**Parameters**

- s:** the string to be parsed.
- sep:** (optional) lets you choose the character that separates attribute-value pairs.

**Return Value**

dictionary containing a=v pairs. One special entry for comments: "\_COMMENT".

**Raises**

`BadFormatError` when *s* is not formatted properly.

**concoct**(*s*)

Converts a dictionary to a line in the form a=v;a=v;#comment. It returns a string. The entries in the dictionary will be converted to a string representation by `fwd()`.

**read**(*fd*, *sep*=';', *line*=None)

Read a file in a=v; format. If *line* is set, put the line number into a slot with the specified name.

**Return Value**

(data, comments) where data=[{a:v, ...}, ...] and comments=[str, ...]  
(*type=tuple(list(dict), list(str))*)

**Raises**

`BadFormatError` when *s* is not formatted properly.

**test1()**

**read\_hdc**(*fd*, *sep*=';', *line*=None)

This emulates `fiatio.read()`.

**Return Value**

(header, data, comments) where data=[{a:v, ...}, ...] and  
comments=[str, ...]

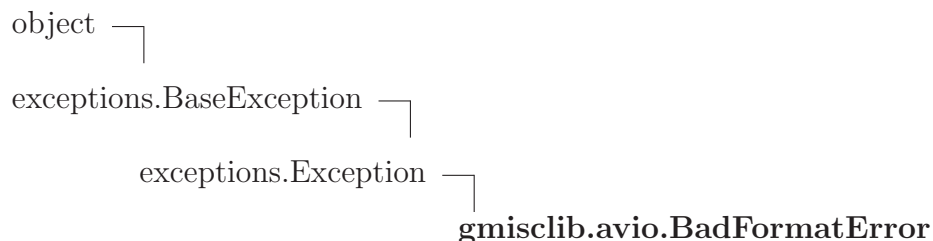
(*type=tuple(dict, list(dict), list(str))*)

**test2()**

## 11.2 Variables

Name	Description
<code>--package--</code>	<b>Value:</b> 'gmisclib'

## 11.3 Class `BadFormatError`



### 11.3.1 Methods

**\_\_init\_\_**(*self*, \**x*)

`x.__init__(...)` initializes `x`; see `help(type(x))` for signature

Overrides: `object.__init__` `extit`(inherited documentation)

*Inherited from `exceptions.Exception`*

`__new__`()

*Inherited from `exceptions.BaseException`*

`__delattr__`(), `__getattr__`(), `__getitem__`(), `__getslice__`(), `__reduce__`(), `__repr__`(),

`__setattr__()`, `__setstate__()`, `__str__()`, `__unicode__()`

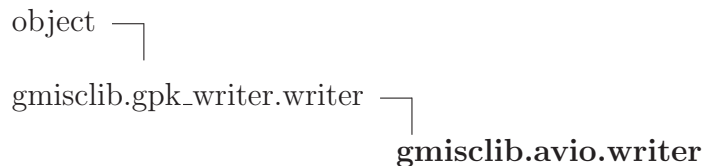
### ***Inherited from object***

`__format__()`, `__hash__()`, `__reduce_ex__()`, `__sizeof__()`, `__subclasshook__()`

#### **11.3.2 Properties**

Name	Description
<i>Inherited from exceptions.BaseException</i>	
args, message	
<i>Inherited from object</i>	
<code>__class__</code>	

## **11.4 Class writer**



Write a file in a=v; format.

#### **11.4.1 Methods**

<b>comment</b> ( <i>self</i> , <i>comment</i> ) <hr/> Add a comment to the data file. Overrides: gmisclib.gpk_writer.writer.comment
<b>header</b> ( <i>self</i> , <i>k</i> , <i>v</i> ) <hr/> NOTE: this is not a fully general function. Keys may not have spaces or equals signs in them. Note also that reading of headers from a=v; files is not supported. Overrides: gmisclib.gpk_writer.writer.header
<b>__init__</b> ( <i>self</i> , <i>fd</i> ) <hr/> <i>x.__init__(...)</i> initializes <i>x</i> ; see <code>help(type(x))</code> for signature Overrides: <code>object.__init__</code> <code>exitit</code> (inherited documentation)

<b>datum</b> ( <i>self</i> , <i>data_item</i> )
---

Overrides: gmisclib.gpk_writer.writer.datum
---

<b>append</b> ( <i>self</i> , <i>d</i> )
--

<b>close</b> ( <i>self</i> )
------------------------------

<b>comments</b> ( <i>self</i> , <i>comments</i> )
---

Add comments to the data file. Comments can appear anywhere.
--

<b>data</b> ( <i>self</i> , <i>dataset</i> )
--

Write a series of lines to the output file.
---

<b>extend</b> ( <i>self</i> , <i>d</i> )
--

<b>flush</b> ( <i>self</i> )
------------------------------

<b>headers</b> ( <i>self</i> , <i>h</i> )
---

### *Inherited from object*

`__delattr__()`, `__format__()`, `__getattr__()`, `__hash__()`, `__new__()`, `__reduce__()`, `__reduce_ex__()`,  
`__repr__()`, `__setattr__()`, `__sizeof__()`, `__str__()`, `__subclasshook__()`

#### 11.4.2 Properties

Name	Description
<i>Inherited from object</i>	
<code>__class__</code>	

#### 11.4.3 Class Variables

Name	Description
<code>__doc__</code>	<b>Value:</b> <code>""Write a file in a=v; form...</code>



## 12 Module *gmisclib.bark\_scale*

**Version:** \$Revision: 1.3 \$

### 12.1 Functions

<b>asinh</b> ( $x$ )
----------------------

<b>acosh</b> ( $x$ )
----------------------

<b>f_to_bark</b> ( $f$ )
--------------------------

frequency to critical band number. See Schroeder et al. (1979).
---

<b>bark_to_f</b> ( $b$ )
--------------------------

critical band number -> frequency M. R. Schroeder, B.S. Atal, J.L. Hall (1979), J.Acoust.Soc.Am. 66(6) 1647-1652. Title: "Optimizing Digital Speech Coders by exploiting masking properties of the Human Ear."
--

<b>cbw</b> ( $b$ )
--------------------

Critical bandwidth (in Hz) at frequency $b$ (barks).
--

### 12.2 Variables

Name	Description
<code>--package--</code>	<b>Value:</b> 'gmisclib'

## 13 Module *gmisclib.beamsearch*

### 13.1 Functions

**beamsearch**(*cost*, *extra*, *initial*, *B*, *E*)

A breadth-first beam search. B = max number of options to keep, E = max cost difference between best and worst threads in beam. initial = [ starting positions ] extra = arbitrary information for cost function. cost = fn(state, extra) -> (total\_cost, [next states], output\_if\_goal)

### 13.2 Variables

Name	Description
--package--	Value: None

## 14 Module *gmisclib.blue\_data\_attributes*

This chooses samples such that the specified attributes are broadly distributed.

### 14.1 Functions

<b>test()</b>
---------------

### 14.2 Variables

Name	Description
<code>__package__</code>	<b>Value:</b> <code>'gmisclib'</code>

### 14.3 Class *blue\_attributes*

object  `gmisclib.blue_data_attributes.blue_attributes`

#### 14.3.1 Methods

<b><code>__init__(self, inspector, data)</code></b>
<code>x.__init__(...)</code> initializes <code>x</code> ; see <code>help(type(x))</code> for signature
<b>Parameters</b>
<code>inspector</code> : a function on a datum that returns a list of attributes.
Overrides: <code>object.__init__</code>

<b><code>pick(self, n)</code></b>
-----------------------------------

<b><code>pick_one(self)</code></b>
------------------------------------

Pick one item from the data set.
----------------------------------

**peek**(*self*)

Inspect (but do not remove) the next item to be picked.

**Return Value**

the next item to be picked.

(*type=whatever (not a list!)*)

**add**(*self*, *datum*)

Add another datum to be sampled.

**Parameters**

**datum:** thing to be added. It has a probability of  $1/\text{len}(\text{self})$  of being the next sample.

(*type=whatever*)

**\_\_len\_\_**(*self*)

**reset**(*self*)

Forget prior history of usage. Choices after this call are uncorrelated with choices before this call.

***Inherited from object***

`__delattr__()`, `__format__()`, `__getattr__()`, `__hash__()`, `__new__()`, `__reduce__()`, `__reduce_ex__()`, `__repr__()`, `__setattr__()`, `__sizeof__()`, `__str__()`, `__subclasshook__()`

**14.3.2 Properties**

Name	Description
<i>Inherited from object</i>	
<code>__class__</code>	

## 15 Module `gmisclib.blue_data_selector`

This takes anticorrelated samples from a sequence of data. In other words, it tends to choose data that have not been seen frequently before. With `blueness>=1`, it never shows a sample `n` times until all samples have been shown `n-1` times.

(Note that it is possible for an item to appear twice in succession, if it appears as the `N`th item in a set of `N`, then appears as the first item in the second set of `N`.)

### 15.1 Functions

<code>test()</code>
---------------------

### 15.2 Variables

Name	Description
<code>--package--</code>	Value: <code>'gmisclib'</code>

### 15.3 Class `bluedata`

object  `gmisclib.blue_data_selector.bluedata`

## 15.3.1 Methods

---

**\_\_init\_\_**(*self*, *data*, *blueness*=2.0, *rng*=None)

---

*x*.\_\_init\_\_(...) initializes *x*; see `help(type(x))` for signature

**Parameters**

**data:** data to sample

(*type=list(whatever)*)

**blueness:** How much should you avoid choosing the same item in succession? If  $-0.5 < \text{blueness} < 0$ , then you are more likely to choose the same item several times in a row. If  $0 < \text{blueness} < 1$ , you are unlikely to choose the same item twice in succession (though it is possible). If  $\text{blueness} \geq 1$ , you will never get the same item twice until you've run through the entire list of data.

(*type=float*)

**rng:** a random number generator, per the `random` module. You can generate repeated sequences of data by repeatedly passing it a random number generator in the same state.

Overrides: `object.__init__`

---

**check**(*self*, *fcn*)

---

This exists to let you run some kind of read-only check on the stored data.

---

**add**(*self*, *datum*)

---

Add another datum to be sampled.

**Parameters**

**datum:** thing to be added. It has a probability of  $1/\text{len}(\text{self})$  of being the next sample.

(*type=whatever*)

---

**pickiter**(*self*, *n*)

---

Pick *n* items from the data set.

**Parameters**

**n:** how many items to pick.

(*type=int*)

**pick1**(*self*)

**pick**(*self*, *n*)

Pick *n* items from the data set.

**Parameters**

*n*: how many items to pick.

(*type=int*)

**Return Value**

list(whatever), with length==*n*

**peek**(*self*)

Inspect (but do not remove) the next item to be picked.

**Return Value**

the next item to be picked.

(*type=whatever (not a list!)*)

**split**(*self*, *n*)

Split the data set into two parts, of size *n* and len(data)-*n*. Successive calls to split will avoid having the same items appear in the first part. For instance, given a bluedata of size 100, with ten calls to split(10), each datum will appear in the first part of the return tuple exactly once.

**Return Value**

tuple(list, list)

**\_\_iter\_\_**(*self*)

This iterator will produce samples forever.

**\_\_len\_\_**(*self*)

**reset**(*self*)

Forget prior history of usage. Choices after this call are uncorrelated with choices before this call.

**Inherited from object**

`__delattr__()`, `__format__()`, `__getattr__()`, `__hash__()`, `__new__()`, `__reduce__()`, `__reduce_ex__()`, `__repr__()`, `__setattr__()`, `__sizeof__()`, `__str__()`, `__subclasshook__()`

### 15.3.2 Properties

Name	Description
<i>Inherited from object</i> __class__	



## 16 Module `gmisclib.cache`

### 16.1 Functions

**cachepath**(*f*, *tail*='', *root*='')

OBSOLETE: Return a pathname suitable for cacheing some result.

**Parameters**

- f**: An arbitrary key, could be a pathname or a tuple of information about a file.  
*(type=often `str`, but could be anything convertible to a `str` via `repr`.)*
- tail**: something to add at the end of the constructed path.  
*(type=`str` or `None`)*

**Return Value**

(`path_to_root`, `path_with_tail`). `Path_to_root` is/will be a directory; `path_with_tail` is a path to a data file within that directory. Normally, the actual cache is at the location `os.path.join(path_to_root, path_with_tail)` on the disk; that is what you would pass as the `fname` argument to `load_cache` or `dump_cache`.  
*(type=`tuple(str, str)`)*

**Raises**

`ValueError` if `suffix_to_del` is specified and `f` doesn't end that way.

**fileinfo**(*fname*, \**other*)

Collect enough information about a file to determine whether or not the cache can be used.

**modFileInfo**(*fname*, \**other*)

Collect enough information about a file to determine whether or not the cache can be used.

**modinfo**(*m*, *seen*)

**modinfo\_guts**(*m*, *seen*)

**namedModInfo**(*nm*)

<b>walkcache</b> ( <i>top</i> )
---------------------------------

This is to help humans read the cache.
--

<b>test_errs</b> ()
---------------------

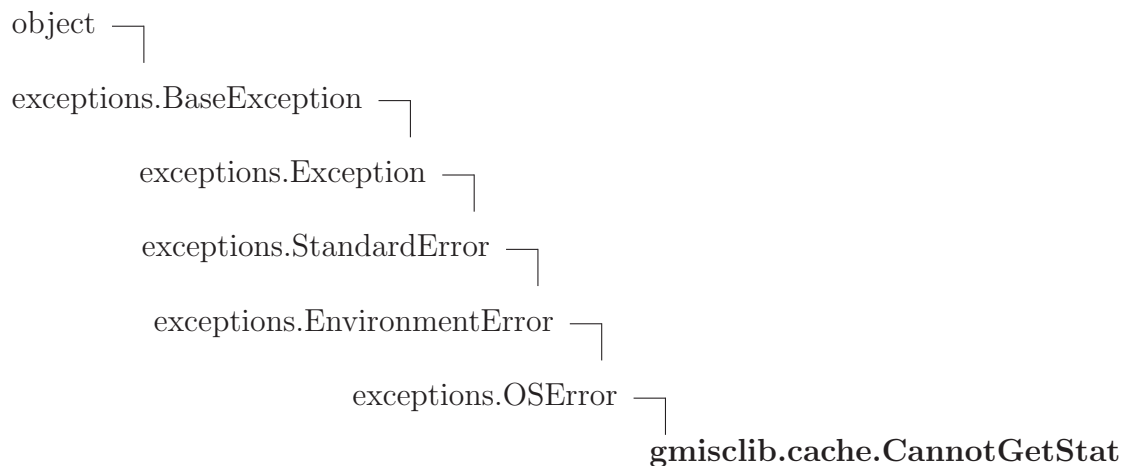
<b>test_normal</b> ()
-----------------------

<b>test</b> ()
----------------

## 16.2 Variables

Name	Description
DEBUG	Value: 0
__package__	Value: 'gmisclib'

## 16.3 Class CannotGetStat



### 16.3.1 Methods

<b>__init__</b> ( <i>self</i> , * <i>s</i> )
--

<i>x</i> .__init__(...) initializes <i>x</i> ; see help(type( <i>x</i> )) for signature
---

Overrides: object.__init__ extit(inherited documentation)
---

*Inherited from exceptions.OSError*

`__new__()`

*Inherited from `exceptions.EnvironmentError`*

`__reduce__()`, `__str__()`

*Inherited from `exceptions.BaseException`*

`__delattr__()`, `__getattr__()`, `__getitem__()`, `__getslice__()`, `__repr__()`, `__setattr__()`,  
`__setstate__()`, `__unicode__()`

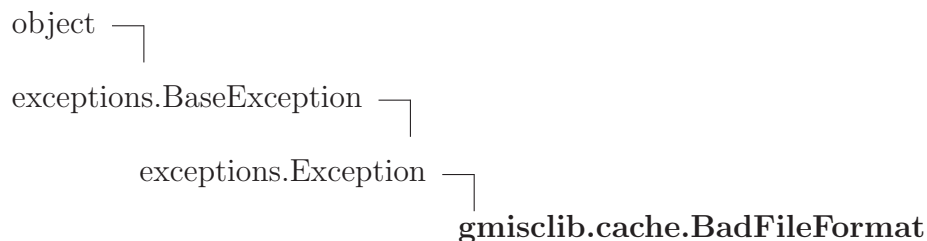
*Inherited from `object`*

`__format__()`, `__hash__()`, `__reduce_ex__()`, `__sizeof__()`, `__subclasshook__()`

### 16.3.2 Properties

Name	Description
<i>Inherited from <code>exceptions.EnvironmentError</code></i> <code>errno</code> , <code>filename</code> , <code>strerror</code>	
<i>Inherited from <code>exceptions.BaseException</code></i> <code>args</code> , <code>message</code>	
<i>Inherited from <code>object</code></i> <code>__class__</code>	

## 16.4 Class `BadFileFormat`



### 16.4.1 Methods

`__init__(self, *s)`  
`x.__init__(...)` initializes `x`; see `help(type(x))` for signature  
 Overrides: `object.__init__` `exitit` (inherited documentation)

*Inherited from `exceptions.Exception`*

`__new__()`

*Inherited from `exceptions.BaseException`*

`__delattr__()`, `__getattr__()`, `__getitem__()`, `__getslice__()`, `__reduce__()`, `__repr__()`,  
`__setattr__()`, `__setstate__()`, `__str__()`, `__unicode__()`

*Inherited from `object`*

`__format__()`, `__hash__()`, `__reduce_ex__()`, `__sizeof__()`, `__subclasshook__()`

#### 16.4.2 Properties

Name	Description
<i>Inherited from <code>exceptions.BaseException</code></i>	
<code>args</code> , <code>message</code>	
<i>Inherited from <code>object</code></i>	
<code>__class__</code>	

### 16.5 Class `cache_info`

object └─ **`gmisclib.cache.cache_info`**

This class manages a disk cache of arbitrary objects. It first constructs a unique name, based on information that you give, then you can **dump** data to that path, or **load** data from that path. An attempt to load data will either succeed or raise an exception; an attempt to dump will either succeed or silently fail.

Typical use:

```
def cached_f(parameters):
    ci = cache_info(info=tuple(parameters))
    if ci is not None:
        try:
            return ci.load()
        except (BadFileFormat, IOError, OSError):
            pass
    o = f(parameters)
    if ci is not None:
        ci.dump(o)
    return o
```

**Note:** This class assumes that the results it is cacheing are generated by a function `f(parameters)`.

You need to be careful to give all of the relevant parameters to `cache_info`, otherwise you can get the wrong results back. For instance, if you have five parameters and you forget to give `parameters[2]` to `cache_info`, it will happily store values obtained with all different values of `parameters[2]` in the same slot, and when you later call `load`, you'll get whatever you asked for, even if it is not what you wanted.



## 16.5.1 Methods

**`__init__(self, root, info=(), fname=None, modname=None, mod=None)`**

`x.__init__(...)` initializes `x`; see `help(type(x))` for signature

**Parameters**

**`info`:** This is where you specify the parameters from which the cached value can be computed. It is essentially a look-up key for the value.

*(type=tuple(anything))*

**`fname`:** You can specify that the cached value depends on the contents of a file (in addition to other parameters). See `fileinfo` for details.

*(type=str)*

**`modname`:** You can specify that the value depends on a module (or a list of modules). (You give the names of the modules here.) In which case, it tries to detect changes to the specified modules. See `namedModInfo` for details. You use this argument to protect yourself against changes to the code used to compute the cached value. Obviously, you don't want to load a value from last weeks, buggy implementation.

*(type=str or tuple(str))*

**`modname`:** You can specify that the value depends on a module (or a list of modules). (You give the module itself here.) In which case, it tries to detect changes to the specified modules. See `modinfo` for details.

*(type=str or tuple(str))*

**`mod`:** *(type=module or tuple(module).)*

Overrides: `object.__init__`

**Notes:**

- Certain compromises were made in the handling of `modname` and `mod`. Even if you use them, you are not 100% guaranteed to be protected from all changes to the code used to compute the cached values. To be entirely safe, you should manually clear the cache when ever you change your code. However, this will probably save your tail if you forget to clear the cache. See `modinfo` for details.
- If there is an error when reading files (i.e. if `fname`, `modname`, or `mod` is specified), then the object will be constructed with `info=None`. This will lead to a `OSError` if you then call `load` on the object, which is what you'd get from a cache miss. Calling `dump` or `bg.dump` will silently do nothing.

```
--repr--(self)
```

```
repr(x)
```

Overrides: `object.__repr__` extit(inherited documentation)

```
copy(self)
```

```
addinfo(self, *s, **kv)
```

**Note:** This does *not* modify `self`! It creates a new object.

```
makespace(self, avoid)
```

```
dump(self, e)
```

Cache some data on the disk.

**Parameters**

`e`: the data to write.

*(type=anything picklable.)*

**Return Value**

whatever was passed as `e`.

*(type=could be anything picklable.)*

**Note:** This function might quietly fail. Since this is a cache, failure to write is not considered a major problem. In my experience, failure to write is often caused by intermittent network problems, and you don't want it to crash a long-running computation.

```
bg_dump(self, e)
```

```
load(self)
```

Pull in some data from the disk.

**Return Value**

whatever was cached on disk.

*(type=could be anything picklable.)*

**Raises**

`BadFileFormat` when the data isn't valid.

`OSError` on cache miss.

`IOError` e.g. network problems.



**cachepath(*self*)**

Return a pathname suitable for cacheing some result.

**Return Value**

(*path\_to\_root*, *path\_with\_tail*). *Path\_to\_root* is/will be a directory; *path\_with\_tail* is a path to a data file within that directory. Normally, the actual cache is at the location `os.path.join(path_to_root, path_with_tail)` on the disk; that is what you would pass as the *fname* argument to `load_cache` or `dump_cache`.

(*type*=*tuple(str, str)*)

**Raises**

`ValueError` if you haven't specified any *info* yet.

***Inherited from object***

`__delattr__()`, `__format__()`, `__getattr__()`, `__hash__()`, `__new__()`, `__reduce__()`, `__reduce_ex__()`, `__setattr__()`, `__sizeof__()`, `__str__()`, `__subclasshook__()`

**16.5.2 Properties**

Name	Description
<i>Inherited from object</i>	
<code>__class__</code>	

**16.5.3 Class Variables**

Name	Description
Age	<b>Value:</b> 864000
NumObj	<b>Value:</b> 10000
Errors	<b>Value:</b> (<type 'exceptions.IOError'>, <type 'exceptions.EOFError'>...)

## 17 Module *gmisclib.chunkio*

These are I/O routines to allow you to write stuff like arrays and dictionaries (and arrays of dictionaries) to a human-readable file.

The format is normally STARTMARKER LENGTH.INFO DATA ENDMARKER, where STARTMARKER is something like "a{}", that specifies you're at the beginning of an array. LENGTH.INFO can depend on the data type, but it's normally an integer. Then comes the data, and finally as a check, you encounter the ENDMARKER, which is normally "". All of this is recursive, of course.

**Version:** \$Revision: 1.24 \$

### 17.1 Functions

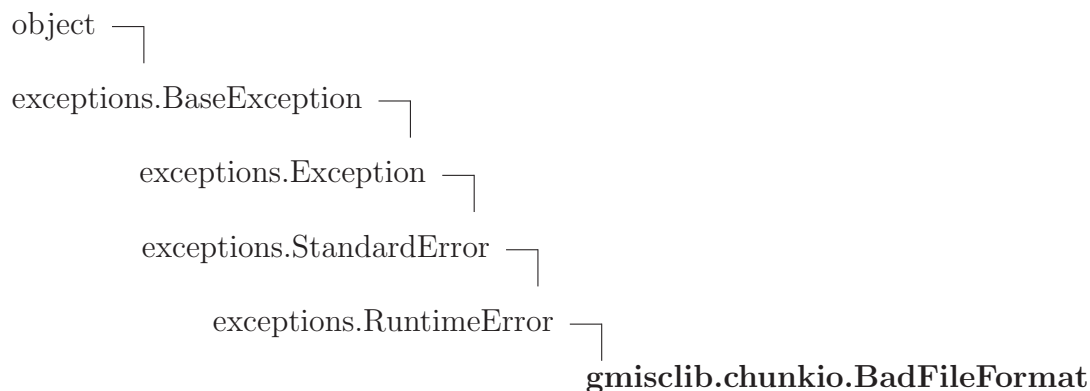
<code>test_e()</code>
-----------------------

<code>test()</code>
---------------------

### 17.2 Variables

Name	Description
<code>--package--</code>	<b>Value:</b> 'gmisclib'

### 17.3 Class *BadFileFormat*



**17.3.1 Methods**

**`__init__(self, s)`**  
`x.__init__(...)` initializes `x`; see `help(type(x))` for signature  
 Overrides: `object.__init__` extit(inherited documentation)

***Inherited from exceptions.RuntimeError***`__new__()`***Inherited from exceptions.BaseException***

`__delattr__()`, `__getattr__()`, `__getitem__()`, `__getslice__()`, `__reduce__()`, `__repr__()`,  
`__setattr__()`, `__setstate__()`, `__str__()`, `__unicode__()`

***Inherited from object***

`__format__()`, `__hash__()`, `__reduce_ex__()`, `__sizeof__()`, `__subclasshook__()`

**17.3.2 Properties**

Name	Description
<i>Inherited from exceptions.BaseException</i>	
<code>args</code> , <code>message</code>	
<i>Inherited from object</i>	
<code>__class__</code>	

**17.4 Class *chunk*****Known Subclasses:** `gmislib.chunkio.datachunk`, `gmislib.chunkio.stringchunk`

Low level file I/O operations. This class represents a sequence of white-space separated chunks of data.

**17.4.1 Methods**

**`__init__(self)`**  
 Constructor.

**`more(self)`**  
 Returns zero if the data source is empty. Returns nonzero if there is more data.

<b>readchunk</b> ( <i>self</i> )
----------------------------------

Read in the next white-space delimited chunk of text.
---

<b>read_float</b> ( <i>self</i> )
-----------------------------------

<b>read_array</b> ( <i>self</i> , <i>fcn</i> )
--

Read an array of data values. Raw text is converted to finished array values by the specified fcn.
--

<b>read_tuple</b> ( <i>self</i> , <i>fcn</i> )
--

Read a tuple of data values. Raw text is converted to finished array values by the specified fcn.
---

<b>read_array_of</b> ( <i>self</i> , <i>fcn</i> )
---

Read an array of data values. Values are read in by the specified fcn.
--

<b>read_dict</b> ( <i>self</i> , <i>fcn</i> )
---

Read a dictionary of data values. Raw text is converted to finished values by the specified fcn. Keys are strings.
--

<b>read_dict_of</b> ( <i>self</i> , <i>fcn</i> )
--

Read a dictionary of data values. Values are read in by the specified fcn. This allows dictionaries of X, where X can be a complex datatype like an array. Keys are strings.
--

<b>groupstart</b> ( <i>self</i> )
-----------------------------------

<b>groupend</b> ( <i>self</i> )
---------------------------------

<b>read_NumArray</b> ( <i>self</i> )
--------------------------------------

#### 17.4.2 Class Variables

Name	Description
<code>--doc--</code>	<b>Value:</b> ""Low level file I/O operatio...

## 17.5 Class *datachunk*

gmisclib.chunkio.chunk └─ **gmisclib.chunkio.datachunk**

Low level file I/O operations. This class represents a file as a sequence of white-space separated chunks of data.

### 17.5.1 Methods

**\_\_init\_\_**(*self*, *fd*)

Constructor.

Overrides: gmisclib.chunkio.chunk.\_\_init\_\_

**more**(*self*)

Returns False if the data source is empty.

**Return Value**

True if there is more data.

Overrides: gmisclib.chunkio.chunk.more

**readchunk**(*self*)

Read in the next white-space delimited chunk of text.

Overrides: gmisclib.chunkio.chunk.readchunk

**groupend**(*self*)

**groupstart**(*self*)

**read\_NumArray**(*self*)

**read\_array**(*self*, *fcn*)

Read an array of data values. Raw text is converted to finished array values by the specified fcn.

**read\_array\_of**(*self*, *fcn*)

Read an array of data values. Values are read in by the specified fcn.

---

**read\_dict**(*self*, *fcn*)

---

Read a dictionary of data values. Raw text is converted to finished values by the specified fcn. Keys are strings.

---

**read\_dict\_of**(*self*, *fcn*)

---

Read a dictionary of data values. Values are read in by the specified fcn. This allows dictionaries of X, where X can be a complex datatype like an array. Keys are strings.

---

**read\_float**(*self*)

---

**read\_tuple**(*self*, *fcn*)

---

Read a tuple of data values. Raw text is converted to finished array values by the specified fcn.

### 17.5.2 Class Variables

Name	Description
<code>__doc__</code>	<b>Value:</b> <code>""</code> "Low level file I/O operatio..."

## 17.6 Class *stringchunk*

```
gmisclib.chunkio.chunk └─ gmisclib.chunkio.stringchunk
```

Low level operations: splitting a string into chunks. This class represents a string as a sequence of white-space separated chunks of data.

### 17.6.1 Methods

---

**\_\_init\_\_**(*self*, *s*)

---

Constructor.

---

Overrides: `gmisclib.chunkio.chunk.__init__`

**more**(*self*)

Returns zero if the data source is empty. Returns nonzero if there is more data.

**Return Value**

zero if the data source is empty; nonzero if there is more data.

Overrides: *gmisclib.chunkio.chunk.more*

**readchunk**(*self*)

Read in the next white-space delimited chunk of text.

Overrides: *gmisclib.chunkio.chunk.readchunk*

**groupend**(*self*)**groupstart**(*self*)**read\_NumArray**(*self*)**read\_array**(*self*, *fcn*)

Read an array of data values. Raw text is converted to finished array values by the specified *fcn*.

**read\_array\_of**(*self*, *fcn*)

Read an array of data values. Values are read in by the specified *fcn*.

**read\_dict**(*self*, *fcn*)

Read a dictionary of data values. Raw text is converted to finished values by the specified *fcn*. Keys are strings.

**read\_dict\_of**(*self*, *fcn*)

Read a dictionary of data values. Values are read in by the specified *fcn*. This allows dictionaries of X, where X can be a complex datatype like an array. Keys are strings.

**read\_float**(*self*)**read\_tuple**(*self*, *fcn*)

Read a tuple of data values. Raw text is converted to finished array values by the specified *fcn*.

### 17.6.2 Class Variables

Name	Description
<code>__doc__</code>	<b>Value:</b> <code>"""Low level operations: splitting a string into chun...</code>

## 17.7 Class *chunk\_w*

**Known Subclasses:** *gmisclib.chunkio.chunkstring\_w*, *gmisclib.chunkio.datachunk\_w*

### 17.7.1 Methods

```
__init__(self)
```

```
writetchunk(self, ch, b=0)
```

```
write(self, ch, b=0)
```

```
nl(self)
```

```
comment(self, comment)
```

```
close(self)
```

```
write_None(self)
```

```
write_array_of(self, data, writer, b=0)
```

```
write_array(self, data, b=0, converter=<type 'str'>)
```

```
write_dict_of(self, data, writer, b=1)
```

```
write_dict(self, data, b=0, converter=<type 'str'>)
```



```
groupstart(self, contents, b=1, comment=None)
```

The 'contents' argument is conventionally a string containing 'a' for an internal array, 'g' for a group, 'd' for a dictionary... It describes the contents of the group. This is just used by the application to check what is in the group, so it could contain any chunk.

```
groupend(self, b=1)
```

```
write_NumArray(self, d, b=0, converter=<type 'str'>)
```

```
write_float(self, d, b=0)
```

```
stringwrite(self, ch, b=0)
```

## 17.8 Class `datachunk_w`

```
gmisclib.chunkio.chunk_w └─ gmisclib.chunkio.datachunk_w
```

This writes stuff to a file.

### 17.8.1 Methods

```
__init__(self, fd, width=80)
```

Overrides: `gmisclib.chunkio.chunk_w.__init__`

```
stringwrite(self, ch, b=0)
```

*ch* is the chunk of text. *b*=1 to begin a new line.

Overrides: `gmisclib.chunkio.chunk_w.stringwrite`

```
nl(self)
```

Overrides: `gmisclib.chunkio.chunk_w.nl`

```
comment(self, comment)
```

Overrides: `gmisclib.chunkio.chunk_w.comment`

**close**(*self*)Overrides: *gmisclib.chunkio.chunk\_w.close***\_\_del\_\_**(*self*)**groupend**(*self*, *b=1*)**groupstart**(*self*, *contents*, *b=1*, *comment=None*)

The 'contents' argument is conventionally a string containing 'a' for an internal array, 'g' for a group, 'd' for a dictionary... It describes the contents of the group. This is just used by the application to check what is in the group, so it could contain any chunk.

**write**(*self*, *ch*, *b=0*)**write\_None**(*self*)**write\_NumArray**(*self*, *d*, *b=0*, *converter=<type 'str'>*)**write\_array**(*self*, *data*, *b=0*, *converter=<type 'str'>*)**write\_array\_of**(*self*, *data*, *writer*, *b=0*)**write\_dict**(*self*, *data*, *b=0*, *converter=<type 'str'>*)**write\_dict\_of**(*self*, *data*, *writer*, *b=1*)**write\_float**(*self*, *d*, *b=0*)**writetechunk**(*self*, *ch*, *b=0*)

### 17.8.2 Class Variables

Name	Description
<code>__doc__</code>	Value: <code>"""This writes stuff to a file."""</code>

## 17.9 Class `chunkstring_w`

`gmisclib.chunkio.chunk_w` └─ **`gmisclib.chunkio.chunkstring_w`**

This accumulates stuff in memory, and returns a string when `close()` is called.

### 17.9.1 Methods

**`__init__(self)`**

Overrides: `gmisclib.chunkio.chunk_w.__init__`

**`stringwrite(self, ch, b=0)`**

Overrides: `gmisclib.chunkio.chunk_w.stringwrite`

**`nl(self)`**

Overrides: `gmisclib.chunkio.chunk_w.nl`

**`comment(self, comment)`**

Overrides: `gmisclib.chunkio.chunk_w.comment`

**`close(self)`**

Overrides: `gmisclib.chunkio.chunk_w.close`

**`groupend(self, b=1)`**

**`groupstart(self, contents, b=1, comment=None)`**

The 'contents' argument is conventionally a string containing 'a' for an internal array, 'g' for a group, 'd' for a dictionary... It describes the contents of the group. This is just used by the application to check what is in the group, so it could contain any chunk.

**`write(self, ch, b=0)`**

**`write_None(self)`**

**`write_NumArray(self, d, b=0, converter=<type 'str'>)`**

```
write_array(self, data, b=0, converter=<type 'str'>)
```

```
write_array_of(self, data, writer, b=0)
```

```
write_dict(self, data, b=0, converter=<type 'str'>)
```

```
write_dict_of(self, data, writer, b=1)
```

```
write_float(self, d, b=0)
```

```
writetechunk(self, ch, b=0)
```

### 17.9.2 Class Variables

Name	Description
<code>__doc__</code>	<b>Value:</b> <code>""</code> This accumulates stuff in memory, and returns a str...

## 18 Module `gmisclib.convex_hull2d`

`convexhull.py`

Calculate the convex hull of a set of  $n$  2D-points in  $O(n \log n)$  time. Taken from Berg et al., Computational Geometry, Springer-Verlag, 1997. Prints output as EPS file.

When run from the command line it generates a random set of points inside a square of given length and finds the convex hull for those, printing the result as an EPS file.

Usage: `convexhull.py <numPoints> <squareLength> <outFile>`

Dinu C. Gherman

Small Bug: Only works with a list of UNIQUE points, Evan Jones, 2005/05/18 If the list of points passed to this function is not unique, it will raise an assertion. To fix this, remove these lines from the beginning of the `convexHull` function:

Taken from <http://aspn.activestate.com/ASPN/Cookbook/Python/Recipe/66527> and modified to work with complex numbers.

### 18.1 Functions

<b><code>saveAsEps(<math>P</math>, <math>H</math>, <math>boxSize</math>, <math>path</math>)</code></b>
--

Save some points and their convex hull into an EPS file.
--

<b><code>convexHull(<math>P</math>)</code></b>
--

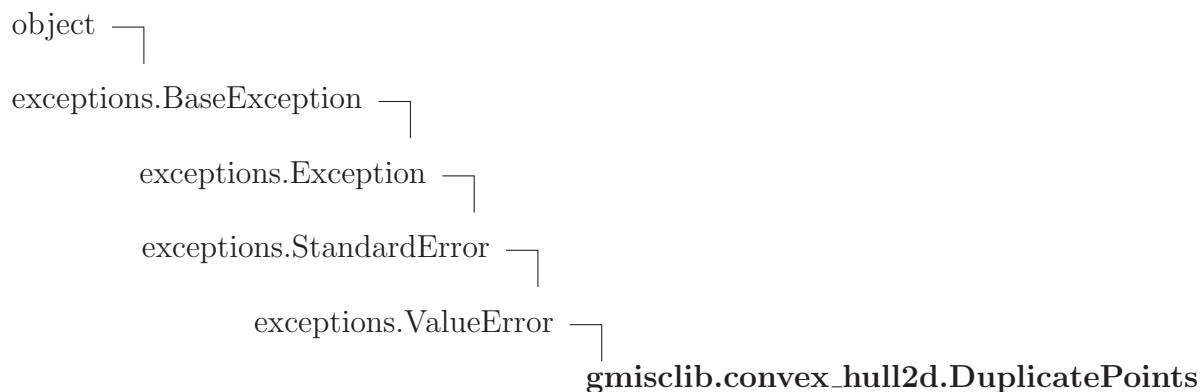
Calculate the convex hull of a set of complex points. If the hull has a duplicate point, an exception will be raised. It is up to the application not to provide duplicates.
--

<b><code>test()</code></b>
----------------------------

### 18.2 Variables

Name	Description
<code>epsHeader</code>	<b>Value:</b> <code>'%!PS-Adobe-2.0 EPSF-2.0\n%%%%BoundingBox: %d %d %d %d\n...'</code>
<code>__package__</code>	<b>Value:</b> <code>'gmisclib'</code>

### 18.3 Class *DuplicatePoints*



#### 18.3.1 Methods

**`__init__(self, s)`**  
`x.__init__(...)` initializes `x`; see `help(type(x))` for signature  
 Overrides: `object.__init__` `exitit`(inherited documentation)

*Inherited from `exceptions.ValueError`*

`__new__()`

*Inherited from `exceptions.BaseException`*

`__delattr__()`, `__getattr__()`, `__getitem__()`, `__getslice__()`, `__reduce__()`, `__repr__()`,  
`__setattr__()`, `__setstate__()`, `__str__()`, `__unicode__()`

*Inherited from `object`*

`__format__()`, `__hash__()`, `__reduce_ex__()`, `__sizeof__()`, `__subclasshook__()`

#### 18.3.2 Properties

Name	Description
<i>Inherited from <code>exceptions.BaseException</code></i>	
<code>args</code> , <code>message</code>	
<i>Inherited from <code>object</code></i>	
<code>__class__</code>	

## 19 Module *gmisclib.dict\_vector*

Vectors of numbers, but indexed as a dictionary.

### 19.1 Functions

<code>round(<i>x</i>)</code>
------------------------------

<code>log(<i>x</i>)</code>
----------------------------

<code>min_within(<i>x</i>)</code>
-----------------------------------

<code>max_within(<i>x</i>)</code>
-----------------------------------

<code>min_between(*<i>x</i>)</code>
-------------------------------------

<code>max_between(*<i>x</i>)</code>
-------------------------------------

<code>to_float(<i>d</i>)</code>
---------------------------------

Ensure that all the values are float.

**Parameters**

*d*: a dictionary

*(type=dict(str:anything))*

**Return Value**

a dict\_vector made of floats.

<code>to_numpy_float_1(<i>d</i>)</code>
---

Ensure that all the values are float.

**Parameters**

*d*: a dictionary

*(type=dict(str:anything))*

**Return Value**

a dict\_vector made of floats.

**to\_numpy\_float\_2(*d*)**

Ensure that all the values are float.

**Parameters**

**d**: a dictionary  
*(type=dict(str:anything))*

**Return Value**

a `dict_vector` made of floats.

**to\_int(*d*)**

Ensure that all the values are int.

**Parameters**

**d**: a dictionary  
*(type=dict(str:anything))*

**Return Value**

a `dict_vector` made of int.

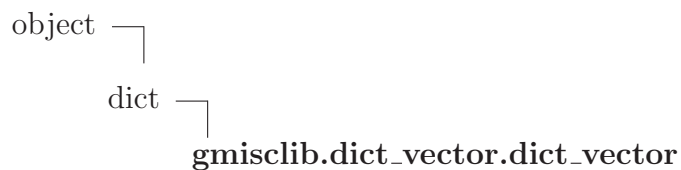
**outer(*a*, *b*)****Return Value**

an outer product.

## 19.2 Variables

Name	Description
<code>--package--</code>	<b>Value:</b> <code>'gmisclib'</code>

## 19.3 Class `dict_vector`





### 19.3.1 Methods

**`copy(self)`**

**Return Value**

a shallow copy of D

Overrides: `dict.copy` `exitit`(inherited documentation)

**`incr(self, k, v)`**

Increment slot `k` by `v`.

**`__add__(self, other)`**

**`__gt__(self, other)`**

`x > y`

Overrides: `dict.__gt__` `exitit`(inherited documentation)

**`__lt__(self, other)`**

`x < y`

Overrides: `dict.__lt__` `exitit`(inherited documentation)

**`__ge__(self, other)`**

`x >= y`

Overrides: `dict.__ge__` `exitit`(inherited documentation)

**`__le__(self, other)`**

`x <= y`

Overrides: `dict.__le__` `exitit`(inherited documentation)

**`__mul__(self, other)`**

**`__div__(self, other)`**

**`__iadd__(self, other)`**

**`__isub__(self, other)`**

`--sub--(self, other)`

`--radd--(self, other)`

`--rmul--(self, other)`

`--rsub--(self, other)`

`--abs--(self)`

`--invert--(self)`

`sum(self)`

Sum all the items in a vector.

`median(self)`

Take the median value of a vector.

`all(self)`

**Return Value**

true if all values are true.

(*type=bool*)

`float(self)`

**Return Value**

a `dict_vector` with float values.

`int(self)`

**Return Value**

a `dict_vector` with int values.

`iint(self)`

Convert to integer, in place.

`ifloat(self)`

Convert to float, in place.

**sign**(*self*)

**Return Value**

a dict\_vector which is -1, 0, or 1, showing the sign of each value.

(*type=dict\_vector of int.*)

**round**(*self*)

Round to the nearest int.

**Return Value**

rounded dict vector.

(*type=dict\_vector of ints.*)

**iround**(*self*)

Round to the nearest int, in place.

**idropzero**(*self*)

Drop all zeros, in place.

**dropzero**(*self*)

**Return Value**

a dict\_vector without any zeros.

**Inherited from dict**

`__cmp__()`, `__contains__()`, `__delitem__()`, `__eq__()`, `__getattr__()`, `__getitem__()`, `__init__()`, `__iter__()`, `__len__()`, `__ne__()`, `__new__()`, `__repr__()`, `__setitem__()`, `__sizeof__()`, `clear()`, `fromkeys()`, `get()`, `has_key()`, `items()`, `iteritems()`, `iterkeys()`, `itervalues()`, `keys()`, `pop()`, `popitem()`, `setdefault()`, `update()`, `values()`, `viewitems()`, `viewkeys()`, `viewvalues()`

**Inherited from object**

`__delattr__()`, `__format__()`, `__reduce__()`, `__reduce_ex__()`, `__setattr__()`, `__str__()`, `__subclasshook__()`

### 19.3.2 Properties

Name	Description
<i>Inherited from object</i>	
<code>__class__</code>	

### 19.3.3 Class Variables

Name	Description
SCALARS	Value: (<type 'int'>, <type 'float'>)
ZERO	Value: 0
<i>Inherited from dict</i>	
__hash__	

## 20 Module *gmisclib.dictops*

Operations on dictionaries.

**Version:** \$Revision: 1.9 \$

### 20.1 Functions

<b>filter</b> ( <i>d</i> , <i>l</i> )
Passes through items listed in <i>l</i> .
<b>Parameters</b>
<i>l</i> : list of keys to preserve ( <i>type=list</i> )
<i>d</i> : dictionary to copy/modify. ( <i>type=mapping</i> )
<b>Return Value</b>
a dictionary where all keys not listed in <i>l</i> are removed; all entries listed in <i>l</i> are copied into the (new) output dictionary. ( <i>type=dict</i> )

<b>remove</b> ( <i>d</i> , <i>l</i> )
Removes items listed in <i>l</i> .
<b>Parameters</b>
<i>l</i> : list of keys to remove ( <i>type=list</i> )
<i>d</i> : dictionary to copy/modify. ( <i>type=mapping</i> )
<b>Return Value</b>
a dictionary where all keys listed in <i>l</i> are removed; all entries not listed in <i>l</i> are copied into the (new) output dictionary. ( <i>type=dict</i> )

**intersection**(\**dlist*)**Parameters***dlist*: dictionaries.*(type=list(dict))***Return Value**

a dictionary containing those key=value pairs that are common to all the dictionaries.

*(type=dict)***test\_intersection()****rev1to1**(*d*)

Reverse a 1-1 mapping so it maps values to keys instead of keys to values.

**Parameters***d*: *(type=dict)***Return Value**

M such that if  $m[k]=v$ ,  $M[v]=k$

*(type=dict)***Raises**

`ValueError` if the mapping is many-to-one.

**compose**(*d2*, *d1*)

Compose two mappings (dictionaries) into one.

**Parameters***d2*: second mapping*d1*: first mapping**Return Value**

*d* such that  $d[k] = d2[d1[k]]$

*(type=dict)*

**read2c(*f*)**

Read in a dictionary from a two-column file; columns are separated by whitespace. This is not a completely general format, because keys cannot contain whitespace and the values cannot start or end in whitespace.

The format can contain comments at the top; these are lines that begin with a hash mark (#). Empty lines are ignored.

**Parameters**

**f**: an open file

**Return Value**

a dictionary and a list of comments

(*type=tuple(dict(str:str), list(str))*)

**Note:** You can get into trouble if you have a key that starts with a hash mark (#). If it happens to be the first key, and if there are no comments or if it directly follows a comment, it will be interpreted as a comment. To avoid this possibility, begin your file with a comment line, then a blank line.

**argmax(*d*)**

For a dictionary of key=value pairs, return the key with the largest value.

**Raises**

**ValueError** when the input dictionary is empty.

**Note:** When there are several equal values, this will return a single, arbitrary choice.

**add\_dol(*dict, key, value*)**

OBSOLETE: replace with class **dict\_of\_lists**. Add an entry to a dictionary of lists. A new list is created if necessary; else the value is appended to an existing list. Obsolete: replace with **dict\_of\_lists**.

**add\_doc(*dict, key, value*)**

OBSOLETE: replace with class **dict\_of\_accums**. Add an entry to a dictionary of counters. A new counter is created if necessary; else the value is added to an existing counter. Obsolete: replace with **dict\_of\_accums**.

**test()**

## 20.2 Variables

Name	Description
<code>--package--</code>	<b>Value:</b> None

## 20.3 Class *BadFileFormatError*



This indicates a problem when reading in a dictionary via `read2c`.

### 20.3.1 Methods

```

__init__(self, *s)

x.__init__(...) initializes x; see help(type(x)) for signature
Overrides: object.__init__ extit(inherited documentation)
  
```

*Inherited from `exceptions.Exception`*

```
__new__()
```

*Inherited from `exceptions.BaseException`*

```

__delattr__(), __getattr__(), __getitem__(), __getslice__(), __reduce__(), __repr__(),
__setattr__(), __setstate__(), __str__(), __unicode__()
  
```

*Inherited from `object`*

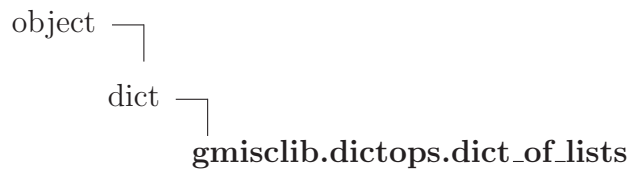
```
__format__(), __hash__(), __reduce_ex__(), __sizeof__(), __subclasshook__()
```

### 20.3.2 Properties

Name	Description
<i>Inherited from <code>exceptions.BaseException</code></i>	
args, message	
<i>Inherited from <code>object</code></i>	
<code>--class--</code>	



## 20.4 Class `dict_of_lists`



A dictionary of lists.

### 20.4.1 Methods

<b><code>add(self, key, value)</code></b>
---

Append value to the list indexed by key.
--

<b><code>addgroup(self, key, values)</code></b>
---

Append values to the list indexed by key.
---

<b><code>add_ifdifferent(self, key, value)</code></b>
---

Append value to the list indexed by key.
--

<b><code>copy(self)</code></b>
--------------------------------

This does a shallow copy.
---------------------------

<b>Return Value</b>
---------------------

a shallow copy of D
---------------------

Overrides: <code>dict.copy</code>
-----------------------------------

<b><code>merge(self, other)</code></b>
--

#### *Inherited from `dict`*

`__cmp__()`, `__contains__()`, `__delitem__()`, `__eq__()`, `__ge__()`, `__getattribute__()`, `__getitem__()`, `__gt__()`, `__init__()`, `__iter__()`, `__le__()`, `__len__()`, `__lt__()`, `__ne__()`, `__new__()`, `__repr__()`, `__setitem__()`, `__sizeof__()`, `clear()`, `fromkeys()`, `get()`, `has_key()`, `items()`, `iteritems()`, `iterkeys()`, `itervalues()`, `keys()`, `pop()`, `popitem()`, `setdefault()`, `update()`, `values()`, `viewitems()`, `viewkeys()`, `viewvalues()`

#### *Inherited from `object`*

`__delattr__()`, `__format__()`, `__reduce__()`, `__reduce_ex__()`, `__setattr__()`, `__str__()`, `__subclasshook__()`

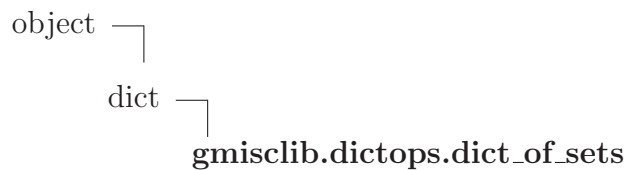
### 20.4.2 Properties

Name	Description
<i>Inherited from object</i> <code>__class__</code>	

### 20.4.3 Class Variables

Name	Description
<i>Inherited from dict</i> <code>__hash__</code>	

## 20.5 Class `dict_of_sets`



A dictionary of lists.

### 20.5.1 Methods

<b><code>add(self, key, value)</code></b>
Append value to the set indexed by key.
<b><code>addgroup(self, key, values)</code></b>
Append values to the list indexed by key.
<b><code>copy(self)</code></b>
This does a shallow copy.
<b>Return Value</b>
a shallow copy of D
Overrides: <code>dict.copy</code>
<b><code>merge(self, other)</code></b>

*Inherited from dict*

`__cmp__()`, `__contains__()`, `__delitem__()`, `__eq__()`, `__ge__()`, `__getattr__()`, `__getitem__()`, `__gt__()`, `__init__()`, `__iter__()`, `__le__()`, `__len__()`, `__lt__()`, `__ne__()`, `__new__()`, `__repr__()`, `__setitem__()`, `__sizeof__()`, `clear()`, `fromkeys()`, `get()`, `has_key()`, `items()`, `iteritems()`, `iterkeys()`, `itervalues()`, `keys()`, `pop()`, `popitem()`, `setdefault()`, `update()`, `values()`, `viewitems()`, `viewkeys()`, `viewvalues()`

### ***Inherited from object***

`__delattr__()`, `__format__()`, `__reduce__()`, `__reduce_ex__()`, `__setattr__()`, `__str__()`, `__subclasshook__()`

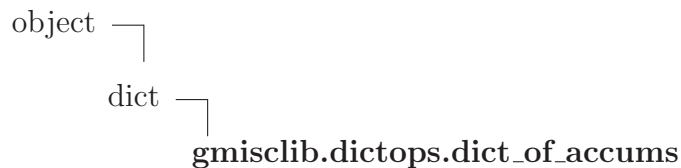
#### **20.5.2 Properties**

Name	Description
<i>Inherited from object</i>	
<code>__class__</code>	

#### **20.5.3 Class Variables**

Name	Description
<i>Inherited from dict</i>	
<code>__hash__</code>	

## **20.6 Class `dict_of_accums`**



A dictionary of accumulators. Note that the `copy()` function produces a `dict_of_accums`.

#### **20.6.1 Methods**

<b><code>add(self, key, value)</code></b>
Add value to the value indexed by key.

**`copy(self)`**

**Return Value**

a shallow copy of D

Overrides: `dict.copy` `exitit`(inherited documentation)

**`merge(self, other)`**

***Inherited from dict***

`__cmp__()`, `__contains__()`, `__delitem__()`, `__eq__()`, `__ge__()`, `__getattribute__()`, `__getitem__()`, `__gt__()`, `__init__()`, `__iter__()`, `__le__()`, `__len__()`, `__lt__()`, `__ne__()`, `__new__()`, `__repr__()`, `__setitem__()`, `__sizeof__()`, `clear()`, `fromkeys()`, `get()`, `has_key()`, `items()`, `iteritems()`, `iterkeys()`, `itervalues()`, `keys()`, `pop()`, `popitem()`, `setdefault()`, `update()`, `values()`, `viewitems()`, `viewkeys()`, `viewvalues()`

***Inherited from object***

`__delattr__()`, `__format__()`, `__reduce__()`, `__reduce_ex__()`, `__setattr__()`, `__str__()`, `__subclasshook__()`

### 20.6.2 Properties

Name	Description
<i>Inherited from object</i>	
<code>__class__</code>	

### 20.6.3 Class Variables

Name	Description
<i>Inherited from dict</i>	
<code>__hash__</code>	

## 20.7 Class `dict_of_averages`



**Known Subclasses:** `gmisclib.spread_jobs.PastPerformance`

A dictionary of accumulators. Note that the `copy()` function produces a `dict_of_averages`.

### 20.7.1 Methods

<code>add(self, key, value, weight=1.0)</code>
--

Add value to the value indexed by key.
--

<code>get_avg(self, key)</code>
---------------------------------

<code>get_avgs(self)</code>
-----------------------------

<code>get_both(self, key)</code>
----------------------------------

<code>copy(self)</code>
-------------------------

**Return Value**

a shallow copy of D
---------------------

Overrides: <code>dict.copy</code> <code>extit</code> (inherited documentation)
--

<code>merge(self, other)</code>
---------------------------------

***Inherited from dict***

`__cmp__()`, `__contains__()`, `__delitem__()`, `__eq__()`, `__ge__()`, `__getattr__()`, `__getitem__()`, `__gt__()`, `__init__()`, `__iter__()`, `__le__()`, `__len__()`, `__lt__()`, `__ne__()`, `__new__()`, `__repr__()`, `__setitem__()`, `__sizeof__()`, `clear()`, `fromkeys()`, `get()`, `has_key()`, `items()`, `iteritems()`, `iterkeys()`, `itervalues()`, `keys()`, `pop()`, `popitem()`, `setdefault()`, `update()`, `values()`, `viewitems()`, `viewkeys()`, `viewvalues()`

***Inherited from object***

`__delattr__()`, `__format__()`, `__reduce__()`, `__reduce_ex__()`, `__setattr__()`, `__str__()`, `__subclasshook__()`

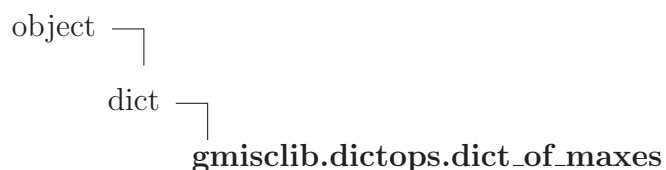
### 20.7.2 Properties

Name	Description
<i>Inherited from object</i>	
<code>__class__</code>	

### 20.7.3 Class Variables

Name	Description
<i>Inherited from dict</i>	
<code>__hash__</code>	

## 20.8 Class `dict_of_maxes`



A dictionary of maxima. Add a new number and the stored maximum will increase iff the new value is bigger. Note that the `copy()` function produces a `dict_of_accums`.

### 20.8.1 Methods

<b><code>add(self, key, value)</code></b>
---

Add value to the value indexed by key.
--

<b><code>copy(self)</code></b>
--------------------------------

**Return Value**

a shallow copy of D

Overrides: `dict.copy` extit(inherited documentation)

<b><code>merge(self, other)</code></b>
--

#### *Inherited from `dict`*

`__cmp__()`, `__contains__()`, `__delitem__()`, `__eq__()`, `__ge__()`, `__getattr__()`, `__getitem__()`, `__gt__()`, `__init__()`, `__iter__()`, `__le__()`, `__len__()`, `__lt__()`, `__ne__()`, `__new__()`, `__repr__()`, `__setitem__()`, `__sizeof__()`, `clear()`, `fromkeys()`, `get()`, `has_key()`, `items()`, `iteritems()`, `iterkeys()`, `itervalues()`, `keys()`, `pop()`, `popitem()`, `setdefault()`, `update()`, `values()`, `viewitems()`, `viewkeys()`, `viewvalues()`

#### *Inherited from `object`*

`__delattr__()`, `__format__()`, `__reduce__()`, `__reduce_ex__()`, `__setattr__()`, `__str__()`, `__subclasshook__()`

### 20.8.2 Properties

Name	Description
<i>Inherited from <code>object</code></i>	
<code>__class__</code>	

### 20.8.3 Class Variables

Name	Description
<i>Inherited from dict</i>	
<code>__hash__</code>	

## 20.9 Class `dict_of_X`



A dictionary of arbitrary things. Note that the `copy()` function produces a `dict_of_X`.

### 20.9.1 Methods

```
__init__(self, constructor, incrementer)
```

`x.__init__(...)` initializes `x`; see `help(type(x))` for signature

**Return Value**  
new empty dictionary

Overrides: `object.__init__` extit(inherited documentation)

```
add(self, key, *value)
```

---

Add value to the thing indexed by key.

#### *Inherited from dict*

```
__cmp__(), __contains__(), __delitem__(), __eq__(), __ge__(), __getattr__(), __getitem__(),
__gt__(), __iter__(), __le__(), __len__(), __lt__(), __ne__(), __new__(), __repr__(), __setitem__(),
__sizeof__(), clear(), copy(), fromkeys(), get(), has_key(), items(), iteritems(), iterkeys(),
itervalues(), keys(), pop(), popitem(), setdefault(), update(), values(), viewitems(),
viewkeys(), viewvalues()
```

#### *Inherited from object*

```
__delattr__(), __format__(), __reduce__(), __reduce_ex__(), __setattr__(), __str__(), __subclasshook__()
```

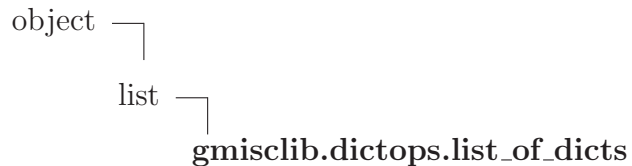
### 20.9.2 Properties

Name	Description
<i>Inherited from object</i>	
<code>__class__</code>	

### 20.9.3 Class Variables

Name	Description
<i>Inherited from dict</i>	
<code>__hash__</code>	

## 20.10 Class `list_of_dicts`



### 20.10.1 Methods

<code>__init__(self, *arg)</code>
<code>x.__init__(...)</code> initializes <code>x</code> ; see <code>help(type(x))</code> for signature
<b>Return Value</b>
new empty list
Overrides: <code>object.__init__</code> extit(inherited documentation)

<code>lookup1(self, key, value)</code>
--

#### *Inherited from list*

`__add__()`, `__contains__()`, `__delitem__()`, `__delslice__()`, `__eq__()`, `__ge__()`, `__getattribute__()`, `__getitem__()`, `__getslice__()`, `__gt__()`, `__iadd__()`, `__imul__()`, `__iter__()`, `__le__()`, `__len__()`, `__lt__()`, `__mul__()`, `__ne__()`, `__new__()`, `__repr__()`, `__reversed__()`, `__rmul__()`, `__setitem__()`, `__setslice__()`, `__sizeof__()`, `append()`, `count()`, `extend()`, `index()`, `insert()`, `pop()`, `remove()`, `reverse()`, `sort()`

#### *Inherited from object*

`__delattr__()`, `__format__()`, `__reduce__()`, `__reduce_ex__()`, `__setattr__()`, `__str__()`, `__subclasshook__()`



### 20.10.2 Properties

Name	Description
<i>Inherited from object</i> <code>__class__</code>	

### 20.10.3 Class Variables

Name	Description
<i>Inherited from list</i> <code>__hash__</code>	

## 21 Module *gmisclib.die*

### 21.1 Functions

**die**(*s*)

Output a fatal error message and terminate. Before the error message, it summarizes all the calls to **note**.

**Parameters**

**s**: error message

(*type=**str*)

**warn**(*s*)

Output a non-fatal warning. Before the warning, it summarizes all the calls to **note**.

**Parameters**

**s**: warning message

(*type=**str*)

**info**(*s*)

Output useful information. Before the message, it summarizes all the calls to **note**.

**Parameters**

**s**: message

(*type=**str*)

**catch**(*extext=***None**)

Call this inside an except statement. It will report the exception and any other information it has.

**catchexit**(*extext=***None**, *n=***1**, *text=***None**)

Call this inside an except statement. It will report all information and then exit.

**dbg(*s*)**

Output debugging information, if `debug` is nonzero. Before the debugging info, it prints all information given to `note`.

**Parameters**

**s:** debug message  
(*type=*`str`)

**exit(*n*, *text=None*)**

Exit, after dumping accumulated `notes`.

**Parameters**

**n:** the processes' exit code  
(*type=*`int`)  
**text:** something to print  
(*type=*`str` or `None`)

**flush()**

Flush out any accumulated messages. Should be called shortly before exit.

**Note:** This matters when `nocompress=False` under conditions where `sys.stderr` or `sys.stdout` may be closed before the program terminates. Under those conditions, you might get a "last message repeated %d times" message produced into a closed file descriptor, which would lead to an `IOError`.

**note(*key*, *value*)**

Memorize a note, which will be output along with the next error/warning/info message. These will be printed in the form '`#NOTE key = value`', with value converted to a string at the time of printing. Note that only the most recent note is kept for each key.

**Parameters**

**value:** (*type=*`anything`)  
**key:** (*type=*`str`)

**get**(*key*)Get the most recent value memorized by a call to **note**.**Parameters****key**: (*type=**str*)**Return Value**

whatever was memorized

**uprint**(*s*)

Like a print statement, except it prints in utf-8 encoding instead of ASCII.

**se\_write**(*s*)

Like sys.stderr.write(), except it prints in utf-8 encoding instead of ASCII.

**so\_write**(*s*)

Like sys.stdout.write(), except it prints in utf-8 encoding instead of ASCII.

## 21.2 Variables

Name	Description
debug	Set to False to disable dbg() messages. <b>Value:</b> True
nocompress	False means that only the first of repeated messages are displayed; True displays all. <b>Value:</b> False
Threads	<b>Value:</b> False
__package__	<b>Value:</b> 'gmisclib'

## 21.3 Class counter

### 21.3.1 Methods

**\_\_init\_\_**(*self*)**set\_out\_err**(*self*, *stdout*, *stderr*)**incoming**(*self*, *msgtype*, *name*, *s*)

<code>showreps(<i>self</i>)</code>
------------------------------------

<code>showdump(<i>self</i>, <i>s</i>)</code>
--

<code>__del__(<i>self</i>)</code>
-----------------------------------

<code>clearmem(<i>self</i>)</code>
------------------------------------

<code>dumpmem(<i>self</i>)</code>
-----------------------------------

<code>memorize(<i>self</i>, <i>key</i>, <i>value</i>)</code>
--

### 21.3.2 Class Variables

Name	Description
lock	<b>Value:</b> threading.Lock()

## 22 Module `gmisclib.die2`

### 22.1 Functions

<b>note</b> ( <i>name</i> , <i>val</i> )
--

<b>show</b> ( <i>name</i> )
-----------------------------

<b>die</b> ( <i>s</i> )
-------------------------

Output a fatal error message and terminate.
---

<b>warn</b> ( <i>s</i> )
--------------------------

Output a non-fatal warning.
-----------------------------

<b>info</b> ( <i>s</i> )
--------------------------

Output useful information.
----------------------------

<b>catch</b> ( <i>extext</i> =None)
-------------------------------------

Call this inside an except statement. It will report the exception and any other information it has.
--

<b>catchexit</b> ( <i>extext</i> =None, <i>n</i> =1, <i>text</i> =None)
---

Call this inside an except statement. It will report all information and then exit.
---

<b>dbg</b> ( <i>s</i> )
-------------------------

Output debugging information, if debug is nonzero.
--

<b>exit</b> ( <i>n</i> , <i>text</i> =None)
---

Exit, after dumping accumulated messages.
---

<b>get</b> ( <i>key</i> )
---------------------------

### 22.2 Variables

Name	Description
debug	Value: True

*continued on next page*

Name	Description
compress	<b>Value:</b> True
logfd	<b>Value:</b> sys.stderr
__package__	<b>Value:</b> 'gmisclib'

## 23 Module gmisclib.do\_exec

Execute a \*nix command and capture the output.

**Version:** \$Revision: 1.5 \$

### 23.1 Functions

#### **getall(*s*)**

Read a list of lines from a process, after dropping junk like comments (beginning with whitespace then '#' ) or blank lines. Raises an exception if if the process returns a line beginning with 'ERR:' The argument *s* is a string containing the command and its arguments. *S* is normally passed to a shell.

#### **get(*s*)**

Read a single line from a process, after dropping junk like comments (#) or blank lines. Returns a trouble indication if the process returns 'ERR:'

### 23.2 Variables

Name	Description
__package__	<b>Value:</b> 'gmisclib'



## 24 Module *gmisclib.ds9\_region*

### 24.1 Variables

Name	Description
<code>--package--</code>	Value: <code>'gmisclib'</code>

### 24.2 Class writer

#### 24.2.1 Methods

```
--init--(self, fd)
```

```
header(self, d)
```

```
text(self, x, y, text)
```

```
close(self)
```

```
--del--(self)
```

## 25 Module gmisclib.dtw4

### 25.1 Functions

**check\_names(...)**

**dtw(...)**

A standard Dynamic Time Warp match of two sequences of feature vectors. The allowed time increments are (0,1), (1,0), and (1,1).

#### Parameters

- x:** one array of feature vectors  
*(type=numpy.ndarray, 2-dimensional. First index is the time axis, second index is the feature axis.)*
- y:** the other array of feature vectors  
*(type=numpy.ndarray, 2-dimensional. First index is the time axis, second index is the feature axis.)*
- alpha:** exponent for differences in features. 2=Euclidean, 1=city block  
*(type=float)*
- beta:** exponent for summing along time axis. Normally 1.  
*(type=float)*

#### Return Value

An array that maps from indices in x to indices into y.  $x[r[i,0]]$  matches  $y[r[i,1]]$  for all i.  
*(type=numpy.ndarray, 2-dimensional array of integers with shape [n,2]. [i,0] is the index (time) in x, [i,1] is the index (time) in y.)*

**dtw\_ti(...)**

Does a DTW on a pair of signals, but allows you to specify a region of each signal to match. It returns the alignment in time units, rather than in integer indices.

**Parameters**

**y**: (*type=gpkimgclass.gpk\_img*)

**x**: (*type=gpkimgclass.gpk\_img*)

**Return Value**

An array of corresponding times. The time `rv[i,0]` in utterance `x` and `rv[i,1]` in utterance `y` correspond to similar acoustic events. Output has units of seconds.

(*type=numpy.ndarray - 2-dimensional array of floats.*)

**dtwjump(...)**

A constrained Dynamic Time Warp match of two sequences of feature vectors. It constrains the time mapping to be a piecewise linear function with a corner every `mm` samples.

**Parameters**

**x**: A sequence of feature vectors.

(*type=numpy 2-dimensional array of floats. The first index is time, the second index is within each feature vector.*)

**y**: A sequence of feature vectors.

(*type=numpy 2-dimensional array of floats. The first index is time, the second index is within each feature vector.*)

**map2dist(...)****map2index(...)**

**map2shared(...)**

This is used after `dtw()` to map the two sequences onto the same time axis.

**Parameters**

- x:** a sequence of feature vectors, typically the same data you pass into `dtw()`.
- y:** a sequence of feature vectors, typically the same data you pass into `dtw()`.
- tmap:** a time mapping as returned from `dtw()` or `dtwjump()`.  
(*type=*`numpy.ndarray[integer, 2-dimensional, shape is Nx2]`)

**Return Value**

a tuple with a time axis, the x-feature vectors mapped onto that time axis, and the y-feature vectors mapped onto the same axis. Analogous to x and y, the first index is the pseudotime index.

(*type=*`tuple(numpy.ndarray(float, 1-d), numpy.ndarray(float, 2-d), numpy.ndarray(float, 2-d))`)

**namelist(...)****parse\_args(...)****Parameters**

- args:** an argument list, much like you might feed to a command line program.  
(*type=*`list(things)` *Things that are flags need to be strings. Things that are numbers can be either floats, ints, or strings. The last two things (corresponding to the "filenames") can be anything.*)

**scale\_tmap(...)****shift\_idxmap(...)****test(...)****test\_partial(...)****test\_same(...)****test\_scaled(...)**

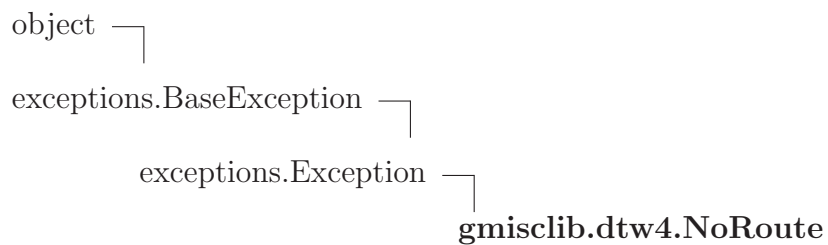
```
test_stretched(...)
```

```
time_it(...)
```

## 25.2 Variables

Name	Description
S	Value: '.'
__package__	Value: 'gmisclib'
__test__	Value: {}

## 25.3 Class NoRoute



### 25.3.1 Methods

```
__init__(...)
x.__init__(...) initializes x; see help(type(x)) for signature
Overrides: object.__init__ extit(inherited documentation)
```

#### *Inherited from exceptions.Exception*

```
__new__()
```

#### *Inherited from exceptions.BaseException*

```
__delattr__(), __getattr__(), __getitem__(), __getslice__(), __reduce__(), __repr__(),
__setattr__(), __setstate__(), __str__(), __unicode__()
```

#### *Inherited from object*

```
__format__(), __hash__(), __reduce_ex__(), __sizeof__(), __subclasshook__()
```

### 25.3.2 Properties

Name	Description
<i>Inherited from exceptions.BaseException</i>	
args, message	
<i>Inherited from object</i>	
__class__	

## 25.4 Class slice\_c



### 25.4.1 Methods

<b>__init__</b> (...)
x.__init__(...) initializes x; see help(type(x)) for signature
Overrides: object.__init__

<b>__new__</b> (T, S, ...)
<b>Return Value</b>
a new object with type S, a subtype of T
Overrides: object.__new__

### *Inherited from object*

\_\_delattr\_\_(), \_\_format\_\_(), \_\_getattr\_\_(), \_\_hash\_\_(), \_\_reduce\_\_(), \_\_reduce\_ex\_\_(),  
 \_\_repr\_\_(), \_\_setattr\_\_(), \_\_sizeof\_\_(), \_\_str\_\_(), \_\_subclasshook\_\_()

### 25.4.2 Properties

Name	Description
<i>Inherited from object</i>	
__class__	

## 25.5 Class `state_c`



### 25.5.1 Methods

**`--init--(...)`**

`x.--init--(...)` initializes `x`; see `help(type(x))` for signature

Overrides: `object.--init--`

**`--new--(T, S, ...)`**

**Return Value**

a new object with type `S`, a subtype of `T`

Overrides: `object.--new--`

#### *Inherited from `object`*

`--delattr--()`, `--format--()`, `--getattr--()`, `--hash--()`, `--reduce--()`, `--reduce_ex--()`,  
`--repr--()`, `--setattr--()`, `--sizeof--()`, `--str--()`, `--subclasshook--()`

### 25.5.2 Properties

Name	Description
<i>Inherited from <code>object</code></i>	
<code>--class--</code>	

## 25.6 Class `state_jump_c`



**25.6.1 Methods****`__init__(...)`**`x.__init__(...)` initializes `x`; see `help(type(x))` for signatureOverrides: `object.__init__`**`__new__(T, S, ...)`****Return Value**a new object with type `S`, a subtype of `T`Overrides: `object.__new__`***Inherited from object***`__delattr__()`, `__format__()`, `__getattr__()`, `__hash__()`, `__reduce__()`, `__reduce_ex__()`,  
`__repr__()`, `__setattr__()`, `__sizeof__()`, `__str__()`, `__subclasshook__()`**25.6.2 Properties**

Name	Description
<i>Inherited from object</i>	
<code>__class__</code>	



## 26 Module *gmisclib.dyn\_prog*

Viterbi search

### 26.1 Functions

<code>path(<i>nodecost</i>, <i>linkcost</i>)</code>
---

<code>test1()</code>
----------------------

<code>test2()</code>
----------------------

<code>test3()</code>
----------------------

### 26.2 Variables

Name	Description
<code>--package--</code>	Value: 'gmisclib'

## 27 Module `gmisclib.edit_distance`

Levenshtein (edit) distance between two strings of symbols.

### 27.1 Functions

**`dist(s, t, csub, cinsert, cdel)`**

Cost for converting string `s` into string `t`. This function takes three dictionaries that contain the cost of insertion, deletion, and substitutions.

**Parameters**

**`s`:** starting string  
*(type=string or array)*

**`t`:** ending string  
*(type=string or array)*

**`csub`:** cost of converting `a` to `b`  
*(type=dict((a,b) : float))*

**`cinsert`:** cost of insertion  
*(type=dict(a: float))*

**`cdel`:** cost of deletion  
*(type=dict(a: float))*

**`distf1(s, t, csub)`**

Cost for converting string `s` into string `t`. This function takes a function that computes the cost of insertions, deletions, or substitutions. (This is an alternative implementation for `distf2()`.)

**Parameters**

**`s`:** starting string  
*(type=string or array)*

**`t`:** ending string  
*(type=string or array)*

**`csub`:** cost of converting `a` to `b`  
*(type=function(a,b) : float)*

**distf2**(*s*, *t*, *csub*)

Cost for converting string *s* into string *t*. This function takes a function that computes the cost of insertions, deletions, or substitutions.

**Parameters**

- s:** starting string  
(*type=string or array*)
- t:** ending string  
(*type=string or array*)
- csub:** cost of converting *a* to *b*  
(*type=function(a,b) : float*)

**distf**(*s*, *t*, *csub*)

Cost for converting string *s* into string *t*. This function takes a function that computes the cost of insertions, deletions, or substitutions.

**Parameters**

- s:** starting string  
(*type=string or array*)
- t:** ending string  
(*type=string or array*)
- csub:** cost of converting *a* to *b*  
(*type=function(a,b) : float*)

**def\_cost**(*a*, *b*)

This is the default cost function for converting *a* into *b*. Insertions, deletions, and substitutions all have a unit cost. Normally, this is passed as an arg to `distf`.

**Parameters**

- a:** a symbol (or None to indicate an insertion).
- b:** a symbol (or None to indicate an deletion).

**Return Value**

- the cost of changing *a* into *b*  
(*type=int*)

**free\_sub(*a*, *b*)**

This is a sample cost function for converting **a** into **b**. Insertions and deletions have a unit cost; substitutions are free. Normally, this is passed as an arg to `distf`.

**Parameters**

**a**: a symbol (or `None` to indicate an insertion).

**b**: a symbol (or `None` to indicate an deletion).

**Return Value**

the cost of changing **a** into **b**

(*type=int*)

**free\_del(*a*, *b*)**

This is a sample cost function for converting **a** into **b**. Insertions and substitutions have a unit cost; deletion are free. Normally, this is passed as an arg to `distf`.

**Parameters**

**a**: a symbol (or `None` to indicate an insertion).

**b**: a symbol (or `None` to indicate an deletion).

**Return Value**

the cost of changing **a** into **b**

(*type=int*)

**test()****test2()**

## 27.2 Variables

Name	Description
<code>--package--</code>	<b>Value:</b> <code>'gmisclib'</code>

## 27.3 Class `text_cost`

object └─ `gmisclib.edit_distance.text_cost`

This is useful for differencing documents that have been parsed into lists of words. It computes the cost of a change of words by an edit distance computation on the characters within the words. (It caches the costs of commonly encountered words for speed.) Instances of this class can be passed as a cost function to `distf`.

### 27.3.1 Methods

---

**`__init__(self, costfac=None, cachesize=None, word_cost=None, frac_exp=0.0)`**

---

Create an instance.

**Parameters**

- costfac:** None, or an arbitrary function that multiplies the scaled edit distance between the two words. Called as `self.costfac(a,b)` where either `a` or `b` can be None for insertions or deletions.
- (type=function(str,str): float or None)*
- cachesize:** how large a cache of word-to-word distances should be kept?
- (type=None (meaning unlimited) or int)*
- word\_cost:** cost function to use inside words: this is a cost for insertion, deletion, or substitution of letters.
- (type=function(str,str): float or None (meaning use def\_cost).)*
- frac\_exp:** An exponent for scaling the edit distance by the length of words.
- (type=float)*

Overrides: `object.__init__`

---



---

**`__call__(self, a, b)`**

---

To be used by `distf`.

**Parameters**

- a:** *(type=str or None)*
- b:** *(type=str or None)*

**Return Value**

- a cost
- (type=presumably a float)*
- 

*Inherited from object*

`__delattr__()`, `__format__()`, `__getattr__()`, `__hash__()`, `__new__()`, `__reduce__()`, `__reduce_ex__()`,  
`__repr__()`, `__setattr__()`, `__sizeof__()`, `__str__()`, `__subclasshook__()`

### 27.3.2 Properties

Name	Description
<i>Inherited from object</i>	
<code>__class__</code>	

## 28 Module `gmisclib.entropy`

### 28.1 Functions

$P(x)$
--------

<code>multinomial_logp(<math>x</math>, <math>cF</math>)</code>
--

<code>multinomial_fixer(<math>x</math>, <math>c</math>)</code>
--

<code>information_gained(<math>c</math>, <math>pc=None</math>)</code>
---

Suppose there is an experiment where you have  $N$  conditions and in each conditions there are  $M$  possible measurements. The outcomes are represented by  $c[N,M]$  matrix of counts: how many times you observe each possible measurement for each condition.

We want to ask what's the information gained by taking a measurement. We assume the probability of each condition is a  $pc[N]$  vector, which defaults to the frequencies in  $c$ . (We assume that the choice of condition is not a random variable, but is made in advance, like most experiments.)

### 28.2 Variables

Name	Description
<code>--package--</code>	Value: <code>'gmisclib'</code>

## 29 Module *gmisclib.erb\_scale*

ERB Perceptual frequency scale.

**Version:** \$Revision: 1.6 \$

### 29.1 Functions

<b>f_to_erb</b> ( <i>f</i> )
------------------------------

Frequency to ERB band number. <a href="http://ccrma-www.stanford.edu/~jos/bbt/Equivalent_Rectangular_Bandwidth.html">http://ccrma-www.stanford.edu/~jos/bbt/Equivalent_Rectangular_Bandwidth.html</a>
---

<b>erb_to_f</b> ( <i>e</i> )
------------------------------

ERB number -> frequency
-------------------------

<b>ebw</b> ( <i>e</i> )
-------------------------

Critical bandwidth (in Hz) at frequency <i>e</i> (erbs).
--

### 29.2 Variables

Name	Description
<code>--package--</code>	<b>Value:</b> <code>'gmisclib'</code>



## 30 Module *gmisclib.evolution*

### 30.1 Variables

Name	Description
<code>--package--</code>	<b>Value:</b> None

## 31 Module gmisclib.fake\_file

A class that implements a simple file in memory. It's not an exact simulation of a real file: it assumes that no seeking is done, and that all reads are line-at-a-time. It also (intentionally) allow reading from files opened in mode 'r', and allows writing for files opened in mode 'w'.

**Version:** \$Revision: 1.4 \$

### 31.1 Functions

<b>open</b> ( <i>name</i> , <i>mode</i> )
---

### 31.2 Variables

Name	Description
<code>--package--</code>	<b>Value:</b> 'gmisclib'

### 31.3 Class file

StringIO.StringIO —  
gmisclib.fake\_file.file

#### 31.3.1 Methods

<b>__init__</b> ( <i>self</i> , <i>mode</i> ='rw', <i>name</i> =None) Overrides: StringIO.StringIO.__init__
--

**Inherited from StringIO.StringIO**

`__iter__()`, `close()`, `flush()`, `getvalue()`, `isatty()`, `next()`, `read()`, `readline()`, `readlines()`, `seek()`, `tell()`, `truncate()`, `write()`, `writelines()`

## 32 Module *gmisclib.fiat\_merge*

### 32.1 Functions

`merge(d1, d2, key1, key2, grab, unique=True)`

### 32.2 Variables

Name	Description
DEBUG	Value: 0
__package__	Value: 'gmisclib'

### 33 Module *gmisclib.fiatio*

Fiatio reads and writes an extension of the FIAT file format originally defined by David Wittman (UC Davis). This reads and writes the FIAT 1.2 format, defined at <http://kochanski.org/gpk/paper> (FIAT 1.0 is defined at <http://dls.physics.ucdavis.edu/fiat/fiat.html>.)

Nice FIAT features:

- header information looks like a comment to most programs, so they will treat a FIAT file as simple multi-column ASCII.
- since it has column names in the header, you can add columns at will, and your existing scripts will continue to run.
- Simple to parse.
- Easy to generate.

This describes fiat 1.2 format, which is nearly 100% upwards compatible with fiat 1.0 format. It is defined as follows:

1. Lines are separated by newlines.
2. All values are encoded by replacing newline and other difficult characters by a percent character (%) followed by a hex code on writing, and the reverse on reading. (There are also some more human-friendly codes which can be used, instead of pure hex: see `g_encode_specials` for their definitions. Notably, %S is space, %L is newline, %R is carriage return, %t is tab, and %T is percent.)
3. At the top of the file, you have a line identifying the format: "# fiat 1.2" (regexp: "# fiat 1\.[0-9.]+").
4. Then, you typically have a number of header lines beginning with "#". Header lines are in the form `# attribute = value` (where white space is optional and can be a mixture of spaces and tabs). The attribute must match the regular expression `[a-zA-Z_][a-zA-Z_0-9]*`. The value is whatever follows the equals sign, after leading and following white space is stripped. If the value begins and ends with the same quote character, either ' or ", the quotes are also stripped off. Values may contain any character except newline and the chosen quote.
  - Note that you must quote or encode a value if it begins or ends with whitespace.
  - Note also that header lines can also appear further down in the file; they are not restricted to the top.
  - Any other header lines are just treated as comments and ignored.
1. There may be header lines of the form "# TTYPE1 = name" or "#ttype4 = name" which name the columns of the data (the leftmost column is TTYPE1). If you don't name the Ith column, its name will be I. When writing, this module adds an attribute `COL_SEPARATOR` which contains the numeric code(s) (ASCII) of the column separator character(s). This defaults to 9, ASCII tab. The module also adds a `COL_EMPTY` attribute with the string used to mark an blank (nonexistent) item. (This defaults to

`%na`.) Note that nonexistent is not the same as a zero-length string.

- These lines may also appear anywhere in the file. They take effect immediately.
- All attributes and names are optional.

- Typically, the header is followed by multicolumn ASCII data. Columns are separated (by default) with any white space, but if there is a `COL_SEPARATOR` attribute, it is used instead. Empty entries for columns should be indicated by whatever code is specified in `COL_EMPTY`, if that is set. Otherwise, if `COL_SEPARATOR` is set, `COL_SEPARATOR` strings separate items, some of which may simply be empty. (In all cases, a completely blank line is treated as a datum which has all columns blank (nonexistent).)

If there is no `DATE` attribute, the write routine adds one, using the current date and time, in the form `ccyy-mm-ddThh:mm:ss` (as defined in the NASA FITS format). Note that all attributes are optional.

This is not quite David Wittman FIAT (1.0), which forces value to either be quoted or to contain no white space. Dwittman FIAT will take a line in the form `"#a=b c"`, and interpret `c` as a comment, whereas fiat 1.2 will interpret the value as `"b c"`. However, almost all files will be interpreted the same way as Fiat 1.0.

Here's an example:

```
# fiat 1.2
# TTYPE1 = b
# TTYPE2 = a
# SAMPRATE = 2.3
# DATE = 2001-09-21T21:32:32
# COL_EMPTY = "%na"
# COL_SEPARATOR = "9"
# Comment1
# Comment2
# b      a
2        1
3        2
3        %na
%na      3
%na      %na
0        1
```

### 33.1 Functions

<code>col_order(<i>a</i>, <i>b</i>)</code>
--

```
write_array(fd, adata, columns=None, comments=None, hdr=None,
sep='\t', numeric=False)
```

Write a rectangular array in FIAT format. Adata is a 2-D numpy array or a sequence of sequences.

```
write(fd, ldata, comments=None, hdr=None, sep='\t', blank='%na',
fixed_order=0)
```

Write a file in FIAT format. Ldata is a list of dictionaries. Each dictionary corresponds to one line in the file. Each unique key generates a column, and the values are printed in the data section of the FIAT file. Note that the TTYPE header lines will be automatically generated from ldata. Hdr is a dictionary of information that will be put in the header. Comments is a list of comment lines for the header. Fd is a file descriptor where the data should be written. Sep is a string used to separate data columns. Blank is a string to use when a data value is missing.

```
shared_data_values(data_items)
```

Takes a list of data and pulls out all the items that have the same value in each line. The idea is that you can then put them into the header via:

```
hdr, data, c = read(fd) htmp, data = shared_data_values(data)
hdr.update(htmp)
```

#### Parameters

`data_items:` (*type=list(dict(str: anything))*)

#### Return Value

It returns a tuple of (1) a dictionary of header items, and (2) a list of data. The list has the same length as `data_items`, but the dictionaries within it may have fewer entries.

(*type=tuple(dict(str: anything),  
list(dict(str: anything)))*)

**read(*fd*)**

Read a fiat format data file. Each line in the FIAT file is represented by a dictionary that maps column name into the data (data is a string). Lines without data in a certain column will not have the corresponding entry in the dictionary for that line.

You can use this function as follows:

```
hdr, data, comments = read(fd)
for datum in data:
    print datum['X']
```

**Parameters**

**fd**: The data source: typically a file descriptor.

*(type=An iterator that generates strings. Typically a **file** object.)*

**Return Value**

Three items: header, data, and comments. Header is the collected dictionary of header information data is a list of dictionaries, one for each line in the file, and comments is a list of strings.

*(type=tuple(dict, list(dict(str:str)), list(str)))*

**read\_merged(*fd*)**

Read in a fiat file and return a list of dictionaries, one for each line in the file. (Also a list of comment lines.) Each line in the input FIAT file is represented by a dictionary that maps column name into the data (data is a string). The header data in the FIAT file is merged into the per-column data, so that the header data is used as a default value for the column of the same name. As a result, all the information in the file (both header and data) is in the resulting list of dictionaries.

NB: this is a bit of a specialized routine. Normally, one uses **read**.

E.g. if there is a header line "`# X = Y`" and no data column called "`X`", then this will succeed:

```
data, comments = read_merged(fd):
for datum in data:
    assert datum['X']=='Y'
```

That this routine does not require header lines to precede data lines. If header lines appear in the middle, then a new column will be created from that point onwards.

**Parameters**

**fd**: The data source: typically a file descriptor.

*(type=An iterator that generates strings. Typically a **file** object.)*

**Return Value**

(data, comments)

*(type=(list(dict), list(str)))*



**readiter(*fd*)**

Read in a fiat file. Each line in the file is represented by a dictionary that maps column name into the data (data is a string). Lines without data in a certain column will not have an entry in that line's dictionary for that column name.

Lines beginning with '#' are either header or comment lines. A fiat file can mix header lines amongst the data lines. (Although, typically, all the header info is at the top.)

You can use this function as follows:

```
for (hdr, datum, comments) in readiter(fd):
    print datum['X']
```

NB: this is a bit of a specialized routine. Normally, one uses `read`.

**Parameters**

**fd**: The data source: typically a *file*. Not a filename.

*(type=Anything that supports iteration. Typically a file object.)*

**Return Value**

Three items: header, data, and comments. Header is the collected dictionary of header information since the last iteration, data is a dictionary of the data on the current line, and comments is a list of comment string seen so far. The end of the file yields None for the last data, along with any header info or comments after the last datum.

*(type=(dict(str:str), dict(str:str), list(str)))*

**read\_as\_float\_array(*fd*, loose=False, baddata=None)**

Read in a fiat file. Return (header, data, comments), where header is a dictionary of header information, data is a numpy array, and comments is a list of strings. Two special entries are added to header: `__COLUMNS` points to a mapping from column numbers (the order in which they appeared in the file, left to right, starting with 0) to names, and `__NAME_TO_COL` holds the reverse mapping.

Empty values are set to NaN.

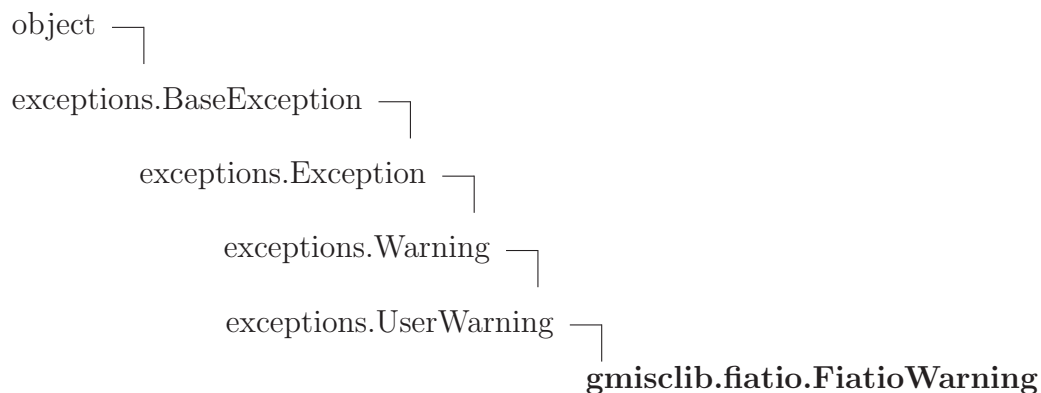
If `loose==False`, then all entries must be either floating point numbers or empty entries or equal to `baddata` (as a string, before conversion to a float). If `loose==True`, all non-floats will simply be masked out.

`test1()``test2()``test3()``test4()``test()`

### 33.2 Variables

Name	Description
<code>TABLEN</code>	<b>Value:</b> 8
<code>__package__</code>	<b>Value:</b> 'gmisclib'

### 33.3 Class *FiatioWarning*



#### 33.3.1 Methods

```

__init__(self, *s)

x.__init__(...) initializes x; see help(type(x)) for signature
Overrides: object.__init__ extit(inherited documentation)

```

*Inherited from `exceptions.UserWarning`*

`__new__()`

***Inherited from exceptions.BaseException***

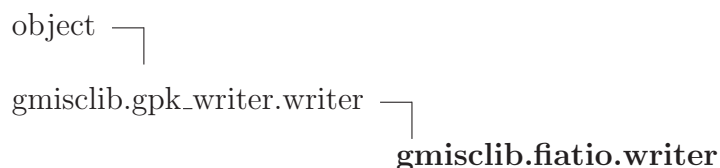
`__delattr__()`, `__getattr__()`, `__getitem__()`, `__getslice__()`, `__reduce__()`, `__repr__()`,  
`__setattr__()`, `__setstate__()`, `__str__()`, `__unicode__()`

***Inherited from object***

`__format__()`, `__hash__()`, `__reduce_ex__()`, `__sizeof__()`, `__subclasshook__()`

**33.3.2 Properties**

Name	Description
<i>Inherited from exceptions.BaseException</i>	
args, message	
<i>Inherited from object</i>	
__class__	

**33.4 Class writer**

**Known Subclasses:** `gmisclib.fatio.merged_writer`

Write a file in FIAT format. This class represents an open file, and you call member functions to write data into the file. This automatically generates much of the header information.

Column names are set from the keys passed in the `datum()` method. Each unique key generates a column, and the values are printed in the data section of the FIAT file. The TTYPE header lines will also be automatically generated.

**33.4.1 Methods**

<b><code>comment(self, comment)</code></b>
Add a comment to the data file.
<b>Parameters</b>
<b>comment:</b> the comment
<i>(type=str)</i>
Overrides: <code>gmisclib.gpk_writer.writer.comment</code>

---

**header**(*self*, *k*, *v*)
 

---

Add a single **key=value** line to the header of the data file.

**Parameters**

**k**: key

(*type=**str*)

**v**: value

(*type=**str*)

Overrides: gmisclib.gpk\_writer.writer.header

---

**\_\_init\_\_**(*self*, *fd*, *sep*='\\t', *blank*='%na')
 

---

x.\_\_init\_\_(...) initializes x; see help(type(x)) for signature

**Parameters**

**fd**: where to write the data

(*type=**file*)

**sep**: what separates columns?

(*type=**str*)

**blank**: what marks a spot where there isn't data?

(*type=**str*)

Overrides: object.\_\_init\_\_

---

**add\_cols**(*self*, *colnames*)
 

---



---

**datum**(*self*, *data\_item*)
 

---

Write a line into a fiat file. They column names will be set from the keys.

**Parameters**

**data\_item**: a dictionary of **key=value** pairs.

(*type=**dict(str: anything)*)

Overrides: gmisclib.gpk\_writer.writer.datum

---

**datavec**(*self*, *vector*, *numeric*=False)
 

---

This assumes that you've already called **add\_cols()** to set the column names. It is an error to have a vector whose length doesn't match the number of column names.

---

**append**(*self*, *d*)
 

---

<b>close</b> ( <i>self</i> )
------------------------------

<b>comments</b> ( <i>self</i> , <i>comments</i> )
---

Add comments to the data file. Comments can appear anywhere.
--

<b>data</b> ( <i>self</i> , <i>dataset</i> )
--

Write a series of lines to the output file.
---

<b>extend</b> ( <i>self</i> , <i>d</i> )
--

<b>flush</b> ( <i>self</i> )
------------------------------

<b>headers</b> ( <i>self</i> , <i>h</i> )
---

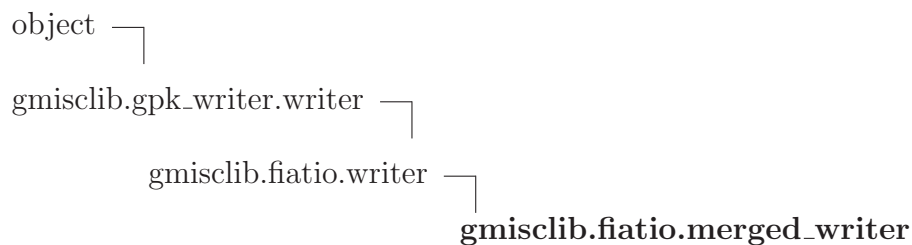
### *Inherited from object*

`--delattr--()`, `--format--()`, `--getattr--()`, `--hash--()`, `--new--()`, `--reduce--()`, `--reduce_ex--()`,  
`--repr--()`, `--setattr--()`, `--sizeof--()`, `--str--()`, `--subclasshook--()`

### 33.4.2 Properties

Name	Description
<i>Inherited from object</i>	
<code>--class--</code>	

## 33.5 Class `merged_writer`



Assumes that the data will be read with `read_merged()`, so that header values will supply default values for each column.

**33.5.1 Methods**

**`__init__(self, fd, sep='\t', blank='%na')`**

`x.__init__(...)` initializes `x`; see `help(type(x))` for signature

**Parameters**

**`fd`:** where to write the data

**`sep`:** what separates columns?

**`blank`:** what marks a spot where there isn't data?

Overrides: `object.__init__` `exitit` (inherited documentation)

**`header(self, k, v)`**

Add a single **key=value** line to the header of the data file.

**Parameters**

**`k`:** key

**`v`:** value

Overrides: `gmisclib.gpk_writer.writer.header`

**`datum(self, data_item)`**

Assumes that the data will be read with `read_merged()`, so that header values will supply default values for each column. This writes a line in the fiat file, but first it deletes any values that already exist as a header item of the same name.

**Parameters**

**`data_item`:** a dictionary of **key=value** pairs.

Overrides: `gmisclib.gpk_writer.writer.datum`

**`add_cols(self, colnames)`**

**`append(self, d)`**

**`close(self)`**

**comment**(*self*, *comment*)

Add a comment to the data file.

**Parameters**

**comment**: the comment  
(*type=**str*)

Overrides: *gmisclib.gpk\_writer.writer.comment*

**comments**(*self*, *comments*)

Add comments to the data file. Comments can appear anywhere.

**data**(*self*, *dataset*)

Write a series of lines to the output file.

**datavec**(*self*, *vector*, *numeric=False*)

This assumes that you've already called *add\_cols()* to set the column names. It is an error to have a vector whose length doesn't match the number of column names.

**extend**(*self*, *d*)

**flush**(*self*)

**headers**(*self*, *h*)

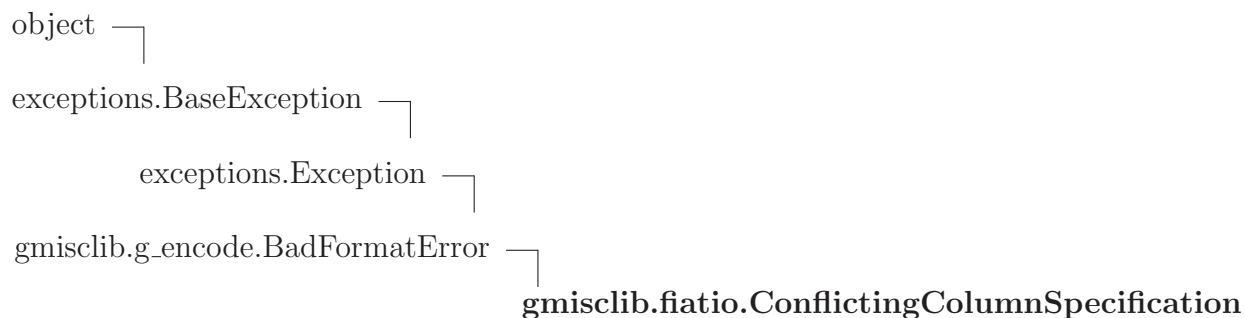
***Inherited from object***

*\_\_delattr\_\_()*, *\_\_format\_\_()*, *\_\_getattr\_\_()*, *\_\_hash\_\_()*, *\_\_new\_\_()*, *\_\_reduce\_\_()*, *\_\_reduce\_ex\_\_()*, *\_\_repr\_\_()*, *\_\_setattr\_\_()*, *\_\_sizeof\_\_()*, *\_\_str\_\_()*, *\_\_subclasshook\_\_()*

**33.5.2 Properties**

Name	Description
<i>Inherited from object</i>	
<i>__class__</i>	

### 33.6 Class *ConflictingColumnSpecification*



#### 33.6.1 Methods

<b><code>--init--(self, s)</code></b> <code>x.--init--(...)</code> initializes x; see <code>help(type(x))</code> for signature Overrides: <code>object.--init--</code> extit(inherited documentation)
---

#### *Inherited from exceptions.Exception*

`--new--()`

#### *Inherited from exceptions.BaseException*

`--delattr--()`, `--getattr__()`, `--getitem--()`, `--getslice--()`, `--reduce--()`, `--repr--()`,  
`--setattr--()`, `--setstate--()`, `--str--()`, `--unicode--()`

#### *Inherited from object*

`--format--()`, `--hash--()`, `--reduce_ex--()`, `--sizeof--()`, `--subclasshook--()`

#### 33.6.2 Properties

Name	Description
<i>Inherited from exceptions.BaseException</i>	
args, message	
<i>Inherited from object</i>	
<code>--class--</code>	



## 34 Module *gmisclib.find\_home*

This lets you find files with just a path name relative to where the program's executable code script sits.

**Version:** \$Revision: 1.6 \$

### 34.1 Functions

<b>executable</b> ( <i>s</i> , <i>general</i> =True)
--

Find an executable program. It searches first in the directory of the currently executing python script. Optionally, it then looks at PATH. If <i>general</i> =False, the function should fail unless the executable is in the same directory as the currently executing python script.
---

<b>module</b> ( <i>s</i> , <i>general</i> =True)
--

Find a module. Returns pathname.
----------------------------------

<b>os_prgm</b> ( <i>nm</i> , <i>general</i> =1)
---

Find an operating-system dependent executable program.
--

<b>data</b> ( <i>s</i> )
--------------------------

Find a data file.
-------------------

<b>directory</b> ( <i>s</i> )
-------------------------------

Find a directory.
-------------------

### 34.2 Variables

Name	Description
<code>--package--</code>	<b>Value:</b> 'gmisclib'

## 35 Module *gmisclib.find\_ngram*

This module lets you search through label files to find particular ngrams.

**Version:** \$Revision: 1.7 \$

### 35.1 Functions

<b>matches</b> ( <i>data</i> , <i>pattern</i> )
---

<b>find_mark</b> ( <i>ngram</i> , <i>fname</i> , <i>datatype</i> =1)
--

*ngram* = list of labels. *fname* = file name for *xwaves\_mark* datafile. *datatype* = *xwaves\_mark*.PHONE or *xwaves\_mark*.WORD

This function returns all instances (even overlapping instances) of the specified N-gram in the file. The return format is [ (end\_time, label), ... ], where the zeroth entry in the list is the symbol before the start of the N-gram. It's end time is the beginning of the N-gram.

In the argument list, the N-gram is an array of labels; the labels need to match the file's labels exactly.

<b>find_lab</b> ( <i>ngram</i> , <i>fname</i> , <i>loose</i> =0)
--

*ngram* = list of labels. *fname* = file name for *xwaves\_lab* datafile.

This function returns all instances (even overlapping instances) of the specified N-gram in the file. The return format is [ (end\_time, label), ... ], where the zeroth entry in the list is the symbol before the start of the N-gram. It's end time is the beginning of the N-gram.

In the argument list, the N-gram is an array of labels; the labels need to match the file's labels exactly.

**find**(*ngram*, *data*)

*ngram* = list of labels. *data* = list of (time, label, ...) as produced by *xwaves\_lab.py* or similar.

This function returns all instances (even overlapping instances) of the specified N-gram in the file. The return format is [ [ label, ...], ... ]. It is a list of n-grams, and each n-gram is a list of entities, and each entities is a tuple which marks when it ends, what it is specifically, and perhaps other things.

In the argument list, the N-gram is an array of labels; the labels need to match the file's labels exactly.

## 35.2 Variables

Name	Description
WildAnyOne	<b>Value:</b> <code>re.compile(r'.')</code>
<code>--package--</code>	<b>Value:</b> <code>'gmisclib'</code>

## 36 Module `gmisclib.findleak`

**Author:** Samual M. Rushing

### 36.1 Functions

```
get_refcounts()
```

```
print_top_N( $N=100$ , prefix='#TOPref')
```

```
readvm()
```

```
alloc1k()
```

```
vm(comment='')
```

```
lookvmstat(fname)
```

### 36.2 Variables

Name	Description
<code>--package--</code>	<b>Value:</b> <code>'gmisclib'</code>

## 37 Module *gmisclib.ftest*

### 37.1 Functions

**fprob**(*dfnum*, *dfden*, *F*)

Returns the (1-tailed) significance level (p-value) of an F statistic given the degrees of freedom for the numerator (dfR-dfF) and the degrees of freedom for the denominator (dfF).

Usage: `lfprob(dfnum, dfden, F)` where usually `dfnum=dfbn`, `dfden=dfwn`

**betai**(*a*, *b*, *x*)

Returns the incomplete beta function:

$I\text{-sub-}x(a,b) = 1/B(a,b) * (\text{Integral}(0,x) \text{ of } t^{(a-1)}(1-t)^{(b-1)} dt)$

where  $a,b > 0$  and  $B(a,b) = G(a)*G(b)/(G(a+b))$  where  $G(a)$  is the gamma function of  $a$ . The continued fraction formulation is implemented here, using the `betacf` function. (Adapted from: Numerical Recipies in C.)

Usage: `lbetai(a,b,x)`

**betacf**(*a*, *b*, *x*)

This function evaluates the continued fraction form of the incomplete Beta function, `betai`. (Adapted from: Numerical Recipies in C.)

Usage: `lbetacf(a,b,x)`

**gammln**(*xx*)

Returns the gamma function of *xx*.  $\Gamma(z) = \text{Integral}(0,\text{infinity}) \text{ of } t^{(z-1)}\exp(-t) dt$ . (Adapted from: Numerical Recipies in C.)

**Parameters**

**xx**: float

**Return Value**

float

**agammln**(*xx*)

Returns the gamma function of *xx*.  $\text{Gamma}(z) = \text{Integral}(0, \text{infinity}) \text{ of } t^{(z-1)} \exp(-t) dt$ . Adapted from: Numerical Recipies in C. Can handle multiple dims ... but probably doesn't normally have to.

Usage: `agammln(xx)`

**37.2 Variables**

Name	Description
--package--	<b>Value:</b> 'gmisclib'

## 38 Module gmisclib.fuzzygraph

### 38.1 Functions

<code>addcurve(<i>xyee</i>, <i>x_range</i>, <i>y_range</i>, <i>bexp</i>, <i>brms</i>, <i>imopfd</i>, <i>xsx</i>, <i>ysz</i>)</code>
---

### 38.2 Variables

Name	Description
IB	Value: 0.7
__package__	Value: 'gmisclib'

## 39 Module *gmisclib.g2\_select*

### 39.1 Functions

**filterlist**(*s*, *d*, *verbose*=False)

This filters a list of dictionaries, passing ones where the *selector* returns true.

**Parameters**

**s**: the selector – a little snippet of python

(*type*=*str*)

**d**: the list of dictionaries. Each dictionary is temporarily used as the local variable space and the selector is evaluated.

(*type*= *list(dict(str:value))* )

**Return Value**

a subset of the input list, whichever dictionaries cause **s** to return True when evaluated.

(*type*= *list(dict(str: value))* )

**filter\_iter**(*s*, *d*, *verbose*=False)

This filters a list of lines, passing ones where the selector returns true. *S* is the selector – a little snippet of python, and *d* is the list of data : a list of {k:v} mappings.

**accept**(*s*, *d*)

This checks a single dictionary, and returns the result of the selector. Errors in the evaluation are trapped and cause *accept()* to return False. *S* is the selector – a little snippet of python, and *d* is the data : a {k:v} mapping.

**whynot**(*s*, *d*)

Returns an explanation of why *d* was not accepted, given *s*, or None if *d* was accepted.

**why**(*s*, *d*)

Returns an explanation of why *d* was accepted, given *s*, or None if *d* was not accepted.



**evaluate**(*s*, *d*)

This checks a single dictionary, and returns the result of the selector. S is the selector – a little snippet of python, and d is the data : a {k:v} mapping.

**test**()

## 39.2 Variables

Name	Description
CCSZ	<b>Value:</b> 100
__package__	<b>Value:</b> 'gmisclib'

## 39.3 Class selector\_c

object └─ gmisclib.g2\_select.selector\_c

### 39.3.1 Methods

**\_\_init\_\_**(*self*, *code*, *global\_values*=None)

Code = a string containing python code. Global\_values = a dictionary containing values to be used by that python code. Note that the dictionary is not copied, so that it is shared and changes will be noticed.

Overrides: object.\_\_init\_\_

**set\_code**(*self*, *code*)

**set\_trap**(*self*, *exc*, *result*)

This adds to a list of exceptions. If any of those exceptions happen later, when you run **self.eval()**, then they will be caught and turned into the specified result.

For instance, if you call **set\_trap(ValueError, False)**, and when you execute **self.cc** in **self.eval** a **ValueError** is raised, then **self.eval()** will return **False**. You have mapped a **ValueError** exception into a returned value of **False**.

<code>eval(self, locals)</code>
---------------------------------

<code>globals(self, s)</code>
-------------------------------

***Inherited from object***

`__delattr__()`, `__format__()`, `__getattr__()`, `__hash__()`, `__new__()`, `__reduce__()`, `__reduce_ex__()`,  
`__repr__()`, `__setattr__()`, `__sizeof__()`, `__str__()`, `__subclasshook__()`

**39.3.2 Properties**

Name	Description
<i>Inherited from object</i>	
<code>__class__</code>	

## 40 Module gmisclib.g\_closure

This module defines closures. Closures are functions that have been supplied with some arguments, but not all of them.

### 40.1 Variables

Name	Description
<code>__package__</code>	<b>Value:</b> None

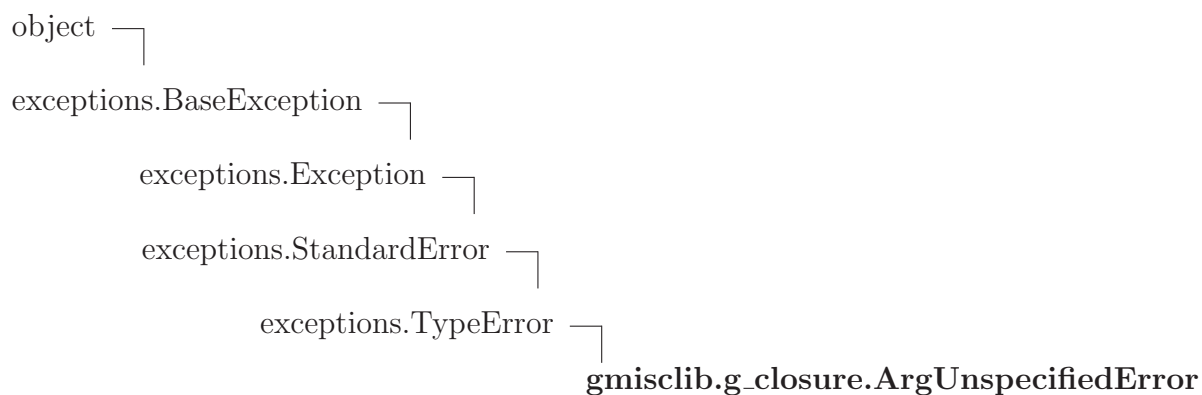
### 40.2 Class NotYet

Just a marker, indicating that a certain argument is not yet present, and the actual call should be deferred until later.

#### 40.2.1 Methods

<code>__repr__(self)</code>
-----------------------------

### 40.3 Class ArgUnspecifiedError



### 40.3.1 Methods

**`--init--`**(*self*, *s*)

*x*.`--init--`(...) initializes *x*; see `help(type(x))` for signature

Overrides: `object.--init--` `exitit`(inherited documentation)

*Inherited from exceptions.TypeError*

`--new--`()

*Inherited from exceptions.BaseException*

`--delattr--`(), `--getattr--`(), `--getitem--`(), `--getslice--`(), `--reduce--`(), `--repr--`(),  
`--setattr--`(), `--setstate--`(), `--str--`(), `--unicode--`()

*Inherited from object*

`--format--`(), `--hash--`(), `--reduce_ex--`(), `--sizeof--`(), `--subclasshook--`()

### 40.3.2 Properties

Name	Description
<i>Inherited from exceptions.BaseException</i>	
<code>args</code> , <code>message</code>	
<i>Inherited from object</i>	
<code>--class--</code>	

## 40.4 Class Closure

### 40.4.1 Methods

**`--init--`**(*self*, *fcn*, \**argv*, \*\**kwarg*)

---

Create a closure by memorizing a function and its arguments. If all arguments are supplied, you can get the result by calling the closure without arguments. Missing arguments are replaced by instances of class `NotYet`; such arguments will need to be supplied when the closure is called.

As an additional feature, you can supply a list of users of the function's value. When the value is (eventually) produced, the user function(s) will be called with the closure's return value. Users must be functions of one argument.

**`--repr--`**(*self*)

---

**`--call--`**(*self*, \**argv*, \*\**argd*)

This evaluates the closure to either produce a value or raise an exception.

---

**`partial`**(*self*, \**argv*, \*\**argd*)

This evaluates the closure to produce another (more complete) closure. Users are not preserved.

---

**`defer`**(*self*, *user*)

This will call the specified function when the closure is completed. It's a bit of a kluge.

## 41 Module *gmisclib.g\_datetime*

### 41.1 Functions

<code>ISO2datetime(<i>s</i>)</code>
-------------------------------------

<code>timedelta2float(<i>tdel</i>)</code>
---

### 41.2 Variables

Name	Description
<code>--package--</code>	Value: <code>'gmisclib'</code>

## 42 Module *gmisclib.g\_ds9*

Handles interactions with ds9 (<http://hea-www.harvard.edu/RD/ds9/>). Ds9 is an image display and analysis tool commonly used in astrophysics, but it's good for any kind of greyscale image.

Note: there is an extra, special property *Coordsys*. This is not fed to ds9. It is merely something that you can use to remember what coordinate system the regions are in.

### 42.1 Functions

**copy**(*r*, **\*\*kwargs**)

Copy a region and update some of the keyword arguments (e.g. *color*). The original object is unchanged, and you can copy any subclass of *Region*.

**dequote**(*s*)

**make\_chunks**(*s*)

**property\_parser**(*ap*)

**test\_property\_parser**()

**region\_parser**(*s*, *defprop*)

**test\_region\_parser**()

**text\_kluge**(*s*)

**test\_text\_kluge**()

**r\_list\_factory**(*slist*, *Coordsys*)

**read\_regions\_iter**(*fd*, *Coordsys*=None)

Read regions from a file. This is an iterator. *Fd* is a file descriptor.

**read\_regions**(*fd*, *Coordsys*=None)

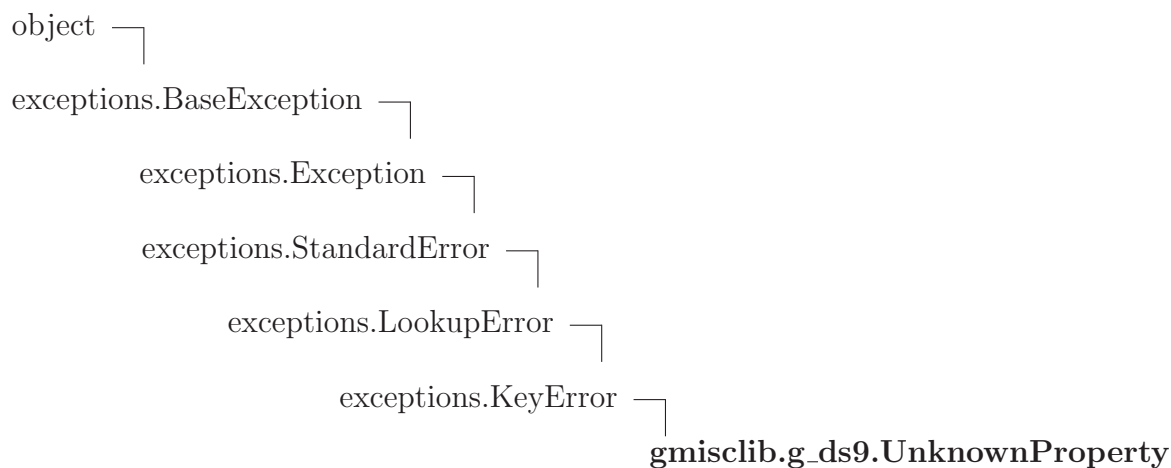
Read regions from a file. *Fd* is a file descriptor.

`test_io()``test_tags()``test_plot()`

## 42.2 Variables

Name	Description
<code>COLORS</code>	<b>Value:</b> ('white', 'black', 'red', 'green', 'blue', 'cyan', 'magen...
<code>PLOTSYMS</code>	<b>Value:</b> ('circle', 'diamond', 'plus', 'cross')
<code>PROPin</code>	<b>Value:</b> {'color': <type 'str'>, 'delete': <type 'int'>, 'edit': <...
<code>PROPout</code>	<b>Value:</b> {'color': _kvstr, 'text': _kvquote, 'font': _kvquote, 'se...
<code>PLOTSTYLES</code>	<b>Value:</b> ('discrete', 'line', 'step', 'quadratic', 'errorbar')
<code>--package--</code>	<b>Value:</b> 'gmisclib'

## 42.3 Class `UnknownProperty`





### 42.3.1 Methods

**`__init__`**(*self*, \**s*)

`x.__init__(...)` initializes `x`; see `help(type(x))` for signature

Overrides: `object.__init__` `extit`(inherited documentation)

*Inherited from `exceptions.KeyError`*

`__new__`(), `__str__`()

*Inherited from `exceptions.BaseException`*

`__delattr__`(), `__getattr__`(), `__getitem__`(), `__getslice__`(), `__reduce__`(), `__repr__`(),  
`__setattr__`(), `__setstate__`(), `__unicode__`()

*Inherited from `object`*

`__format__`(), `__hash__`(), `__reduce_ex__`(), `__sizeof__`(), `__subclasshook__`()

### 42.3.2 Properties

Name	Description
<i>Inherited from <code>exceptions.BaseException</code></i>	
<code>args</code> , <code>message</code>	
<i>Inherited from <code>object</code></i>	
<code>__class__</code>	

## 42.4 Class `execwait`

```

object └─ gmisclib.g_ds9.execwait

```

### 42.4.1 Methods

**`__init__`**(*self*, *s*)

`x.__init__(...)` initializes `x`; see `help(type(x))` for signature

Overrides: `object.__init__` `extit`(inherited documentation)

**`wait`**(*self*)

***Inherited from object***

`__delattr__()`, `__format__()`, `__getattr__()`, `__hash__()`, `__new__()`, `__reduce__()`, `__reduce_ex__()`,  
`__repr__()`, `__setattr__()`, `__sizeof__()`, `__str__()`, `__subclasshook__()`

**42.4.2 Properties**

Name	Description
<i>Inherited from object</i>	
<code>__class__</code>	

**42.5 Class *ds9\_io***

```

object └─
          gmisclib.g_ds9.ds9_io

```

**Known Subclasses:** *gmisclib.g\_ds9.ds9*

**42.5.1 Methods**

```
__init__(self, ds9start=None, xpaname='ds9', xaset=['xaset'],
          xaset_p=['xaset', '-p'], xaget=['xaget'])
```

`x.__init__(...)` initializes `x`; see `help(type(x))` for signature

Overrides: `object.__init__` extit(inherited documentation)

```
set(self, *args)
```

```
set_with_input(self, input, *args)
```

Run the *xaset* command and feed it data on its standard input. Each element in the input list becomes one line of data to *xaset*.

**Parameters**

**input:** (*type=list or iterator*)

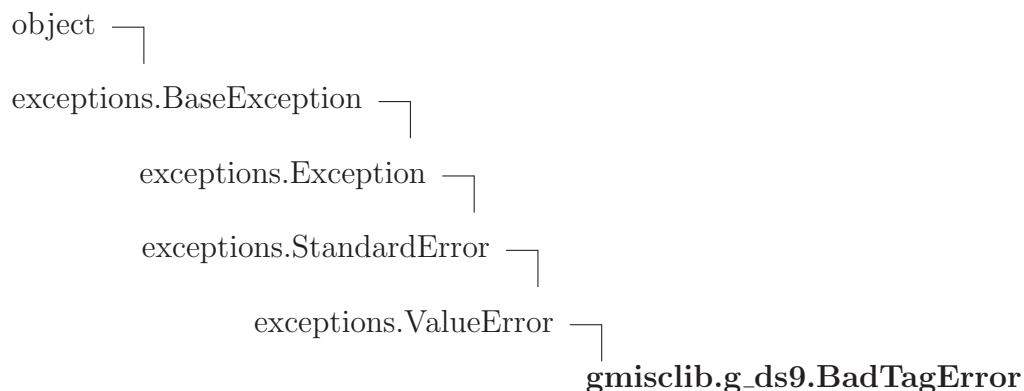
**args:** The command line used to execute *xaset*.

(*type=[ "xaset", ...]*)

```
getlast(self, *args)
```

`getall(self, *args)``getiter(self, *args)``close(self)``__del__(self)`***Inherited from object***`__delattr__()`, `__format__()`, `__getattr__()`, `__hash__()`, `__new__()`, `__reduce__()`, `__reduce_ex__()`,  
`__repr__()`, `__setattr__()`, `__sizeof__()`, `__str__()`, `__subclasshook__()`**42.5.2 Properties**

Name	Description
<i>Inherited from object</i>	
<code>__class__</code>	

**42.6 Class *BadTagError*****42.6.1 Methods**`__init__(self, *args)``x.__init__(...)` initializes `x`; see `help(type(x))` for signatureOverrides: `object.__init__` `extit`(inherited documentation)***Inherited from exceptions.ValueError***

`__new__()`

**Inherited from *exceptions.BaseException***

`__delattr__()`, `__getattr__()`, `__getitem__()`, `__getslice__()`, `__reduce__()`, `__repr__()`,  
`__setattr__()`, `__setstate__()`, `__str__()`, `__unicode__()`

**Inherited from *object***

`__format__()`, `__hash__()`, `__reduce_ex__()`, `__sizeof__()`, `__subclasshook__()`

#### 42.6.2 Properties

Name	Description
<i>Inherited from exceptions.BaseException</i>	
args, message	
<i>Inherited from object</i>	
__class__	

### 42.7 Class Region

object └─ **gmisclib.g\_ds9.Region**

**Known Subclasses:** gmisclib.g\_ds9.circle, gmisclib.g\_ds9.line, gmisclib.g\_ds9.point, gmisclib.g\_ds9.text

#### 42.7.1 Methods

**\_\_init\_\_**(*self*, *kwargs*, *coordsys*)  
*x*.\_\_init\_\_(...) initializes *x*; see help(type(*x*)) for signature  
 Overrides: object.\_\_init\_\_ extit(inherited documentation)

**\_\_str\_\_**(*self*)  
 str(*x*)  
 Overrides: object.\_\_str\_\_ extit(inherited documentation)

**addprop**(*self*, *k*, *v*)

**has\_prop**(*self*, *k*)

```
getprop(self, *args)
```

```
props(self)
```

```
chgprops(self, **kwargs)
```

```
compatible_coord(self, coordsys)
```

### *Inherited from object*

```
__delattr__(), __format__(), __getattr__(), __hash__(), __new__(), __reduce__(), __reduce_ex__(),
__repr__(), __setattr__(), __sizeof__(), __subclasshook__()
```

#### 42.7.2 Properties

Name	Description
<i>Inherited from object</i>	
__class__	

## 42.8 Class line



#### 42.8.1 Methods

```
__init__(self, x0, y0, x1, y1, Coordsys=None, **kwargs)
x.__init__(...) initializes x; see help(type(x)) for signature
Overrides: object.__init__ extit(inherited documentation)
```

```
__str__(self)
str(x)
Overrides: object.__str__ extit(inherited documentation)
```

```
end(self, whichend)
```

`center(self)``addprop(self, k, v)``chgprops(self, **kwargs)``compatible_coord(self, coordsys)``getprop(self, *args)``has_prop(self, k)``props(self)`***Inherited from object***

`__delattr__()`, `__format__()`, `__getattr__()`, `__hash__()`, `__new__()`, `__reduce__()`, `__reduce_ex__()`,  
`__repr__()`, `__setattr__()`, `__sizeof__()`, `__subclasshook__()`

**42.8.2 Properties**

Name	Description
<i>Inherited from object</i>	
<code>__class__</code>	

**42.9 Class *circle*****42.9.1 Methods**`__init__(self, x, y, r, Coordsys=None, **kwargs)`

`x.__init__(...)` initializes x; see `help(type(x))` for signature

Overrides: `object.__init__` `extit`(inherited documentation)

`__str__(self)``str(x)`Overrides: `object.__str__` `exitit`(inherited documentation)`center(self)``addprop(self, k, v)``chgprops(self, **kwargs)``compatible_coord(self, coordsys)``getprop(self, *args)``has_prop(self, k)``props(self)`***Inherited from object***`__delattr__()`, `__format__()`, `__getattr__()`, `__hash__()`, `__new__()`, `__reduce__()`, `__reduce_ex__()`, `__repr__()`, `__setattr__()`, `__sizeof__()`, `__subclasshook__()`**42.9.2 Properties**

Name	Description
<i>Inherited from object</i>	
<code>__class__</code>	

**42.10 Class point**

## 42.10.1 Methods

**`__init__(self, x, y, Coordsys=None, **kwargs)`**  
`x.__init__(...)` initializes `x`; see `help(type(x))` for signature  
 Overrides: `object.__init__` extit(inherited documentation)

**`__str__(self)`**  
`str(x)`  
 Overrides: `object.__str__` extit(inherited documentation)

**`addprop(self, k, v)`**

**`chgprops(self, **kwargs)`**

**`compatible_coord(self, coordsys)`**

**`getprop(self, *args)`**

**`has_prop(self, k)`**

**`props(self)`**

*Inherited from object*

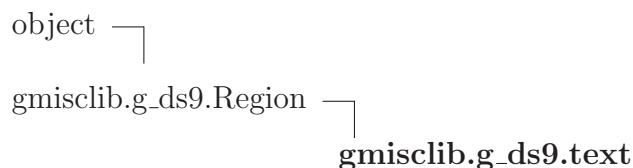
`__delattr__()`, `__format__()`, `__getattribute__()`, `__hash__()`, `__new__()`, `__reduce__()`, `__reduce_ex__()`,  
`__repr__()`, `__setattr__()`, `__sizeof__()`, `__subclasshook__()`

## 42.10.2 Properties

Name	Description
<i>Inherited from object</i>	
<code>__class__</code>	



## 42.11 Class text



### 42.11.1 Methods

```
__init__(self, x, y, text, Coordsys=None, **kwargs)
```

*x*.**\_\_init\_\_**(...) initializes *x*; see help(type(*x*)) for signature

Overrides: object.**\_\_init\_\_** extit(inherited documentation)

```
__str__(self)
```

str(*x*)

Overrides: object.**\_\_str\_\_** extit(inherited documentation)

```
addprop(self, k, v)
```

```
chgprops(self, **kwargs)
```

```
compatible_coord(self, coordsys)
```

```
getprop(self, *args)
```

```
has_prop(self, k)
```

```
props(self)
```

### *Inherited from object*

```
__delattr__(), __format__(), __getattribute__(), __hash__(), __new__(), __reduce__(), __reduce_ex__(),  
__repr__(), __setattr__(), __sizeof__(), __subclasshook__()
```

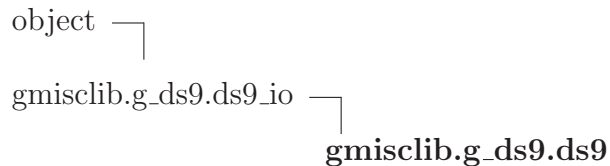
### 42.11.2 Properties

Name	Description
<i>Inherited from object</i>	

*continued on next page*

Name	Description
<code>--class--</code>	

## 42.12 Class *ds9*



### 42.12.1 Methods

**`--init--`**(*self*, *ds9start*=None)

*x*.**`--init--`**(...) initializes *x*; see `help(type(x))` for signature

Overrides: `object.--init--` `exitit`(inherited documentation)

**`load`**(*self*, *fn*)

**`get_current_fname`**(*self*)

**`get_frames`**(*self*)

**`frame_cmd`**(*self*, \**cmds*)

A catch-all command for anything that's not otherwise implemented.

**`del_frames`**(*self*, *framelist*)

Note: deleting all the frames crashes *ds9*.

**`select_frame`**(*self*, *i*)

Select a frame on the attached *ds9* image display. The frame will be created if it didn't already exist.

#### Parameters

*i*: Which frame? Apparently, any integer is OK, even negative.

(*type*=*int*)

**`get_regions_iter`**(*self*)

```
get_regions(self)
```

```
delete_all_regions(self)
```

```
delete_region_group(self, *group)
```

```
add_region(self, *region)
```

Add a list of regions to the current frame.

#### Parameters

**region:** one or more regions.

(*type*=[ **region** ], where **region** is normally an instance of **Region**, but can be anything that produces a string suitable for ds9. See

<http://hea-www.harvard.edu/RD/ds9/ref/xpa.html#regions>.)

```
region_cmd(self, *cmds)
```

A catch-all command for anything that's not otherwise implemented. This does `xpaset -p`, so you cannot use it for region definitions that need multi-line input via `set_with_input`.

```
zoom(self, z)
```

```
scale_cmd(self, *cmds)
```

A catch-all command for anything that's not otherwise implemented.

```
cmap_cmd(self, *cmds)
```

A catch-all command for anything that's not otherwise implemented.

```
set_mode(self, m)
```

```
pan_to(self, x, y, coordsys)
```

```
plot(self, name, xylist, title='', xlabel='', ylabel='', format='(x,y)',  
line='solid', color='black', symbol=None)
```

```
close_plot(self, name)
```

```
list_plots(self)
```

<code>comment(self, s)</code>
-------------------------------

<code>flush(self)</code>
--------------------------

<code>__del__(self)</code>
----------------------------

<code>close(self)</code>
--------------------------

<code>getall(self, *args)</code>
----------------------------------

<code>getiter(self, *args)</code>
-----------------------------------

<code>getlast(self, *args)</code>
-----------------------------------

<code>set(self, *args)</code>
-------------------------------

<code>set_with_input(self, input, *args)</code>
---

Run the *xpaset* command and feed it data on its standard input. Each element in the input list becomes one line of data to *xpaset*.

**Parameters**

**input:** (*type=list* or *iterator*)

**args:** The command line used to execute *xpaset*.

(*type*=[ "*xpaset*", ...])

**Inherited from object**

`__delattr__()`, `__format__()`, `__getattr__()`, `__hash__()`, `__new__()`, `__reduce__()`, `__reduce_ex__()`, `__repr__()`, `__setattr__()`, `__sizeof__()`, `__str__()`, `__subclasshook__()`

**42.12.2 Properties**

Name	Description
<i>Inherited from object</i>	
<code>__class__</code>	

**42.12.3 Class Variables**

Name	Description
<code>plotname_ok</code>	<b>Value:</b> <code>re.compile(r'[a-z][a-zA-Z]*')</code>

## 42.13 Class *ds9\_file*



This is just a convenient way to write regions to a file, instead of writing them to a ds9 session. Note that plots and other interactive things are simply no-ops.

### 42.13.1 Methods

```
__init__(self, fd)
```

*x*.**\_\_init\_\_**(...) initializes *x*; see `help(type(x))` for signature

Overrides: `object.__init__` `exitit`(inherited documentation)

```
select_frame(self, i)
```

```
add_region(self, region)
```

```
comment(self, s)
```

```
flush(self)
```

```
plot(self, name, *args, **kwargs)
```

```
list_plots(self)
```

### *Inherited from object*

`__delattr__()`, `__format__()`, `__getattr__()`, `__hash__()`, `__new__()`, `__reduce__()`, `__reduce_ex__()`, `__repr__()`, `__setattr__()`, `__sizeof__()`, `__str__()`, `__subclasshook__()`

### 42.13.2 Properties

Name	Description
<i>Inherited from object</i>	
<code>__class__</code>	

## 43 Module `gmisclib.g_encode`

This module allows strings to be encoded into a reduced subset. It is designed to work for `avio.py`, and to do a minimal mapping, so that the resulting text is human-readable. It is similar to Quoted-printable encoding, but is not specialized to e-mail limitations and is rather more flexible.

**Version:** \$Revision: 1.10 \$

### 43.1 Functions

<code>test()</code>
---------------------

### 43.2 Variables

Name	Description
<code>--package--</code>	<b>Value:</b> <code>'gmisclib'</code>

### 43.3 Class `BadFormatError`



**Known Subclasses:** `gmisclib.fiatio.ConflictingColumnSpecification`

#### 43.3.1 Methods

<code>--init--(self, *x)</code> <code>x.--init--(...)</code> initializes x; see <code>help(type(x))</code> for signature Overrides: <code>object.--init--</code> extit(inherited documentation)
---

*Inherited from `exceptions.Exception`*

`--new--()`

***Inherited from exceptions.BaseException***

`__delattr__()`, `__getattr__()`, `__getitem__()`, `__getslice__()`, `__reduce__()`, `__repr__()`,  
`__setattr__()`, `__setstate__()`, `__str__()`, `__unicode__()`

***Inherited from object***

`__format__()`, `__hash__()`, `__reduce_ex__()`, `__sizeof__()`, `__subclasshook__()`

**43.3.2 Properties**

Name	Description
<i>Inherited from exceptions.BaseException</i>	
<code>args</code> , <code>message</code>	
<i>Inherited from object</i>	
<code>__class__</code>	

**43.4 Class encoder****43.4.1 Methods**

**`__init__(self, allowed=None, notallowed=None, regex=None, eschar='%')`**

Note that there are some twiddly points in defining encoders – the `notallowed` and `allowed` arguments need to be thought through carefully, as they are passed into the `re` module as part of a regular expression. Certain characters may give surprising results.

**`back(self, x)`**

Converts back from a string containing `%xx` escape sequences to an unencoded string.

**`fwd(self, x)`**

Escapes a string so it is suitable for `a=v;` form. Nonprinting characters, along with `[:#]` are converted to `%xx` escapes (hexadecimal). Non-strings will be converted to strings with `repr()`, and can be fed back into the python interpreter.

## 44 Module *gmisclib.g\_entropy*

This returns the entropy of a probability distribution that produced a given sample.

### 44.1 Functions

<b>entropy_probs(<i>p</i>)</b>
--------------------------------

Entropy of a probability distribution.
--

<b>entropy_vec(<i>p</i>, <i>N</i>=None, <i>F</i>=1.0, <i>Clip</i>=0.01)</b>
---

Entropy of a frequency distribution <i>p</i> . Here, we assume that <i>p</i> is counts derived from multinomial distributed data; they are not normalized to one.
---

### 44.2 Variables

Name	Description
<code>--package--</code>	<b>Value:</b> 'gmisclib'



## 45 Module *gmisclib.g\_exec*

Run a Linux command and capture the result. This module is thread-safe.

Other than the \*\_raw functions, these functions ignore comments lines (e.g. ones starting with '#'), and if they see a line beginning with an error prefix, they raise an **ExecError** exception. The default error prefix is **ERR**, but you can set it for each call. Likewise, the default comment prefix is **COMMENT**, but you can also set it with the **comment** argument of any call.

The basic idea is that

- you start a subprocess.
- you feed in a bunch of strings (one at a time) to its standard input. (These come from the **input** argument. You need to know, in advance, how much data to feed it.) Note that newlines are not added to the **input** entries, so if you want the strings to be lines, they ought to end in a newline.
- then loop over this:
  - if **perline** is not None, it feeds the subprocess's standard input a string, taken from **perline**. (Note that newlines are not added; you need to include them if you want them, and you probably want them.)
  - it produces an output line
  - repeat
  - keep reading until the subprocess closes its standard output.

When **perline** is empty, the subprocess's standard input is closed. These functions continue reading the subprocess's output as long as it keeps producing.

So, you can do this:

```
get(None, ['pwd']) # "/home/gpk/whatever"
getall(None, ['ls']) # Yields a listing of your directory
get(None, ['sum'], input=['wurgle']) # Same as typing "echo wurgle | sum" to bash
getall(None, ['cat'], input=['wurgle
'], perline=['biffle ']) == ['wurgle ', 'biffle ']
```

## 45.1 Functions

**getiter\_raw**(*s*, *argv*, *input*=None, *perline*=None, *debug*=False)

Read a list of lines from a subprocess.

**Parameters**

- s:** the name of the program to execute. (Or `None`, in which case, it is taken from `argv[0]`).  
(*type*=`str` or `None`)
- argv:** an array of argument to execute.  
(*type*=`list(str)`)
- input:** strings to feed to the subprocess on startup, before the first output is read. These are sent to the subprocess's standard input.  
(*type*=array, sequence or iterator, containing strings)
- perline:** is a sequence/iterator of strings to feed in, one at a time, as the subprocess is producing data. (These are fed to the subprocess's standard input, one after each output line that it produces.)  
(*type*=array, sequence or iterator, containing strings)
- debug:** Should it print some debugging information?  
(*type*=`bool`)

**Return Value**

a sequence of the lines of output that the program produced.  
(Newlines are not removed.)  
(*type*=a generator of strings.)

**Raises**

`ExecError` ZZZ

**Note:** If `input` or `perline` is badly chosen, one can produce a locked loop of pipes. (Locked loops happen when you have too much stuff waiting to be processed.)

```
getiter(s, argv, input=None, perline=None, debug=False, err='ERR: ',  
comment='#')
```

Read a list of lines from a subprocess, after dropping junk like comments. (Comment lines begin with the `comment` argument). Raises an exception if the subprocess returns an error line (the beginning of an error line is specified in the `err` argument).

### Parameters

- s:** the name of the program to execute. (Or `None`, in which case, it is taken from `argv[0]`).  
(*type=*`str` *or* `None`*)*
- argv:** an array of argument to execute.  
(*type=*`list(str)`*)*
- input:** strings to feed to the subprocess on startup, before the first output is read. These are sent to the subprocess's standard input.  
(*type=*`array, sequence or iterator, containing strings)`
- perline:** is a sequence/iterator of strings to feed in, one at a time, as the subprocess is producing data. (These are fed to the subprocess's standard input, one after each output line that it produces.)  
(*type=*`array, sequence or iterator, containing strings)`
- debug:** Should it print some debugging information?  
(*type=*`bool`*)*

### Return Value

- a sequence of the lines of output that the program produced. (Newlines are not removed.)  
(*type=a generator of strings.*)

### Raises

- `ExecError ZZZ`

**Note:** If `input` or `perline` is badly chosen, one can produce a locked loop of pipes. (Locked loops happen when you have too much stuff waiting to be processed.)

```
get(s, argv=None, input=None, perline=None, debug=False, err='ERR:',  
comment='#')
```

Read a single line from a subprocess, after dropping junk like comments.  
Raises an exception if the subprocess produces an error line (see `getiter`,  
ERR) or no output. See `getiter` for arguments.

**Return Value**

the subprocess's first output line (after dropping comment lines).  
(*type*=*str*)

**Raises**

ExecError if the subprocess produces no output, or an error line.

```
get_raw(s, argv=None, input=None, perline=False, debug=False)
```

Read a single line from a subprocess. (One normally uses this for processes  
that are always supposed to produce a single line of output.) See `getiter_raw`  
for arguments.

**Return Value**

the subprocess's first output line.  
(*type*=*str*)

**Raises**

ExecError if the subprocess produces no output.

```
getlast(s, argv=None, input=None, perline=False, debug=False,  
err='ERR:', comment='#')
```

Read the last line from a subprocess, after dropping junk like comments.  
Raises an exception if the subprocess produces an error line (see `getiter`,  
ERR) or no output. See `getiter` for arguments.

**Return Value**

the subprocess's first output line (after dropping comment lines).  
(*type*=*str*)

**Raises**

ExecError if the subprocess produces no output, or an error line.

```
getall(s, argv, input=None, perline=None, debug=False, err='ERR:',  
comment='#')
```

Read the text lines produced by a subprocess, after dropping junk like comments. Raises an exception if the process produces an error line (see `getiter`, `ERR`). See `getiter` for arguments.

**Return Value**

the subprocess's first output line (after dropping comment lines).  
(*type=list(str)*)

**Raises**

`ExecError` if the process produces an error line.

```
test()
```

## 45.2 Variables

Name	Description
<code>ERR</code>	This is the default for the prefix for error messages from the subprocess. <b>Value:</b> <code>'ERR:'</code>
<code>COMMENT</code>	This is the default for the prefix for comments from the subprocess. <b>Value:</b> <code>'#'</code>
<code>__package__</code>	<b>Value:</b> <code>'gmisclib'</code>

## 45.3 Class `ExecError`

```
object └─
exceptions.BaseException └─
    exceptions.Exception └─
        gmisclib.g_exec.ExecError
```

Something went wrong. The subprocess did not produce output when some was expected, or it generated an error line (see `ERR`, `err` argument). Perhaps, some of the input that you supplied could not be used. Also, pipe creation might have failed.

**45.3.1 Methods**

**`--init--`**(*self*, \**s*)

*x*.`--init--`(...) initializes *x*; see `help(type(x))` for signature

Overrides: `object.--init--` `exit`(inherited documentation)

***Inherited from exceptions.Exception***

`--new--`()

***Inherited from exceptions.BaseException***

`--delattr--`(), `--getattr--`(), `--getitem--`(), `--getslice--`(), `--reduce--`(), `--repr--`(),  
`--setattr--`(), `--setstate--`(), `--str--`(), `--unicode--`()

***Inherited from object***

`--format--`(), `--hash--`(), `--reduce_ex--`(), `--sizeof--`(), `--subclasshook--`()

**45.3.2 Properties**

Name	Description
<i>Inherited from exceptions.BaseException</i>	
<code>args</code> , <code>message</code>	
<i>Inherited from object</i>	
<code>--class--</code>	

## 46 Module *gmisclib.g\_implements*

This module tells you if an object's signature matches a class. It lets you know if one class can substitute for another, even if they are unrelated in an OO hierarchy.

This allows you to begin to check that a foreign object will have the necessary properties to be used in your code. It is more of a functional check than `isimplements()`.

For instance, if your code needs a `write()` method, you can check for it like this:

```
class some_template(object):
    def write(self, somearg):
        pass

#
g_implements.check(x, some_template)
#... at this point, we can be assured that a
# call to x.write(3) is possible.
# (Note that we don't know what the call will *do*,
# merely that the proper method of x exists and that
# it can take an argument.
```

Semi-kluge: if you have an instance, `i` that needs to convince `g_implements` that it can actually behave as class `C`, then set `i.__g_implements__ = ['C']`. Generally, this can be a list or set of class names. This trick is useful for C-implementations of python classes (where introspection cannot get all the necessary information) or for fancy python classes that use `__getattr__` and similar.

### 46.1 Functions

<b>Vartype</b> ( <i>mem</i> , <i>com</i> )
--

<b>Strict</b> ( <i>mem</i> , <i>com</i> )
---

<b>why</b> ( <i>instance</i> , <i>classobj</i> , <i>ignore=None</i> )
---

Explains why instance doesn't implement the classobj.

#### Parameters

**instance**: an instance of a class  
**classobj**: a class object

#### Return Value

None (if it does) or a string explanation of why it doesn't @rtype  
 None or str

**impl**(*instance*, *classobj*)

Tells you if an instance of an object implements a class. By implements, I mean that the instance supplies every member that the class supplies, and every member has the same type. The instance may have \*more\* members, of course. Functions require that the argument names must match, too at least as far as the required arguments in the classobj's function.

The match may be made looser by adding a g\_implements attribute to various class members. Possibilities for the value are Optional, Strict, Vartype, Varargs, or you can give a two-argument function, and that function will be called to decide whether the match is acceptable or not.

**make\_optional**(*x*)

This is a decorator for a function. make\_optional implies make\_varargs.

**make\_varargs**(*x*)

This is a decorator for a function.

**make\_vartype**(*x*)

This is a decorator for a function.

**make\_strict**(*x*)

This is a decorator for a function.

**check**(*instance*, *classobj*, *ignore=None*)

Require that **instance** has the attributes defined by **classobj**. This function either quietly succeeds or it raises a `GTypeError` exception.

**Parameters**

**instance**: the thing to check

*(type=any instance of a class (i.e. anything with a `--class--` attribute).)*

**classobj**: a class that provides a pattern of attributes.

*(type=any class object (i.e. not an instance, typically).)*

**Raises**

`GTypeError` **instance** doesn't provide all the features of **classobj**.

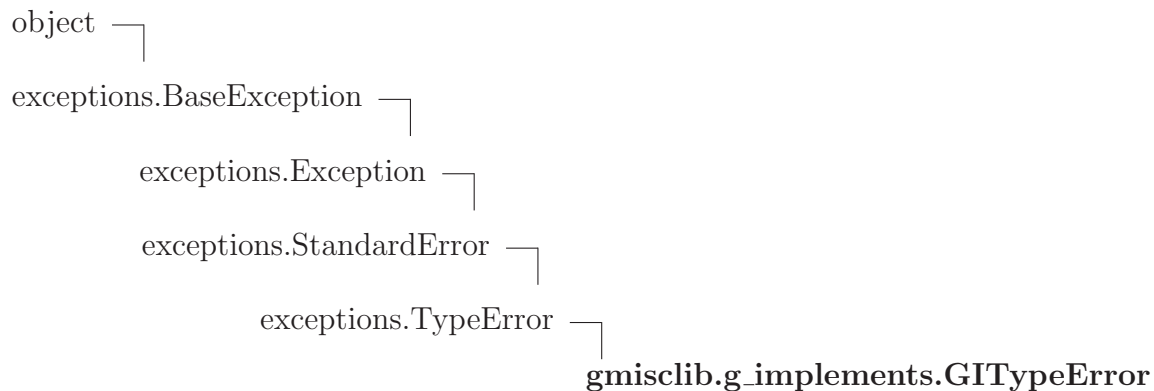
**test**()



## 46.2 Variables

Name	Description
<code>MAXCACHE</code>	<b>Value:</b> 100
<code>Optional</code>	<b>Value:</b> 'optional'
<code>Varargs</code>	<b>Value:</b> 'varargs'
<code>Ignore</code>	<b>Value:</b> set(['__class__', '__delattr__', '__dict__', '__doc__', '...'])
<code>__package__</code>	<b>Value:</b> 'gmisclib'

## 46.3 Class `GTypeError`



Exception raised by `check` to indicate failure.

### 46.3.1 Methods

```

__init__(self, s)

x.__init__(...) initializes x; see help(type(x)) for signature
Overrides: object.__init__ extit(inherited documentation)

```

*Inherited from `exceptions.TypeError`*

```
__new__()
```

*Inherited from `exceptions.BaseException`*

```

__delattr__(), __getattr__(), __getitem__(), __getslice__(), __reduce__(), __repr__(),
__setattr__(), __setstate__(), __str__(), __unicode__()

```

*Inherited from `object`*

`__format__()`, `__hash__()`, `__reduce_ex__()`, `__sizeof__()`, `__subclasshook__()`

#### 46.3.2 Properties

Name	Description
	<i>Inherited from <code>exceptions.BaseException</code></i>
	args, message
	<i>Inherited from object</i>
<code>__class__</code>	

## 47 Module `gmisclib.g_keyed_accum`

### 47.1 Functions

<code>test()</code>
---------------------

### 47.2 Variables

Name	Description
<code>--package--</code>	Value: <code>'gmisclib'</code>

### 47.3 Class `keyed_avg`

`gmisclib.g_keyed_accum.keyed_thing` — `gmisclib.g_keyed_accum.keyed_avg`

#### 47.3.1 Methods

<code>--init__(self)</code>
-----------------------------

Overrides: `gmisclib.g_keyed_accum.keyed_thing.__init__`

<code>add(self, key, value)</code>
------------------------------------

*Inherited from `gmisclib.g_keyed_accum.keyed_thing`*

`--contains--()`, `--delitem--()`, `--iter--()`, `--len--()`, `clear()`, `get()`, `has_key()`, `items()`, `iterkeys()`, `keys()`, `values()`

## 48 Module *gmisclib.g\_lin\_fit*

Generic linear best fits.

### 48.1 Functions

**fit**(*idata*, *fitting\_fcns*, *ff\_info*=None, *wt*=None)

Fits data to a linear combination of fitting\_fcns.

**Parameters**

**fitting\_fcns:** the basis functions for the linear regression. The fitting functions return vectors (which must match the data). Fitting functions are called as *f*(*info*, *ff\_info*, *len*(*vector*)). Vectors may have different lengths, as long as the corresponding data and fit vectors agree.

*(type=an array of functions.)*

**idata:** *(type=an array of (vector, info).)*

**Return Value**

The returned value is a tuple of: the coefficients of the optimal solution (as an array), the residual (a single float number), the rank of the fit (int), an array of the singular values, and the best fit to the data. The returned value follows Num.LA.linear\_least\_squares(), except that residual is always calculated and is a scalar.

*(type=tuple)*

**test**()

**test1**()

**test2**()

**test3**()

**test4**()

**test5**()

**test6**()

## 48.2 Variables

Name	Description
<code>--package--</code>	<b>Value:</b> 'gmisc.lib'

## 49 Module *gmisclib.g\_localfit*

Fit a linear transform to a bunch of tt input/output vectors.

### Notes:

- If you compute Pearson's  $R^2$  via `1-localfit()/err_before_fit()`, you get the "un-adjusted"  $R^2$  value. There is also an "adjusted"  $R^2$  value, which is  $R_a^2 = 1 - (1 - R^2) * (n - 1) / (n - p - 1)$  where  $n$  is the number of data and  $p$  is the number of adjustable parameters of the linear regression.
- When you are computing Pearson's  $r^2$  by way of `localfit()/err_before_fit()` the two function calls *MUST* have the same data and flags. Specifically, **constant** must be equal in both calls.

### 49.1 Functions

**err\_before\_fit**(*data*, *minsv*=None, *minsvr*=None, *constant*=True)

How much variation did the data have before the fit? This is used to compare with the error after the fit, to allow a F-test or ANOVA. Strictly speaking, we compute the mean-squared error about a constant.

#### Parameters

**data:** See *pack*.

(*type*=See *pack*.)

#### Return Value

Returns the per-coordinate sum-squared-error of the output coordinates.

(*type*=*numpy.ndarray*)

**pack**(*data*, *constant*=True)

Prepare the data array and (optionally) weight the data.

**Parameters**

**data:** [(input\_coordinates, output\_coordinates), ...] or [(input\_coordinates, output\_coordinates, weight), ...]. A list of (in,out) tuples corresponding to the independent and dependent parameters of the linear transform. Both **in** and **out** are one-dimensional numpy vectors. They don't need to have the same length, though (obviously) all instances of "in" need to have the same length and all instances of "out" also need to match one another. If **weight** is given, it must be a scalar.

(*type*=[(numpy.ndarray, numpy.ndarray), ...] or [(numpy.ndarray, numpy.ndarray, float), ...])

**localfit**(*data*, *minsv*=None, *minsvr*=None, *constant*=True)

Does a linear fit to data via a singular value decomposition algorithm. It returns the matrix A and vector B such that  $A \cdot \text{input\_coordinates} + B$  is the best fit to `output_coordinates`.

#### Parameters

- minsv:** sets the minimum useable singular value;  
(*type*=float or None)
- minsvr:** sets the minimum useable s.v. in terms of the largest s.v.  
(*type*=float or None)
- constant:** Do you want a constant built into the linear equations?  
(*type*=Boolean)
- data:** list of tuples. See `pack`.  
(*type*=See `pack`.)

#### Return Value

(A, B, errs, sv, rank) where

- A is the matrix of coefficients
- B is a vector of constants. (Or None) One constant per component of the `out` vector in `pack`.
- errs Each component gives the sum of squared residuals for the corresponding component of the `out` vector.
- sv are the singular values, sorted into decreasing order.

(*type*=(A, B, errs, sv, rank) where

- A is a 2-D `numpy.ndarray` matrix.
- B is a 1-D `numpy.ndarray` vector (if constant, else None).
- errs is a `numpy.ndarray` vector, one value for each output coordinate. The length is the same as the `out` vector (see `pack`).
- sv is a `numpy.ndarray` vector. The length is the same as the `in` vector (see `pack`).

)



---

**reg\_localfit**(*data*, *regstr*=0.0, *regtgt*=None, *rscaler*=None, *constant*=True)

Does a linear fit to data via a singular value decomposition algorithm. It returns the matrix A and vector B such that  $A \cdot \text{input\_coordinates} + B$  is the best fit to *output\_coordinates*.

**Parameters**

**constant:** Do you want a constant built into the linear equations?

(*type*=*Boolean*)

**data:** list of tuples. See **pack**.

(*type*=*See pack.*)

**Return Value**

(A, B, errs, sv, rank) where

- A is a 2-D `numpy.ndarray` matrix.
- B is a 1-D `numpy.ndarray` vector (if constant, else `None`).
- errs is a `numpy.ndarray` vector, one value for each output coordinate. It gives the sum of squared residuals.
- sv are the singular values, sorted into decreasing order.

---

**robust\_localfit**(*data*, *minsv*=None, *minsvr*=None, *constant*=True)

Data is [ (input\_coordinates, output\_coordinates), ... ] *minsv* sets the minimum useable singular value; *minsvr* sets the minimum useable s.v. in terms of the largest s.v.. It returns the matrix A and vector B such that the best fit is  $A \cdot \text{input\_coordinates} + B$  in a tuple (A, B, errs, rank). *errs* is a vector, one value for each output coordinate. Rank is the minimum rank of the various fits.

Warning! Not tested.

**fit\_giving\_sigmas**(*data*, *minsv*=None, *minsvr*=None, *constant*=True)

Does a linear fit to data via a singular value decomposition algorithm. It returns the matrix A and vector B such that  $A \cdot \text{input\_coordinates} + B$  is the best fit to `output_coordinates`.

**Parameters**

- minsv:** sets the minimum useable singular value;  
(*type*=float or None)
- minsvr:** sets the minimum useable s.v. in terms of the largest s.v.  
(*type*=float or None)
- constant:** Do you want a constant built into the linear equations?  
(*type*=Boolean)
- data:** [(input\_coordinates, output\_coordinates), ...]. A list of (in,out) tuples corresponding to the independent and dependent parameters of the linear transform. Both `in` and `out` are one-dimensional **numpy vectors**. They don't need to have the same length, though (obviously) all instances of "in" need to have the same length and all instances of "out" also need to match one another.  
(*type*=[(numpy.ndarray, numpy.ndarray), ...])

**Return Value**

(A, B, sigmaA, sigmaB) where

- A is a 2-D `numpy.ndarray` matrix.
- B is a 1-D `numpy.ndarray` vector (if constant, else None). )

**leaktest**()

**test0**()

**test\_localfit11**(*r*)

**test\_localfit11e**()

**test\_localfit21**()

**test\_localfit21u**()

**test\_localfit22**(*r*)

<code>test_fgs1()</code>
--------------------------

<code>test_wt()</code>
------------------------

## 49.2 Variables

Name	Description
<code>--package--</code>	<b>Value:</b> 'gmisclib'

## 50 Module gmisclib.g\_pipe

A multithreaded version of `os.popen2()`. Note that the argument list isn't quite the same.

### 50.1 Functions

**popen2**(*path, args, bufsize=0*)

Forks off a process, and returns the processes input and output file descriptors. The latter is really a pfd (defined above).

Path and args are passed directly into `os.execvp()`.

**test**()

### 50.2 Variables

Name	Description
<code>__package__</code>	<b>Value:</b> 'gmisclib'

### 50.3 Class pfd

object —  
gmisclib.g\_pipe.pfd

Pseudo-file descriptor. Just like a FD, except that it waits for the process at the end of the pipe to terminate when you close it.

#### 50.3.1 Methods

**\_\_init\_\_**(*self, p*)

`x.__init__(...)` initializes `x`; see `help(type(x))` for signature

Overrides: `object.__init__` `exitit` (inherited documentation)

**read**(*self*)

**readline**(*self*)

`readlines(self)``next(self)``__iter__(self)``flush(self)``xreadlines(self, sizehint=-1)``close(self)``__del__(self)`***Inherited from object***

`__delattr__()`, `__format__()`, `__getattr__()`, `__hash__()`, `__new__()`, `__reduce__()`, `__reduce_ex__()`,  
`__repr__()`, `__setattr__()`, `__sizeof__()`, `__str__()`, `__subclasshook__()`

**50.3.2 Properties**

Name	Description
<i>Inherited from object</i>	
<code>__class__</code>	

**50.3.3 Class Variables**

Name	Description
<code>__doc__</code>	<b>Value:</b> <code>"""Pseudo-file descriptor. Just like a FD, exc...</code>

## 51 Module *gmisclib.g\_pipe\_old*

A multithreaded version of `os.popen2()`. Note that the argument list isn't quite the same.

### 51.1 Functions

**popen2**(*path*, *args*)

Forks off a process, and returns the processes input and output file descriptors. The latter is really a *pfid* (defined above).

Path and args are passed directly into `os.execvp()`.

**test**()

### 51.2 Variables

Name	Description
<code>__package__</code>	<b>Value:</b> <code>'gmisclib'</code>

### 51.3 Class *pfid*

object —  
     *gmisclib.g\_pipe\_old.pfid*

Pseudo-file descriptor. Just like a FD, except that it waits for the process at the end of the pipe to terminate when you close it.

#### 51.3.1 Methods

**\_\_init\_\_**(*self*, *fd*, *cpid*)

`x.__init__(...)` initializes `x`; see `help(type(x))` for signature

Overrides: `object.__init__` `exitit` (inherited documentation)

**read**(*self*)

**readline**(*self*)

`readlines(self)``next(self)``__iter__(self)``flush()``xreadlines(self, sizehint=-1)``close(self)``__del__(self)`***Inherited from object***

`__delattr__()`, `__format__()`, `__getattr__()`, `__hash__()`, `__new__()`, `__reduce__()`, `__reduce_ex__()`,  
`__repr__()`, `__setattr__()`, `__sizeof__()`, `__str__()`, `__subclasshook__()`

**51.3.2 Properties**

Name	Description
<i>Inherited from object</i>	
<code>__class__</code>	

**51.3.3 Class Variables**

Name	Description
<code>__doc__</code>	<b>Value:</b> <code>"""Pseudo-file descriptor. Just like a FD, exc...</code>

## 52 Module gmisclib.g\_place\_label

### 52.1 Functions

<b>Gauss</b> ( <i>y</i> , <i>ybar</i> , <i>yvar</i> )
---

<b>gauss</b> ( <i>y</i> , <i>ybar</i> , <i>yvar</i> )
---

<b>q_gauss</b> ( <i>y</i> , <i>ybar</i> , <i>yvar</i> )
---

### 52.2 Variables

Name	Description
<code>--package--</code>	<b>Value:</b> 'gmisclib'

### 52.3 Class boxc

object —  
     **gmisclib.g\_place\_label.boxc**

**Known Subclasses:** gmisclib.g\_place\_label.viewport

#### 52.3.1 Methods

<b>--init--</b> ( <i>self</i> , <i>xc</i> , <i>yc</i> , <i>xw</i> , <i>yw</i> )
---

*x*.--init--(...) initializes *x*; see help(type(*x*)) for signature

Overrides: object.--init-- extit(inherited documentation)

<b>xmin</b> ( <i>self</i> )
-----------------------------

<b>xmax</b> ( <i>self</i> )
-----------------------------

<b>ymin</b> ( <i>self</i> )
-----------------------------

<b>ymax</b> ( <i>self</i> )
-----------------------------



`w(self)``h(self)``yc(self)``xc(self)``shift(self, x, y)`***Inherited from object***

`__delattr__()`, `__format__()`, `__getattr__()`, `__hash__()`, `__new__()`, `__reduce__()`, `__reduce_ex__()`,  
`__repr__()`, `__setattr__()`, `__sizeof__()`, `__str__()`, `__subclasshook__()`

**52.3.2 Properties**

Name	Description
<i>Inherited from object</i>	
<code>__class__</code>	

**52.4 Class `text_template`**

object └─ `gmisclib.g_place_label.text_template`

**52.4.1 Methods**`__init__(self, text, H, W)`

(0,0) is the lower left corner of the block of text.

Overrides: `object.__init__``addline(self, x0, y0, xlength, yht, txtline)`

Specify LLC.

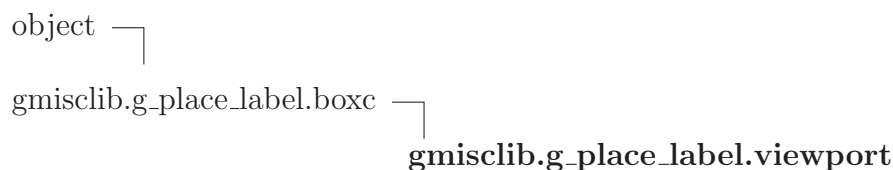
`nlines(self)`***Inherited from object***

`--delattr--()`, `--format--()`, `--getattr--()`, `--hash--()`, `--new--()`, `--reduce--()`, `--reduce_ex--()`,  
`--repr--()`, `--setattr--()`, `--sizeof--()`, `--str--()`, `--subclasshook--()`

### 52.4.2 Properties

Name	Description
<i>Inherited from object</i>	
<code>--class--</code>	

## 52.5 Class viewport



### 52.5.1 Methods

**`--init--`**(*self*, *fracCharHt*, *fracCharW*, *xmin*, *xmax*, *ymin*, *ymax*)

`x.--init--(...)` initializes x; see `help(type(x))` for signature

Overrides: `object.--init--` extit(inherited documentation)

**`dataset`**(*self*, *x*, *y*)

**`find_y`**(*self*, *x0*, *ttm*)

Figures out a good place to put a label. The label is at a specified tc, but can be at any vertical position. This finds a good vertical position. Returns the bottom of the block of text.

**`textplace`**(*self*, *x0*, *text*)

**`draw_hor_line`**(*self*, *xc*, *yc*, *w*)

**`h`**(*self*)

**`shift`**(*self*, *x*, *y*)

`w(self)``xc(self)``xmax(self)``xmin(self)``yc(self)``ymax(self)``ymin(self)`***Inherited from object***

`__delattr__()`, `__format__()`, `__getattr__()`, `__hash__()`, `__new__()`, `__reduce__()`, `__reduce_ex__()`,  
`__repr__()`, `__setattr__()`, `__sizeof__()`, `__str__()`, `__subclasshook__()`

**52.5.2 Properties**

Name	Description
<i>Inherited from object</i>	
<code>__class__</code>	

## 53 Module gmisclib.g\_pylab

This class works in concert with `bin/pylab_server.py` to allow you to display simple pylab/matplotlib graphics on another machine. It's most useful if you need to do plots from a machine where pylab isn't or cannot be installed.

## 54 Module gmisclib.g\_selector

### 54.1 Functions

```
selector_not(selector)
```

```
test()
```

### 54.2 Variables

Name	Description
<code>--package--</code>	<b>Value:</b> 'gmisclib'

### 54.3 Class selector

**Known Subclasses:** gmisclib.g\_selector.false, gmisclib.g\_selector.one\_selector, gmisclib.g\_selector.selector, gmisclib.g\_selector.true

#### 54.3.1 Methods

```
accept(self, item)
```

```
--init--(self)
```

```
and_another(self, txt)
```

```
select(self, txt)
```

```
or_another(self, txt)
```

```
And(self, s)
```

```
Or(self, s)
```

```
Not(self)
```

```
leaf(self, txt)
```

**filterlist**(*self*, *x*)

## 54.4 Class **true**

gmisclib.g\_selector.selector └─ **gmisclib.g\_selector.true**

### 54.4.1 Methods

**\_\_init\_\_**(*self*)

Overrides: gmisclib.g\_selector.selector.\_\_init\_\_

**accept**(*self*, *item*)

Overrides: gmisclib.g\_selector.selector.accept

**And**(*self*, *s*)

**Not**(*self*)

**Or**(*self*, *s*)

**and\_another**(*self*, *txt*)

**filterlist**(*self*, *x*)

**leaf**(*self*, *txt*)

**or\_another**(*self*, *txt*)

**select**(*self*, *txt*)

## 54.5 Class **false**

gmisclib.g\_selector.selector └─ **gmisclib.g\_selector.false**

### 54.5.1 Methods

**`__init__`**(*self*)Overrides: `gmisclib.g_selector.selector.__init__`**`accept`**(*self*, *item*)Overrides: `gmisclib.g_selector.selector.accept`**`And`**(*self*, *s*)**`Not`**(*self*)**`Or`**(*self*, *s*)**`and_another`**(*self*, *txt*)**`filterlist`**(*self*, *x*)**`leaf`**(*self*, *txt*)**`or_another`**(*self*, *txt*)**`select`**(*self*, *txt*)

## 54.6 Class `selector_op`

`gmisclib.g_selector.selector``gmisclib.g_selector.selector_op`

### 54.6.1 Methods

**`__init__`**(*self*, *l*, *op*, *r*)Overrides: `gmisclib.g_selector.selector.__init__`**`accept`**(*self*, *x*)Overrides: `gmisclib.g_selector.selector.accept`

`And(self, s)``Not(self)``Or(self, s)``and_another(self, txt)``filterlist(self, x)``leaf(self, txt)``or_another(self, txt)``select(self, txt)`

## 54.7 Class `one_selector`

`gmisclib.g_selector.selector` └─ `gmisclib.g_selector.one_selector`

### 54.7.1 Methods

`__init__(self, txt)`Overrides: `gmisclib.g_selector.selector.__init__``accept(self, item)`Overrides: `gmisclib.g_selector.selector.accept``And(self, s)``Not(self)``Or(self, s)``and_another(self, txt)`



```
filterlist(self, x)
```

```
leaf(self, txt)
```

```
or_another(self, txt)
```

```
select(self, txt)
```

## 55 Module `gmisclib.g_unicode`

Functions to make unicode handling easier for Python 2.X.

### 55.1 Functions

**`u(s, encoding='utf-8')`**

Convert string-like objects to unicode. The general idea is that, as soon as data comes into your program, you call `u()` on it to make sure it is unicode.

**Note:** `u()` should be idempotent: i.e. `u(u(x)) == u(x)`

**`test()`**

**`e(s)`**

Encode unicode into a bytestring, for printing. The idea is that, just before you print anything, you call `e()` on the data to be printed.

### 55.2 Variables

Name	Description
<code>--package--</code>	<b>Value:</b> <code>'gmisclib'</code>

## 56 Module *gmisclib.gpk\_getopt*

### 56.1 Functions

**parse**(*arglist*, *shortoptions*, *longoptions*)

Splits apart an command line argument list, and puts the arguments into a dictionary. The arguments are a list to be parsed, a string of short one character options, and a list of long gnu-style options. The function returns a dictionary mapping between option names and values (if any). Names in the dictionary have had leading dashes removed.

### 56.2 Variables

Name	Description
<code>__package__</code>	<b>Value:</b> <code>'gmisclib'</code>

## 57 Module *gmisclib.gpk\_hdr*

### 57.1 Functions

<code>uq(<i>s</i>, <i>qc</i>)</code>
--------------------------------------

### 57.2 Variables

Name	Description
<code>--package--</code>	<b>Value:</b> <code>'gmisclib'</code>

### 57.3 Class *hdr*

#### 57.3.1 Methods

<code>--init--(<i>self</i>, <i>fn</i>)</code>
---

<code>has_key(<i>self</i>, <i>key</i>)</code>
---

<code>items(<i>self</i>)</code>
---------------------------------

<code>get(<i>self</i>, <i>key</i>, <i>default</i>=None, <i>qc</i>=None)</code>
--

<code>set(<i>self</i>, <i>key</i>, <i>value</i>, <i>qc</i>=None)</code>
---

<code>--del--(<i>self</i>)</code>
-----------------------------------

<code>close(<i>self</i>)</code>
---------------------------------

<code>write(<i>self</i>)</code>
---------------------------------

<code>read(<i>self</i>)</code>
--------------------------------

## 58 Module `gmisclib.gpk_lapack`

### 58.1 Functions

<code>solve(<i>a</i>, <i>b</i>)</code>
--

solves $a \cdot x = b$ .
--------------------------

<code>test1()</code>
----------------------

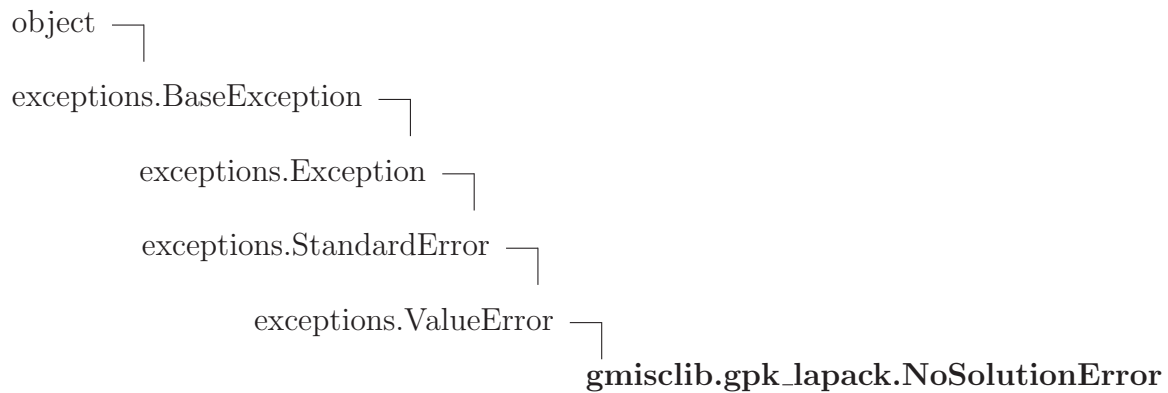
<code>err(<i>a</i>, <i>b</i>)</code>
--------------------------------------

<code>test2()</code>
----------------------

<code>testa()</code>
----------------------

<code>testbdi()</code>
------------------------

### 58.2 Class `NoSolutionError`



#### 58.2.1 Methods

<code>__init__(<i>self</i>, <i>s</i>)</code>
--

<code>x.__init__(...)</code> initializes <code>x</code> ; see <code>help(type(x))</code> for signature
--

Overrides: <code>object.__init__</code> <code>exitit</code> (inherited documentation)
---

*Inherited from `exceptions.ValueError`*

`__new__()`

***Inherited from exceptions.BaseException***

`__delattr__()`, `__getattr__()`, `__getitem__()`, `__getslice__()`, `__reduce__()`, `__repr__()`,  
`__setattr__()`, `__setstate__()`, `__str__()`, `__unicode__()`

***Inherited from object***

`__format__()`, `__hash__()`, `__reduce_ex__()`, `__sizeof__()`, `__subclasshook__()`

### 58.2.2 Properties

Name	Description
<i>Inherited from exceptions.BaseException</i>	
args, message	
<i>Inherited from object</i>	
__class__	

## 59 Module *gmisclib.gpk\_lsq*

### 59.1 Functions

<code>test_svd()</code>
-------------------------

<code>all_between(<i>a</i>, <i>vec</i>, <i>b</i>)</code>
--

<code>test0()</code>
----------------------

<code>test_vec()</code>
-------------------------

<code>test_vec2()</code>
--------------------------

<code>test_m1()</code>
------------------------

<code>test_m2()</code>
------------------------

<code>test_m2r()</code>
-------------------------

<code>test_m2rR()</code>
--------------------------

<code>test_hat()</code>
-------------------------

Make sure that the hat function meets its definition.
---

### 59.2 Variables

Name	Description
<code>--package--</code>	<b>Value:</b> <code>'gmisclib'</code>

### 59.3 Class *lls\_base*

object  `gmisclib.gpk_lsq.lls_base`

**Known Subclasses:** `gmisclib.gpk_lsq.linear_least_squares`, `gmisclib.gpk_lsq.reg_linear_least_squares`

## 59.3.1 Methods

---

**`__init__(self, a, copy=True)`**

`x.__init__(...)` initializes `x`; see `help(type(x))` for signature

Overrides: `object.__init__` `extit`(inherited documentation)

---

**`set_y(self, y, copy=True)`**


---

**`y(self, copy=True)`**


---

**`hat(self, copy=True)`**


---

**`x(self, y=None, copy=True)`**

**Raises**

`numpy.linalg.linalg.LinAlgError` when the matrix is singular.

---

**`fit(self, copy=False)`**

**Raises**

`numpy.linalg.linalg.LinAlgError` when the matrix is singular.

---

**`residual(self)`**


---

**`variance_about_fit(self)`**

Returns the estimator of the standard deviation of the data about the fit.

**Return Value**

`numpy.ndarray` with `shape=(q,)`. Each entry corresponds to one of the `q` sets of equations that are being fit.

---

**`eff_n(self)`**

Returns something like the number of data, except that it looks at their weighting and the structure of the problem. It counts how many data have a relatively large effect on the solution, and if a datum has little influence, it doesn't count for much.

**Return Value**

float



**eff\_rank**(*self*)

Returns something like the rank of the solution, but rather than counting how many dimensions can be solved at all, it reports how many dimensions can be solved with a relatively good accuracy.

**Return Value**

float

*Inherited from object*

`--delattr--()`, `--format--()`, `--getattr--()`, `--hash--()`, `--new--()`, `--reduce--()`, `--reduce_ex--()`, `--repr--()`, `--setattr--()`, `--sizeof--()`, `--str--()`, `--subclasshook--()`

**59.3.2 Properties**

Name	Description
<i>Inherited from object</i>	
<code>--class--</code>	

**59.4 Class `linear_least_squares`**

object └

gmisclib.gpk\_lsq.lls\_base └

**gmisclib.gpk\_lsq.linear\_least\_squares****Known Subclasses:** `gmisclib.gpk_rlsq.w_linear_least_squares`

## 59.4.1 Methods

**`__init__(self, a, y=None, minsv=None, minsvr=None, copy=True)`**

This solves the set of linear equations  $a \cdot x = y$ , and allows you to get properties of the fit via methods. Normally,  $a.shape == (m, n)$  and  $y.shape == (m, q)$ , and the returned  $x.shape == (n, q)$ . where  $m$  is the number of constraints provided by the data,  $n$  is the number of parameters to use in a fit (equivalently, the number of basis functions), and  $q$  is the number of separate sets of equations that you are fitting. Then, `self.x()` has shape  $(n, q)$  and `self.the_fit()` has shape  $(m, q)$ . Interpreting this as a linear regression, there are  $n$  parameters in the model, and  $m$  measurements.  $Q$  is the number of times you apply the model to a different data set; each on yields a different solution.

The procedure uses a singular value decomposition algorithm, and treats all singular values that are smaller than `minsv` as zero (i.e. drops them). If `minsvr` is specified, it treats all singular values that are smaller than `minsvr` times the largest s.v. as zero. The returned rank is the rank of the solution, which is normally the number of nonzero elements in  $x$ . Note that the shape of the solution vector or matrix is defined by  $a$  and  $y$ , and the rank can be smaller than  $m$ .

Overrides: `object.__init__`

**Note:**  $Y$  may be a 1-D matrix (a vector), in which case the fit is a vector. This is the normal case where you are fitting one equation. If  $y$  is a 2-D matrix, each column (second index) in  $y$  is a separate fit, and each column in the solution is a separate result.

**`sv(self)`**

**`rank(self)`**

---

**hat**(*self*, *copy*=True)

---

Hat Matrix Diagonal Data points that are far from the centroid of the X-space are potentially influential. A measure of the distance between a data point,  $x_i$ , and the centroid of the X-space is the data point's associated diagonal element  $h_i$  in the hat matrix. Belsley, Kuh, and Welsch (1980) propose a cutoff of  $2p/n$  for the diagonal elements of the hat matrix, where  $n$  is the number of observations used to fit the model, and  $p$  is the number of parameters in the model. Observations with  $h_i$  values above this cutoff should be investigated. For linear models, the hat matrix

$$H = X \text{ inv}(X'X) X'$$

can be used as a projection matrix. The hat matrix diagonal variable contains the diagonal elements of the hat matrix

$$h_i = x_i \text{ inv}(X'X) x_i'$$

Overrides: `gmisclib.gpk_lsq.lls_base.hat`

---

**x\_variances**(*self*)

---

Estimated standard deviations of the solution. This is the diagonal of the solution covariance matrix.

---

**eff\_rank**(*self*)

---

Returns something like the rank of the solution, but rather than counting how many dimensions can be solved at all, it reports how many dimensions can be solved with a relatively good accuracy.

**Return Value**

float

Overrides: `gmisclib.gpk_lsq.lls_base.eff_rank` `exitit`(inherited documentation)

---

**eff\_n**(*self*)

---

Returns something like the number of data, except that it looks at their weighting and the structure of the problem. It counts how many data have a relatively large effect on the solution, and if a datum has little influence, it doesn't count for much.

**Return Value**

float

**fit**(*self*, *copy*=False)

**Raises**

`numpy.linalg.linalg.LinAlgError` when the matrix is singular.

**residual**(*self*)

**set\_y**(*self*, *y*, *copy*=True)

**variance\_about\_fit**(*self*)

Returns the estimator of the standard deviation of the data about the fit.

**Return Value**

`numpy.ndarray` with shape=(q,). Each entry corresponds to one of the q sets of equations that are being fit.

**x**(*self*, *y*=None, *copy*=True)

**Raises**

`numpy.linalg.linalg.LinAlgError` when the matrix is singular.

**y**(*self*, *copy*=True)

*Inherited from object*

`__delattr__()`, `__format__()`, `__getattr__()`, `__hash__()`, `__new__()`, `__reduce__()`, `__reduce_ex__()`, `__repr__()`, `__setattr__()`, `__sizeof__()`, `__str__()`, `__subclasshook__()`

#### 59.4.2 Properties

Name	Description
<i>Inherited from object</i>	
<code>__class__</code>	

### 59.5 Class `reg_linear_least_squares`



**59.5.1 Methods**

**`__init__(self, a, y, regstr=0.0, regtgt=None, rscale=None, copy=True)`**

This solves  $\min \|a^*x - y\|^2 + \|\text{regstr}^*(x - \text{regtgt})\|^2$ , and returns (x, the\_fit, rank, s). Normally, `a.shape==(m,n)` and `y.shape==(m,q)`, where `m` is the number of data to be fit, `n` is the number of parameters to use in a fit (equivalently, the number of basis functions), and `q` is the number of separate sets of equations that you are fitting. Then, `x` has shape `(n,q)` and `the_fit` has shape `(m,q)`.

The regularization target, `regtgt` is the same shape as `x`, that is `(n,q)`. (It must be a vector if and only if `y` is a vector.) `Regstr`, the strength of the regularization is normally an `(n,n)` matrix, though `(*,n)` will work, as will a scalar.

`Y` may be a 1-D matrix (a vector), in which case the fit is a vector. This is the normal case where you are fitting one equation. If `y` is a 2-D matrix, each column (second index) in `y` is a separate fit, and each column in the solution is a separate result.

Overrides: `object.__init__`

**`sv_reg(self)`**

Singular values of the regularized problem.

**`sv_unreg(self)`**

Singular values of the unregularized problem.

**`eff_rank(self)`**

Returns something like the rank of the solution, but rather than counting how many dimensions can be solved at all, it reports how many dimensions can be solved with a relatively good accuracy.

**Return Value**

float

Overrides: `gmisclib.gpk_lsq.lls_base.eff_rank` extit(inherited documentation)

**hat**(*self*, *copy*=True)

Hat Matrix Diagonal Data points that are far from the centroid of the X-space are potentially influential. A measure of the distance between a data point,  $x_i$ , and the centroid of the X-space is the data point's associated diagonal element  $h_i$  in the hat matrix. Belsley, Kuh, and Welsch (1980) propose a cutoff of  $2p/n$  for the diagonal elements of the hat matrix, where  $n$  is the number of observations used to fit the model, and  $p$  is the number of parameters in the model. Observations with  $h_i$  values above this cutoff should be investigated. For linear models, the hat matrix

$$H = X (X'X)^{-1} X'$$

can be used as a projection matrix. The hat matrix diagonal variable contains the diagonal elements of the hat matrix

$$h_i = x_i (X'X)^{-1} x_i'$$

Overrides: `gmisclib.gpk_lsq.lls_base.hat`

**eff\_n**(*self*)

Returns something like the number of data, except that it looks at their weighting and the structure of the problem. It counts how many data have a relatively large effect on the solution, and if a datum has little influence, it doesn't count for much.

**Return Value**

float

**fit**(*self*, *copy*=False)

**Raises**

`numpy.linalg.linalg.LinAlgError` when the matrix is singular.

**residual**(*self*)

**set\_y**(*self*, *y*, *copy*=True)

**variance\_about\_fit**(*self*)

Returns the estimator of the standard deviation of the data about the fit.

**Return Value**

`numpy.ndarray` with shape=( $q$ ). Each entry corresponds to one of the  $q$  sets of equations that are being fit.

<code>x(self, y=None, copy=True)</code>
---

<b>Raises</b>
---------------

<code>numpy.linalg.linalg.LinAlgError</code> when the matrix is singular.
---

<code>y(self, copy=True)</code>
---------------------------------

***Inherited from object***

`__delattr__()`, `__format__()`, `__getattr__()`, `__hash__()`, `__new__()`, `__reduce__()`, `__reduce_ex__()`,  
`__repr__()`, `__setattr__()`, `__sizeof__()`, `__str__()`, `__subclasshook__()`

**59.5.2 Properties**

Name	Description
<i>Inherited from object</i>	
<code>__class__</code>	

## 60 Module *gmisclib.gpk\_rlsq*

### 60.1 Functions

<code>test_w1()</code>
------------------------

<code>test_w1b()</code>
-------------------------

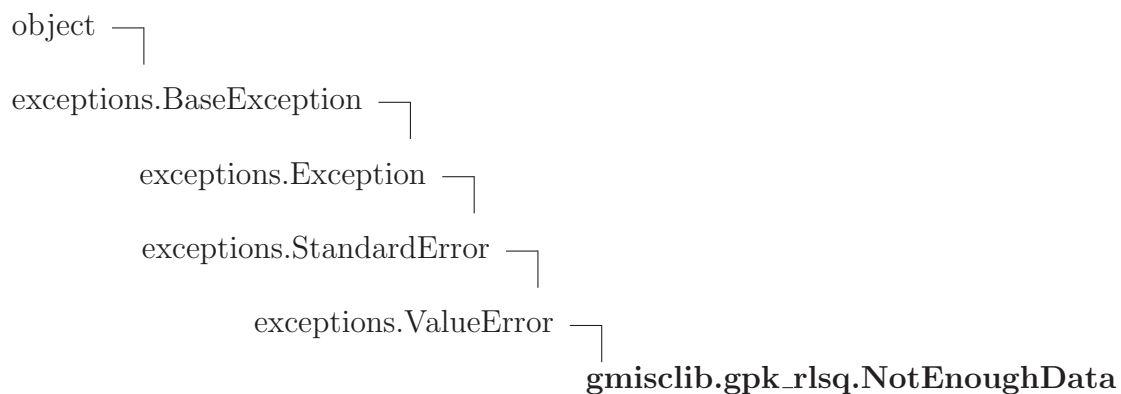
<code>test_r1()</code>
------------------------

<code>test_r1hat()</code>
---------------------------

### 60.2 Variables

Name	Description
<code>--package--</code>	<b>Value:</b> <code>'gmisclib'</code>

### 60.3 Class *NotEnoughData*



#### 60.3.1 Methods

<code>--init--(self, s)</code> <code>x.__init--(...)</code> initializes x; see <code>help(type(x))</code> for signature Overrides: <code>object.__init--</code> <code>extit(inherited documentation)</code>
---

*Inherited from `exceptions.ValueError`*



`__new__()`

**Inherited from `exceptions.BaseException`**

`__delattr__()`, `__getattr__()`, `__getitem__()`, `__getslice__()`, `__reduce__()`, `__repr__()`,  
`__setattr__()`, `__setstate__()`, `__str__()`, `__unicode__()`

**Inherited from `object`**

`__format__()`, `__hash__()`, `__reduce_ex__()`, `__sizeof__()`, `__subclasshook__()`

### 60.3.2 Properties

Name	Description
<i>Inherited from <code>exceptions.BaseException</code></i>	
<code>args</code> , <code>message</code>	
<i>Inherited from <code>object</code></i>	
<code>__class__</code>	

## 60.4 Class `w_linear_least_squares`



**Known Subclasses:** `gmisclib.gpk_rlsq.robust_linear_fit`

This solves the set of linear equations  $a \cdot x = y$ , with weights `w`. Normally, `a.shape==(m,n)` and `y.shape==(m,q)`, and the returned `x.shape==(n,q)`. where `m` is the number of constraints provided by the data, `n` is the number of parameters to use in a fit (equivalently, the number of basis functions), and `q` is the number of separate sets of equations that you are fitting. Then, `x` has shape `(n,q)` and the fit has shape `(m,q)`. Interpreting this as a linear regression, there are `n` parameters in the model, and `m` measurements. `Q` is the number of times you apply the model to a different data set; each one yields a different solution.

This uses `gpk_lsq.linear_least_squares` as the underlying algorithm.

**Note:** `Y` may be a 1-D matrix (a vector), in which case the fit is a vector. This is the normal case where you are fitting one equation. If `y` is a 2-D matrix, each column (second index) in `y` is a separate fit, and each column in the solution is a separate result.

## 60.4.1 Methods

**`__init__(self, a, y, w, minsv=None, minsvr=None, copy=True)`**

This solves the set of linear equations  $a \cdot x = y$ , and allows you to get properties of the fit via methods. Normally, `a.shape==(m,n)` and `y.shape==(m,q)`, and the returned `x.shape==(n,q)`. where `m` is the number of constraints provided by the data, `n` is the number of parameters to use in a fit (equivalently, the number of basis functions), and `q` is the number of separate sets of equations that you are fitting. Then, `self.x()` has shape `(n,q)` and `self.the_fit()` has shape `(m,q)`. Interpreting this as a linear regression, there are `n` parameters in the model, and `m` measurements. `Q` is the number of times you apply the model to a different data set; each on yields a different solution.

The procedure uses a singular value decomposition algorithm, and treats all singular values that are smaller than `minsv` as zero (i.e. drops them). If `minsvr` is specified, it treats all singular values that are smaller than `minsvr` times the largest s.v. as zero. The returned rank is the rank of the solution, which is normally the number of nonzero elements in `x`. Note that the shape of the solution vector or matrix is defined by `a` and `y`, and the rank can be smaller than `m`.

Overrides: `object.__init__` `extit(inherited documentation)`

**`fit(self, copy=False)`**

**Raises**

`numpy.linalg.linalg.LinAlgError` when the matrix is singular.

Overrides: `gmisclib.gpk_lsq.lls_base.fit`

**Note:** all elements of `w` must be nonzero.

**`y(self, copy=True)`**

Overrides: `gmisclib.gpk_lsq.lls_base.y`

**`set_y(self, y, copy=True)`**

Overrides: `gmisclib.gpk_lsq.lls_base.set_y`

**`residual(self)`**

Overrides: `gmisclib.gpk_lsq.lls_base.residual`

**eff\_n**(*self*)

Returns something like the number of data, except that it looks at their weighting and the structure of the problem. It counts how many data have a relatively large effect on the solution, and if a datum has little influence, it doesn't count for much.

**Return Value**

float

**eff\_rank**(*self*)

Returns something like the rank of the solution, but rather than counting how many dimensions can be solved at all, it reports how many dimensions can be solved with a relatively good accuracy.

**Return Value**

float

Overrides: `gmisclib.gpk_olsq.ols_base.eff_rank` `extit`(inherited documentation)

**hat**(*self*, *copy*=True)

Hat Matrix Diagonal Data points that are far from the centroid of the X-space are potentially influential. A measure of the distance between a data point,  $x_i$ , and the centroid of the X-space is the data point's associated diagonal element  $h_i$  in the hat matrix. Belsley, Kuh, and Welsch (1980) propose a cutoff of  $2p/n$  for the diagonal elements of the hat matrix, where  $n$  is the number of observations used to fit the model, and  $p$  is the number of parameters in the model. Observations with  $h_i$  values above this cutoff should be investigated. For linear models, the hat matrix

$$H = X \text{ inv}(X'X) X'$$

can be used as a projection matrix. The hat matrix diagonal variable contains the diagonal elements of the hat matrix

$$h_i = x_i \text{ inv}(X'X) x_i'$$

Overrides: `gmisclib.gpk_olsq.ols_base.hat`

**rank**(*self*)**sv**(*self*)

**variance\_about\_fit**(*self*)

Returns the estimator of the standard deviation of the data about the fit.

**Return Value**

`numpy.ndarray` with shape=(q,). Each entry corresponds to one of the q sets of equations that are being fit.

**x**(*self*, *y=None*, *copy=True*)**Raises**

`numpy.linalg.linalg.LinAlgError` when the matrix is singular.

**x\_variances**(*self*)

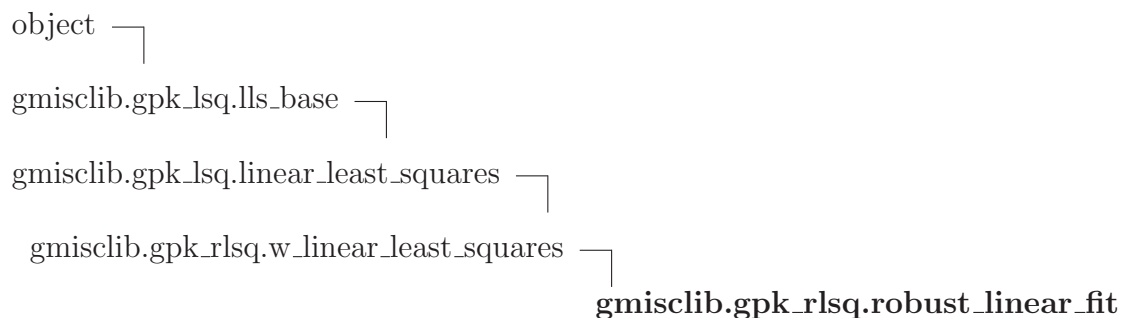
Estimated standard deviations of the solution. This is the diagonal of the solution covariance matrix.

**Inherited from object**

`__delattr__()`, `__format__()`, `__getattr__()`, `__hash__()`, `__new__()`, `__reduce__()`, `__reduce_ex__()`, `__repr__()`, `__setattr__()`, `__sizeof__()`, `__str__()`, `__subclasshook__()`

**60.4.2 Properties**

Name	Description
<i>Inherited from object</i>	
<code>__class__</code>	

**60.5 Class *robust\_linear\_fit***

**60.5.1 Methods**


---

**`__init__(self, a, y, sigma_y=None, minsv=None, minsvr=None, copy=True)`**


---

This does bounded influence regression, with a robust M-estimator in the y-direction.

Overrides: `object.__init__`

---

**`eff_n(self)`**


---

Returns something like the number of data, except that it looks at their weighting and the structure of the problem. It counts how many data have a relatively large effect on the solution, and if a datum has little influence, it doesn't count for much.

**Return Value**

float

---

**`eff_rank(self)`**


---

Returns something like the rank of the solution, but rather than counting how many dimensions can be solved at all, it reports how many dimensions can be solved with a relatively good accuracy.

**Return Value**

float

Overrides: `gmisclib.gpk_lsq.lls_base.eff_rank` extit(inherited documentation)

---

**`fit(self, copy=False)`**


---

**Raises**

`numpy.linalg.linalg.LinAlgError` when the matrix is singular.

Overrides: `gmisclib.gpk_lsq.lls_base.fit`

**Note:** all elements of `w` must be nonzero.

**hat**(*self*, *copy*=True)

Hat Matrix Diagonal Data points that are far from the centroid of the X-space are potentially influential. A measure of the distance between a data point,  $x_i$ , and the centroid of the X-space is the data point's associated diagonal element  $h_i$  in the hat matrix. Belsley, Kuh, and Welsch (1980) propose a cutoff of  $2p/n$  for the diagonal elements of the hat matrix, where  $n$  is the number of observations used to fit the model, and  $p$  is the number of parameters in the model. Observations with  $h_i$  values above this cutoff should be investigated. For linear models, the hat matrix

$$H = X \text{ inv}(X'X) X'$$

can be used as a projection matrix. The hat matrix diagonal variable contains the diagonal elements of the hat matrix

$$h_i = x_i \text{ inv}(X'X) x_i'$$

Overrides: *gmisclib.gpk\_olsq.ols\_base.hat*

**rank**(*self*)

**residual**(*self*)

Overrides: *gmisclib.gpk\_olsq.ols\_base.residual*

**set\_y**(*self*, *y*, *copy*=True)

Overrides: *gmisclib.gpk\_olsq.ols\_base.set\_y*

**sv**(*self*)

**variance\_about\_fit**(*self*)

Returns the estimator of the standard deviation of the data about the fit.

**Return Value**

`numpy.ndarray` with shape=( $q$ ). Each entry corresponds to one of the  $q$  sets of equations that are being fit.

**x**(*self*, *y*=None, *copy*=True)

**Raises**

`numpy.linalg.linalg.LinAlgError` when the matrix is singular.

<b>x_variances</b> ( <i>self</i> )
------------------------------------

Estimated standard deviations of the solution. This is the diagonal of the solution covariance matrix.
--

<b>y</b> ( <i>self</i> , <i>copy</i> =True)
---

Overrides: <i>gmisclib.gpk_lsq.lls_base.y</i>
---

### *Inherited from object*

`__delattr__()`, `__format__()`, `__getattr__()`, `__hash__()`, `__new__()`, `__reduce__()`, `__reduce_ex__()`, `__repr__()`, `__setattr__()`, `__sizeof__()`, `__str__()`, `__subclasshook__()`

#### 60.5.2 Properties

Name	Description
<i>Inherited from object</i>	
<code>__class__</code>	

#### 60.5.3 Class Variables

Name	Description
HTMIN	<b>Value:</b> 3.0
STMIN	<b>Value:</b> 1.5

## 61 Module *gmisclib.gpk\_writer*

### 61.1 Variables

Name	Description
<code>--package--</code>	<b>Value:</b> None

### 61.2 Class *writer*

object └─ ***gmisclib.gpk\_writer.writer***

**Known Subclasses:** *gmisclib.avio.writer*, *gmisclib.fiatio.writer*, *gmisclib.gpk\_writer.null\_writer*, *select\_fiat\_entries.writer*

#### 61.2.1 Methods

***comments***(*self*, *comments*)

Add comments to the data file. Comments can appear anywhere.

***comment***(*self*, *comment*)

***header***(*self*, *k*, *v*)

***headers***(*self*, *h*)

***\_\_init\_\_***(*self*, *fd*)

*x.\_\_init\_\_*(...) initializes *x*; see *help(type(x))* for signature

Overrides: *object.\_\_init\_\_* extit(inherited documentation)

***data***(*self*, *dataset*)

Write a series of lines to the output file.

***extend***(*self*, *d*)

***append***(*self*, *d*)



**datum**(*self*, *data\_item*)

**flush**(*self*)

**close**(*self*)

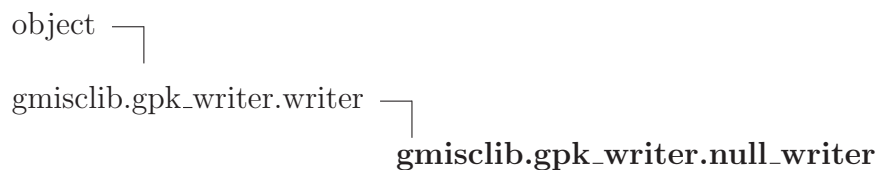
### *Inherited from object*

`__delattr__()`, `__format__()`, `__getattr__()`, `__hash__()`, `__new__()`, `__reduce__()`, `__reduce_ex__()`, `__repr__()`, `__setattr__()`, `__sizeof__()`, `__str__()`, `__subclasshook__()`

#### 61.2.2 Properties

Name	Description
<i>Inherited from object</i>	
<code>__class__</code>	

### 61.3 Class `null_writer`



#### 61.3.1 Methods

**comments**(*self*, *comments*)

Add comments to the data file. Comments can appear anywhere.

Overrides: `gmisclib.gpk_writer.writer.comments` `exitit`(inherited documentation)

**comment**(*self*, *comment*)

Overrides: `gmisclib.gpk_writer.writer.comment`

**header**(*self*, *k*, *v*)

Overrides: `gmisclib.gpk_writer.writer.header`

**headers**(*self*, *h*)Overrides: *gmisclib.gpk\_writer.writer.headers***\_\_init\_\_**(*self*)*x.\_\_init\_\_*(...) initializes *x*; see *help(type(x))* for signatureOverrides: *object.\_\_init\_\_* *exitit*(inherited documentation)**data**(*self*, *dataset*)

Write a series of lines to the output file.

Overrides: *gmisclib.gpk\_writer.writer.data* *exitit*(inherited documentation)**datum**(*self*, *data\_item*)Overrides: *gmisclib.gpk\_writer.writer.datum***flush**(*self*)Overrides: *gmisclib.gpk\_writer.writer.flush***close**(*self*)Overrides: *gmisclib.gpk\_writer.writer.close***append**(*self*, *d*)**extend**(*self*, *d*)***Inherited from object***

*\_\_delattr\_\_*(), *\_\_format\_\_*(), *\_\_getattr\_\_*(), *\_\_hash\_\_*(), *\_\_new\_\_*(), *\_\_reduce\_\_*(), *\_\_reduce\_ex\_\_*(),  
*\_\_repr\_\_*(), *\_\_setattr\_\_*(), *\_\_sizeof\_\_*(), *\_\_str\_\_*(), *\_\_subclasshook\_\_*()

**61.3.2 Properties**

Name	Description
<i>Inherited from object</i>	
<i>__class__</i>	

## 62 Module `gmisclib.gpkmisc`

### 62.1 Functions

**median**( $x$ )

**Raises**

`ValueError` if the input list is zero length.

**median\_across**( $xl$ )

**Raises**

`ValueError` If the input vectors are different lengths.

`ValueError` see `median`.

**Note:** There is a version of this in `Numeric_gpk` that is more efficient when the input is a list of `numpy.ndarray` vectors.

**avg**( $x$ )

**median\_ad**( $x$ )

Median absolute deviation

**mad**( $x$ )

Median absolute deviation

**mean**( $x$ )

**mean\_stdev**( $x$ )

**Parameters**

**x:** A list of data to average.  
(*type=list(float), typically.*)

**Return Value**

the mean and standard deviation of the input list. If there is only one datum, the standard deviation is reported as `None`.

(*type=(float, float) or (float, None)*)

**Raises**

`ValueError` If there is no data.

**mean\_ad**( $x$ )

---

**geo\_mean**(\**d*)

---

**Return Value**

Geometric mean of its arguments.

(*type=float*)

---

**Raises**

**ValueError** if any argument is negative.

---

**entropy**(*x*)

---

Compute the entropy of a list of probabilities.

---

**resample**(*d*)

---

Bootstrap resampling. Call this many times: each one returns a random resampling.

---

**jackknife**(*d*)

---

Jackknife resampling. Call this once. It returns a list of deleted lists.

---

**Student\_t\_dens**(*x*, *n*)

---

From p.337 Statistical Theory by Bernard Lindgren.

---

**log\_Student\_t\_dens**(*x*, *n*)

---

From p.337 Statistical Theory by Bernard Lindgren.

---

**log\_factorial**(*n*)

---

**log\_Combinations**(*n*, *m*)

---

**ComplexMedian**(*P*)

---

*P* is a list of complex numbers. This algorithm works by repeatedly stripping off the convex hull of the points.

---

**testCM**()

---

**thr\_iter\_read**(*fd*)

---

Read the contents of a file as an iterator. The read is two-threaded, so that one thread can be waiting on disk I/O while the other thread is processing the results.

**makedirs**(*fname*, *mode*=509)

This makes the specified directory, including all necessary directories above it. It is like `os.makedirs()`, except that if the directory already exists it does not raise an exception.

**Parameters**

**fname:** Name of the directory to create.

(*type*=*str*)

**mode:** Linux file permissions for any directories it needs to create.

(*type*=*int*)

**Raises**

**OSError** If it cannot create a part of the directory chain.

**Note:** If the directory already exists, it does not force it to have the specified **mode**.

**shuffle\_Nrep**(*y*, *n*=1, *compare*=None)

Shuffle a list, *y*, so that no item occurs more than *n* times in a row. Equality is determined by the comparison function *compare* returning zero.

**testSNR**()

---

**open\_nowipe**(*nm1*, *nm2*, *mode*='w')
 

---

Open a file (typically for writing) but make sure that the file doesn't already exist. The name is constructed from *nm1*, a sequence number, and *nm2*. The sequence number gets incremented until a name is found that doesn't exist. This works by creating a directory as a lock file; it should be safe across NFS.

**Parameters**

**nm1**: the part of the name to the left of the sequence number

(*type*=*str*)

**nm2**: the part of the name to the right of the sequence number

Typically, *nm2* is a suffix like ".wav". This may not contain a slash.

(*type*=*str*)

**mode**: The way to open the file – passed to `open()`.

(*type*=*str @rtype file*)

**Return Value**

The opened `file` object. (Its name can be gotten from the **name** attribute.)

**Note:** The directory containing **nm1** must exist and be writeable.

---

**dropfront**(*prefix*, *s*)
 

---

Drops the prefix from the string and raises an exception if it is not there to be dropped.

---

**open\_compressed**(*fn*)
 

---



---

**gammaIn**(*x*)
 

---



---

**a\_factor**(*n*)
 

---

Finds the smallest prime factor of a number.

---

**primes**()
 

---

This is a generator that produces an infinite list of primes.

**factor**(*n*)

Factor a number into a list of prime factors, in increasing order.

**Parameters**

**n:** input number  
(*type=int*)

**Return Value**

prime factors  
(*type=list*)

**test\_primes**()**gcd**(*a, b*)

Greatest common factor/denominator.

**Parameters**

**a:** (*type=int*)  
**b:** (*type=int*)

**Return Value**

the greatest common factor of a and b.  
(*type=int*)

**find\_in\_PATH**(*progrname*)

Search PATH to find where a program resides.

**Parameters**

**progrname:** the program to look for.  
(*type=str*)

**Return Value**

the full path name.  
(*type=str*)

**get\_mtime(*fn*)**

Paired with `need_to_recompute()`. These implement something like make, where we figure out if we need to compute things based on the age of files. This is used to get the age of the pre-requisites.

**Parameters**

**fn:** a filename or a file  
*(type=str or file)*

**Return Value**

None (if the file doesn't exist) or its modification time.  
*(type=None or float)*

**prereq\_mtime(\**tlist*)****need\_to\_recompute(*fn*, *lazytime*, *size*=-1)**

Paired with `get_mtime()`. These implement something like make, where we figure out if we need to compute things.

**Parameters**

**fn:** a filename or a file  
*(type=str or file)*

**lazytime:** a time (as obtained from `ST_MTIME` in `os.stat()`). If the file modification time of **fn** is older than **lazytime**, recompute.  
*(type=None or float)*

**size:** recompute the file if it is smaller than **size**. Normally, this is used to recompute on empty output files by setting **size**=0.  
*(type=int)*

**Return Value**

True if **fn** needs to be recomputed or if it doesn't exist.  
*(type=bool)*

**truncate(*s*, *maxlen*)****erf(*x*)**

$\text{erf}(x) = (2/\sqrt{\pi}) * \int_0^x \exp(-t^2) dt$



<b><code>asinh(<i>x</i>)</code></b>
Inverse hyperbolic sine.

<b><code>chooseP(<i>x</i>, <i>p</i>)</code></b>
Sample from a list with specified probabilities.
<b>Parameters</b>
<b><i>x</i></b> : a list of things from which to sample <i>(type=list(something))</i>
<b><i>p</i></b> : a list of probabilities for sampling the corresponding item of <b><i>x</i></b> . <i>(type=list(float))</i>
<b>Return Value</b>
a sample from <b><i>x</i></b> <i>(type=whatever is inside <i>x</i>)</i>
<b>Raises</b>
<b>AssertionError</b> Will (sometimes) detect negative probabilities, or probabilities that sum to something other than one.

<b><code>misc_mode(<i>lx</i>)</code></b>
<b>Parameters</b>
<b><i>lx</i></b> : a sequence of hashable objects.
<b>Return Value</b>
The most common object from a list of arbitrary objects.

<b><code>distrib(<i>key</i>)</code></b>
<b>Return Value</b>
The release name of the linux distribution that you're running. <i>(type=str)</i>

## 62.2 Variables

Name	Description
<code>--package--</code>	<b>Value:</b> <code>'gmisc.lib'</code>

## 62.3 Class `threaded_readable_file`



### 62.3.1 Methods

<b><code>__init__(self, fd)</code></b> <code>x.__init__(...)</code> initializes <code>x</code> ; see <code>help(type(x))</code> for signature Overrides: <code>object.__init__</code> <code>extit</code> (inherited documentation)
--

<b><code>readline(self)</code></b>
------------------------------------

<b><code>readlines(self)</code></b>
-------------------------------------

<b><code>read_iter(self)</code></b>
-------------------------------------

<b><code>__iter__(self)</code></b>
------------------------------------

### *Inherited from object*

`__delattr__()`, `__format__()`, `__getattr__()`, `__hash__()`, `__new__()`, `__reduce__()`, `__reduce_ex__()`,  
`__repr__()`, `__setattr__()`, `__sizeof__()`, `__str__()`, `__subclasshook__()`

### 62.3.2 Properties

Name	Description
<i>Inherited from object</i>	
<code>__class__</code>	

### 62.3.3 Class Variables

Name	Description
<code>QSIZE</code>	<b>Value:</b> 100

## 62.4 Class dir\_lock

object └─  
           gmisclib.gpkmisc.dir\_lock

### 62.4.1 Methods

```
__init__(self, lockname)
x.__init__(...) initializes x; see help(type(x)) for signature
Overrides: object.__init__ extit(inherited documentation)
```

```
__enter__(self)
```

```
__exit__(self, exc_type, exc_value, traceback)
```

#### *Inherited from object*

```
__delattr__(), __format__(), __getattr__(), __hash__(), __new__(), __reduce__(), __reduce_ex__(),
__repr__(), __setattr__(), __sizeof__(), __str__(), __subclasshook__()
```

### 62.4.2 Properties

Name	Description
<i>Inherited from object</i>	
<code>__class__</code>	

## 62.5 Class PrereqError

```

object └─
exceptions.BaseException └─
    exceptions.Exception └─
        exceptions.StandardError └─
            exceptions.ValueError └─
                gmisclib.gpkmisc.PrereqError
```

**62.5.1 Methods**

```
__init__(self, *s)
x.__init__(...) initializes x; see help(type(x)) for signature
Overrides: object.__init__ extit(inherited documentation)
```

```
repl(self, x)
```

*Inherited from exceptions.ValueError*

```
__new__()
```

*Inherited from exceptions.BaseException*

```
__delattr__(), __getattr__(), __getitem__(), __getslice__(), __reduce__(), __repr__(),
__setattr__(), __setstate__(), __str__(), __unicode__()
```

*Inherited from object*

```
__format__(), __hash__(), __reduce_ex__(), __sizeof__(), __subclasshook__()
```

**62.5.2 Properties**

Name	Description
<i>Inherited from exceptions.BaseException</i> args, message	
<i>Inherited from object</i> __class__	

## 63 Module *gmisclib.hilbert\_xform*

### 63.1 Functions

<b>nextpow2</b> ( <i>i</i> )
------------------------------

<b>hilbert</b> ( <i>x</i> , <i>cutoff</i> )
---

This is a Hilbert transform when cutoff=0.
--

### 63.2 Variables

Name	Description
--package--	Value: 'gmisclib'

## 64 Module `gmisclib.kl_dist`

Suppose there is a random variable with true distribution  $p$ . Then (as we will see) we could represent that random variable with a code that has average length  $H(p)$ . However, due to incomplete information we do not know  $p$ ; instead we assume that the distribution of the random variable is  $q$ . Then (as we will see) the code would need more bits to represent the random variable. The difference in the number of bits is denoted as  $D(p|q)$ . The quantity  $D(p|q)$  comes up often enough that it has a name: it is known as the relative entropy:

The relative entropy or Kullback-Leibler distance between two probability mass functions  $p(x)$  and  $q(x)$  is defined as  
 $D(p||q) = \text{Sum}\{x \text{ in } X\} p(x) \log(p(x)/q(x))$

Note that this is not symmetric, and the  $q$  (the second argument) appears only in the denominator.

### 64.1 Functions

<b>P</b> ( $x$ )
------------------

<b>multinomial_logp</b> ( $x, cF$ )
-------------------------------------

<b>multinomial_fixer</b> ( $x, c$ )
-------------------------------------

<b>kl_nonzero_probs</b> ( $p, q$ )
------------------------------------

Kullback-Liebler distance between two normalized, nonzero probability distributions.
--

<b>kl_nonzero_prob_m</b> ( $p, q$ )
-------------------------------------

<b>kl_dist_vec</b> ( $p, q, N=\text{None}, Fp=1.0, Fq=1.0, Clip=0.01$ )
---

Relative entropy or Kullback-Liebler distance between two frequency distributions $p$ and $q$ . Here, we assume that both $p$ and $q$ are counts derived from multinomial distributed data; they are not normalized to one.
---

<b>tr_from_obs</b> ( $x$ )
----------------------------

Given a matrix of $P(i \text{ and } j)$ as $x[i,j]$ , we compute $P(j \text{ given } i)$ and return it as $y[i,j]$ . The result is a transition probability matrix where the first index is the input state, and the second index marks the result state.
---

<b>estimate_tr_probs</b> ( <i>counts</i> , <i>N</i> , <i>F</i> =1.0)
--

<b>solve_for_pi</b> ( <i>p</i> )
----------------------------------

Given a transition probability matrix <i>p</i> , where the first index is the initial state and the second index is the resultant state, compute the steady-state probability distribution, assuming a Markov process.
--

<b>kl_nonzero_tr_probs</b> ( <i>pp</i> , <i>qq</i> )
--

KL distance, given a matrix of nonzero transition probabilities. Each matrix indexes states as <i>pp</i> [from,to], and contains $P(\text{to given from})$ as a conditional probability, where for any from, the Sum over to( <i>pp</i> [from,to]) = 1.
---

<b>kl_nonzero_tr_prob_m</b> ( <i>pp</i> , <i>qq</i> )
---

<b>cross</b> ( <i>a</i> , <i>b</i> )
--------------------------------------

<b>kldist_Markov</b> ( <i>p</i> , <i>q</i> , <i>N</i> =None)
--

Kullback-Liebler distance between two matrices of bigram counts.
--

<b>kldist_Markov_m</b> ( <i>p</i> , <i>q</i> , <i>N</i> =None)
--

Kullback-Liebler distance between two matrices of bigram counts.
--

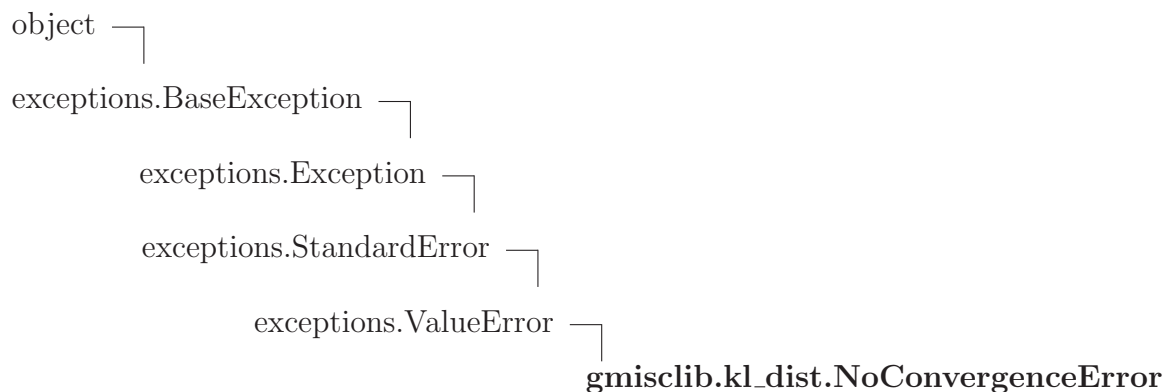
<b>kldist_Markov_mm</b> (* <i>p</i> )
---------------------------------------

List of Kullback-Liebler distances between all combinations of pairs of matrices of bigram counts. It returns a list of matrices of all the distances. Each item on the list is a sample of the distance histogram.
---

## 64.2 Variables

Name	Description
<code>__package__</code>	<b>Value:</b> 'gmisclib'

### 64.3 Class NoConvergenceError



#### 64.3.1 Methods

**`__init__(self, s)`**  
`x.__init__(...)` initializes `x`; see `help(type(x))` for signature  
 Overrides: `object.__init__` extit(inherited documentation)

*Inherited from `exceptions.ValueError`*

`__new__()`

*Inherited from `exceptions.BaseException`*

`__delattr__()`, `__getattr__()`, `__getitem__()`, `__getslice__()`, `__reduce__()`, `__repr__()`,  
`__setattr__()`, `__setstate__()`, `__str__()`, `__unicode__()`

*Inherited from `object`*

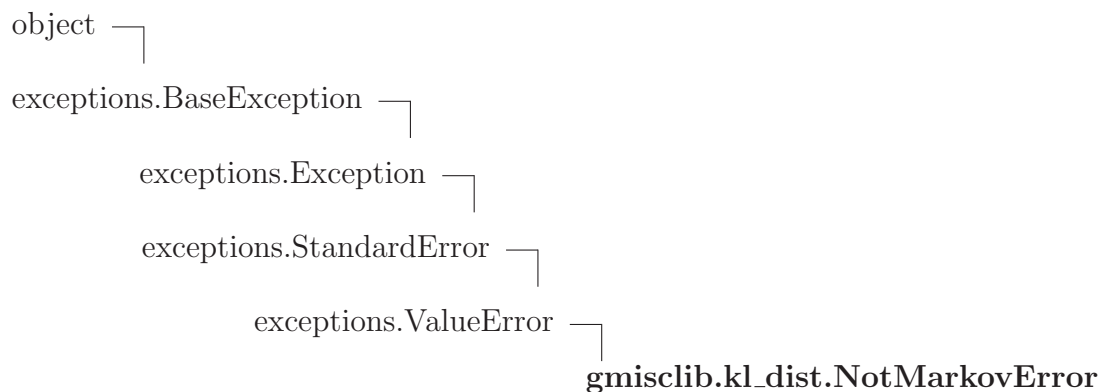
`__format__()`, `__hash__()`, `__reduce_ex__()`, `__sizeof__()`, `__subclasshook__()`

#### 64.3.2 Properties

Name	Description
<i>Inherited from <code>exceptions.BaseException</code></i>	
<code>args</code> , <code>message</code>	
<i>Inherited from <code>object</code></i>	
<code>__class__</code>	



## 64.4 Class `NotMarkovError`



### 64.4.1 Methods

**`__init__`**(*self*, *s*)  
*x*.**`__init__`**(...) initializes *x*; see `help(type(x))` for signature  
 Overrides: `object.__init__` `extit`(inherited documentation)

*Inherited from `exceptions.ValueError`*

`__new__`()

*Inherited from `exceptions.BaseException`*

`__delattr__`(), `__getattr__`(), `__getitem__`(), `__getslice__`(), `__reduce__`(), `__repr__`(),  
`__setattr__`(), `__setstate__`(), `__str__`(), `__unicode__`()

*Inherited from `object`*

`__format__`(), `__hash__`(), `__reduce_ex__`(), `__sizeof__`(), `__subclasshook__`()

### 64.4.2 Properties

Name	Description
<i>Inherited from <code>exceptions.BaseException</code></i>	
<code>args</code> , <code>message</code>	
<i>Inherited from <code>object</code></i>	
<code>__class__</code>	

## 65 Module `gmisclib.load_mod`

This module has functions that help you dynamically import modules.

### 65.1 Functions

**`split_name(name)`**

Split a name in the form `a/b.c` into `a`, `b`, `c`, where `a` is a search path, `b` is a module (package) name, and `c` is a name in the module.

**`load(name, path)`**

Load a module from the specified list of paths. It returns the module, but does not import it. If `path` is `None`, only look in `sys.path` and builtins. If `path` is an array containing `None`, replace the `None` with `sys.path`.

**`load_mod(name, path)`**

Load a module from the specified list of paths. It returns the module, but does not import it. If `path` is `None`, only look in `sys.path` and builtins. If `path` is an array containing `None`, replace the `None` with `sys.path`.

**`load_inc_path(name, path)`**

Load a module from the specified list of paths. It returns the module, but does not import it. If `path` is `None`, only look in `sys.path` and builtins. If `path` is an array containing `None`, replace the `None` with `sys.path`.

**`load_mod_inc_path(name, path)`**

Load a module from the specified list of paths. It returns the module, but does not import it. If `path` is `None`, only look in `sys.path` and builtins. If `path` is an array containing `None`, replace the `None` with `sys.path`.

**`load_named(name, use_sys_path=True)`**

Load a module. If the module name is in the form `a/b`, it looks in directory `"a"` first. If `use_sys_path` is true, it searches the entire Python path

It returns the module, but does not import it. This version handles importing packages and functions nicely, but with less control over the search path.

Usage:

- `load_named_module('/dir/my_module')`, or
- `load_named_module('foo/my_module')`, or
- `load_named_module('foo/my_module.submodule.function')`, or
- various combinations.

**`load_named_module(name, use_sys_path=True)`**

Load a module. If the module name is in the form `a/b`, it looks in directory `"a"` first. If `use_sys_path` is true, it searches the entire Python path

It returns the module, but does not import it. This version handles importing packages and functions nicely, but with less control over the search path.

Usage:

- `load_named_module('/dir/my_module')`, or
- `load_named_module('foo/my_module')`, or
- `load_named_module('foo/my_module.submodule.function')`, or
- various combinations.

**`load_named_fcn(name, use_sys_path=True)`**

Load a module. If the module name is in the form `a/b`, it looks in directory `"a"` first. If `use_sys_path` is true, it searches the entire Python path

It returns the module, but does not import it. This version handles importing packages and functions nicely, but with less control over the search path.

Usage:

- `load_named_module('/dir/my_module')`, or
- `load_named_module('foo/my_module')`, or
- `load_named_module('foo/my_module.submodule.function')`, or
- various combinations.

**load\_fcn**(*name*, *use\_sys\_path*=True)

Load a module. If the module name is in the form a/b, it looks in directory "a" first. If *use\_sys\_path* is true, it searches the entire Python path

It returns the module, but does not import it. This version handles importing packages and functions nicely, but with less control over the search path.

Usage:

- `load_named_module('/dir/my_module')`, or
- `load_named_module('foo/my_module')`, or
- `load_named_module('foo/my_module.submodule.function')`, or
- various combinations.

## 65.2 Variables

Name	Description
<code>--package--</code>	<b>Value:</b> 'gmisclib'

## 66 Module *gmisclib.lreg\_fill*

### 66.1 Functions

<code>fill(<i>f</i>, <i>wt</i>, <i>s</i>)</code>
--

### 66.2 Variables

Name	Description
LREG	<b>Value:</b> <code>'/home/gpk/misctts/lreg_fit2'</code>
<code>--package--</code>	<b>Value:</b> <code>'gmisclib'</code>

## 67 Module **gmisclib.makemake**

This module is designed to build makefiles.

### 67.1 Functions

<b>maketemp</b> ( <i>dir</i> =' /tmp', <i>prefix</i> ='maketemp', <i>suffix</i> ='')
--

<b>setlog</b> ( <i>f</i> )
----------------------------

<b>log</b> (* <i>s</i> )
--------------------------

Log some strings, one per line.
---------------------------------

<b>quote</b> ( <i>s</i> )
---------------------------

<b>var</b> ( <i>k</i> , <i>v</i> )
------------------------------------

Pass a variable to make.
--------------------------

<b>rule</b> ( <i>a</i> , * <i>b</i> )
---------------------------------------

Writes a rule into a makefile. All lines except the first are indented.
---

<b>blank</b> ()
-----------------

<b>set_debug</b> ()
---------------------

<b>finish</b> ()
------------------

<b>date</b> ()
----------------

<b>path_to</b> ( <i>s</i> )
-----------------------------

<b>set_make_prog</b> (* <i>s</i> )
------------------------------------

**`ncpu()`**

**Return Value**

a string representation of the integer number of cores that the computer has.

(*type=str !Not an integer!*)

**`read(fn)`**

## 67.2 Variables

Name	Description
<code>makeflags</code>	<b>Value:</b> <code>[]</code>
<code>makefile</code>	<b>Value:</b> <code>None</code>
<code>DB_format</code>	<b>Value:</b> <code>'fiatio'</code>
<code>__package__</code>	<b>Value:</b> <code>'gmisclib'</code>

## 67.3 Class `FileNotFound`



### 67.3.1 Methods

**`__init__(self, *s)`**

`x.__init__(...)` initializes `x`; see `help(type(x))` for signature

Overrides: `object.__init__` extit(inherited documentation)

***Inherited from `exceptions.Exception`***

`__new__()`

***Inherited from `exceptions.BaseException`***

`__delattr__()`, `__getattr__()`, `__getitem__()`, `__getslice__()`, `__reduce__()`, `__repr__()`,  
`__setattr__()`, `__setstate__()`, `__str__()`, `__unicode__()`

***Inherited from object***

`__format__()`, `__hash__()`, `__reduce_ex__()`, `__sizeof__()`, `__subclasshook__()`

**67.3.2 Properties**

Name	Description
	<i>Inherited from exceptions.BaseException</i>
	args, message
	<i>Inherited from object</i>
<code>__class__</code>	



## 68 Module `gmisclib.matrix_arrange`

Take a covariance-like matrix, and re-order it to be close to a diagonal matrix. We calculate a map that renames the indices (the same map for both left and right), to bring large off-diagonal elements close to the diagonal.

### 68.1 Functions

<code>diagonalize(<i>a</i><math>\theta</math>)</code>
---

<code>map_array(<i>a</i>, <i>m</i>)</code>
--

<code>test()</code>
---------------------

### 68.2 Variables

Name	Description
<code>--package--</code>	<b>Value:</b> <code>'gmisclib'</code>

## 69 Module `gmisclib.matrix_arrange_entropy`

This rotates a matrix by multiplying with a unitary matrix so that the resulting elements are either nearly zero or relatively large. It minimizes sum of  $|x_{ij}| \log(|x_{ij}|)$ , i.e. the entropy (sortof).

### 69.1 Functions

<b><code>make_min_entropy</code></b> ( <i>x</i> , <i>extra_entropy</i> =0.0)
--

<i>extra_entropy</i> is a vector that helps choose what to optimize.
--

<b><code>test2</code></b> ()
------------------------------

<b><code>test3</code></b> ()
------------------------------

### 69.2 Variables

Name	Description
<code>--package--</code>	<b>Value:</b> <code>'gmisclib'</code>

## 70 Module `gmisc.lib.matrix_rearrange_labels`

This module helps you plot confusion matrices or similar images where the axis labels do not have a natural order. So, the probability of confusing two phonemes is a perfect example: phonemes do not naturally fall onto a 1-dimensional sequence, so one is free to put them in any order one likes. Given that, one might as well put them into an order that reveals something interesting about the probabilities.

All the functions ending in "2" work on rectangular arrays. All the functions without "2" work only on square arrays, and they assume that both axes remain in the same order.

### 70.1 Functions

<b><code>symm_swap_toward_minimal_fom</code></b> <i>(<code>m</code>, <code>fom</code>, <code>maxtries</code>)</i>
Used internally. Finds an ordering of the labels that minimizes whatever <code>fom</code> you supply. Both axes will have the same order.

<b><code>swap_toward_minimal_fom</code></b> <i>(<code>m</code>, <code>fom</code>, <code>maxtries</code>)</i>
Used internally. Finds an ordering of the labels that minimizes whatever <code>fom</code> you supply. The ordering of the two axes may be different.

**swap\_toward\_diag**(*m*, *lbls*, *maxtries*=None, *sign*=1)

Swap the rows and columns of a matrix to bring it closer to a diagonal matrix: i.e. entries with large absolute values on the main diagonal and small entries away from the main diagonal. Rows and columns are swapped together, so that the ordering of rows will match the ordering of columns.

**Parameters**

- m:** a 2-dimensional array  
(*type*=`numpy.ndarray`)
- lbls:** a list of arbitrary labels for each row or column.  
(*type*=`list(anything)`)
- maxtries:** how hard to work at optimizing the matrix layout? None gives a resonable default value. Appropriate values are a few times the number of elements in the matrix.  
(*type*=`int` or `None`)
- sign:** (default=1) if **sign**=-1, do the opposite: put the small absolute values on the main diagonal.  
(*type*=`int`)

**Return Value**

A tuple containing the swapped matrix and a swapped list of values.  
(*type*=`tuple(numpy.ndarray, list(something))`)

---

**swap\_toward\_positive**(*m*, *lbls*, *maxtries*=None, *sign*=1)

Swap the rows and columns of a matrix to put the most positive values on the main diagonal. Rows and columns are swapped together, so that the ordering of rows will match the ordering of columns.

**Parameters**

- m:** a 2-dimensional array  
(*type*=*numpy.ndarray*)
- lbls:** a list of arbitrary labels for each row or column.  
(*type*=*list(anything)*)
- maxtries:** how hard to work at optimizing the matrix layout? None gives a resonable default value. Appropriate values are a few times the number of elements in the matrix.  
(*type*=*int* or *None*)
- sign:** (default=1) if **sign**=-1, do the opposite: put the negative entries on the main diagonal.  
(*type*=*int*)

**Return Value**

A tuple containing the swapped matrix and a swapped list of labels.  
(*type*=*tuple(numpy.ndarray, list(something))*)

**swap\_toward\_diag2**(*m*, *lbl1*, *lbl2*, *maxtries*=None)

Swap the rows and columns of a matrix to make it roughly diagonal: i.e. large entries on the main diagonal and small entries away from the main diagonal. Note that this will work even if the matrix is not square.

#### Parameters

- m:** a 2-dimensional array  
(*type*=`numpy.ndarray`)
- lbl1:** a list of arbitrary labels for the first index of the matrix *m*.  
(*type*=`list(anything)`)
- lbl2:** a list of arbitrary labels for the second index of the matrix *m*.  
(*type*=`list(anything)`)
- maxtries:** how hard to work at optimizing the matrix layout? None gives a resonable default value. Appropriate values are a few times the number of elements in the matrix.  
(*type*=`int` or `None`)

#### Return Value

A tuple containing the swapped matrix and the two swapped lists of labels, one for the first and one for the second axis.  
(*type*=`tuple(numpy.ndarray, list(something), list(something))`)

**pos\_near\_diag2**(*m*, *lbl1*, *lbl2*, *maxtries*=None)

**neg\_near\_diag2**(*m*, *lbl1*, *lbl2*, *maxtries*=None)

**swap\_toward\_blocks2**(*m*, *lbl1*, *lbl2*, *maxtries*=None)

Swap rows and columns of a matrix to bring it closer to a block form, where similar values occur together in blocks.

**Parameters**

- m:** a 2-dimensional array  
(*type*=*numpy.ndarray*)
- lbl1:** a list of arbitrary labels for the first axis of m  
(*type*=*list(anything)*)
- lbl2:** a list of arbitrary labels for the second axis of m  
(*type*=*list(anything)*)
- maxtries:** how hard to work at optimizing the matrix layout? None gives a resonable default value. Appropriate values are a few times the number of elements in the matrix.  
(*type*=*int* or *None*)

**Return Value**

A tuple containing the swapped matrix and the two swapped lists of values, one for the first and one for the second axis.

(*type*=*tuple(numpy.ndarray, list(something), list(something))*)

**swap\_toward\_blocks**(*m*, *lbls*, *maxtries*=None)

Swap rows and columns of a matrix to bring it closer to a block form, where similar values occur together in blocks. It minimizes the sum of changes when proceeding along a row or column.

**Parameters**

- m:** a 2-dimensional array  
(*type*=*numpy.ndarray*)
- lbls:** a list of arbitrary labels for the first axis of m  
(*type*=*list(anything)*)
- maxtries:** how hard to work at optimizing the matrix layout? None gives a resonable default value. Appropriate values are a few times the number of elements in the matrix.  
(*type*=*int* or *None*)

**Return Value**

A tuple containing the swapped matrix and a swapped lists of values.

(*type*=*tuple(numpy.ndarray, list(something))*)

<b>test1()</b>
----------------

Two test cases on 2x2 matrices.
---------------------------------

<b>test()</b>
---------------

## 70.2 Variables

Name	Description
<code>--package--</code>	<b>Value:</b> 'gmisclib'

## 70.3 Class diagfom

object └─  
           gmisclib.matrix\_rearrange\_labels.diagfom

A class that defines a figure-of-merit for a matrix. It tries to put the largest entries on the diagonal. Instances of this class are passed to `symm.swap_toward_minimal_fom`.

### 70.3.1 Methods

<b><code>--init--</code></b> ( <i>self</i> , <i>n</i> , <i>sign</i> =1)
---

<code>x.__init__</code> (...) initializes x; see <code>help(type(x))</code> for signature
---

#### Parameters

- sign:**   • If `sign=1`, the f-o-m returned by `eval` will be minimal when the most positive matrix elements are on the diagonal.
- If `sign=-1`, you'll get minimal fom with the the most positive elements off diagonal.

(*type=int*)

**n:**       dimension of matrix.

(*type=int*)

Overrides: <code>object.__init__</code>
---



```
eval(self, swv, m)
```

Evaluate the figure of merit for a matrix m with the specified swap vector.

**Parameters**

**m:** matrix

**swv:** swap vector

**Return Value**

figure of merit

(type=int or float (according to the type of m).)

**Inherited from object**

`__delattr__()`, `__format__()`, `__getattr__()`, `__hash__()`, `__new__()`, `__reduce__()`, `__reduce_ex__()`, `__repr__()`, `__setattr__()`, `__sizeof__()`, `__str__()`, `__subclasshook__()`

**70.3.2 Properties**

Name	Description
<i>Inherited from object</i>	
<code>__class__</code>	

**70.4 Class diagfom2**

```
object └─ gmsclib.matrix_rearrange_labels.diagfom2
```

This tries to make your matrix approximately diagonal for rectangular matrices.

**70.4.1 Methods**

```
__init__(self, n1, n2)
```

x.`__init__`(...) initializes x; see `help(type(x))` for signature

Overrides: `object.__init__` extit(inherited documentation)

```
eval(self, sw1, sw2, m)
```

**Inherited from object**

`__delattr__()`, `__format__()`, `__getattr__()`, `__hash__()`, `__new__()`, `__reduce__()`, `__reduce_ex__()`,  
`__repr__()`, `__setattr__()`, `__sizeof__()`, `__str__()`, `__subclasshook__()`

#### 70.4.2 Properties

Name	Description
<i>Inherited from object</i>	
<code>__class__</code>	

### 70.5 Class *blockfom*

object └─ **`gmisc.lib.matrix_rearrange_labels.blockfom`**

This tries to make your matrix into a block form. The idea is to minimize the sum of entry-to-entry differences along a row (same with columns).

#### 70.5.1 Methods

<b><code>__init__(self, n1, n2)</code></b>
<code>x.__init__(...)</code> initializes x; see <code>help(type(x))</code> for signature
Overrides: <code>object.__init__</code> <code>extit</code> (inherited documentation)
<b><code>eval(self, sw1, sw2, m)</code></b>

#### *Inherited from object*

`__delattr__()`, `__format__()`, `__getattr__()`, `__hash__()`, `__new__()`, `__reduce__()`, `__reduce_ex__()`,  
`__repr__()`, `__setattr__()`, `__sizeof__()`, `__str__()`, `__subclasshook__()`

#### 70.5.2 Properties

Name	Description
<i>Inherited from object</i>	
<code>__class__</code>	

## 71 Module `gmisclib.mcmc`

Bootstrap Markov-Chain Monte-Carlo algorithm. This can be used to generate samples from a probability distribution, or also as a simulated annealing algorithm for maximization. This can be imported and its functions and classes can be used.

The central interfaces are the `BootStepper` class, and within it, the `BootStepper.step` method is used iteratively to take a Markov step. To use this module, the user should derive a class from the `problem_definition` class and redefine (at least) the `problem_definition.logp` method.

It was originally inspired by `amoeba.anneal` (in Numerical Recipes, Press et al.). The essential feature is that it keeps a large archive of previous positions (possibly many times more than  $N$  of them). It samples two positions from the archive and subtracts them to generate candidate steps. This has the nice property that when sampling from a multivariate Gaussian distribution, the candidate steps match the distribution nicely.

It can be operated in two modes (or set to automatically switch). One is optimization mode where it heads for the maximum of the probability distribution. The other mode is sampling mode, where it asymptotically follows a Markov sampling procedure and has the proper statistical properties.

The algorithm has been described in:

- “Bootstrap Markov chain Monte Carlo and optimal solutions for the Law of Categorical Judgment (Corrected)” Greg Kochanski and Burton S. Rosner (2010), `\emph{arXiv:1008.1596}` available from `\url{http://arxiv.org/abs/1008.1596}`.

An earlier version has been used in:

- “Rhythm measures and dimensions of durational variation in speech”, Anastassia Loukina, Greg Kochanski, Burton Rosner, and Elinor Keane and Chilin Shih (Submitted 2010 to J. Acoustical Society of America). Preprint available at <http://kochanski.org/gpk/papers/2010/class>
- “Image quality in non-gated versus gated reconstruction of tongue motion using magnetic resonance imaging: a comparison using automated image processing”, Christopher Alvey, C. Orphanidou, J. Coleman, A. McIntyre, S. Golding and G. Kochanski, *Intl. J. of Computer Assisted Radiology and Surgery* v3(5), 2008, pp. 457–464, doi:10.1007/s11548-008-0218-5<sup>1</sup>.
- “Evidence for attractors in English intonation”, Braun, B., Kochanski, G., Grabe, E., Rosner, B. S., 2006, *J. Acoustical Society of America* 119(6) 4006–4015 <http://dx.doi.org/10.1121/1.21>

<sup>1</sup><http://dx.doi.org/10.1007/s11548-008-0218-5>

## 71.1 Functions

**start\_is\_list\_a**(*start*)

Is the argument a sequence of numpy arrays?

**start\_is\_list\_p**(*start*)

Is the argument a sequence of position\_base objects?

**make\_list\_of\_positions**(*x*, *PositionClass*, *problem\_def*)

Turn almost anything into a list of position\_base objects. You can hand it a sequence of numpy vectors or a single 1-dimensional numpy vector; a sequence of position\_base objects or a single 1-dimensional position\_base object.

**Precondition:** This depends on PositionClass being callable as PositionClass(vector\_of\_doubles, problem\_definition).

**bootstepper**(*logp*, *x*, *v*, *c=None*, *strategy='intermediate'*, *fixer=None*, *repeatable=True*)

This is (essentially) another interface to the class constructor. It's really there for backwards compatibility.

**test**\_(*stepper*)

**diag\_variance**(*start*)

Hand this a list of vectors and it will compute the variance of each component, then return a diagonal covariance matrix.

**test**()

## 71.2 Variables

Name	Description
Debug	<b>Value:</b> 0
MEMORYS_WORTH_OF_PARAMETERS	How many bytes can we use to store the archives? <b>Value:</b> 100000000.0
__package__	<b>Value:</b> 'gmisclib'

### 71.3 Class `problem_definition`

object —  
**`gmisclib.mcmc.problem_definition`**

**Known Subclasses:** `gmisclib.mcmc.problem_definition.F`, `gmisclib.mcmc_idxr.problem_definition`

This class implements the problem to be solved. It's main function in life is to compute the probability that a given parameter vector is acceptable. `Mcmc.py` then uses that to run a Markov-Chain Monte-Carlo sampling. You probably want to derive a class from this and override most of the functions defined here, or implement your own class with the same functions. (Of course, in either case, it can contain extra functions also.)

For instance, it can be a good idea to define a function "model" that computes the model that you are fitting to data (if that is your plan). Then `logp()` can be something like `return -numpy.sum((self.model()-self.data)**2)`. Also, it can be good to define a "guess" function that computes a reasonable initial guess to the parameters, somehow.

#### 71.3.1 Methods

**`__init__(self)`**

`x.__init__(...)` initializes `x`; see `help(type(x))` for signature

Overrides: `object.__init__` extit(inherited documentation)

**`logp(self, x)`**

Compute the log of the probability density at `x`.

**Parameters**

`x`: a parameter vector

(*type=*`numpy.ndarray`)

**Return Value**

The log of the probability that the model assigns to parameters `x`.

(*type=*`float`)

**Raises**

**NotGoodPosition** This exception is used to indicate that the position `x` is not valid. This is equivalent to returning an extremely negative `logp`.

**fixer**(*self*, *x*)

This is called on each candidate position vector. Generally, it is used to restrict the possible solution space by folding position vectors that escape outside the solution space back into the solution space. It can also allow for symmetries in equations.

Formally, it defines a convex region. All vectors outside the region are mapped into the region, and the mapping must be continuous at the boundary. (More precisely, `logp(fixer(x))` must be continuous everywhere that `logp(x)` is continuous, including the boundary.) For instance, mapping `x[0]` into `abs(x[0])` defines a convex region (the positive half-space), and the mapping is continuous near `x[0]=0`.

Additionally, it may re-normalize parameters at will subject to the restriction that `logp(fixer(x))==logp(x)`. For instance, it can implement a constraint that `sum(x)==0` by mapping `x` into `x - average(x)`, so long as the value of `logp()` is unaffected by that substitution. Other folds can sometimes lead to problems.

**Parameters**

`x`: a parameter vector  
(*type=*`numpy.ndarray`)

**Return Value**

a (possibly modified) parameter vector.  
(*type=*`numpy.ndarray`)

**Raises**

**NotGoodPosition** This exception is used to indicate that the position `x` is not valid. Fixer has the option of either mapping invalid parameter vectors into valid ones or raising this exception.

**Attention:** Within a convex region (presumably one that contains the optimal `x`), fixer must *not* change the value of `logp()`: `logp(fixer(x)) == logp(x)`.

**log**(*self*, *p*, *i*)

Some code calls this function every iteration to log the current state of the MCMC process.

**Parameters**

`p`: the current parameter vector, and  
`i`: an integer iteration counter.

**Return Value**

nothing.

***Inherited from object***

`__delattr__()`, `__format__()`, `__getattr__()`, `__hash__()`, `__new__()`, `__reduce__()`, `__reduce_ex__()`,  
`__repr__()`, `__setattr__()`, `__sizeof__()`, `__str__()`, `__subclasshook__()`

**71.3.2 Properties**

Name	Description
<i>Inherited from object</i>	
<code>__class__</code>	

**71.4 Class `problem_definition_F`**

```

object └─
gmisclib.mcmc.problem_definition └─
                                     gmisclib.mcmc.problem_definition_F

```

This is a variant that is used bin the `bin/mcmc.py` script.

**71.4.1 Methods**

<p><code>__init__(self, logp_fcn, c, fixer=None)</code></p> <p><code>x.__init__(...)</code> initializes <code>x</code>; see <code>help(type(x))</code> for signature</p> <p>Overrides: <code>object.__init__</code> <code>exitit</code>(inherited documentation)</p>
--

**logp**(*self*, *x*)

Compute the log of the probability density at **x**.

**Parameters**

**x**: a parameter vector

(*type=numpy.ndarray*)

**Return Value**

The log of the probability that the model assigns to parameters **x**.

(*type=float*)

**Raises**

**NotGoodPosition** This exception is used to indicate that the position **x** is not valid. This is equivalent to returning an extremely negative logp.

Overrides: *gmisclib.mcmc.problem\_definition.logp*



**fixer**(*self*, *x*)

This is called on each candidate position vector. Generally, it is used to restrict the possible solution space by folding position vectors that escape outside the solution space back into the solution space. It can also allow for symmetries in equations.

Formally, it defines a convex region. All vectors outside the region are mapped into the region, and the mapping must be continuous at the boundary. (More precisely, `logp(fixer(x))` must be continuous everywhere that `logp(x)` is continuous, including the boundary.) For instance, mapping `x[0]` into `abs(x[0])` defines a convex region (the positive half-space), and the mapping is continuous near `x[0]=0`.

Additionally, it may re-normalize parameters at will subject to the restriction that `logp(fixer(x)) == logp(x)`. For instance, it can implement a constraint that `sum(x) == 0` by mapping `x` into `x - average(x)`, so long as the value of `logp()` is unaffected by that substitution. Other folds can sometimes lead to problems.

#### Parameters

`x`: a parameter vector  
(*type=*`numpy.ndarray`)

#### Return Value

a (possibly modified) parameter vector.  
(*type=*`numpy.ndarray`)

#### Raises

**NotGoodPosition** This exception is used to indicate that the position `x` is not valid. Fixer has the option of either mapping invalid parameter vectors into valid ones or raising this exception.

Overrides: `gmisc.lib.mcmc.problem_definition.fixer`

**Attention:** Within a convex region (presumably one that contains the optimal `x`), `fixer` must *not* change the value of `logp()`: `logp(fixer(x)) == logp(x)`.

**`log(self, p, i)`**

Some code calls this function every iteration to log the current state of the MCMC process.

**Parameters**

`p`: the current parameter vector, and  
`i`: an integer iteration counter.

**Return Value**

nothing.

***Inherited from object***

`__delattr__()`, `__format__()`, `__getattr__()`, `__hash__()`, `__new__()`, `__reduce__()`, `__reduce_ex__()`, `__repr__()`, `__setattr__()`, `__sizeof__()`, `__str__()`, `__subclasshook__()`

**71.4.2 Properties**

Name	Description
<i>Inherited from object</i>	
<code>__class__</code>	

**71.5 Class `position_base`**

object —  
**`gmisclib.mcmc.position_base`**

**Known Subclasses:** `gmisclib.mcmc.position_nonrepeatable`, `gmisclib.mcmc.position_repeatable`

This class is used internally in the MCMC sampling process to represent a position. It stores a parameter vector, a reference to the problem definition, and (optionally) caches computed values of  $\log(P)$ .

**71.5.1 Methods**

**`__init__(self, x, problem_def)`**

You need this function and this signature if you are going to pass it to `make_list_of_positions()`, but not necessarily otherwise.

Overrides: `object.__init__`

---

**logp**(*self*)

---

 Compute the log of the probability for the position.

---

**logp\_nocompute**(*self*)

---

 Shows a recent logp value. It should not compute a value unless necessary. The intent is to look up a value in a cache.

---

**new**(*self*, *shift*, *logp*=None)

---

 Returns a new **position**, shifted by the specified amount.

**Parameters**

**shift**: How much of a move to make to the new position?

(*type*=*numpy.ndarray*)

**logp**: (optional) If this is supplied, it is used to set the **log(P)** value for the newly created position structure.

(*type*=*float* or *None*)

**Return Value**

a new position

(*type*=**position\_base** or a subclass)

---

**prms**(*self*)

---

 The result of this function is to be handed to `problem_definition.logp()`. This result must contain all the information specifying the position. Normally, this is a vector of floats, but conceivably, you could include other information.

---

**vec**(*self*)

---

 Returns a numpy vector. The result should contain all the information specifying the position; if not all, it should at least contain all the information that can be usefully expressed as a vector of floating point numbers.

Normally, **vec()** and **prms()** are identical.

---

**\_\_repr\_\_**(*self*)

**repr**(x)

Overrides: `object.__repr__` `exitit`(inherited documentation)

---

**\_\_cmp\_\_**(*self*, *other*)

---

 This is used when the archive is sorted.

**uid**(*self*)

*Inherited from `object`*

`__delattr__()`, `__format__()`, `__getattribute__()`, `__hash__()`, `__new__()`, `__reduce__()`, `__reduce_ex__()`,  
`__setattr__()`, `__sizeof__()`, `__str__()`, `__subclasshook__()`

### 71.5.2 Properties

Name	Description
<i>Inherited from <code>object</code></i>	
<code>__class__</code>	

## 71.6 Class `position_repeatable`



This is for the common case where `logp` is a well-behaved function of its arguments. It caches positions and their corresponding values of  $\log(P)$ .

### 71.6.1 Methods

**`__init__`**(*self*, *x*, *problem\_def*, *logp*=None)

You need this function and this signature if you are going to pass it to `make_list_of_positions()`, but not necessarily otherwise.

#### Parameters

**`x`:** *(type=*`numpy.ndarray`*)*

**`logp`:** the value of  $\log(P)$  at *x*, or `None` to indicate that it hasn't been computed yet.  
*(type=*`float` *or* `None`*)*

**`problem_def`:** *(type=***`problem_definition`** *or a subclass thereof.)*

#### Raises

`ValueError` if sanity check is failed. @raise `NotGoodPosition`: from inside `problem_definition.logp`.

Overrides: `object.__init__`

**invalidate\_cache**(*self*)

This can be called when the mapping between parameters (*x*) and value changes. You might use it if you wanted to change the probability distribution (i.e.  $\log(P)$ ).

**logp**(*self*)

Compute the log of the probability for the position.

Overrides: *gmisclib.mcmc.position\_base.logp* extit(inherited documentation)

**logp\_nocompute**(*self*)

Shows a recent logp value. It should not compute a value unless necessary. The intent is to look up a value in a cache.

Overrides: *gmisclib.mcmc.position\_base.logp\_nocompute* extit(inherited documentation)

**new**(*self*, *shift*, *logp*=None)

Returns a new position, shifted by the specified amount.

**Parameters**

**shift**: How much of a move to make to the new position?

**logp**: (optional) If this is supplied, is is used to set the  $\log(P)$  value for the newly created position structure.

**Return Value**

a new position

(*type=position\_base or a subclass*)

Overrides: *gmisclib.mcmc.position\_base.new*

**\_\_repr\_\_**(*self*)

repr(*x*)

Overrides: *object.\_\_repr\_\_* extit(inherited documentation)

**\_\_cmp\_\_**(*self*, *other*)

This is used when the archive is sorted.

**prms(*self*)**

The result of this function is to be handed to `problem_definition.logp()`. This result must contain all the information specifying the position. Normally, this is a vector of floats, but conceivably, you could include other information.

**uid(*self*)****vec(*self*)**

Returns a numpy vector. The result should contain all the information specifying the position; if not all, it should at least contain all the information that can be usefully expressed as a vector of floating point numbers.

Normally, `vec()` and `prms()` are identical.

***Inherited from object***

`__delattr__()`, `__format__()`, `__getattribute__()`, `__hash__()`, `__new__()`, `__reduce__()`, `__reduce_ex__()`, `__setattr__()`, `__sizeof__()`, `__str__()`, `__subclasshook__()`

**71.6.2 Properties**

Name	Description
<i>Inherited from object</i>	
<code>__class__</code>	

**71.6.3 Class Variables**

Name	Description
EPS	When we check for reproducibility, how closely must answers match? <b>Value:</b> 1e-07
HUGE	<b>Value:</b> 1e+38
CACHE_LIFE	<b>Value:</b> 50
FIXER_CHECK	How often should we recompute cached values to make sure that the same parameters always lead to the same results? <b>Value:</b> 100

## 71.7 Class `position_nonrepeatable`



This is for the (unfortunately common) case where `logp` is an independent random function of its arguments. It does not cache as much as `position_repeatable`. Arguments are analogous to `position_repeatable`.

**Note:** I think this class is obsolete, given the call to `T_acceptor.probe()` that is not inserted in to `BootStepper.step()`.

### 71.7.1 Methods

**`__init__(self, x, problem_def, logp=None)`**

You need this function and this signature if you are going to pass it to `make_list_of_positions()`, but not necessarily otherwise.

Overrides: `object.__init__` extit(inherited documentation)

**`logp(self)`**

Compute the log of the probability for the position.

Overrides: `gmisclib.mcmc.position_base.logp` extit(inherited documentation)

**`logp_nocompute(self)`**

Shows a recent `logp` value. It should not compute a value unless necessary. The intent is to look up a value in a cache.

Overrides: `gmisclib.mcmc.position_base.logp_nocompute` extit(inherited documentation)

**new**(*self*, *shift*, *logp*=None)

Returns a new position, shifted by the specified amount.

**Parameters**

**shift**: How much of a move to make to the new position?  
**logp**: (optional) If this is supplied, is is used to set the **log(P)** value for the newly created position structure.

**Return Value**

a new position  
*(type=position\_base or a subclass)*

Overrides: *gmisclib.mcmc.position\_base.new*

**\_\_repr\_\_**(*self*)

*repr*(x)

Overrides: *object.\_\_repr\_\_* *exitit*(inherited documentation)

**\_\_cmp\_\_**(*self*, *other*)

This is used when the archive is sorted.

**prms**(*self*)

The result of this function is to be handed to *problem\_definition.logp()*. This result must contain all the information specifying the position. Normally, this is a vector of floats, but conceivably, you could include other information.

**uid**(*self*)

**vec**(*self*)

Returns a numpy vector. The result should contain all the information specifying the position; if not all, it should at least contain all the information that can be usefully expressed as a vector of floating point numbers.

Normally, *vec()* and *prms()* are identical.

**Inherited from object**

*\_\_delattr\_\_()*, *\_\_format\_\_()*, *\_\_getattr\_\_()*, *\_\_hash\_\_()*, *\_\_new\_\_()*, *\_\_reduce\_\_()*, *\_\_reduce\_ex\_\_()*, *\_\_setattr\_\_()*, *\_\_sizeof\_\_()*, *\_\_str\_\_()*, *\_\_subclasshook\_\_()*

**71.7.2 Properties**



Name	Description
<i>Inherited from object</i>	
<code>--class--</code>	

### 71.7.3 Class Variables

Name	Description
HUGE	<b>Value:</b> 1e+38

## 71.8 Class `acceptor_base`



**Known Subclasses:** `gmisclib.mcmc.T_acceptor`, `gmisclib.mcmc.rough_acceptor_base`

### 71.8.1 Methods

<code>--init--(<i>self</i>)</code>
<code>x.__init__(...)</code> initializes <code>x</code> ; see <code>help(type(x))</code> for signature
Overrides: <code>object.__init__</code> <code>extit</code> (inherited documentation)

<code>T(<i>self</i>)</code>
<b>Return Value</b>
A decent approximation to the system temperature.
( <i>type=</i> <code>float</code> )

---

**\_\_call\_\_**(*self*, *delta*)

---

Accept a step or not?

**Parameters**

**delta**: The proposed step gives **delta** as the change in  $\log(\text{probability})$ .

(*type=**float*)

**Return Value**

should it be accepted or not?

(*type=**bool*)

---



---

**probe**(*self*, *currentpos*, *stepper*)

---

This is a hook for a special case where you are optimizing a random function or one that is deterministic, but very rough, and you don't care about the small-scale structure. In such a case, you might use this to probe the local variability and use that knowledge to change the temperature.

---



---

**jitter**(*self*)

---

In the acoustic distances paper, it is called  $\bar{\delta}$ .

**Return Value**

An estimate of how rough the function is, on a small scale.

(*type=**float*)

---



---

**reset**(*self*)

---

In order to make the overall algorithm asymptotically a Markovian, we need to make sure that (asymptotically) the acceptor function depends only on long-term averages, not on recent history. The problem is the jitter measurement, of course. So, one needs to average the jitter over a time longer than the recurrence time, asymptotically. But that makes it horribly unresponsive in the early stages of the optimization when things are rapidly changing. The solution is to pass along resets, so that it can (temporarily) revert to a short averaging time.

---

***Inherited from object***

`__delattr__()`, `__format__()`, `__getattr__()`, `__hash__()`, `__new__()`, `__reduce__()`, `__reduce_ex__()`, `__repr__()`, `__setattr__()`, `__sizeof__()`, `__str__()`, `__subclasshook__()`

**71.8.2 Properties**

Name	Description
<i>Inherited from object</i>	
<code>__class__</code>	

### 71.8.3 Instance Variables

Name	Description
<code>last_probe</code>	A pair of <code>position_base</code> instances showing the position and logP at each end of the most recent probe for a discontinuity. Or, it could be None. This value is not used in <code>mcmc.py</code> . Rather, it is there for the logging modules.

## 71.9 Class `T_acceptor`



This class implements a normal Metropolis-Hastings acceptance of steps.

### 71.9.1 Methods

<b><code>__init__(self, T=1.0)</code></b>
You can change the temperature to do simulated annealing.
<b>Parameters</b>
<b>T</b> : temperature
( <i>type=float</i> )
Overrides: <code>object.__init__</code>

<b><code>T(self)</code></b>
<b>Return Value</b>
the system temperature.
( <i>type=float</i> )
Overrides: <code>gmislib.mcmc.acceptor_base.T</code>

---

**\_\_call\_\_(self, delta)**


---

Accept a step or not?

**Parameters**

**delta:** The proposed step gives **delta** as the change in *log(probability)*.  
(*type=float*)

**Return Value**

should it be accepted or not?  
(*type=bool*)

Overrides: *gmisclib.mcmc.acceptor\_base.\_\_call\_\_*

---

**jitter(self)**


---

In the acoustic distances paper, it is called  $\bar{\delta}$ .

**Return Value**

An estimate of how rough the function is, on a small scale.  
(*type=float*)

---

**probe(self, currentpos, stepper)**


---

This is a hook for a special case where you are optimizing a random function or one that is deterministic, but very rough, and you don't care about the small-scale structure. In such a case, you might use this to probe the local variability and use that knowledge to change the temperature.

---

**reset(self)**


---

In order to make the overall algorithm asymptotically a Markovian, we need to make sure that (asymptotically) the acceptor function depends only on long-term averages, not on recent history. The problem is the jitter measurement, of course. So, one needs to average the jitter over a time longer than the recurrence time, asymptotically. But that makes it horribly unresponsive in the early stages of the optimization when things are rapidly changing. The solution is to pass along resets, so that it can (temporarily) revert to a short averaging time.

***Inherited from object***

*\_\_delattr\_\_()*, *\_\_format\_\_()*, *\_\_getattr\_\_()*, *\_\_hash\_\_()*, *\_\_new\_\_()*, *\_\_reduce\_\_()*, *\_\_reduce\_ex\_\_()*, *\_\_repr\_\_()*, *\_\_setattr\_\_()*, *\_\_sizeof\_\_()*, *\_\_str\_\_()*, *\_\_subclasshook\_\_()*

**71.9.2 Properties**

Name	Description
<i>Inherited from object</i> <code>--class--</code>	

### 71.9.3 Instance Variables

Name	Description
<code>last_probe</code>	A pair of <code>position_base</code> instances showing the position and <code>logP</code> at each end of the most recent probe for a discontinuity. Or, it could be <code>None</code> . This value is not used in <code>mcmc.py</code> . Rather, it is there for the logging modules.

## 71.10 Class `rough_acceptor_base`



**Known Subclasses:** `gmislib.mcmc.rough_T_acceptor`, `gmislib.mcmc_helper.step_acceptor`

### 71.10.1 Methods

<b><code>probe(self, currentpos, stepper)</code></b>
This computes <code>logP()</code> near the current step position, and sees how much that differs from <code>logP()</code> <code>_at_</code> the current step position. That difference is used as a measure of the "roughness" or "randomness" of the function. That extra roughness is added onto the temperature to ensure that the MCMC stepper doesn't get trapped in an unimportant local valley. Essentially, we are making the statement that any differences within the region covered by <code>probestep</code> are unimportant.
Overrides: <code>gmislib.mcmc.acceptor_base.probe</code>

**jitter**(*self*)

In the acoustic distances paper, it is called  $\bar{\delta}$ .

**Return Value**

An estimate of how rough the function is, on a small scale.

(*type=**float*)

Overrides: *gmisclib.mcmc.acceptor\_base.jitter* *exitit*(inherited documentation)

**reset**(*self*)

In order to make the overall algorithm asymptotically a Markovian, we need to make sure that (asymptotically) the acceptor function depends only on long-term averages, not on recent history. The problem is the jitter measurement, of course. So, one needs to average the jitter over a time longer than the recurrence time, asymptotically. But that makes it horribly unresponsive in the early stages of the optimization when things are rapidly changing. The solution is to pass along resets, so that it can (temporarily) revert to a short averaging time.

Overrides: *gmisclib.mcmc.acceptor\_base.reset* *exitit*(inherited documentation)

**\_\_init\_\_**(*self*, *probestep*, *tau*, *droop*)

*x.\_\_init\_\_*(...) initializes *x*; see *help*(*type*(*x*)) for signature

Overrides: *object.\_\_init\_\_* *exitit*(inherited documentation)

**T**(*self*)**Return Value**

A decent approximation to the system temperature.

(*type=**float*)

**\_\_call\_\_**(*self*, *delta*)

Accept a step or not?

**Parameters**

**delta**: The proposed step gives **delta** as the change in *log(probability)*.

(*type=**float*)

**Return Value**

should it be accepted or not?

(*type=**bool*)

***Inherited from object***

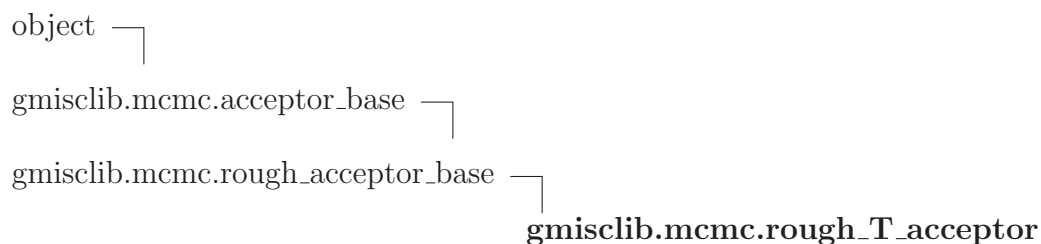
`__delattr__()`, `__format__()`, `__getattr__()`, `__hash__()`, `__new__()`, `__reduce__()`, `__reduce_ex__()`, `__repr__()`, `__setattr__()`, `__sizeof__()`, `__str__()`, `__subclasshook__()`

**71.10.2 Properties**

Name	Description
<i>Inherited from object</i>	
<code>__class__</code>	

**71.10.3 Instance Variables**

Name	Description
<code>last_probe</code>	A pair of <code>position_base</code> instances showing the position and logP at each end of the most recent probe for a discontinuity. Or, it could be None. This value is not used in <code>mcmc.py</code> . Rather, it is there for the logging modules.
<code>since_last_reset</code>	Number of iterations since the last overall MCMC reset.

**71.11 Class `rough_T_acceptor`**

This class implements a normal Metropolis-Hastings acceptance of steps.

**71.11.1 Methods**


---

**`__init__`**(*self*, *probestep*, *tau\_probe*=None, *T*=1.0, *jitterdroop*=None)

---

You can change the temperature to do simulated annealing.

**Parameters**

**T:** temperature  
(*type*=float)

**probestep:** something callable that generates small offsets within a voxel. This is called with the currently active stepper as an argument.

(*type*=function(*stepper*) -> *numpy.ndarray*  
(*length*=*BootStepper.np*).)

Overrides: `object.__init__`

---

**`T`**(*self*)

---

**Return Value**

the system temperature.

(*type*=float)

Overrides: `gmisclib.mcmc.acceptor_base.T`

---

**`__call__`**(*self*, *delta*)

---

Accept a step or not?

**Parameters**

**delta:** The proposed step gives **delta** as the change in *log(probability)*.  
(*type*=float)

**Return Value**

should it be accepted or not?

(*type*=bool)

Overrides: `gmisclib.mcmc.acceptor_base.__call__`

---



**jitter**(*self*)

In the acoustic distances paper, it is called  $\bar{\delta}$ .

**Return Value**

An estimate of how rough the function is, on a small scale.

(*type=float*)

Overrides: *gmisclib.mcmc.acceptor\_base.jitter* *exitit*(inherited documentation)

**probe**(*self*, *currentpos*, *stepper*)

This computes  $\log P()$  near the current step position, and sees how much that differs from  $\log P()$  at the current step position. That difference is used as a measure of the "roughness" or "randomness" of the function. That extra roughness is added onto the temperature to ensure that the MCMC stepper doesn't get trapped in an unimportant local valley. Essentially, we are making the statement that any differences within the region covered by **probestep** are unimportant.

Overrides: *gmisclib.mcmc.acceptor\_base.probe*

**reset**(*self*)

In order to make the overall algorithm asymptotically a Markovian, we need to make sure that (asymptotically) the acceptor function depends only on long-term averages, not on recent history. The problem is the jitter measurement, of course. So, one needs to average the jitter over a time longer than the recurrence time, asymptotically. But that makes it horribly unresponsive in the early stages of the optimization when things are rapidly changing. The solution is to pass along resets, so that it can (temporarily) revert to a short averaging time.

Overrides: *gmisclib.mcmc.acceptor\_base.reset* *exitit*(inherited documentation)

**Inherited from object**

*\_\_delattr\_\_()*, *\_\_format\_\_()*, *\_\_getattribute\_\_()*, *\_\_hash\_\_()*, *\_\_new\_\_()*, *\_\_reduce\_\_()*, *\_\_reduce\_ex\_\_()*, *\_\_repr\_\_()*, *\_\_setattr\_\_()*, *\_\_sizeof\_\_()*, *\_\_str\_\_()*, *\_\_subclasshook\_\_()*

**71.11.2 Properties**

Name	Description
<i>Inherited from object</i>	
<i>__class__</i>	

**71.11.3 Instance Variables**

Name	Description
<code>last_probe</code>	A pair of <code>position_base</code> instances showing the position and logP at each end of the most recent probe for a discontinuity. Or, it could be None. This value is not used in <code>mcmc.py</code> . Rather, it is there for the logging modules.
<code>since_last_reset</code>	Number of iterations since the last overall MCMC reset.

**71.12 Class *stepper***

object └─ **`gmisclib.mcmc.stepper`**

**Known Subclasses:** `gmisclib.mcmc.BootStepper`

This is your basic stepper class. It's a base class containing common features; it is not used directly.

**71.12.1 Methods**

**`__init__(self)`**

`x.__init__(...)` initializes `x`; see `help(type(x))` for signature

Overrides: `object.__init__` `extit`(inherited documentation)

**`step(self)`**

In subclasses, this takes a step and returns 0 or 1, depending on whether the step was accepted or not.

**`prms(self)`**

**Return Value**

The current parameters

(*type*=`numpy.ndarray`)

**`status(self)`**

Provides some printable status information in `a=v;` format.

**reset**(*self*)

Called internally to mark when the optimization has found a new minimum.

**Note:** You might also call it if the function you are minimizing changes.

**reset\_id**(*self*)

Use this to tell if the stepper has been reset since you last looked at it.

**Return Value**

an integer that's different for each reset.

(*type=int*)

**needs\_a\_reset**(*self*)

Decides if we we need a reset. This checks to see if we have a new  $\log P$  that exceeds the old record. It keeps track of the necessary paperwork.

**current**(*self*)**Return Value**

the current position.

(*type=position\_base*)

**Inherited from object**

`__delattr__()`, `__format__()`, `__getattr__()`, `__hash__()`, `__new__()`, `__reduce__()`, `__reduce_ex__()`, `__repr__()`, `__setattr__()`, `__sizeof__()`, `__str__()`, `__subclasshook__()`

**71.12.2 Properties**

Name	Description
<i>Inherited from object</i>	
<code>__class__</code>	

**71.12.3 Instance Variables**

Name	Description
<code>acceptable</code>	Acceptable is a function that decides whether or not a step is OK. You can replace it if you want, but the class should be equivalent to <code>T_acceptor</code> .

*continued on next page*

Name	Description
<code>last_failed</code>	It should reflect the success or failure of the most recently completed step. It is <code>None</code> if the last step succeeded; it is the most recently rejected <code>position_base</code> object if the last step failed. This is intended for debugging mostly. It is not used by code in this module.

### 71.13 Class *adjuster*

object —  
     **gmisclib.mcmc.adjuster**

The `adjuster` class controls the step size.

#### 71.13.1 Methods

**`__init__(self, F, vscale, vse=0.0, vmax=1e+30)`**

`x.__init__(...)` initializes `x`; see `help(type(x))` for signature

Overrides: `object.__init__` `extit`(inherited documentation)

**`reset(self)`**

Called when old habits should be forgotten.

**`f(self)`**

This allows the desired fraction of accepted steps to depend on `self.vscale`.

**Requires:** This method should only be called with `self.vse != 0` where `self.vscale` can normally be expected to be fairly close to unity, and where small values of `self.vse` indicate trouble. In the `mcmc.Bootstepper.step_boot` case this is true.

**`inctry(self, accepted)`**

**vs(*self*)**

We stick in the factor of `random.lognormvariate()` so that all sizes of move are possible and thus we can prove that we can random-walk to any point in a connected region. This makes the proof of ergodicity simpler.

**Return Value**

a scale factor for the step size.

(*type=float*)

**status(*self*)****Return Value**

misc. status information for debugging purposes.

***Inherited from object***

`__delattr__()`, `__format__()`, `__getattr__()`, `__hash__()`, `__new__()`, `__reduce__()`, `__reduce_ex__()`, `__repr__()`, `__setattr__()`, `__sizeof__()`, `__str__()`, `__subclasshook__()`

**71.13.2 Properties**

Name	Description
<i>Inherited from object</i>	
<code>__class__</code>	

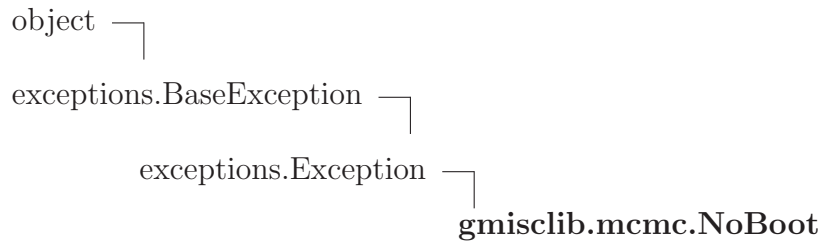
**71.13.3 Class Variables**

Name	Description
Z0	<b>Value:</b> 1.5
DZ	<b>Value:</b> 0.3
TOL	<b>Value:</b> 0.2
STABEXP	<b>Value:</b> 1.0

**71.13.4 Instance Variables**

Name	Description
<code>vsmax</code>	Used when the acceptance probability is larger than 25%. Large acceptance probabilities can happen if the probability is everywhere about equal. (E.g. a data fitting problem with almost no data)

## 71.14 Class NoBoot



This is used internally to signify that some particular method of selecting a step has decided that it is not suitable. For instance, a method that depends on an archive of previous steps might see that the archive isn't long enough.

### 71.14.1 Methods

**`--init--`**(*self*, \**s*)

`x.--init--(...)` initializes `x`; see `help(type(x))` for signature

Overrides: `object.--init--` extit(inherited documentation)

*Inherited from exceptions.Exception*

`--new--()`

*Inherited from exceptions.BaseException*

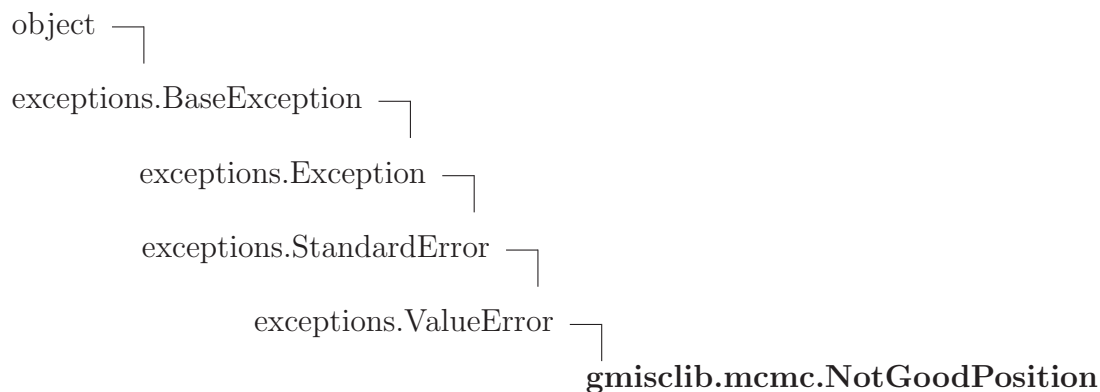
`--delattr--()`, `--getattr--()`, `--getitem--()`, `--getslice--()`, `--reduce--()`, `--repr--()`,  
`--setattr--()`, `--setstate--()`, `--str--()`, `--unicode--()`

*Inherited from object*

`--format--()`, `--hash--()`, `--reduce.ex--()`, `--sizeof--()`, `--subclasshook--()`

### 71.14.2 Properties

Name	Description
<i>Inherited from exceptions.BaseException</i>	
<code>args</code> , <code>message</code>	
<i>Inherited from object</i>	
<code>--class--</code>	

**71.15 Class NotGoodPosition**

This is raised by user code (e.g. in the `problem_definition.logP` method or in `problem_definition.fix`) to indicate that the current position is not computable or is unacceptable in some other way. This is assumed to be a permanent problem: i.e. the mathematical model blows up for this set of parameters.

**71.15.1 Methods**

```
__init__(self, *s)
```

`x.__init__(...)` initializes `x`; see `help(type(x))` for signature

Overrides: `object.__init__` extit(inherited documentation)

```
add_args(self, *s)
```

Typical usage:

```

try:
    something
except NotGoodPosition, ex:
    raise ex.add_args(more, args, added, here)
  
```

**Parameters**

**s**: a tuple of arguments to add onto the tail of the exception's args list.

**Return Value**

The modified exception.

(*type=NotGoodPosition*)

**Note:** This modifies the original object and returns it.

*Inherited from exceptions.ValueError*

`__new__()`

*Inherited from `exceptions.BaseException`*

`__delattr__()`, `__getattr__()`, `__getitem__()`, `__getslice__()`, `__reduce__()`, `__repr__()`,  
`__setattr__()`, `__setstate__()`, `__str__()`, `__unicode__()`

*Inherited from `object`*

`__format__()`, `__hash__()`, `__reduce_ex__()`, `__sizeof__()`, `__subclasshook__()`

### 71.15.2 Properties

Name	Description
<i>Inherited from <code>exceptions.BaseException</code></i>	
<code>args</code> , <code>message</code>	
<i>Inherited from <code>object</code></i>	
<code>__class__</code>	

## 71.16 Class `hashcounter_c`



Keep a count of how many times we've see various items.

### 71.16.1 Methods

**`incr(self, x)`**

**`decr(self, x)`**

**Raises**

`ValueError` if you ever see something less than zero times.

*Inherited from `dict`*

`__cmp__()`, `__contains__()`, `__delitem__()`, `__eq__()`, `__ge__()`, `__getattr__()`, `__getitem__()`,  
`__gt__()`, `__init__()`, `__iter__()`, `__le__()`, `__len__()`, `__lt__()`, `__ne__()`, `__new__()`, `__repr__()`,  
`__setitem__()`, `__sizeof__()`, `clear()`, `copy()`, `fromkeys()`, `get()`, `has_key()`, `items()`,



iteritems(), iterkeys(), itervalues(), keys(), pop(), popitem(), setdefault(), update(), values(), viewitems(), viewkeys(), viewvalues()

### *Inherited from object*

`__delattr__()`, `__format__()`, `__reduce__()`, `__reduce_ex__()`, `__setattr__()`, `__str__()`, `__subclasshook__()`

#### 71.16.2 Properties

Name	Description
<i>Inherited from object</i>	
<code>__class__</code>	

#### 71.16.3 Class Variables

Name	Description
<i>Inherited from dict</i>	
<code>__hash__</code>	

### 71.17 Class Archive

object └─ **gmisclib.mcmc.Archive**

**Known Subclasses:** gmisclib.mcmc.ContPrmArchive

This maintains a list of all the recent accepted positions.

#### 71.17.1 Methods

<b><code>__init__(self, lop, np_eff, strategy='intermediate', maxArchSize=None, alpha=None)</code></b> <code>x.__init__(...)</code> initializes x; see <code>help(type(x))</code> for signature Overrides: <code>object.__init__</code> <code>exitit</code> (inherited documentation)
<b><code>distinct_count(self)</code></b> How many distinct values of the parameters are there in the archive?
<b><code>prmlist(self, n)</code></b>

---

**sort**(*self*)

---

Called under lock. Sort the archive into order of `logp`.

---

**reset**(*self*)

---

We sort the archive to speed the convergence to the best solution. After all, if you've just gotten a reset, it is likely that you're not at the bottom yet, so statistical properties of the distribution are likely to be irrelevant.

---

**\_\_len\_\_**(*self*)

---



---

**choose**(*self*)

---



---

**truncate**(*self*, *desired\_length*)

---

Shortens the archive and updates the various counters and statistics.

**Parameters**

**desired\_length**: Measured in terms of the number of distinct positions.

**Note**: Must be called under lock.

---

**append**(*self*, *x*, *maxdups*)

---

Adds stuff to the archive, possibly sorting the new information into place. It updates all kinds of counters and summary statistics.

**Parameters**

**x**: A position to (possibly) add to the archive.

(*type=position\_base*)

**Return Value**

A one-letter code indicating what happened. 'f' if x is a duplicate and duplicates are forbidden. 'd' if it is a duplicate and there have been too many duplicates lately. 'a' otherwise – x has been added to the archive.

(*type=str*)

---

**append\_hook**(*self*, *x*)

---



---

**truncate\_hook**(*self*, *to\_be\_dropped*)

---

**Inherited from object**

`__delattr__()`, `__format__()`, `__getattr__()`, `__hash__()`, `__new__()`, `__reduce__()`, `__reduce_ex__()`,

`__repr__()`, `__setattr__()`, `__sizeof__()`, `__str__()`, `__subclasshook__()`

### 71.17.2 Properties

Name	Description
<i>Inherited from object</i>	
<code>__class__</code>	

### 71.17.3 Class Variables

Name	Description
SUPHILL	Always keep the list of positions sorted. This improves minimization performance at the cost of a distorted probability distribution. <b>Value:</b> <code>'hillclimb'</code>
SSAMPLE	Never sort the list. Best if you really want an exact Monte Carlo distribution. <b>Value:</b> <code>'sample'</code>
SANNEAL	Sort the list only when <code>logp()</code> is making substantial improvements. <b>Value:</b> <code>'intermediate'</code>
Sfac	<b>Value:</b> <code>{'hillclimb': 100, 'intermediate': 5, 'sample': 2}</code>

### 71.17.4 Instance Variables

Name	Description
<code>min_l</code>	The minimum length for the archive. This is chosen to be big enough so that all the parallel copies probably span the full parameter space.

## 71.18 Class ContPrmArchive



**71.18.1 Methods**

**`__init__(self, lop, np_eff, strategy='intermediate', maxArchSize=None, alpha=None)`**

`Append_hook()` is called for every element of the archive. That function can be replaced in a sub-class to accumulate some kind of summary. Here, it is used to keep track of parameter means and standard deviations.

Overrides: `object.__init__`

**`append_hook(self, x)`**

This accumulates parameter means and standard deviations.

Overrides: `gmisclib.mcmc.Archive.append_hook`

**`truncate_hook(self, to_be_dropped)`**

Overrides: `gmisclib.mcmc.Archive.truncate_hook`

**`variance(self)`**

**`__len__(self)`**

**`append(self, x, maxdups)`**

Adds stuff to the archive, possibly sorting the new information into place. It updates all kinds of counters and summary statistics.

**Parameters**

**x**: A position to (possibly) add to the archive.

(*type=position\_base*)

**Return Value**

A one-letter code indicating what happened. 'f' if x is a duplicate and duplicates are forbidden. 'd' if it is a duplicate and there have been too many duplicates lately. 'a' otherwise – x has been added to the archive.

(*type=str*)

**`choose(self)`**

**`distinct_count(self)`**

How many distinct values of the parameters are there in the archive?

<b>prmlist</b> ( <i>self</i> , <i>n</i> )
---

<b>reset</b> ( <i>self</i> )
------------------------------

We sort the archive to speed the convergence to the best solution. After all, if you've just gotten a reset, it is likely that you're not at the bottom yet, so statistical properties of the distribution are likely to be irrelevant.
---

<b>sort</b> ( <i>self</i> )
-----------------------------

Called under lock. Sort the archive into order of <b>logp</b> .
---

<b>truncate</b> ( <i>self</i> , <i>desired_length</i> )
---

Shortens the archive and updates the various counters and statistics.
---

<b>Parameters</b>
-------------------

<b>desired_length</b> : Measured in terms of the number of distinct positions.
--

<b>Note:</b> Must be called under lock.
---

**Inherited from object**

`__delattr__()`, `__format__()`, `__getattr__()`, `__hash__()`, `__new__()`, `__reduce__()`, `__reduce_ex__()`, `__repr__()`, `__setattr__()`, `__sizeof__()`, `__str__()`, `__subclasshook__()`

**71.18.2 Properties**

Name	Description
<i>Inherited from object</i>	
<code>__class__</code>	

**71.18.3 Class Variables**

Name	Description
SANNEAL	Sort the list only when <code>logp()</code> is making substantial improvements. <b>Value:</b> <code>'intermediate'</code>
SSAMPLE	Never sort the list. Best if you really want an exact Monte Carlo distribution. <b>Value:</b> <code>'sample'</code>

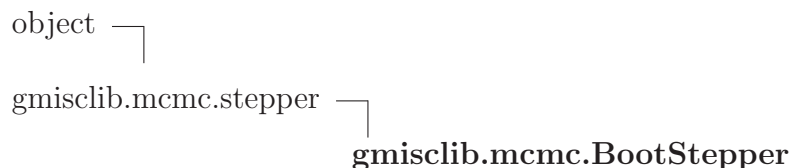
*continued on next page*

Name	Description
SUPHILL	Always keep the list of positions sorted. This improves minimization performance at the cost of a distorted probability distribution. <b>Value:</b> 'hillclimb'
Sfac	<b>Value:</b> {'hillclimb': 100, 'intermediate': 5, 'sample': 2}

#### 71.18.4 Instance Variables

Name	Description
min_l	The minimum length for the archive. This is chosen to be big enough so that all the parallel copies probably span the full parameter space.

### 71.19 Class *BootStepper*



**Known Subclasses:** *gmisclib.mcmc.big.BootStepper*

The *BootStepper* class is the primary interface. It implements a Bootstrap Markov Chain Monte Carlo stepper. The class instance stores the necessary state information, and each call to **step()** takes another step.

**71.19.1 Methods**

**`__init__(self, lop, v, strategy='intermediate', maxArchSize=None, parallelSizeDiv=1)`**

`x.__init__(...)` initializes `x`; see `help(type(x))` for signature

**Parameters**

- `maxArchSize`:** How many position vectors can be stored. This is normally used to (loosely) enforce a memory limitation for large jobs.
- `parallelSizeDiv`:** For use when there are several cooperating MCMC processes that share data. When  $>1$ , this allows each process to have smaller stored lists. Normally, `parallelSizeDiv` is between 1 and the number of cooperating processes.
- `lop`:** These are a list of starting postions that should be evaluated first and used as the basis for further exploration.  
*(type=list of a class derived from position\_base.)*
- `v`:** A covariance matrix for non-bootstrap steps. Mostly, the system takes bootstrap steps that are adaptively derived from previous steps. However, to generate the list of previous steps, it needs a way to generate steps ab initio. So, `v` is used to get things going. `v` is also used occasionally thereafter, just to make sure the stepper doesn't get trapped in some subspace.
- `strategy`:** *(type=One of SSAMPLE, SUPHILL, or SANNEAL.)*

Overrides: `object.__init__`

**`step(self)`**

In subclasses, this takes a step and returns 0 or 1, depending on whether the step was accepted or not.

Overrides: `gmislib.mcmc.stepper.step` `exitit`(inherited documentation)

**`status(self)`**

Provides some printable status information in `a=v;` format.

Overrides: `gmislib.mcmc.stepper.status` `exitit`(inherited documentation)

**stepV**(*self*)

**step\_boot**(*self*)

**step\_probe**(*self*)

**reset\_adjusters**(*self*)

**ergodic**(*self*)

A crude measure of how ergodic the MCMC is.

**Return Value**

The inverse of how many steps it takes to cross the minimum in the slowest direction. Or zero, if it is too soon after some major violation of the MCMC assumptions.

**set\_strategy**(*self*, *ss*)

**set\_sort\_strategy**(*self*, *ss*)

**current**(*self*)

**Return Value**

the current position.

(*type=position\_base*)

**needs\_a\_reset**(*self*)

Decides if we we need a reset. This checks to see if we have a new  $\log P$  that exceeds the old record. It keeps track of the necessary paperwork.

**prms**(*self*)

**Return Value**

The current parameters

(*type=numpy.ndarray*)

**reset**(*self*)

Called internally to mark when the optimization has found a new minimum.

**Note:** You might also call it if the function you are minimizing changes.



**reset\_id**(*self*)

Use this to tell if the stepper has been reset since you last looked at it.

**Return Value**

an integer that's different for each reset.

(*type=int*)

**Inherited from object**

`__delattr__()`, `__format__()`, `__getattribute__()`, `__hash__()`, `__new__()`, `__reduce__()`, `__reduce_ex__()`, `__repr__()`, `__setattr__()`, `__sizeof__()`, `__str__()`, `__subclasshook__()`

**71.19.2 Properties**

Name	Description
<i>Inherited from object</i>	
<code>__class__</code>	

**71.19.3 Class Variables**

Name	Description
F	F is the targeted step acceptance rate. This is from G. O. Roberts, A. Gelman, and W. Gilks (1997) "Weak convergence and optimal scaling of random walk Metropolis algorithm." Ann. Applied. Probability, 7, p. 110-120 and also from G. O. Roberts and J. S. Rosenthal (2001) "Optimal scaling of various Metropolis-Hastings algorithms." Statistical Sci. 16, pp. 351-367. <b>Value:</b> 0.234
alpha	How rapidly should one expand the archive after a reset? <b>Value:</b> 0.1
PBootLim	PBootLim Limits the probability of taking a bootstrap step. This, if the optimization collapses into a subspace, some other kind of step will eventually get it out. <b>Value:</b> 0.9
SSAMPLE	Sampling mode <b>Value:</b> 'sample'
SUPHILL	Go straight uphill <b>Value:</b> 'hillclimb'

*continued on next page*

Name	Description
SANNEAL	Simulated annealing <b>Value:</b> 'intermediate'
SSAUTO	<b>Value:</b> 'intermediate'
SSNEVER	<b>Value:</b> 'sample'
SSLOW	<b>Value:</b> 'intermediate'
SSALWAYS	<b>Value:</b> 'hillclimb'

#### 71.19.4 Instance Variables

Name	Description
np	The number of parameters:
np_eff	In a multiprocessor situation, np_eff tells you how much data do you need to store locally, so that the overall group of processors stores enough variety of data.
acceptable	Acceptable is a function that decides whether or not a step is OK. You can replace it if you want, but the class should be equivalent to <code>T_acceptor</code> .
last_failed	It should reflect the success or failure of the most recently completed step. It is None if the last step succeeded; it is the most recently rejected <code>position_base</code> object if the last step failed. This is intended for debugging mostly. It is not used by code in this module.

## 72 Module `gmisclib.mcmcS2`

Markov-Chain Monte-Carlo algorithms.

Here, we do MCMC when  $-\log(p)$  is a sum of squares.

### 72.1 Functions

<code>go_step(<i>x</i>, <i>showvec</i>)</code>
--

## 73 Module *gmisclib.mcmc\_big*

An extension of *mcmc* that includes new stepping algorithms.

### 73.1 Functions

**N\_maximum**(*a*)

**test**()

**find\_closest\_p**(*v*, *vp*)

Searches a list of (v,p) pairs and finds the one whose v is closest to the first argument. Returns (v,p) of the closest pair.

**bootstepper**(*logp*, *x*, *v*, *c*=None, *strategy*='intermediate', *fixer*=None, *repeatable*=True)

This is (essentially) another interface to the class constructor. It's really there for backwards compatibility.

**test2d**(*stepper*)

### 73.2 Variables

Name	Description
SIGFAC	Value: 3.0
__package__	Value: 'gmisclib'

### 73.3 Class *BootStepper*



**73.3.1 Methods**

```
__init__(self, lop, v, strategy='intermediate', maxArchSize=None,
parallelSizeDiv=1)
```

*x.\_\_init\_\_*(...) initializes *x*; see *help*(*type*(*x*)) for signature

**Parameters**

- maxArchSize:** How many position vectors can be stored. This is normally used to (loosely) enforce a memory limitation for large jobs.
- parallelSizeDiv:** For use when there are several cooperating MCMC processes that share data. When >1, this allows each process to have smaller stored lists. Normally, *parallelSizeDiv* is between 1 and the number of cooperating processes.
- lop:** These are a list of starting positions that should be evaluated first and used as the basis for further exploration.
- v:** A covariance matrix for non-bootstrap steps. Mostly, the system takes bootstrap steps that are adaptively derived from previous steps. However, to generate the list of previous steps, it needs a way to generate steps ab initio. So, *v* is used to get things going. *v* is also used occasionally thereafter, just to make sure the stepper doesn't get trapped in some subspace.

Overrides: *object.\_\_init\_\_* *exitit*(inherited documentation)

```
step(self)
```

In subclasses, this takes a step and returns 0 or 1, depending on whether the step was accepted or not.

Overrides: *gmisclib.mcmc.stepper.step* *exitit*(inherited documentation)

```
step_mixed(self)
```

```
step_parab(self)
```

---

**current**(*self*)

---

**Return Value**

the current position.

*(type=position\_base)*

---

**ergodic**(*self*)

A crude measure of how ergodic the MCMC is.

---

**Return Value**

The inverse of how many steps it takes to cross the minimum in the slowest direction. Or zero, if it is too soon after some major violation of the MCMC assumptions.

---

**needs\_a\_reset**(*self*)

Decides if we we need a reset. This checks to see if we have a new `logP` that exceeds the old record. It keeps track of the necessary paperwork.

---

**prms**(*self*)

---

**Return Value**

The current parameters

*(type=numpy.ndarray)*

---

**reset**(*self*)

Called internally to mark when the optimization has found a new minimum.

**Note:** You might also call it if the function you are minimizing changes.

---

**reset\_adjusters**(*self*)

---

**reset\_id**(*self*)

Use this to tell if the stepper has been reset since you last looked at it.

---

**Return Value**

an integer that's different for each reset.

*(type=int)*

---

**set\_sort\_strategy**(*self*, *ss*)

---

**set\_strategy**(*self*, *ss*)

**status**(*self*)

Provides some printable status information in a=v; format.

Overrides: *gmisc.lib.mcmc.stepper.status* *exitit*(inherited documentation)**stepV**(*self*)**step\_boot**(*self*)**step\_probe**(*self*)***Inherited from object***

`__delattr__()`, `__format__()`, `__getattr__()`, `__hash__()`, `__new__()`, `__reduce__()`, `__reduce_ex__()`,  
`__repr__()`, `__setattr__()`, `__sizeof__()`, `__str__()`, `__subclasshook__()`

**73.3.2 Properties**

Name	Description
<i>Inherited from object</i>	
<code>__class__</code>	

**73.3.3 Class Variables**

Name	Description
F	F is the targeted step acceptance rate. This is from G. O. Roberts, A. Gelman, and W. Gilks (1997) "Weak convergence and optimal scaling of random walk Metropolis algorithm." <i>Ann. Applied. Probability</i> , 7, p. 110-120 and also from G. O. Roberts and J. S. Rosenthal (2001) "Optimal scaling of various Metropolis-Hastings algorithms." <i>Statistical Sci.</i> 16, pp. 351-367. <b>Value:</b> 0.234
PBootLim	PBootLim Limits the probability of taking a bootstrap step. This, if the optimization collapses into a subspace, some other kind of step will eventually get it out. <b>Value:</b> 0.9
SANNEAL	Simulated annealing <b>Value:</b> 'intermediate'
SSALWAYS	<b>Value:</b> 'hillclimb'

*continued on next page*

Name	Description
SSAMPLE	Sampling mode <b>Value:</b> 'sample'
SSAUTO	<b>Value:</b> 'intermediate'
SSLOW	<b>Value:</b> 'intermediate'
SSNEVER	<b>Value:</b> 'sample'
SUPHILL	Go straight uphill <b>Value:</b> 'hillclimb'
alpha	How rapidly should one expand the archive after a reset? <b>Value:</b> 0.1

#### 73.3.4 Instance Variables

Name	Description
acceptable	Acceptable is a function that decides whether or not a step is OK. You can replace it if you want, but the class should be equivalent to <code>T_acceptor</code> .
last_failed	It should reflect the success or failure of the most recently completed step. It is <code>None</code> if the last step succeeded; it is the most recently rejected <code>position_base</code> object if the last step failed. This is intended for debugging mostly. It is not used by code in this module.
np	The number of parameters:
np_eff	In a multiprocessor situation, <code>np_eff</code> tells you how much data do you need to store locally, so that the overall group of processors stores enough variety of data.



## 74 Module `gmisclib.mcmc_cooperate`

### 74.1 Functions

<code>test0()</code>
----------------------

<code>test0s()</code>
-----------------------

<code>test1()</code>
----------------------

<code>test_many(<i>n</i>, <i>fcn</i>)</code>
--

<code>op_string_median(<i>tmp</i>)</code>
---

<code>op_float_median(<i>tmp</i>)</code>
--

### 74.2 Variables

Name	Description
encoder	<b>Value:</b> <code>g_encode.encoder(regex=r"^[a-zA-Z0-9&lt;&gt;?,./:~;'\{\}[\]\!@\$...)</code>
Ops	<b>Value:</b> <code>{'string_median': op_string_median, 'float_median': op_fl...</code>
Unpack	<b>Value:</b> <code>{'float_median': &lt;type 'float'&gt;, 'string_median': &lt;type '...</code>
test_args	<b>Value:</b> <code>('localhost', 8487, 'K', 'job')</code>
<code>--package--</code>	<b>Value:</b> <code>'gmisclib'</code>

### 74.3 Class Barrier

object — `gmisclib.mcmc_cooperate.Barrier`

### 74.3.1 Methods

**`__init__(self, *x)`**

Constructs a barrier from a list of integers.

Overrides: `object.__init__`

**`__cmp__(self, other)`**

**`__iadd__(self, other)`**

**`__repr__(self)`**

`repr(x)`

Overrides: `object.__repr__` `exitit`(inherited documentation)

**`deepen(self, v)`**

#### *Inherited from object*

`__delattr__()`, `__format__()`, `__getattr__()`, `__hash__()`, `__new__()`, `__reduce__()`, `__reduce_ex__()`, `__setattr__()`, `__sizeof__()`, `__str__()`, `__subclasshook__()`

### 74.3.2 Properties

Name	Description
<i>Inherited from object</i>	
<code>__class__</code>	

## 74.4 Class Oops

object └

exceptions.BaseException └

exceptions.Exception └

**gmisclib.mcmc\_cooperate.Oops**

**Known Subclasses:** `gmisclib.mcmc_cooperate.LateToBarrier`

**74.4.1 Methods**

**`--init--`**(*self*, \**s*)

*x*.`--init--`(...) initializes *x*; see `help(type(x))` for signature

Overrides: `object.--init--` `exitit`(inherited documentation)

***Inherited from exceptions.Exception***

`--new--`()

***Inherited from exceptions.BaseException***

`--delattr--`(), `--getattr--`(), `--getitem--`(), `--getslice--`(), `--reduce--`(), `--repr--`(),  
`--setattr--`(), `--setstate--`(), `--str--`(), `--unicode--`()

***Inherited from object***

`--format--`(), `--hash--`(), `--reduce_ex--`(), `--sizeof--`(), `--subclasshook--`()

**74.4.2 Properties**

Name	Description
<i>Inherited from exceptions.BaseException</i>	
<code>args</code> , <code>message</code>	
<i>Inherited from object</i>	
<code>--class--</code>	

**74.5 Class *LateToBarrier***

### 74.5.1 Methods

**`--init--`**(*self*, \**s*)

*x*.`--init--`(...) initializes *x*; see `help(type(x))` for signature

Overrides: `object.--init--` `exitit`(inherited documentation)

*Inherited from exceptions.Exception*

`--new--`()

*Inherited from exceptions.BaseException*

`--delattr--`(), `--getattr--`(), `--getitem--`(), `--getslice--`(), `--reduce--`(), `--repr--`(),  
`--setattr--`(), `--setstate--`(), `--str--`(), `--unicode--`()

*Inherited from object*

`--format--`(), `--hash--`(), `--reduce_ex--`(), `--sizeof--`(), `--subclasshook--`()

### 74.5.2 Properties

Name	Description
<i>Inherited from exceptions.BaseException</i>	
<code>args</code> , <code>message</code>	
<i>Inherited from object</i>	
<code>--class--</code>	

## 74.6 Class connection

```

object └─
          gmisclib.mcmc_cooperate.connection

```

### 74.6.1 Methods

**`--init--`**(*self*, *host*, *port*, *Key*, *jobid*)

*x*.`--init--`(...) initializes *x*; see `help(type(x))` for signature

Overrides: `object.--init--` `exitit`(inherited documentation)

**`send`**(*self*, \**s*)

`flush(self)``recv(self)``list_ops(self)``version(self)``rank(self)`**Return Value**

number\_of\_processes, rank\_of\_this\_process

*(type=(int, int))*`close(self)``__del__(self)``barrier(self, b, nmin=0, exc=True)``swap_vec(self, logp, v)`**Return Value***(logp, vector)*`set(self, *kv)``get_list(self, key)``get_combined(self, key, operation)``spread(self, key, value, barrier, nmin=0)``get_consensus(self, key, value, barrier, operation, nmin=0)`***Inherited from object***`__delattr__(), __format__(), __getattr__(), __hash__(), __new__(), __reduce__(), __reduce_ex__(),  
__repr__(), __setattr__(), __sizeof__(), __str__(), __subclasshook__()`**74.6.2 Properties**

Name	Description
<i>Inherited from object</i> __class__	

#### 74.6.3 Class Variables

Name	Description
e	<b>Value:</b> g_encode.encoder(regex= r""""[ <sup>^</sup> a-zA-Z0-9<>?,./: ";'{}[\]!@\$...

## 75 Module *gmisclib.mcmc\_helper*

This is a helper module to make use of *mcmc.py* and *mcmc\_big.py*. It allows you to conveniently run a Monte-Carlo simulation of any kind until it converges (*stepper.run\_to\_bottom*) or until it has explored a large chunk of parameter space (*stepper.run\_to\_ergodic*).

It also helps you with logging the process.

### 75.1 Functions

```
make_stepper_from_lov(problem_def, vector_generator,
mcmc_mod=<module 'gmisclib.mcmc' from
'/scratch/gpk/working_natty/...', posn_class=<class
'gmisclib.mcmc.position_repeatable'>, n=None)
```

#### Parameters

**vector\_generator**: Initial starting points for the optimization.  
*(type=A sequence of numpy.ndarray, normally a list of them.)*

**n**: The minimum number of samples to use. None means number of parameters+2.  
*(type=None or int)*

#### Return Value

A *BootStepper*, ready to run, from the specified module.  
*(type=instance of mcmc\_mod, typically BootStepper.)*

```
make_stepper_from_lop(position_generator, mcmc_mod=<module
'gmisclib.mcmc' from '/scratch/gpk/working_natty/...', n=None)
```

#### Parameters

**position\_generator**: Initial starting points for the optimization.  
*(type=A sequence of mcmc.position\_base.)*

**n**: The minimum number of samples to use. None means number of parameters+2.  
*(type=None or int)*

#### Return Value

A *BootStepper*, ready to run, from the specified module.  
*(type=instance of mcmc\_mod, typically BootStepper.)*

```
test1()
```

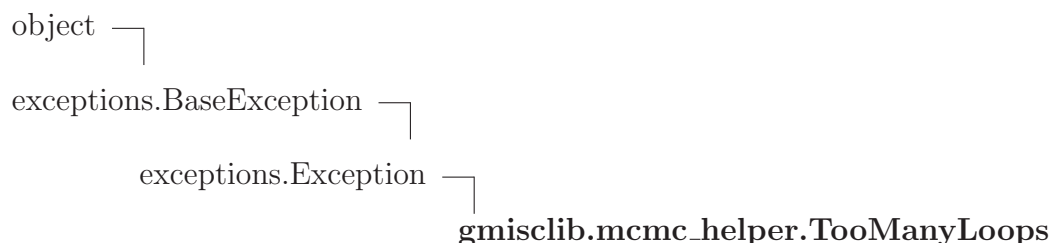
<code>test_probe()</code>
---------------------------

<code>test()</code>
---------------------

## 75.2 Variables

Name	Description
Debug	Value: 0
__package__	Value: 'gmisclib'

## 75.3 Class *TooManyLoops*



### 75.3.1 Methods

<code>__init__(self, *s)</code> <code>x.__init__(...)</code> initializes x; see <code>help(type(x))</code> for signature Overrides: <code>object.__init__</code> extit(inherited documentation)
---

*Inherited from `exceptions.Exception`*

`__new__()`

*Inherited from `exceptions.BaseException`*

`__delattr__()`, `__getattr__()`, `__getitem__()`, `__getslice__()`, `__reduce__()`, `__repr__()`,  
`__setattr__()`, `__setstate__()`, `__str__()`, `__unicode__()`

*Inherited from `object`*

`__format__()`, `__hash__()`, `__reduce_ex__()`, `__sizeof__()`, `__subclasshook__()`

### 75.3.2 Properties



Name	Description
<i>Inherited from <code>exceptions.BaseException</code></i>	
<code>args</code> , <code>message</code>	
<i>Inherited from <code>object</code></i>	
<code>__class__</code>	

## 75.4 Class `warnevery`



### 75.4.1 Methods

`__init__(self, interval)`

`x.__init__(...)` initializes `x`; see `help(type(x))` for signature

Overrides: `object.__init__` `exitit` (inherited documentation)

`inc(self, incr=1)`

`count(self)`

### *Inherited from `object`*

`__delattr__()`, `__format__()`, `__getattr__()`, `__hash__()`, `__new__()`, `__reduce__()`, `__reduce_ex__()`,  
`__repr__()`, `__setattr__()`, `__sizeof__()`, `__str__()`, `__subclasshook__()`

### 75.4.2 Properties

Name	Description
<i>Inherited from <code>object</code></i>	
<code>__class__</code>	

## 75.5 Class `logger_template`



**Known Subclasses:** gmisclib.mcmc\_logger.logger\_c

### 75.5.1 Methods

```
add(self, stepperInstance, iter)
```

```
close(self)
```

```
reset(self)
```

### *Inherited from object*

```
__delattr__(), __format__(), __getattr__(), __hash__(), __init__(), __new__(), __reduce__(),
__reduce_ex__(), __repr__(), __setattr__(), __sizeof__(), __str__(), __subclasshook__()
```

### 75.5.2 Properties

Name	Description
<i>Inherited from object</i>	
__class__	

## 75.6 Class stepper

```
object └─ gmisclib.mcmc_helper.stepper
```

**Known Subclasses:** gmisclib.mcmc\_socket.stepper

### 75.6.1 Methods

```
__init__(self, x, maxloops=-1, logger=None, share=None)
```

x.\_\_init\_\_(...) initializes x; see help(type(x)) for signature

Overrides: object.\_\_init\_\_ extit(inherited documentation)

```
reset_loops(self, maxloops=-1)
```

---

**run\_to\_change**(*self*, *ncw*=1, *acceptable\_step*=None, *update\_T*=False)

Run the *Markov Chain Monte-Carlo* until it finds an acceptable step.

**Parameters**

- ncw:** How many steps should be accepted before it stops?  
(*type*=int)
- acceptable\_step:** A function that decides what steps are acceptable and what are not. This is passed to the MCMC stepper, `mcmc.BootStepper`.  
(*type*=function(float) -> bool, often *mcmc.T\_acceptor* or *step\_acceptor*)
- update\_T:** Obsolete. Must be False.  
(*type*=boolean.)

**Raises**

- TooManyLoops** if it takes a long time before enough steps are accepted.

---

**communicate\_hook**(*self*, *id*)

---

**close**(*self*)

After calling `close()`, it is no longer legal to call `run_to_change()`, `run_to_ergodic()`, or `run_to_bottom()`. Logging is shut down, but the contents of the stepper are still available for inspection.

---

**run\_to\_ergodic**(*self*, *ncw*=1, *T*=1.0)

Run the stepper until it has explored all of parameter space **ncw** times (as best as we can estimate). Note that this is a pretty careful routine. If the stepper finds a better region of parameter space so that log(P) improves part way through the process, the routine will reset itself and begin again.

NOTE: this sets **T** and **sortstrategy** for the **stepper**.

**Parameters**

**ncw**: how many times (or what fraction) of an ergodic exploration to make.

(*type=int.*)

**T**: temperature at which to run the Monte-Carlo process. This is normally 1.0. If it is **None**, then the current temperature is not modified.

(*type=float or None.*)

**Return Value**

a **position** at the end of the process.

**run\_to\_bottom**(*self*, *ns*=3, *acceptable\_step*=None)

Run the *Markov Chain Monte-Carlo* until it converges near a minimum.

The general idea of the termination condition is that it keeps track of the angle between successive sequences of steps, where each sequence contains **ns** steps. If the angle is less than 90 degrees, it is evidence that the solution is still drifting in some consistent direction and therefore not yet at the maximum. Angles of greater than 90 degrees suggest that the optimization is wandering aimlessly near a minimum. It will run until a sufficient number of large angles are seen since the last BMCMC reset. A similar check is made on individual components of the parameter vector.

#### Parameters

**ns**: This controls how carefully it checks that it has really found a minimum. Large values will take longer but will assure more accurate convergence.

(*type*=**int**)

**acceptable\_step**: A callable object (i.e. class or function) that returns **True** or **False**, depending on whether a proposed step is acceptable. See `mcmc.T_acceptor` for an example.

#### Raises

TooManyLoops ?

**synchronize\_start**(*self*, *id*)

**synchronize\_end**(*self*, *id*)

**synchronize\_abort**(*self*, *id*)

**note**(*self*, *s*, *lvl*)

#### Inherited from object

`__delattr__()`, `__format__()`, `__getattr__()`, `__hash__()`, `__new__()`, `__reduce__()`, `__reduce_ex__()`, `__repr__()`, `__setattr__()`, `__sizeof__()`, `__str__()`, `__subclasshook__()`

#### 75.6.2 Properties

Name	Description
<i>Inherited from object</i>	

*continued on next page*

Name	Description
<code>--class--</code>	

## 75.7 Class *step\_acceptor*

object └

*gmisclib.mcmc.acceptor\_base* └

*gmisclib.mcmc.rough\_acceptor\_base* └

***gmisclib.mcmc\_helper.step\_acceptor***

This class defines the annealing schedule when *mcmc* is used as an optimizer. It corresponds to *n* extra degrees of freedom that exchange probability (i.e. ‘energy’) with each other, and the system to be optimized. Over time, the excess  $\log(\text{probability})$  gradually oozes away.

**75.7.1 Methods**


---

```
__init__(self, n, T0, T=1.0, maxT=None, probestep=None, tau_probe=None,
jitterdroop=None)
```

---

Create a `step_acceptor` and define it's properties.

**Parameters**

**n**: How many degrees of freedom should the step acceptor store? Note that the energy of the step acceptor is gradually equilibrated to the desired temperature. Each step, there is a 1/n chance of equilibrating one DOF. So, when n=1, this is exactly a `T_acceptor`. When n>>1, the acceptor is very nearly leak-free, and exchanges energy with the system to be optimized, but hardly at all with the heat bath. So, for n=1, the temperature is fixed at T. When n>>1, the temperature can rise as logP goes up, and will fall when logP goes down.

(*type*=*int*)

**T**: What's the temperature of the heat bath? This is the final temperature that the annealing schedule will eventually approach. Default: 1.0.

(*type*=*float*)

**T0**: The starting temperature.

(*type*=*float*)

**maxT**: In optimization mode, it's normal for the system being optimized to increase it's log(P). That dumps log(P) or 'energy' into the `step_acceptor`'s degrees of freedom and raises its temperature. This parameter lets you define an approximate maximum temperature. Setting this may speed up convergence, though there is a corresponding risk of getting trapped in a local minimum.

(*type*=*float* or *None*)

Overrides: `object.__init__`

---

```
T(self)
```

---

**Return Value**

the current effective temperature.

(*type*=*float*)

Overrides: `gmisclib.mcmc.acceptor_base.T`

---

---

**\_\_call\_\_**(*self*, *delta*)

---

 Accept a step or not?

**Parameters**

**delta**: How much did the candidate step change  $\log(P)$ ?  
*(type=float)*

**Return Value**

Whether to accept the candidate step or not.  
*(type=bool)*

 Overrides: *gmisclib.mcmc.acceptor\_base.\_\_call\_\_*


---

**\_\_repr\_\_**(*self*)

---

 repr(x)

 Overrides: *object.\_\_repr\_\_* extit(inherited documentation)

---

**is\_root**(*self*)

---

 This is a stub for compatibility with MPI code.

---

**size**(*self*)

---

 This is a stub for compatibility with MPI code.

---

**rank**(*self*)

---

 This is a stub for compatibility with MPI code.

---

**jitter**(*self*)

 In the acoustic distances paper, it is called  $\bar{\delta}$ .

**Return Value**

An estimate of how rough the function is, on a small scale.  
*(type=float)*

 Overrides: *gmisclib.mcmc.acceptor\_base.jitter* extit(inherited documentation)



**probe**(*self*, *currentpos*, *stepper*)

This computes  $\log P()$  near the current step position, and sees how much that differs from  $\log P()$  *\_at\_* the current step position. That difference is used as a measure of the "roughness" or "randomness" of the function. That extra roughness is added onto the temperature to ensure that the MCMC stepper doesn't get trapped in an unimportant local valley. Essentially, we are making the statement that any differences within the region covered by **probestep** are unimportant.

Overrides: `gmisclib.mcmc.acceptor_base.probe`

**reset**(*self*)

In order to make the overall algorithm asymptotically a Markovian, we need to make sure that (asymptotically) the acceptor function depends only on long-term averages, not on recent history. The problem is the jitter measurement, of course. So, one needs to average the jitter over a time longer than the recurrence time, asymptotically. But that makes it horribly unresponsive in the early stages of the optimization when things are rapidly changing. The solution is to pass along resets, so that it can (temporarily) revert to a short averaging time.

Overrides: `gmisclib.mcmc.acceptor_base.reset` `exitit`(inherited documentation)

### *Inherited from object*

`__delattr__()`, `__format__()`, `__getattr__()`, `__hash__()`, `__new__()`, `__reduce__()`, `__reduce_ex__()`, `__setattr__()`, `__sizeof__()`, `__str__()`, `__subclasshook__()`

### 75.7.2 Properties

Name	Description
<i>Inherited from object</i>	
<code>__class__</code>	

### 75.7.3 Instance Variables

Name	Description
<code>last_probe</code>	A pair of <code>position_base</code> instances showing the position and $\log P$ at each end of the most recent probe for a discontinuity. Or, it could be <code>None</code> . This value is not used in <code>mcmc.py</code> . Rather, it is there for the logging modules.

*continued on next page*

Name	Description
<code>since_last_reset</code>	Number of iterations since the last overall MCMC reset.

## 76 Module *gmisclib.mcmc\_idxr*

### 76.1 Functions

**logp\_prior\_normalized**(*x*, *guess\_prob\_dist*)

**Parameters**

**guess\_prob\_dist:** (*type=list((str, probdist\_c))*)  
**x:** (*type=newstem2.indexclass.index\_base*)

**Return Value**

the log of the prior probability. @except *mcmc.NotGoodPosition*:  
 When raised by any of the underlying probability distributions.  
*(type=float)*

### 76.2 Variables

Name	Description
<code>--package--</code>	<b>Value:</b> 'gmisclib'

### 76.3 Class *probdist\_c*

object └─ **gmisclib.mcmc\_idxr.probdist\_c**

**Known Subclasses:** *gmisclib.mcmc\_idxr.Expo*, *gmisclib.mcmc\_idxr.Fixed*, *gmisclib.mcmc\_idxr.LogNormal*, *gmisclib.mcmc\_idxr.Normal*, *gmisclib.mcmc\_idxr.Uniform*, *gmisclib.mcmc\_idxr.Weibull*

#### 76.3.1 Methods

**\_\_call\_\_**(*self*)

Sample from the probability distribution.

**Return Value**

the sample  
*(type=float)*

**logdens**(*self*, *x*)

Return the log(density) of the probability distribution at *x*.

**Parameters**

*x*: the position where the density should be evaluated.

(*type=float*)

**Return Value**

log(density)

(*type=float*)

**Raises**

`mcmc.NotGoodPosition` when the density is not defined at *x*.

**Inherited from object**

`__delattr__()`, `__format__()`, `__getattribute__()`, `__hash__()`, `__init__()`, `__new__()`, `__reduce__()`,  
`__reduce_ex__()`, `__repr__()`, `__setattr__()`, `__sizeof__()`, `__str__()`, `__subclasshook__()`

### 76.3.2 Properties

Name	Description
<i>Inherited from object</i>	
<code>__class__</code>	

## 76.4 Class Fixed

object └

gmisclib.mcmc\_idxr.probdist\_c └

**gmisclib.mcmc\_idxr.Fixed**

This is essentially a delta-function probability distribution. Use this for fixed parameters.

**Note:** Software that uses this should ignore `logdens`.

### 76.4.1 Methods

**\_\_init\_\_**(*self*, *value*)

*x*.`__init__`(...) initializes *x*; see `help(type(x))` for signature

Overrides: `object.__init__` `extit`(inherited documentation)

```
--call--(self)
```

Sample from the probability distribution.

**Return Value**

the sample

(*type=float*)

Overrides: gmisclib.mcmc\_idxr.probdist\_c.\_\_call\_\_ extit(inherited documentation)

```
--doc--(self)
```

*Inherited from object*

\_\_delattr\_\_(), \_\_format\_\_(), \_\_getattr\_\_(), \_\_hash\_\_(), \_\_new\_\_(), \_\_reduce\_\_(), \_\_reduce\_ex\_\_(), \_\_repr\_\_(), \_\_setattr\_\_(), \_\_sizeof\_\_(), \_\_str\_\_(), \_\_subclasshook\_\_()

#### 76.4.2 Properties

Name	Description
<i>Inherited from object</i>	
__class__	

#### 76.4.3 Class Variables

Name	Description
logdens	<b>Value:</b> None

### 76.5 Class Weibull

object └

gmisclib.mcmc\_idxr.probdist\_c └

**gmisclib.mcmc\_idxr.Weibull**

Weibull distribution

## 76.5.1 Methods

**`__init__(self, scale, shape)`**

`x.__init__(...)` initializes `x`; see `help(type(x))` for signature

Overrides: `object.__init__` `exitit`(inherited documentation)

**`__call__(self)`**

Sample from the probability distribution.

**Return Value**

the sample

(*type=*`float`)

Overrides: `gmisclib.mcmc_idxr.probdist_c.__call__` `exitit`(inherited documentation)

**`logdens(self, x)`**

Return the `log(density)` of the probability distribution at `x`.

**Parameters**

`x`: the position where the density should be evaluated.

**Return Value**

`log(density)`

(*type=*`float`)

**Raises**

`mcmc.NotGoodPosition` when the density is not defined at `x`.

Overrides: `gmisclib.mcmc_idxr.probdist_c.logdens` `exitit`(inherited documentation)

**`__doc__(self)`**

*Inherited from object*

`__delattr__()`, `__format__()`, `__getattr__()`, `__hash__()`, `__new__()`, `__reduce__()`, `__reduce_ex__()`, `__repr__()`, `__setattr__()`, `__sizeof__()`, `__str__()`, `__subclasshook__()`

## 76.5.2 Properties

Name	Description
<i>Inherited from object</i>	

*continued on next page*

Name	Description
<code>--class--</code>	

## 76.6 Class Expo

object └─

gmisclib.mcmc\_idxr.probdist\_c └─  
**gmisclib.mcmc\_idxr.Expo**

Exponential distribution

### 76.6.1 Methods

<b><code>--init--</code></b> ( <i>self</i> , <i>lambda</i> )
<code>x.__init__(...)</code> initializes x; see <code>help(type(x))</code> for signature
<b>Parameters</b>
<i>lambda</i> : <i>1/the_mean</i> (also <i>1/the_standard_deviation</i> ) ( <i>type=float</i> )
<b>Raises</b>
ValueError Bad parameters for the distribution.
Overrides: <code>object.__init__</code>
<b><code>--call--</code></b> ( <i>self</i> )
Sample from the probability distribution.
<b>Return Value</b>
the sample ( <i>type=float</i> )
Overrides: <code>gmisclib.mcmc_idxr.probdist_c.__call__</code> <code>exitit</code> (inherited documentation)

**logdens**(*self*, *x*)

Return the log(density) of the probability distribution at *x*.

**Parameters**

*x*: the position where the density should be evaluated.

**Return Value**

log(density)

(*type=float*)

**Raises**

`mcmc.NotGoodPosition` when the density is not defined at *x*.

Overrides: `gmisclib.mcmc_idxr.probdist_c.logdens` `exitit`(inherited documentation)

**\_\_doc\_\_**(*self*)

**Inherited from object**

`__delattr__()`, `__format__()`, `__getattr__()`, `__hash__()`, `__new__()`, `__reduce__()`, `__reduce_ex__()`, `__repr__()`, `__setattr__()`, `__sizeof__()`, `__str__()`, `__subclasshook__()`

**76.6.2 Properties**

Name	Description
<i>Inherited from object</i>	
<code>__class__</code>	

**76.7 Class Normal**

object └

gmisclib.mcmc\_idxr.probdist\_c └

gmisclib.mcmc\_idxr.Normal

Normal distribution



## 76.7.1 Methods

**`__init__(self, mu, sigma)`**

`x.__init__(...)` initializes `x`; see `help(type(x))` for signature

Overrides: `object.__init__` `exitit`(inherited documentation)

**`__call__(self)`**

Sample from the probability distribution.

**Return Value**

the sample

(*type=float*)

Overrides: `gmisclib.mcmc_idxr.probdist_c.__call__` `exitit`(inherited documentation)

**`logdens(self, x)`**

Return the `log(density)` of the probability distribution at `x`.

**Parameters**

`x`: the position where the density should be evaluated.

**Return Value**

`log(density)`

(*type=float*)

**Raises**

`mcmc.NotGoodPosition` when the density is not defined at `x`.

Overrides: `gmisclib.mcmc_idxr.probdist_c.logdens` `exitit`(inherited documentation)

**`__doc__(self)`**

*Inherited from object*

`__delattr__()`, `__format__()`, `__getattr__()`, `__hash__()`, `__new__()`, `__reduce__()`, `__reduce_ex__()`, `__repr__()`, `__setattr__()`, `__sizeof__()`, `__str__()`, `__subclasshook__()`

## 76.7.2 Properties

Name	Description
<i>Inherited from object</i>	

*continued on next page*

Name	Description
<code>--class--</code>	

## 76.8 Class Uniform

object └

gmsclib.mcmc\_idxr.probdist\_c └  
**gmsclib.mcmc\_idxr.Uniform**

Uniform distribution

### 76.8.1 Methods

**`--init--`**(*self*, *low*, *high*)

*x*.`--init--`(...) initializes *x*; see `help(type(x))` for signature

**Raises**

**ValueError** Bad parameters for the distribution.

Overrides: `object.--init--`

**`--call--`**(*self*)

Sample from the probability distribution.

**Return Value**

the sample

(*type=float*)

Overrides: `gmsclib.mcmc_idxr.probdist_c.--call--` `exitit`(inherited documentation)

**logdens**(*self*, *x*)

Return the log(density) of the probability distribution at *x*.

**Parameters**

*x*: the position where the density should be evaluated.

**Return Value**

log(density)

(*type=float*)

**Raises**

`mcmc.NotGoodPosition` when the density is not defined at *x*.

Overrides: `gmisclib.mcmc_idxr.probdist_c.logdens` `exitit`(inherited documentation)

**\_\_doc\_\_**(*self*)

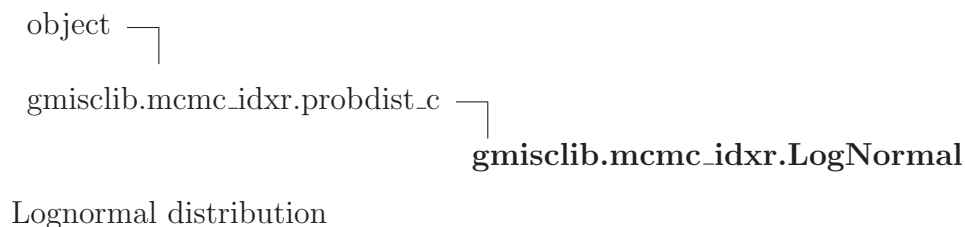
**Inherited from object**

`__delattr__()`, `__format__()`, `__getattr__()`, `__hash__()`, `__new__()`, `__reduce__()`, `__reduce_ex__()`, `__repr__()`, `__setattr__()`, `__sizeof__()`, `__str__()`, `__subclasshook__()`

## 76.8.2 Properties

Name	Description
<i>Inherited from object</i>	
<code>__class__</code>	

## 76.9 Class LogNormal



**76.9.1 Methods**

**`__init__(self, mu, sigma)`**

`x.__init__(...)` initializes `x`; see `help(type(x))` for signature

**Parameters**

**sigma:** The standard deviation of the log of the resulting distribution.

(*type=float*)

**mu:** The median of the resulting distribution, *not* the log of the median.

(*type=float*)

**Raises**

**ValueError** Bad parameters for the distribution.

Overrides: `object.__init__`

**`__call__(self)`**

Sample from the probability distribution.

**Return Value**

the sample

(*type=float*)

Overrides: `gmisclib.mcmc_idxr.probdist_c.__call__` `exitit`(inherited documentation)

**`logdens(self, x)`**

Return the `log(density)` of the probability distribution at `x`.

**Parameters**

**x:** the position where the density should be evaluated.

**Return Value**

`log(density)`

(*type=float*)

**Raises**

**mcmc.NotGoodPosition** when the density is not defined at `x`.

Overrides: `gmisclib.mcmc_idxr.probdist_c.logdens` `exitit`(inherited documentation)

**`__doc__(self)`**

***Inherited from object***

`__delattr__()`, `__format__()`, `__getattr__()`, `__hash__()`, `__new__()`, `__reduce__()`, `__reduce_ex__()`, `__repr__()`, `__setattr__()`, `__sizeof__()`, `__str__()`, `__subclasshook__()`

**76.9.2 Properties**

Name	Description
<i>Inherited from object</i>	
<code>__class__</code>	

**76.10 Class `problem_definition`****76.10.1 Methods**

<b><code>__init__(self)</code></b> <code>x.__init__(...)</code> initializes <code>x</code> ; see <code>help(type(x))</code> for signature Overrides: <code>object.__init__</code> extit(inherited documentation)
<b><code>set_idxr(self, idxr)</code></b>
<b><code>plot(self, idxr, arg, pylab, inum)</code></b>
<b><code>do_print(self, idxr, arg, iter)</code></b>

**logp**(*self*, *x*)

Compute the log of the probability density at **x**.

**Parameters**

**x**: a parameter vector

**Return Value**

The log of the probability that the model assigns to parameters **x**.

(*type=float*)

**Raises**

**NotGoodPosition** This exception is used to indicate that the position **x** is not valid. This is equivalent to returning an extremely negative logp.

Overrides: *gmisclib.mcmc.problem\_definition.logp* *extit*(inherited documentation)

**logp\_data\_normalized**(*self*, *x*)

Only define this if you can compute the actual probability density for the data given model & parameters, not just something proportional to it! To do this function, you need to be able to do the full multidimensional integral over the probability distribution!

NOTE that **x** is an indexer! NOTE that this is not the full logp function: it doesn't contain the prior!

**fixer**(*self*, *x*)

This is called on each candidate position vector. Generally, it is used to restrict the possible solution space by folding position vectors that escape outside the solution space back into the solution space. It can also allow for symmetries in equations.

Formally, it defines a convex region. All vectors outside the region are mapped into the region, and the mapping must be continuous at the boundary. (More precisely, `logp(fixer(x))` must be continuous everywhere that `logp(x)` is continuous, including the boundary.) For instance, mapping `x[0]` into `abs(x[0])` defines a convex region (the positive half-space), and the mapping is continuous near `x[0]=0`.

Additionally, it may re-normalize parameters at will subject to the restriction that `logp(fixer(x))==logp(x)`. For instance, it can implement a constraint that `sum(x)==0` by mapping `x` into `x - average(x)`, so long as the value of `logp()` is unaffected by that substitution. Other folds can sometimes lead to problems.

#### Parameters

`x`: a parameter vector

#### Return Value

a (possibly modified) parameter vector.

(*type=*`numpy.ndarray`)

#### Raises

**NotGoodPosition** This exception is used to indicate that the position `x` is not valid. Fixer has the option of either mapping invalid parameter vectors into valid ones or raising this exception.

Overrides: `gmisclib.mcmc.problem_definition.fixer` `extit`(inherited documentation)

**logp\_prior\_normalized**(*self*, *x*)

**logp\_guts**(*self*, *idxr*, *data\_sink=None*)

**log**(*self*, *p*, *i*)

Some code calls this function every iteration to log the current state of the MCMC process.

**Parameters**

*p*: the current parameter vector, and

*i*: an integer iteration counter.

**Return Value**

nothing.

*Inherited from object*

`__delattr__()`, `__format__()`, `__getattr__()`, `__hash__()`, `__new__()`, `__reduce__()`, `__reduce_ex__()`, `__repr__()`, `__setattr__()`, `__sizeof__()`, `__str__()`, `__subclasshook__()`

**76.10.2 Properties**

Name	Description
<i>Inherited from object</i>	
<code>__class__</code>	

**76.10.3 Class Variables**

Name	Description
PriorProbDist	Data for guessing. <b>Value:</b> None



## 77 Module *gmisclib.mcmc\_indexclass*

This module provides several classes to manage the parameters of an algorithm, particularly so that you can run the *mcmc.py* optimizer on it. Essentially, the main problem that it solves is how to keep track of human-readable names assigned to components of a vector of parameters.

You have an *index\_base* object *i*, and you can get a parameter from it like this: *i.p("velocity", "car", "x")*. That would get out a number that might correspond to the x-velocity of a car. That's the human-readable side, but you can also get a vector of all the parameters via *i.get\_prms()* and get a mapping between names and indices into that vector via *i.map*.

You can also have parameters that do not appear in the vector of adjustable parameters. These are known as "fixed" parameters; their intent is to let you fix some of the parameters of an optimization, and let the optimizer play with the remainder.

Use *guess* first to establish the mapping, write it to a file and guess reasonable starting values for an optimization. Then, use *index* to get parameter values corresponding to names.

### 77.1 Functions

<code>default_mapfp(<i>nmprefix</i>='m')</code>
---

<code>guess_to_indexer(<i>g</i>)</code>
---

<code>reindex(<i>q</i>, <i>ref</i>)</code>
--

This remaps the parameters of *q* so that they are in the same order as in *ref*.

**Return Value**

an indexer that contains the same data as *q*, but shares its parameter mapping with *ref*.

(*type=indexer*)

**Note:** The parameters in *q* must be a superset of the adjustable parameters in *ref*. Extra parameters in *q* will be carried over as fixed parameters.

### 77.2 Variables

Name	Description
Debug	<b>Value:</b> False
__package__	<b>Value:</b> 'gmisclib'

## 77.3 Class *PriorProbDist*

object —  
**gmisclib.mcmc\_indexclass.PriorProbDist**

This is a map from a regular expression to a callable object. It's used to find which element of the *PriorProbDist* applies to a particular parameter.

### 77.3.1 Methods

**\_\_init\_\_**(*self*, *guesses*)

*x.\_\_init\_\_*(...) initializes *x*; see *help(type(x))* for signature

**Parameters**

**guesses**: a list of associations between a pattern and a probability distribution. If *R* exists, it is a float used for probing for function roughness.

*(type=list( (str, mcmc\_idxr.probdist\_c, R) ) where R is an optional float.)*

Overrides: *object.\_\_init\_\_*

**\_\_getitem\_\_**(*self*, *fkey*)

Look up a string against the stored list of patterns. Return the corresponding callable.

**Parameters**

**fkey**: this takes the key formatted as a string, not as a tuple.

*(type=str @rtype whatever is the middle component of a guesses tuple. This is normally an instance of mcmc\_idxr.probdist\_c.)*

**getprobe**(*self*, *fkey*)

**Parameters**

**fkey**: this takes the key formatted as a string, not as a tuple.

*(type=str)*

**Return Value**

whatever you get from the final component of a *guesses* tuple. This is normally a float.

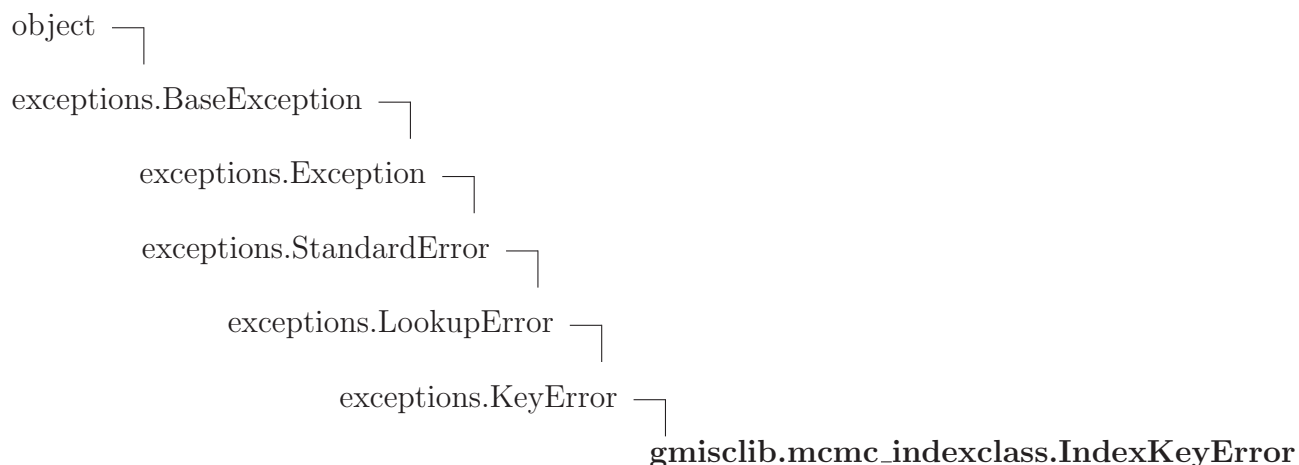
*Inherited from object*

`__delattr__()`, `__format__()`, `__getattr__()`, `__hash__()`, `__new__()`, `__reduce__()`, `__reduce_ex__()`,  
`__repr__()`, `__setattr__()`, `__sizeof__()`, `__str__()`, `__subclasshook__()`

### 77.3.2 Properties

Name	Description
<i>Inherited from object</i>	
<code>__class__</code>	

## 77.4 Class `IndexKeyError`



### 77.4.1 Methods

<code>__init__(self, *a)</code>
<code>x.__init__(...)</code> initializes <code>x</code> ; see <code>help(type(x))</code> for signature
Overrides: <code>object.__init__</code> <code>extit</code> (inherited documentation)

*Inherited from `exceptions.KeyError`*

`__new__()`, `__str__()`

*Inherited from `exceptions.BaseException`*

`__delattr__()`, `__getattr__()`, `__getitem__()`, `__getslice__()`, `__reduce__()`, `__repr__()`,  
`__setattr__()`, `__setstate__()`, `__unicode__()`

*Inherited from `object`*

`__format__()`, `__hash__()`, `__reduce_ex__()`, `__sizeof__()`, `__subclasshook__()`

#### 77.4.2 Properties

Name	Description
<i>Inherited from <code>exceptions.BaseException</code></i>	
<code>args</code> , <code>message</code>	
<i>Inherited from <code>object</code></i>	
<code>__class__</code>	

### 77.5 Class `index_base`

object —  
     `gmisclib.mcmc_indexclass.index_base`

**Known Subclasses:** `gmisclib.mcmc_indexclass.guess`, `gmisclib.mcmc_indexclass.index`

#### 77.5.1 Methods

`__init__(self, keymap, name=None)`

`x.__init__(...)` initializes `x`; see `help(type(x))` for signature

Overrides: `object.__init__` `extit` (inherited documentation)

`clear(self)`

`p(self, *key)`

`p_lower(self, ll, *key)`

`p_range(self, lb, ub, *key)`

`pN(self, num, *key)`

`pNr(self, num, low, high, *key)`

`pNI(self, num, low, high, *key)`

---

`get_prms(self)`


---

**Return Value**

a vector of the adjustable parameters

(*type=*`numpy.ndarray`)

---

`columns(self)`


---



---

`n(self)`


---

**Return Value**

the number of adjustable parameters.

(*type=*`int`)

**Note:** to include the number of fixed parameters, you can take `len(self._p)`.

---

`i_k_val(self)`


---



---

`comment(self, c)`


---



---

`hash(self)`


---



---

`__call__(self)`


---



---

`logdens(self, x)`


---



---

`set_all_fixed(self, kv)`


---

This sets a whole dictionary's worth of fixed parameters.

**Parameters**

**kv:** parameters to set

(*type=*`dict`)

---

`set_fixed(self, v, *k)`


---

You can have "fixed" parameters that are not updated by the MCMC optimizer. This sets one of them.

**Parameters**

**v:** the value

**k:** the key.

<b><code>get_fixed(self)</code></b>
-------------------------------------

Get only the fixed parameters.
--------------------------------

<b>Return Value</b>
---------------------

a dictionary from parameter name to value.
--

<b><code>get_all(self)</code></b>
-----------------------------------

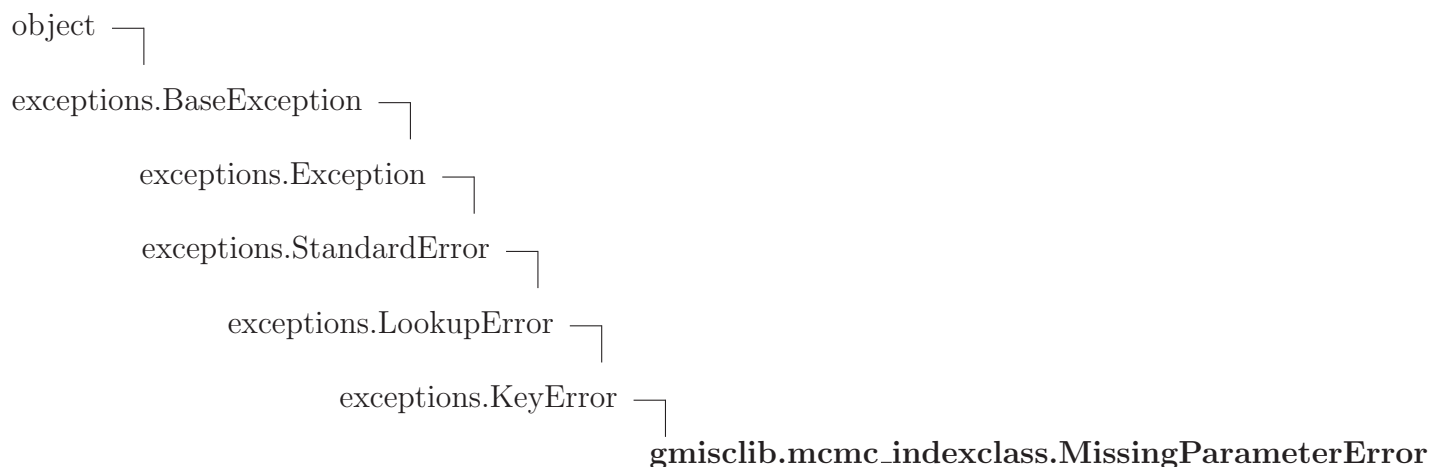
### *Inherited from `object`*

`__delattr__()`, `__format__()`, `__getattribute__()`, `__hash__()`, `__new__()`, `__reduce__()`, `__reduce_ex__()`, `__repr__()`, `__setattr__()`, `__sizeof__()`, `__str__()`, `__subclasshook__()`

#### 77.5.2 Properties

Name	Description
<i>Inherited from object</i>	
<code>__class__</code>	

## 77.6 Class `MissingParameterError`



#### 77.6.1 Methods

<b><code>__init__(self, *s)</code></b>
--

<code>x.__init__(...)</code> initializes <code>x</code> ; see <code>help(type(x))</code> for signature
--

Overrides: <code>object.__init__</code> <code>exitit</code> (inherited documentation)
---

***Inherited from exceptions.KeyError***`__new__()`, `__str__()`***Inherited from exceptions.BaseException***`__delattr__()`, `__getattr__()`, `__getitem__()`, `__getslice__()`, `__reduce__()`, `__repr__()`,  
`__setattr__()`, `__setstate__()`, `__unicode__()`***Inherited from object***`__format__()`, `__hash__()`, `__reduce_ex__()`, `__sizeof__()`, `__subclasshook__()`**77.6.2 Properties**

Name	Description
<i>Inherited from exceptions.BaseException</i> args, message	
<i>Inherited from object</i> __class__	

**77.7 Class sampler****77.7.1 Methods**

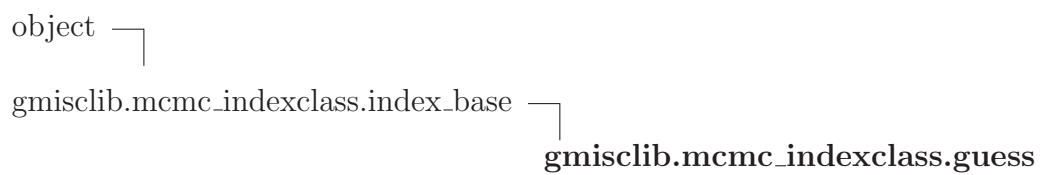
<b><code>__init__(self)</code></b>  x.__init__(...) initializes x; see help(type(x)) for signature Overrides: object.__init__ extit(inherited documentation)
<b><code>__call__(self)</code></b>
<b><code>logdens(self, x)</code></b>

***Inherited from object***`__delattr__()`, `__format__()`, `__getattr__()`, `__hash__()`, `__new__()`, `__reduce__()`, `__reduce_ex__()`,  
`__repr__()`, `__setattr__()`, `__sizeof__()`, `__str__()`, `__subclasshook__()`

### 77.7.2 Properties

Name	Description
<i>Inherited from object</i> <code>__class__</code>	

## 77.8 Class `guess`





## 77.8.1 Methods

**`__init__(self, guesses, fp=None, name='')`**

Create an object that produces the initial guesses at parameter values. Feed it a list of patterns that match parameter names, each with a function to produce a random guess for that parameter. When you request the value of a parameter via `self.p` or similar, it will look up the parameter name, find the first regular expression that matches it, and call the associated function to generate a random value for that parameter.

**Parameters**

**guesses:** a list of (**pattern**, **sampler**) tuples, where **pattern** is an (uncompiled) regular expression, The **pattern** selects for which parameters **sampler** will be used. It must match the entire parameter name; it will be surrounded by "`^`" before use. **sampler** is a function (or class) that produces samples from some probability distribution. **Sampler** is called without arguments, once for each parameter, for each guess that is needed.

*(type=list(tuple(str, sampler)))*

**fp:** an open file which can be used as a log, or None.

*(type=a writeable file object.)*

**name:** a named for the indexer. It is just stored away, and may be accessed as the "name" attribute.

*(type=anything.)*

Overrides: `object.__init__`

**`comment(self, c)`**

Overrides: `gmisclib.mcmc_indexclass.index_base.comment`

**`clear(self)`**

Overrides: `gmisclib.mcmc_indexclass.index_base.clear`

**get\_prms**(*self*)

This **freezes** your set of parameters so no more will be added, and then returns the current guess for all the adjustable parameters.

**Return Value**

a vector of the adjustable parameters

(*type=****numpy.ndarray***)

Overrides: *gmisclib.mcmc\_indexclass.index\_base.get\_prms*

**\_\_repr\_\_**(*self*)

*repr*(*x*)

Overrides: *object.\_\_repr\_\_* *exitit*(inherited documentation)

**set\_user**(*self*, *user*)

**usage**(*self*)

Tells you which users access each key.

**Return Value**

**dictops.dict\_of\_sets** (approximately a dict) indexed by integer indices and mapping to a set of users. e.g. `{('some', 'key'): set(['user1', 'user2']), ...}`.

(*type=****dict(tuple: set(str))***)

**p**(*self*, \**key*)

Overrides: *gmisclib.mcmc\_indexclass.index\_base.p*

**pc**(*self*, \**key*)

**p\_lower**(*self*, *ll*, \**key*)

A range, with a lower limit.

Overrides: *gmisclib.mcmc\_indexclass.index\_base.p\_lower*

**pc\_lower**(*self*, *ll*, \**key*)

A range, with a lower limit.

**p\_range**(*self*, *lb*, *ub*, \**key*)

Overrides: *gmisclib.mcmc\_indexclass.index\_base.p\_range*

**p\_periodic**(*self*, *ub*, \**key*)

A range with periodic boundary conditions 0 to ub.

**freeze**(*self*)

This call means that all parameter names have been seen. Any novel names in calls to **self.p()** will then raise an error.

**set\_prm**(*self*, *v*, \**key*)

This function is used to adjust parameters in non-trivial ways.

**\_\_call\_\_**(*self*)

**columns**(*self*)

**get\_all**(*self*)

**get\_fixed**(*self*)

Get only the fixed parameters.

**Return Value**

a dictionary from parameter name to value.

**hash**(*self*)

**i\_k\_val**(*self*)

**logdens**(*self*, *x*)

**n**(*self*)

**Return Value**

the number of adjustable parameters.

(*type=int*)

**Note:** to include the number of fixed parameters, you can take **len(self.\_p)**.

**pN**(*self*, *num*, \**key*)

**pNI**(*self*, *num*, *low*, *high*, \**key*)

<b>pNr</b> ( <i>self</i> , <i>num</i> , <i>low</i> , <i>high</i> , * <i>key</i> )
---

<b>set_all_fixed</b> ( <i>self</i> , <i>kv</i> )
--

This sets a whole dictionary's worth of fixed parameters.

**Parameters**

**kv**: parameters to set  
(*type=**dict*)

<b>set_fixed</b> ( <i>self</i> , <i>v</i> , * <i>k</i> )
--

You can have "fixed" parameters that are not updated by the MCMC optimizer. This sets one of them.

**Parameters**

**v**: the value  
**k**: the key.

***Inherited from object***

`--delattr--()`, `--format--()`, `--getattr--()`, `--hash--()`, `--new--()`, `--reduce--()`, `--reduce_ex--()`,  
`--setattr--()`, `--sizeof--()`, `--str--()`, `--subclasshook--()`

**77.8.2 Properties**

Name	Description
<i>Inherited from object</i>	
<code>--class--</code>	

**77.9 Class index**

object └

gmislib.mcmc\_indexclass.index\_base └  
gmislib.mcmc\_indexclass.index

**Known Subclasses:** `gmislib.mcmc_indexclass.index_counted`

**77.9.1 Methods**

**`__init__(self, keymap, p=None, name='', fixed=None)`**

`x.__init__(...)` initializes `x`; see `help(type(x))` for signature

Overrides: `object.__init__` `exitit`(inherited documentation)

**`hash(self)`**

Overrides: `gmisclib.mcmc_indexclass.index_base.hash`

**`clear(self)`**

Overrides: `gmisclib.mcmc_indexclass.index_base.clear`

**`set_prms(self, prm)`**

This sets the vector of parameters to be used and modified. Note that `p_lower()` can affect the `prm` argument! No copy is made and this side effect is needed for the `problem_definition.fixer()` function.

This function should only be called after the map is established.

**`set_prm(self, v, *key)`**

This sets one element of the parameters vector. This function should only be called after the map is established; it cannot be used to set fixed parameters.

**`p(self, *key)`**

**Parameters**

**key:** (*type=tuple(str)*)

**Return Value**

the parameter indexed by **\*key**.

(*type=float*)

Overrides: `gmisclib.mcmc_indexclass.index_base.p`

**`p_lower(self, ll, *key)`**

**Raises**

**KeyError** when you apply it to a fixed parameter below the specified `ll`.

Overrides: `gmisclib.mcmc_indexclass.index_base.p_lower`

**Note:** this may have side effects on the array passed to `prms()`.

---

**p\_range**(*self*, *lb*, *ub*, \**key*)

---

**Raises**

**KeyError** when you apply it to a fixed parameter outside specified range (*lb*,*ub*).

Overrides: *gmisclib.mcmc\_indexclass.index\_base.p\_range*

**Note:** this may have side effects on the array passed to *prms()*.

---



---

**p\_periodic**(*self*, *ub*, \**key*)

---

A range with periodic boundary conditions 0 to *ub*.

**Raises**

**KeyError** when you apply it to a fixed parameter outside specified range (*lb*,*ub*).

**Note:** this may have side effects on the array passed to *prms()*.

---



---

**pc**(*self*, \**key*)

---

**Parameters**

**key:** (*type=tuple(str)*)

**Return Value**

the parameter indexed by \**key*.

(*type=float*)

---



---

**pc\_lower**(*self*, *ll*, \**key*)

---

**Raises**

**KeyError** when you apply it to a fixed parameter below the specified *ll*.

**Note:** this may have side effects on the array passed to *prms()*.

---



---

**get\_prms**(*self*)

---

**Return Value**

a vector of the adjustable parameters

(*type=****numpy.ndarray***)

Overrides: *gmisclib.mcmc\_indexclass.index\_base.get\_prms*

---

**toDisplay**(*self*, *reprkeys*=None)

**Note:** This prints the values without the folding or wrapping that is done by `p_range`, `p_periodic`, `p_lower`, or `p_upper`. So, you should not be surprised or disturbed if you see a value outside the expected range. If you extract the values normally, using the `p_*` accessor functions, all will be well.

**\_\_repr\_\_**(*self*)

`repr(x)`

Overrides: `object.__repr__` `exitit`(inherited documentation)

**items**(*self*)

**Return Value**

a sequence of all the parameter, fixed or adjustable.

(*type*=sequence( *key*, *value* ) ) where **key** could be anything hashable, but is usually a tuple of strings, and **value** is a *float*.)

**keys**(*self*)

**\_\_call\_\_**(*self*)

**columns**(*self*)

**comment**(*self*, *c*)

**get\_all**(*self*)

**get\_fixed**(*self*)

Get only the fixed parameters.

**Return Value**

a dictionary from parameter name to value.

**i\_k\_val**(*self*)

**logdens**(*self*, *x*)

**n**(*self*)

**Return Value**

the number of adjustable parameters.

(*type=int*)

**Note:** to include the number of fixed parameters, you can take `len(self._p)`.

**pN**(*self*, *num*, \**key*)

**pNI**(*self*, *num*, *low*, *high*, \**key*)

**pNr**(*self*, *num*, *low*, *high*, \**key*)

**set\_all\_fixed**(*self*, *kv*)

This sets a whole dictionary's worth of fixed parameters.

**Parameters**

**kv:** parameters to set

(*type=dict*)

**set\_fixed**(*self*, *v*, \**k*)

You can have "fixed" parameters that are not updated by the MCMC optimizer. This sets one of them.

**Parameters**

**v:** the value

**k:** the key.

***Inherited from object***

`__delattr__()`, `__format__()`, `__getattr__()`, `__hash__()`, `__new__()`, `__reduce__()`, `__reduce_ex__()`, `__setattr__()`, `__sizeof__()`, `__str__()`, `__subclasshook__()`

**77.9.2 Properties**

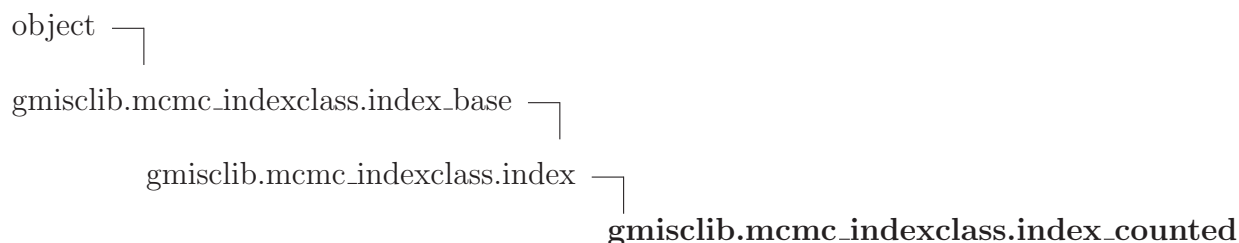
Name	Description
<i>Inherited from object</i>	
<code>__class__</code>	

**77.9.3 Class Variables**



Name	Description
<code>reprkeys</code>	<b>Value:</b> <code>re.compile(r'.')</code>

## 77.10 Class `index_counted`



This is a special-purpose wrapped version of `index`. Its only function is to make sure that all parameters were used, in addition to the normal checking that you do not use any parameters that are undefined.

Use it exactly as you would use `index`, but it is slower.

### 77.10.1 Methods

**`__init__(self, idxr)`**

`x.__init__(...)` initializes `x`; see `help(type(x))` for signature

Overrides: `object.__init__` extit(inherited documentation)

**`p(self, *key)`**

**Return Value**

the parameter indexed by `*key`.

(*type=float*)

Overrides: `gmisclib.mcmc_indexclass.index_base.p`

**`p_lower(self, ll, *key)`**

**Raises**

**KeyError** when you apply it to a fixed parameter below the specified ll.

Overrides: `gmisclib.mcmc_indexclass.index_base.p_lower`

**p\_range**(*self*, *lb*, *ub*, \**key*)

**Raises**

**KeyError** when you apply it to a fixed parameter outside specified range (*lb*,*ub*).

Overrides: `gmisclib.mcmc_indexclass.index_base.p_range`

**p\_periodic**(*self*, *ub*, \**key*)

A range with periodic boundary conditions 0 to *ub*.

**Raises**

**KeyError** when you apply it to a fixed parameter outside specified range (*lb*,*ub*).

Overrides: `gmisclib.mcmc_indexclass.index.p_periodic` extit(inherited documentation)

**confirm\_all\_used**(*self*)

This can be used to check that we aren't using an indexer with a problem that is too small (for instance, if the indexer was read in from a log file).

**Raises**

**IndexKeyError** if there are some keys in the index that have not been used.

**\_\_call\_\_**(*self*)

**\_\_repr\_\_**(*self*)

`repr(x)`

Overrides: `object.__repr__` extit(inherited documentation)

**clear**(*self*)

Overrides: `gmisclib.mcmc_indexclass.index_base.clear`

**columns**(*self*)

**comment**(*self*, *c*)

**get\_all**(*self*)

**get\_fixed**(*self*)

Get only the fixed parameters.

**Return Value**

a dictionary from parameter name to value.

**get\_prms**(*self*)

**Return Value**

a vector of the adjustable parameters

(*type=*`numpy.ndarray`)

Overrides: `gmisclib.mcmc_indexclass.index_base.get_prms`

**hash**(*self*)

Overrides: `gmisclib.mcmc_indexclass.index_base.hash`

**i\_k\_val**(*self*)

**items**(*self*)

**Return Value**

a sequence of all the parameter, fixed or adjustable.

(*type=*`sequence( (key, value) )` where **key** could be anything hashable, but is usually a tuple of strings, and **value** is a `float`.)

**keys**(*self*)

**logdens**(*self*, *x*)

**n**(*self*)

**Return Value**

the number of adjustable parameters.

(*type=*`int`)

**Note:** to include the number of fixed parameters, you can take `len(self._p)`.

**pN**(*self*, *num*, \**key*)

**pNI**(*self*, *num*, *low*, *high*, \**key*)

**pNr**(*self*, *num*, *low*, *high*, \**key*)

---

**pc**(*self*, \**key*)

---

**Parameters***key*: (*type=tuple(str)*)**Return Value**the parameter indexed by \**key*.(*type=float*)

---

**pc\_lower**(*self*, *ll*, \**key*)

---

**Raises****KeyError** when you apply it to a fixed parameter below the specified *ll*.**Note:** this may have side effects on the array passed to `prms()`.

---

**set\_all\_fixed**(*self*, *kv*)

---

This sets a whole dictionary's worth of fixed parameters.

**Parameters***kv*: parameters to set(*type=dict*)

---

**set\_fixed**(*self*, *v*, \**k*)

---

You can have "fixed" parameters that are not updated by the MCMC optimizer. This sets one of them.

**Parameters***v*: the value*k*: the key.

---

**set\_prm**(*self*, *v*, \**key*)

---

This sets one element of the parameters vector. This function should only be called after the map is established; it cannot be used to set fixed parameters.

---

**set\_prms**(*self*, *prm*)

---

This sets the vector of parameters to be used and modified. Note that `p_lower()` can affect the `prm` argument! No copy is made and this side effect is needed for the `problem_definition.fixer()` function.

This function should only be called after the map is established.

<b>toDisplay</b> ( <i>self</i> , <i>reprkeys</i> =None)
---

<p><b>Note:</b> This prints the values without the folding or wrapping that is done by <code>p_range</code>, <code>p_periodic</code>, <code>p_lower</code>, or <code>p_upper</code>. So, you should not be surprised or disturbed if you see a value outside the expected range. If you extract the values normally, using the <code>p_*</code> accessor functions, all will be well.</p>
---

### *Inherited from object*

`__delattr__()`, `__format__()`, `__getattr__()`, `__hash__()`, `__new__()`, `__reduce__()`, `__reduce_ex__()`,  
`__setattr__()`, `__sizeof__()`, `__str__()`, `__subclasshook__()`

#### 77.10.2 Properties

Name	Description
<i>Inherited from object</i>	
<code>__class__</code>	

#### 77.10.3 Class Variables

Name	Description
<code>reprkeys</code>	<b>Value:</b> <code>re.compile(r'.'</code> )

## 78 Module `gmisclib.mcmc_logger`

### 78.1 Functions

**`read_currentlist(fname, trigger=None)`**

Starting from a file name, read in a log file.

**Parameters**

**fname:** name of a (possibly compressed) log file.

(*type=*`str`)

**trigger:** where to start in the log file. See `readiter`.

**Return Value**

a list of data lines and header information. A data line is a dictionary, containing parameters and other information from one iteration.

(*type=*`tuple(list(dict(str:str)), dict(str:str))`)

**Note:** this is very similar to `newstem2.newlogger.read_currentlist` except that returns a more complicated data structure that has one more level of indirection.

**`read_multisample(fname, Nsamp=10, tail=0.0, trigger=None, reader=<function read_currentlist at 0x4aa0ed8>)`**

Read in a log file: it gives you multiple samples from the log. In other words, it provides a time-series of the changes to the parameters.

**Parameters**

**tail:** Where to start in the file? `tail=0` means that you start at the trigger or beginning. `tail=1-epsilon` means you take just the last tiny bit of the file.

**reader:** A function that sucks in the data from the log file, yielding a list of iterations and an overall header.

(*type=*`function(str, trigger) -> (list(dict(str:str)), dict(str:str))`)

**Return Value**

(*type=*`(dict(str:str), list(indexer), list(float))`)

**Raises**

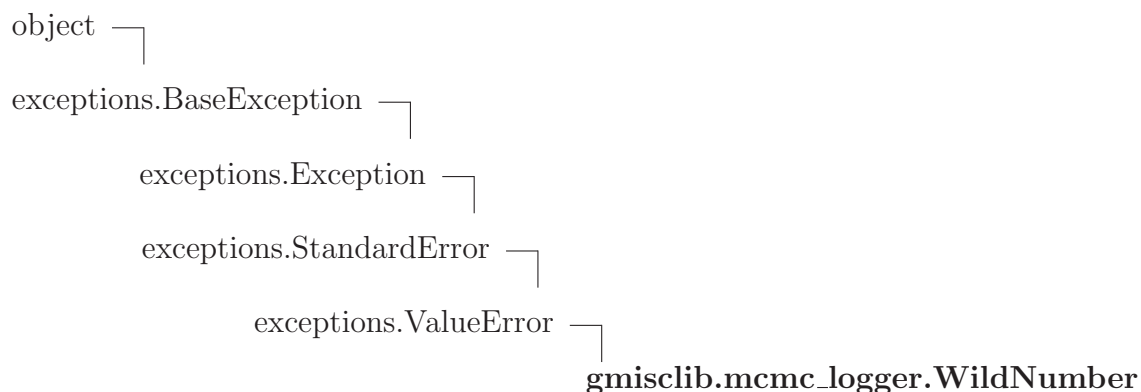
`NoDatError` when the log file is empty.

`BadFormatError` when a parameter appears mid-way through the optimization or an iteration cannot be read.

## 78.2 Variables

Name	Description
<code>--package--</code>	<b>Value:</b> 'gmisclib'

## 78.3 Class WildNumber



### 78.3.1 Methods

```

__init__(self, *s)
x.__init__(...) initializes x; see help(type(x)) for signature
Overrides: object.__init__ extit(inherited documentation)

```

*Inherited from exceptions.ValueError*

```
__new__()
```

*Inherited from exceptions.BaseException*

```

__delattr__(), __getattr__(), __getitem__(), __getslice__(), __reduce__(), __repr__(),
__setattr__(), __setstate__(), __str__(), __unicode__()

```

*Inherited from object*

```
__format__(), __hash__(), __reduce_ex__(), __sizeof__(), __subclasshook__()
```

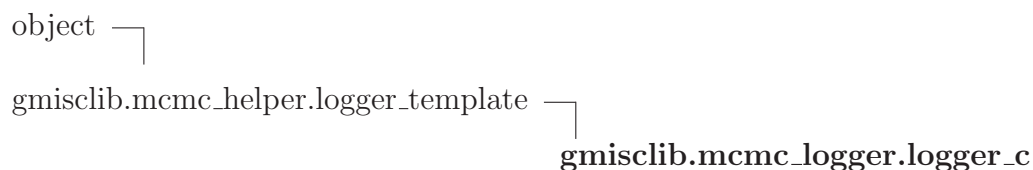
### 78.3.2 Properties

Name	Description
<i>Inherited from exceptions.BaseException</i>	

*continued on next page*

Name	Description
args, message	
<i>Inherited from object</i>	
<code>__class__</code>	

## 78.4 Class `logger_c`



**Known Subclasses:** `gmisclib.mcmc_logger.logger_A`, `gmisclib.mcmc_logger.logger_N`

This class is called from the Markov-Chain stepper and writes the log file.

### 78.4.1 Methods

```
__init__(self, logwriter, iterstep=100, ergstep=0.1)
```

`x.__init__(...)` initializes `x`; see `help(type(x))` for signature

Overrides: `object.__init__` `exitit` (inherited documentation)

```
set_logstep(self, iterstep, ergstep, reason='sampled')
```

```
add(self, stepperInstance, iter)
```

Called every step: this decides whether data should be logged.

Overrides: `gmisclib.mcmc_helper.logger_template.add`

```
Add(self, stepperInstance, iter, reason=None)
```

```
close(self)
```

Overrides: `gmisclib.mcmc_helper.logger_template.close`

```
comment(self, comment)
```

```
header(self, k, v)
```



<code>headers(self, hdict)</code>
-----------------------------------

<code>reset(self)</code>
--------------------------

Overrides: <code>gmisclib.mcmc_helper.logger_template.reset</code>
--

<code>set_trigger_point(self)</code>
--------------------------------------

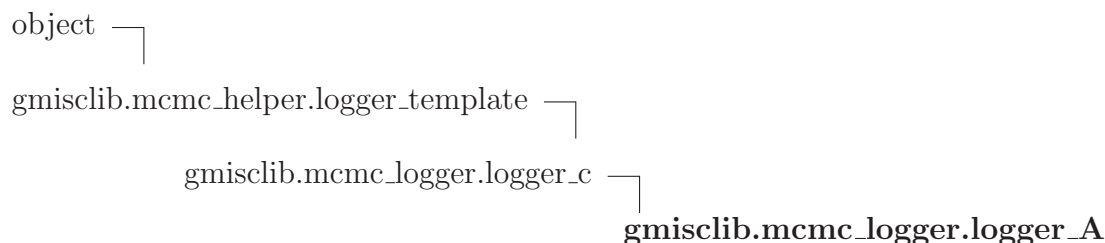
### *Inherited from object*

`__delattr__()`, `__format__()`, `__getattr__()`, `__hash__()`, `__new__()`, `__reduce__()`, `__reduce_ex__()`, `__repr__()`, `__setattr__()`, `__sizeof__()`, `__str__()`, `__subclasshook__()`

#### 78.4.2 Properties

Name	Description
<i>Inherited from object</i>	
<code>__class__</code>	

## 78.5 Class `logger_A`



### 78.5.1 Methods

<code>__init__(self, logwriter, iterstep=100, ergstep=0.1)</code>
---

logwriter is a <code>fiatio.writer</code> or <code>avio.writer</code> or similar instance.
--

Overrides: <code>object.__init__</code>
---

<code>ok_logp(self, logpval)</code>
-------------------------------------

<code>ok_prms(self, prms)</code>
----------------------------------

<code>fnt(self, prmname)</code>
---------------------------------

<code>get_map(self, stepper)</code>
-------------------------------------

<code>Add(self, stepperInstance, iter, reason=None)</code>
--

Overrides: <code>gmisclib.mcmc_logger.logger_c.Add</code>
---

<code>add(self, stepperInstance, iter)</code>
---

Called every step: this decides whether data should be logged.
--

Overrides: <code>gmisclib.mcmc_helper.logger_template.add</code>
--

<code>close(self)</code>
--------------------------

Overrides: <code>gmisclib.mcmc_helper.logger_template.close</code>
--

<code>comment(self, comment)</code>
-------------------------------------

<code>header(self, k, v)</code>
---------------------------------

<code>headers(self, hdict)</code>
-----------------------------------

<code>reset(self)</code>
--------------------------

Overrides: <code>gmisclib.mcmc_helper.logger_template.reset</code>
--

<code>set_logstep(self, iterstep, ergstep, reason='sampled')</code>
---

<code>set_trigger_point(self)</code>
--------------------------------------

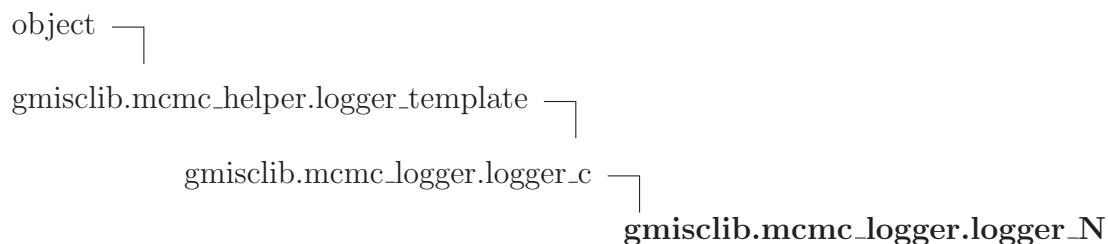
### *Inherited from object*

`__delattr__()`, `__format__()`, `__getattr__()`, `__hash__()`, `__new__()`, `__reduce__()`, `__reduce_ex__()`,  
`__repr__()`, `__setattr__()`, `__sizeof__()`, `__str__()`, `__subclasshook__()`

### 78.5.2 Properties

Name	Description
<i>Inherited from object</i>	
<code>__class__</code>	

## 78.6 Class *logger\_N*



### 78.6.1 Methods

**`__init__(self, logwriter, iterstep=100, ergstep=0.1)`**

logwriter is really a newstem2.newlogger instance.

Overrides: `object.__init__`

**`Add(self, stepperInstance, iter, reason=None)`**

Actually write a set of parameters into the log file.

Overrides: `gmisclib.mcmc_logger.logger_c.Add`

**`add(self, stepperInstance, iter)`**

Called every step: this decides whether data should be logged.

Overrides: `gmisclib.mcmc_helper.logger_template.add`

**`close(self)`**

Overrides: `gmisclib.mcmc_helper.logger_template.close`

**`comment(self, comment)`**

**`header(self, k, v)`**

**`headers(self, hdict)`**

**`reset(self)`**

Overrides: `gmisclib.mcmc_helper.logger_template.reset`

**`set_logstep(self, iterstep, ergstep, reason='sampled')`**

`set_trigger_point(self)`

*Inherited from `object`*

`__delattr__()`, `__format__()`, `__getattribute__()`, `__hash__()`, `__new__()`, `__reduce__()`, `__reduce_ex__()`,  
`__repr__()`, `__setattr__()`, `__sizeof__()`, `__str__()`, `__subclasshook__()`

### 78.6.2 Properties

Name	Description
<i>Inherited from <code>object</code></i>	
<code>__class__</code>	

## 78.7 Class `lti_c`

```

object └─
          gmisclib.mcmc_logger.lti_c
  
```

### 78.7.1 Methods

`__init__(self)`

`x.__init__(...)` initializes `x`; see `help(type(x))` for signature

Overrides: `object.__init__` `extit`(inherited documentation)

`parse(self, d)`

This constructs an `newstem2.indexclass.index` object from a line from a log file.

**Parameters**

`d`: a line from a log file specifying information for an iteration.

(*type=*`dict(str:str)`)

**Return Value**

the parameters for that iteration.

(*type=*`newstem2.indexclass.index`)

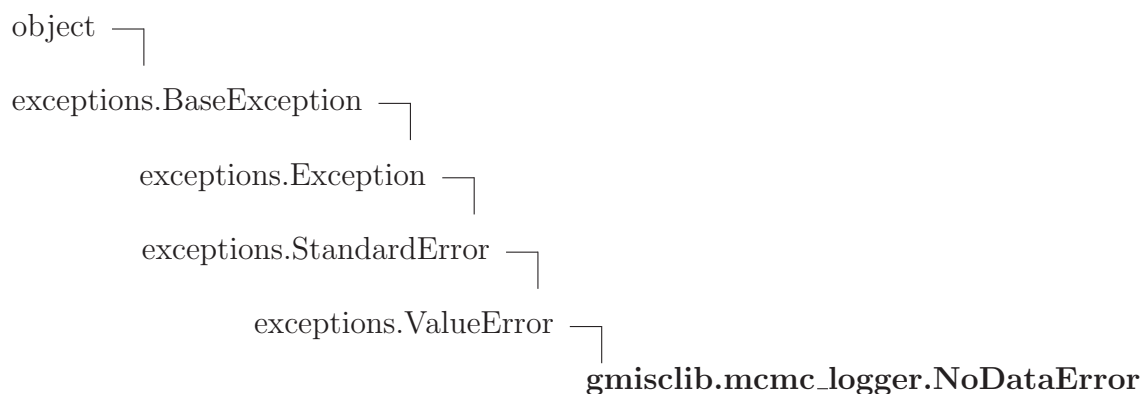
*Inherited from `object`*

`__delattr__()`, `__format__()`, `__getattribute__()`, `__hash__()`, `__new__()`, `__reduce__()`, `__reduce_ex__()`,  
`__repr__()`, `__setattr__()`, `__sizeof__()`, `__str__()`, `__subclasshook__()`

### 78.7.2 Properties

Name	Description
<i>Inherited from object</i>	
<code>__class__</code>	

## 78.8 Class `NoDataError`



### 78.8.1 Methods

```

__init__(self, *s)

x.__init__(...) initializes x; see help(type(x)) for signature
Overrides: object.__init__ extit(inherited documentation)

```

*Inherited from `exceptions.ValueError`*

```
__new__()
```

*Inherited from `exceptions.BaseException`*

```

__delattr__(), __getattr__(), __getitem__(), __getslice__(), __reduce__(), __repr__(),
__setattr__(), __setstate__(), __str__(), __unicode__()

```

*Inherited from `object`*

```
__format__(), __hash__(), __reduce_ex__(), __sizeof__(), __subclasshook__()
```

### 78.8.2 Properties

Name	Description
	<i>Inherited from exceptions.BaseException</i>
	args, message
	<i>Inherited from object</i>
	<code>__class__</code>

## 79 Module *gmisclib.mcmc.logtools*

### 79.1 Functions

**read\_many\_files**(*fnames, uid, Nsamp, tail, trigger*)

OBSOLETE

**Return Value**

a dictionary mapping filenames onto "data" and header information. Header information is from the last file read. The "data" are *onelog* class instances that encapsulate log information from one log file.

(*type=tuple(dict(str:onelog), dict(str:str))*)

**read\_uid\_many\_files**(*fnames, uid, Nsamp, tail, trigger*)

**Return Value**

a dictionary mapping filenames onto "data" and header information. Header information is from the last file read. The "data" are *onelog* class instances that encapsulate log information from one log file.

(*type=tuple(dict(str:onelog), dict(str:str))*)

**get\_pmap**(*per\_fn*)

**Parameters**

*per\_fn*: (*type=dict(str: mcmc\_newlogger.logline)*)

**list\_prm\_samples**(*per\_fn, sample\_selector, out*)

**Parameters**

*per\_fn*: (*type=dict(str: mcmc\_newlogger.logline)*)

**indexer\_covar**(*per\_fn, sample\_selector, weight\_by\_T=False*)

**Parameters**

*per\_fn*: (*type=dict(str: mcmc\_newlogger.logline)*)

**logp\_stdev**(*per\_fn, sample\_selector*)

**Parameters**

*per\_fn*: (*type=dict(str: mcmc\_newlogger.logline)*)

**all**(*per\_fn, source=None*)

This selects all measurements.

**some\_after\_convergence**(*per\_fn*, *source*=None)

This selects which measurements will be used. It looks after convergence, then throws out optimizations that haven't converged.

**near\_each\_max**(*per\_fn*, *source*=None)

This selects which measurements will be used. It looks after convergence, then gives you the best few results from each run.

**Parameters**

*per\_fn*: (*type*=dict(*str*: *mcmc\_newlogger.logline*))

**last**(*per\_fn*, *source*=None)

This selects which measurements will be used.

**Parameters**

*per\_fn*: (*type*=dict(*str*: *mcmc\_newlogger.logline*))

**each\_best**(*per\_fn*, *source*=None)

This selects which measurements will be used.

**overall\_best**(*per\_fn*, *source*=None)

This selects which measurements will be used.

**indexer\_stdev**(*per\_fn*, *selector*, *weight\_by\_T*=False)

Return a summary of the properties of the selected indexers.

**Parameters**

*selector*: A generator that selects some of the loglines in *per\_fn* that meet certain selection criteria.

(*type*=function(dict(*filename*:  
list(*mcmc\_newlogger.logline*))) that yields  
*mcmc\_newlogger.logline*)

**print\_index\_error**(*ke*)

**drop\_files**(*per\_fn*, *FileDropFac*, *Trim*=None, *Stretch*=None)

**Parameters**

*per\_fn*: produced by *LT.read\_uid\_many\_files()*



**ascii\_cmp**(*a*, *b*)

Compares the ASCII form of keys, for sorting purposes. Does a good attempt at ASCII ordering for strings and numeric ordering for numbers.

**key\_cmp**(*a*, *b*)

Compares the tuple form of keys, for sorting purposes. Does a good attempt at ASCII ordering for strings and numeric ordering for numbers.

**P\_bayes\_list**(*per\_fn*, *argv*, *m*, *arg*, *selector*, *hdr*)

$\text{Log}(\text{PosteriorMarginalLikelihood})$  is the posterior marginal likelihood of this model:  $\text{P\_posterior}[\text{Data}|\text{model}] = \text{integral}(\text{P}[\text{D}|\text{params},\text{model}] * \text{P}[\text{params}|\text{data},\text{model}] * \text{d\_params})$ . It's the average of  $\text{P}[\text{D}|\text{params},\text{model}]$  over the posterior distribution of  $\text{P}[\text{params}|\text{data},\text{model}]*\text{Prior}(\text{params})$ .

References are:

- Rampal S. Etienne, Han Olf (2005) Confronting different models of community structure to species-abundance data: a Bayesian model comparison *Ecology Letters* 8 (5) , 493-504  
doi:10.1111/j.1461-0248.2005.00745.x

and that references

- M. Aitkin 1991: Posterior Bayes Factors, *J. of the Royal Statistical Soc. B* 53: 111-142
- P. W. Laud and J. G. Ibrahim 1995: Predictive Model Selection, *J. of the Royal Statistical Soc. B* 57: 247-262.
- F. De Santis and F. Spezzaferri 1997: Alternative Bayes Factors for Model Selection, *Canadian J. of Statistics* 25: 503-515
- S. K. Upadhyay and M. Peswani 2003: Choice between Weibull and Lognormal Models: a simulation based Bayesian Study *Communications in Statistics: Theory and Methods* 32: 318-405
- P. K. Vlachos and A. E. Gelfand 2003: On the calibration of Bayeseian model choice criteria, *J. of Statistical Planning and Inference* 111: 223-234
- R. E. Kass and A. E. Raftery 1995: Bayes Factors, *J. of the American Statistical Assoc.* 90: 773-795

The other thing,  $\text{Log}(\text{BayesWeightedBayes})$  is the average of  $\text{P}(\text{D}|\text{params},\text{M})*\text{Prior}(\text{params})$  over the posterior distribution of  $\text{P}[\text{params}|\text{data},\text{model}]*\text{Prior}(\text{params})$ . It has no real statistical backing, but it's a crude approximation for the Bayes evidence itself, the normalized  $\text{P}[\text{params}|\text{data},\text{model}]*\text{Prior}(\text{params})$ .

**P\_bayes**(*per\_fn*, *argv*, *m*, *arg*, *selector*, *hdr*)

## 79.2 Variables

Name	Description
FILE_DROP_FAC	<b>Value:</b> 0.2
TRIGGER	<b>Value:</b> 'run_to_bottom finished'
__package__	<b>Value:</b> 'gmisclib'

## 79.3 Class *onelog*

### 79.3.1 Methods

**\_\_init\_\_**(*self*, *logps*, *indexers*, *fname*)

## 80 Module *gmisclib.mcmc\_m4p*

This is a helper module to make use of *mcmc.py* and *mcmc\_big.py*. It allows you to conveniently run a Monte-Carlo simulation of any kind until it converges (*stepper.run\_to\_bottom*) or until it has explored a large chunk of parameter space (*stepper.run\_to\_ergodic*).

It also helps you with logging the process.

When run in parallel, each processor does its thing more-or-less independently. However, every few steps, they exchange notes on their current progress. If one finds an improved vertex, it will be passed on to other processors via MPI.

Tested with *mpi4py* version 1.1.0, <http://mpi4py.scipy.org>

### 80.1 Functions

<b>precompute_logp</b> ( <i>lop</i> )
---------------------------------------

Does a parallel evaluation of logp for all items in <i>lop</i> .
--

<b>is_root</b> ()
-------------------

<b>size</b> ()
----------------

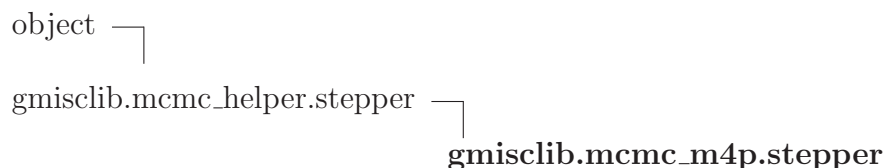
<b>rank</b> ()
----------------

<b>test_</b> ()
-----------------

### 80.2 Variables

Name	Description
Debug	<b>Value:</b> 0
mpi	<b>Value:</b> <code>MPI.COMM_WORLD</code>

## 80.3 Class *stepper*



### 80.3.1 Methods

**`__init__(self, x, maxloops=-1, logger=None)`**

`x.__init__(...)` initializes `x`; see `help(type(x))` for signature

Overrides: `object.__init__` extit(inherited documentation)

**`reset_loops(self, maxloops=-1)`**

Overrides: `gmisclib.mcmc_helper.stepper.reset_loops`

**`communicate_hook(self, id)`**

Overrides: `gmisclib.mcmc_helper.stepper.communicate_hook`

**`join(self, id)`**

**`note(self, s, lvl)`**

Overrides: `gmisclib.mcmc_helper.stepper.note`

**`close(self)`**

After calling `close()`, it is no longer legal to call `run_to_change()`, `run_to_ergodic()`, or `run_to_bottom()`. Logging is shut down, but the contents of the stepper are still available for inspection.

**run\_to\_bottom**(*self*, *ns*=3, *acceptable\_step*=None)

Run the *Markov Chain Monte-Carlo* until it converges near a minimum.

The general idea of the termination condition is that it keeps track of the angle between successive sequences of steps, where each sequence contains **ns** steps. If the angle is less than 90 degrees, it is evidence that the solution is still drifting in some consistent direction and therefore not yet at the maximum. Angles of greater than 90 degrees suggest that the optimization is wandering aimlessly near a minimum. It will run until a sufficient number of large angles are seen since the last BMCMC reset. A similar check is made on individual components of the parameter vector.

**Parameters**

**ns**: This controls how carefully it checks that it has really found a minimum. Large values will take longer but will assure more accurate convergence.

(*type*=**int**)

**acceptable\_step**: A callable object (i.e. class or function) that returns **True** or **False**, depending on whether a proposed step is acceptable. See `mcmc.T_acceptor` for an example.

**Raises**

`TooManyLoops` ?

---

**run\_to\_change**(*self*, *ncw*=1, *acceptable\_step*=None, *update\_T*=False)

---

Run the *Markov Chain Monte-Carlo* until it finds an acceptable step.

#### Parameters

- ncw:** How many steps should be accepted before it stops?  
(*type=int*)
- acceptable\_step:** A function that decides what steps are acceptable and what are not. This is passed to the MCMC stepper, `mcmc.BootStepper`.  
(*type=function(float) -> bool, often `mcmc.T_acceptor` or `step_acceptor`*)
- update\_T:** Obsolete. Must be False.  
(*type=boolean.*)

#### Raises

- TooManyLoops** if it takes a long time before enough steps are accepted.

---

**run\_to\_ergodic**(*self*, *ncw*=1, *T*=1.0)

---

Run the stepper until it has explored all of parameter space **ncw** times (as best as we can estimate). Note that this is a pretty careful routine. If the stepper finds a better region of parameter space so that log(P) improves part way through the process, the routine will reset itself and begin again.

NOTE: this sets **T** and **sortstrategy** for the **stepper**.

#### Parameters

- ncw:** how many times (or what fraction) of an ergodic exploration to make.  
(*type=int.*)
- T:** temperature at which to run the Monte-Carlo process. This is normally 1.0. If it is **None**, then the current temperature is not modified.  
(*type=float or None.*)

#### Return Value

- a **position** at the end of the process.

---

**synchronize\_abort**(*self*, *id*)

---



---

**synchronize\_end**(*self*, *id*)

---

<code>synchronize_start(<i>self</i>, <i>id</i>)</code>
--

*Inherited from object*

`__delattr__()`, `__format__()`, `__getattr__()`, `__hash__()`, `__new__()`, `__reduce__()`, `__reduce_ex__()`,  
`__repr__()`, `__setattr__()`, `__sizeof__()`, `__str__()`, `__subclasshook__()`

### 80.3.2 Properties

Name	Description
<i>Inherited from object</i>	
<code>__class__</code>	

### 80.3.3 Class Variables

Name	Description
<code>MPIID</code>	<b>Value:</b> 1241

## 81 Module `gmisclib.mcmc_mpi`

This is a helper module to make use of `mcmc.py` and `mcmc_big.py`. It allows you to conveniently run a Monte-Carlo simulation of any kind until it converges (`stepper.run_to_bottom`) or until it has explored a large chunk of parameter space (`stepper.run_to_ergodic`).

It also helps you with logging the process.

When run in parallel, each processor does its thing more-or-less independently. However, every few steps, they exchange notes on their current progress. If one finds an improved vertex, it will be passed on to other processors via MPI.

### 81.1 Functions

<b><code>precompute_logp(logp)</code></b>
---

Does a parallel evaluation of <code>logp</code> for all items in <code>logp</code> .
--

<b><code>test_()</code></b>
-----------------------------

### 81.2 Variables

Name	Description
Debug	Value: 0

### 81.3 Class `stepper`

object └─

`gmisclib.mcmc_helper.stepper` └─

**`gmisclib.mcmc_mpi.stepper`**

#### 81.3.1 Methods

<b><code>__init__(self, x, maxloops=-1, logger=None, share=None)</code></b>
---

<code>x.__init__(...)</code> initializes <code>x</code> ; see <code>help(type(x))</code> for signature
--

Overrides: <code>object.__init__</code> <code>exitit</code> (inherited documentation)
---



**reset\_loops**(*self*, *maxloops*=-1)Overrides: *gmisclib.mcmc\_helper.stepper.reset\_loops***communicate\_hook**(*self*, *id*)Overrides: *gmisclib.mcmc\_helper.stepper.communicate\_hook***synchronize\_start**(*self*, *id*)Overrides: *gmisclib.mcmc\_helper.stepper.synchronize\_start***synchronize\_end**(*self*, *id*)Overrides: *gmisclib.mcmc\_helper.stepper.synchronize\_end***synchronize\_abort**(*self*, *id*)Overrides: *gmisclib.mcmc\_helper.stepper.synchronize\_abort***synchronize**(*self*, *id*)**note**(*self*, *s*, *lvl*)Overrides: *gmisclib.mcmc\_helper.stepper.note***size**(*self*)**rank**(*self*)**close**(*self*)

After calling `close()`, it is no longer legal to call `run_to_change()`, `run_to_ergodic()`, or `run_to_bottom()`. Logging is shut down, but the contents of the stepper are still available for inspection.

**run\_to\_bottom**(*self*, *ns*=3, *acceptable\_step*=None)

Run the *Markov Chain Monte-Carlo* until it converges near a minimum.

The general idea of the termination condition is that it keeps track of the angle between successive sequences of steps, where each sequence contains **ns** steps. If the angle is less than 90 degrees, it is evidence that the solution is still drifting in some consistent direction and therefore not yet at the maximum. Angles of greater than 90 degrees suggest that the optimization is wandering aimlessly near a minimum. It will run until a sufficient number of large angles are seen since the last BMCMC reset. A similar check is made on individual components of the parameter vector.

**Parameters**

**ns**: This controls how carefully it checks that it has really found a minimum. Large values will take longer but will assure more accurate convergence.

(*type*=**int**)

**acceptable\_step**: A callable object (i.e. class or function) that returns **True** or **False**, depending on whether a proposed step is acceptable. See `mcmc.T_acceptor` for an example.

**Raises**

`TooManyLoops` ?

---

**run\_to\_change**(*self*, *ncw*=1, *acceptable\_step*=None, *update\_T*=False)

---

Run the *Markov Chain Monte-Carlo* until it finds an acceptable step.

#### Parameters

- ncw:** How many steps should be accepted before it stops?  
(*type*=int)
- acceptable\_step:** A function that decides what steps are acceptable and what are not. This is passed to the MCMC stepper, `mcmc.BootStepper`.  
(*type*=function(float) -> bool, often *mcmc.T\_acceptor* or *step\_acceptor*)
- update\_T:** Obsolete. Must be False.  
(*type*=boolean.)

#### Raises

- TooManyLoops** if it takes a long time before enough steps are accepted.

---

**run\_to\_ergodic**(*self*, *ncw*=1, *T*=1.0)

---

Run the stepper until it has explored all of parameter space **ncw** times (as best as we can estimate). Note that this is a pretty careful routine. If the stepper finds a better region of parameter space so that log(P) improves part way through the process, the routine will reset itself and begin again.

NOTE: this sets **T** and **sortstrategy** for the **stepper**.

#### Parameters

- ncw:** how many times (or what fraction) of an ergodic exploration to make.  
(*type*=int.)
- T:** temperature at which to run the Monte-Carlo process. This is normally 1.0. If it is **None**, then the current temperature is not modified.  
(*type*=float or None.)

#### Return Value

- a **position** at the end of the process.

### *Inherited from object*

`--delattr--()`, `--format--()`, `--getattr--()`, `--hash--()`, `--new--()`, `--reduce--()`, `--reduce_ex--()`, `--repr--()`, `--setattr--()`, `--sizeof--()`, `--str--()`, `--subclasshook--()`

**81.3.2 Properties**

Name	Description
<i>Inherited from object</i> __class__	

**81.3.3 Class Variables**

Name	Description
MPIID	<b>Value:</b> 1241

## 82 Module *gmisclib.mcmc\_newlogger*

### 82.1 Functions

<code>ok(<i>x</i>)</code>
---------------------------

<b><code>readiter(<i>logfd</i>, <i>trigger</i>=None)</code></b>
---

Reads in one line at a time from a log file. This is an iterator. At every non-comment line, it yields an output.

**Parameters**

**logfd:** a file descriptor for a log file produced by this module.

**trigger:** this is None or a regular expression that specifies when to start paying attention to the data. If **trigger** is not None, body lines before **trigger** matches are ignored. Note that **trigger** is matched to a line with trailing whitespace stripped away.

**Return Value**

(Yields) (**hdr**, **x**) where **hdr** is a dictionary of the header information current at that point in the file and **x** is a **logline** class containing the current line's data.

**Raises**

BadFormatError ???

KeyError ???

<b><code>read_raw(<i>logfd</i>)</code></b>
--

Read log information in raw line-by-line form.

**read\_log(*fname*, *which*)**

Read in a log file, returning information that you can use to restart the optimizer. The log file must be produced by `logger.add()`.

**Parameters**

**fname**: the filename to read.

(*type=*`str`)

**which**: a string that tells which log entry to grab.

- "last": the final logged entry;
- "index%d" (i.e. "index" followed directly by an integer): the %dth log entry (negative numbers count backwards from the end), so "index-1" is the final logged entry and "index0" is the first;
- "frac%g" (i.e. "frac" followed by a float): picks a log entry as a fraction of the length of the log. "frac0" is the first, and "frac1" is the final entry.
- "best": picks the iteration with the most positive logP value.

(*type=*`str`)

**Return Value**

(`hdr`, `stepperstate`, `vlists`, `logps`), where

- `hdr` is all the header information from the log file (e.g. metadata).
- `stepperstate` is a dictionary of `IC.index` instances, one per UID and `newstem2.position.OuterUID`. It will evoke a `True` when passed to `newstem2.position.is_statedict()`. Basically, this is a single moment of the stepper's state.
- `vlists` is a dictionary (again, one per UID plus...) whose values are lists of position vectors that were recently encountered. This is helpful for the optimizer to figure out plausible step sizes. Basically, this is many moments of state information near `idx`.
- `logps` is a dictionary (again...) whose values are recently obtained values for logP for that UID. This is mostly useful for display and debugging purposes.

**Raises**

`NoDataError` ???

---

**read\_multilog**(*fname*, *Nsamp*=10, *tail*=0.5)
 

---

Read in a log file, producing a sample of stepperstate information.

**Parameters**

**tail**: Where to start in the file? **tail**=0 means that you start at the trigger or beginning. **tail**=1-**epsilon** means you take just the last tiny bit of the file.

---

**read\_multi\_uid**(*fname*, *uid*, *Nsamp*=10, *tail*=0.0, *trigger*=None)
 

---

Read in a log file, selecting information only for a particular UID. This *\*doesn't\** read in multiple UIDs, no matter what the name says: the "multi" refers to the fact that it gives you multiple samples from the log. In other words, it provides a time-series of the changes to the parameters.

OBSOLETE.

**Parameters**

**Nsamp**: the maximum number of samples to extract (or None, or -1). None means "as many as are available"; -1 means "as many samples as there are parameters."

(*type*=*int* or *None*.)

**tail**: Where to start in the file? **tail**=0 means that you start at the trigger or beginning. **tail**=1-**epsilon** means you take just the last tiny bit of the file.

**Raises**

**NoDataError** When the log file is empty or a trigger is set but not triggered.

**read\_tail\_uid**(*fname*, *uid*, *Nsamp*=10, *tail*=0.0, *trigger*=None)

Read in a log file, selecting information only for a particular UID. It gives you multiple samples from the log. In other words, it provides a time-series of the changes to the parameters.

#### Parameters

**Nsamp**: the maximum number of samples to extract (or None, or -1).  
None means "as many as are available"; -1 means "as many samples as there are parameters."

(*type*=*int* or *None*.)

**tail**: Where to start in the file? **tail**=0 means that you start at the trigger or beginning. **tail**=1-**epsilon** means you take just the last tiny bit of the file.

#### Return Value

The data file's header, and the selected list of **logline** instances.

(*type*=(*dict*{*str*:*str*}, *list*(*logline*)))

#### Raises

**NoDataError** When the log file is empty or a trigger is set but not triggered.

**read\_human\_fmt**(*fname*)

This reads a more human-readable format, as produced by **print\_log**. It can easily be modified with a text editor to change parameters.

**load\_module**(*phdr*)

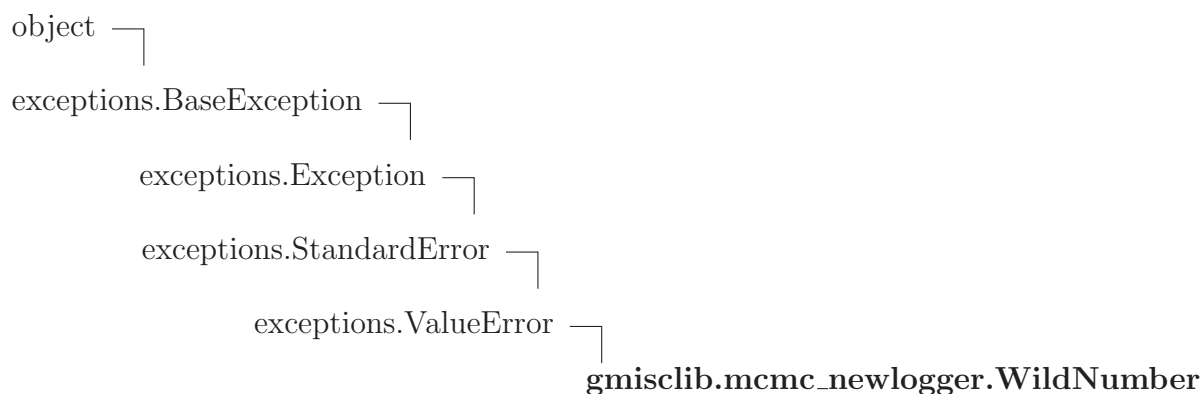
**print\_log**(*fname*, *which*)

## 82.2 Variables

Name	Description
HUGE	<b>Value:</b> 1e+30
__package__	<b>Value:</b> 'gmisclib'



## 82.3 Class WildNumber



### 82.3.1 Methods

**`--init--`**(*self*, \**s*)

*x*.**`--init--`**(...) initializes *x*; see `help(type(x))` for signature

Overrides: `object.--init--` `__init__` (inherited documentation)

#### *Inherited from exceptions.ValueError*

`--new--`()

#### *Inherited from exceptions.BaseException*

`--delattr--`(), `--getattr--`(), `--getitem--`(), `--getslice--`(), `--reduce--`(), `--repr--`(),  
`--setattr--`(), `--setstate--`(), `--str--`(), `--unicode--`()

#### *Inherited from object*

`--format--`(), `--hash--`(), `--reduce.ex--`(), `--sizeof--`(), `--subclasshook--`()

### 82.3.2 Properties

Name	Description
<i>Inherited from exceptions.BaseException</i>	
args, message	
<i>Inherited from object</i>	
<code>--class--</code>	

## 82.4 Class `logger_base`

object └─ `gmisclib.mcmc_newlogger.logger_base`

**Known Subclasses:** `gmisclib.mcmc_newlogger.DBGlogger`, `gmisclib.mcmc_newlogger.logger`

### 82.4.1 Methods

**`__init__(self, fd)`**  
`x.__init__(...)` initializes `x`; see `help(type(x))` for signature  
 Overrides: `object.__init__` `exitit` (inherited documentation)

**`flush(self)`**

**`close(self)`**

**`header(self, k, v)`**

**`headers(self, hdict)`**

**`comment(self, comment)`**

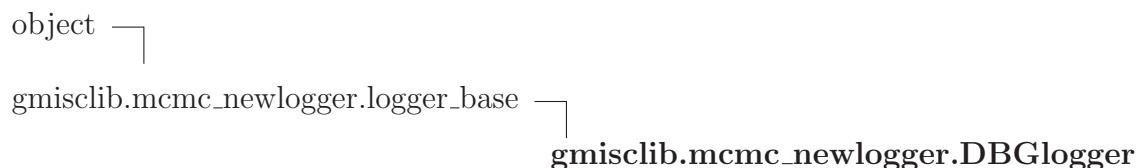
### *Inherited from `object`*

`__delattr__()`, `__format__()`, `__getattribute__()`, `__hash__()`, `__new__()`, `__reduce__()`, `__reduce_ex__()`,  
`__repr__()`, `__setattr__()`, `__sizeof__()`, `__str__()`, `__subclasshook__()`

### 82.4.2 Properties

Name	Description
<i>Inherited from <code>object</code></i>	
<code>__class__</code>	

## 82.5 Class DBGlogger



### 82.5.1 Methods

**`--init--(self, fd)`**

`x.--init--(...)` initializes `x`; see `help(type(x))` for signature

Overrides: `object.--init--` `exitit`(inherited documentation)

**`add(self, uid, prms, map, logp, iter, extra=None, reason=None)`**

Adds parameters to the log file. `uid` – which subset of parameters, `prms` – a vector of the actual numbers, `map` – a dictionary mapping from parameter names to an index into the `prms` array. `logp` – how good is the fit? `log(probability)` `iter` – an integer to keep track of the number of iterations

**`close(self)`**

**`comment(self, comment)`**

**`flush(self)`**

**`header(self, k, v)`**

**`headers(self, hdict)`**

### *Inherited from object*

`--delattr--()`, `--format--()`, `--getattr--()`, `--hash--()`, `--new--()`, `--reduce--()`, `--reduce_ex--()`, `--repr--()`, `--setattr--()`, `--sizeof--()`, `--str--()`, `--subclasshook--()`

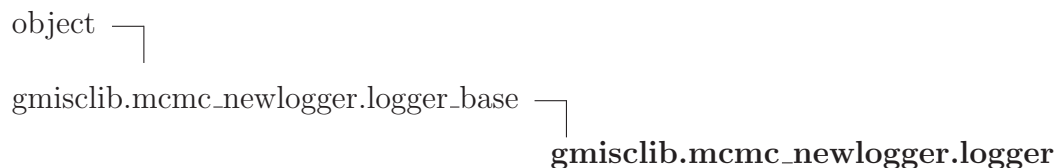
### 82.5.2 Properties

Name	Description
<i>Inherited from object</i>	
<code>--class--</code>	

*continued on next page*

Name	Description
------	-------------

## 82.6 Class *logger*



### 82.6.1 Methods

**`__init__(self, fd, huge=1e+30)`**

Create a data log.

**Parameters**

**fd:** Where to log the results  
(*type=file*)

**huge:** sets a threshold for throwing `WildNumber`. If the absolute value of any parameter gets bigger than `huge`, something is assumed to be wrong.  
(*type=normally float, but could be numpy.ndarray*)

Overrides: `object.__init__`

**add**(*self*, *uid*, *prms*, *map*, *logp*, *iter*, *extra*=None, *reason*=None)

Adds a set of parameters to the log file.

**Parameters**

**uid:** which subset of parameters,  
(*type*=string)

**prms:** a vector of the actual numbers,  
(*type*=numpy.ndarray)

**map:** a dictionary mapping from parameter names to an index  
into the prms array.  
(*type*=dict)

**logp:** how good is the fit? log(probability)  
(*type*=float)

**iter:** which iteration is it?  
(*type*=int)

**extra:** any extra information desired.  
(*type*=None or a dictionary.)

**close**(*self*)

**comment**(*self*, *comment*)

**flush**(*self*)

**header**(*self*, *k*, *v*)

**headers**(*self*, *hdict*)

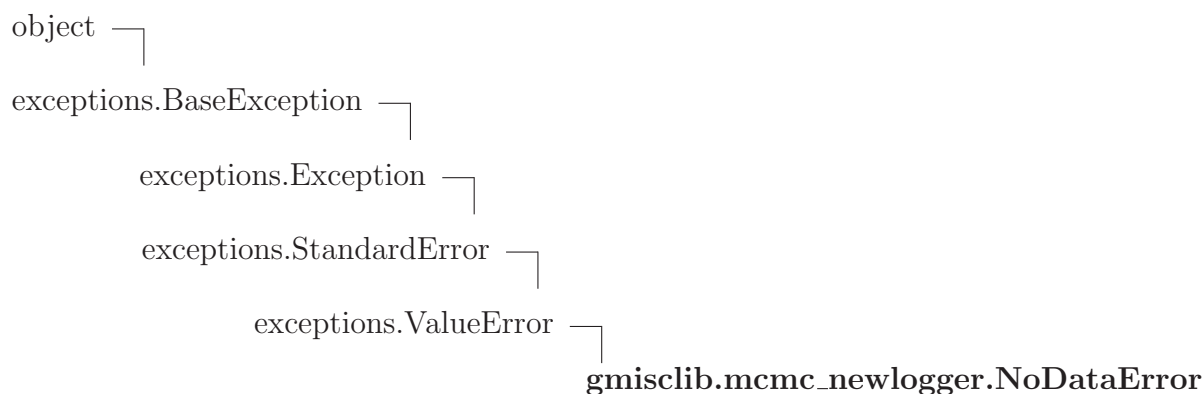
**Inherited from object**

`__delattr__()`, `__format__()`, `__getattr__()`, `__hash__()`, `__new__()`, `__reduce__()`, `__reduce_ex__()`,  
`__repr__()`, `__setattr__()`, `__sizeof__()`, `__str__()`, `__subclasshook__()`

**82.6.2 Properties**

Name	Description
<i>Inherited from object</i>	
<code>__class__</code>	

## 82.7 Class `NoDataError`



There is not a complete set of data in the log file: data from the outer optimizer and at least one full set of inner optimizer results.

### 82.7.1 Methods

`--init--(self, *s)`  
`x.__init__(...)` initializes `x`; see `help(type(x))` for signature  
 Overrides: `object.__init__` extit(inherited documentation)

*Inherited from `exceptions.ValueError`*

`--new--()`

*Inherited from `exceptions.BaseException`*

`--delattr--()`, `--getattr--()`, `--getitem--()`, `--getslice--()`, `--reduce--()`, `--repr--()`,  
`--setattr--()`, `--setstate--()`, `--str--()`, `--unicode--()`

*Inherited from `object`*

`--format--()`, `--hash--()`, `--reduce_ex--()`, `--sizeof--()`, `--subclasshook--()`

### 82.7.2 Properties

Name	Description
<i>Inherited from <code>exceptions.BaseException</code></i>	
<code>args</code> , <code>message</code>	
<i>Inherited from <code>object</code></i>	
<code>--class--</code>	

## 82.8 Class logline

object └─  
           **gmisclib.mcmc\_newlogger.logline**

This holds the information from one line of a log file. For efficiency reasons, some of the data may be stored pickled, and should be accessed through `prms()` or `logp()`.

### 82.8.1 Methods

**`__init__(self, uid, iter, logp, prms, names, extra)`**

`x.__init__(...)` initializes `x`; see `help(type(x))` for signature

Overrides: `object.__init__` `exitit`(inherited documentation)

**`prms(self)`**

**`logp(self)`**

**`idxr(self)`**

**`__repr__(self)`**

`repr(x)`

Overrides: `object.__repr__` `exitit`(inherited documentation)

**`T(self)`**

### *Inherited from object*

`__delattr__()`, `__format__()`, `__getattr__()`, `__hash__()`, `__new__()`, `__reduce__()`, `__reduce_ex__()`,  
`__setattr__()`, `__sizeof__()`, `__str__()`, `__subclasshook__()`

### 82.8.2 Properties

Name	Description
<code>extra</code>	
<code>iter</code>	
<code>names</code>	
<code>uid</code>	
<i>Inherited from object</i>	

*continued on next page*

Name	Description
__class__	



## 83 Module `gmisclib.mcmc_pypar`

This is a helper module to make use of `mcmc.py` and `mcmc_big.py`. It allows you to conveniently run a Monte-Carlo simulation of any kind until it converges (`stepper.run_to_bottom`) or until it has explored a large chunk of parameter space (`stepper.run_to_ergodic`).

It also helps you with logging the process.

### 83.1 Functions

<b>Reduce</b> ( <i>x</i> , <i>root</i> )
--

<b>precompute_logp</b> ( <i>lop</i> )
---------------------------------------

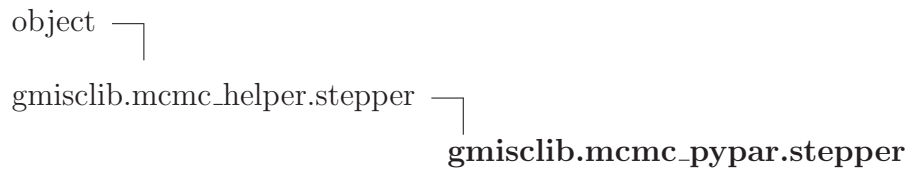
Does a parallel evaluation of logp for all items in <i>lop</i> .
--

<b>is_root</b> ()
-------------------

<b>size</b> ()
----------------

<b>test_opt</b> ()
--------------------

### 83.2 Class `stepper`



#### 83.2.1 Methods

<b>__init__</b> ( <i>self</i> , <i>x</i> , <i>maxloops</i> =-1, <i>logger</i> =None)
--

<i>x</i> . <b>__init__</b> (...) initializes <i>x</i> ; see <code>help(type(x))</code> for signature
--

Overrides: <code>object.__init__</code> <code>exitit</code> (inherited documentation)
---

<b>reset_loops</b> ( <i>self</i> , <i>maxloops</i> =-1)
---

Overrides: <code>gmisclib.mcmc_helper.stepper.reset_loops</code>
--

**communicate\_hook**(*self*)

Overrides: `gmisclib.mcmc_helper.stepper.communicate_hook`

**join**(*self*, *id*)

**note**(*self*, *s*)

Overrides: `gmisclib.mcmc_helper.stepper.note`

**close**(*self*)

After calling `close()`, it is no longer legal to call `run_to_change()`, `run_to_ergodic()`, or `run_to_bottom()`. Logging is shut down, but the contents of the stepper are still available for inspection.

**run\_to\_bottom**(*self*, *ns*=3, *acceptable\_step*=None)

Run the *Markov Chain Monte-Carlo* until it converges near a minimum.

The general idea of the termination condition is that it keeps track of the angle between successive sequences of steps, where each sequence contains **ns** steps. If the angle is less than 90 degrees, it is evidence that the solution is still drifting in some consistent direction and therefore not yet at the maximum. Angles of greater than 90 degrees suggest that the optimization is wandering aimlessly near a minimum. It will run until a sufficient number of large angles are seen since the last BMCMC reset. A similar check is made on individual components of the parameter vector.

#### Parameters

**ns**: This controls how carefully it checks that it has really found a minimum. Large values will take longer but will assure more accurate convergence.

(*type*=*int*)

**acceptable\_step**: A callable object (i.e. class or function) that returns **True** or **False**, depending on whether a proposed step is acceptable. See `mcmc.T_acceptor` for an example.

#### Raises

`TooManyLoops` ?

---

**run\_to\_change**(*self*, *ncw*=1, *acceptable\_step*=None, *update\_T*=False)

---

Run the *Markov Chain Monte-Carlo* until it finds an acceptable step.

#### Parameters

- ncw:** How many steps should be accepted before it stops?  
(*type*=int)
- acceptable\_step:** A function that decides what steps are acceptable and what are not. This is passed to the MCMC stepper, `mcmc.BootStepper`.  
(*type*=function(float) -> bool, often *mcmc.T\_acceptor* or *step\_acceptor*)
- update\_T:** Obsolete. Must be False.  
(*type*=boolean.)

#### Raises

- TooManyLoops** if it takes a long time before enough steps are accepted.

---

**run\_to\_ergodic**(*self*, *ncw*=1, *T*=1.0)

---

Run the stepper until it has explored all of parameter space **ncw** times (as best as we can estimate). Note that this is a pretty careful routine. If the stepper finds a better region of parameter space so that log(P) improves part way through the process, the routine will reset itself and begin again.

NOTE: this sets **T** and **sortstrategy** for the **stepper**.

#### Parameters

- ncw:** how many times (or what fraction) of an ergodic exploration to make.  
(*type*=int.)
- T:** temperature at which to run the Monte-Carlo process. This is normally 1.0. If it is **None**, then the current temperature is not modified.  
(*type*=float or None.)

#### Return Value

- a **position** at the end of the process.

---

**synchronize\_abort**(*self*, *id*)

---



---

**synchronize\_end**(*self*, *id*)

---

<code>synchronize_start(self, id)</code>
--

### *Inherited from object*

`__delattr__()`, `__format__()`, `__getattr__()`, `__hash__()`, `__new__()`, `__reduce__()`, `__reduce_ex__()`,  
`__repr__()`, `__setattr__()`, `__sizeof__()`, `__str__()`, `__subclasshook__()`

### 83.2.2 Properties

Name	Description
<i>Inherited from object</i>	
<code>__class__</code>	

### 83.2.3 Class Variables

Name	Description
<code>MPIID</code>	<b>Value:</b> 1241

## 84 Module `gmisclib.mcmc_restart`

### 84.1 Functions

```
setup(pd_factory, arglist, uid='UID', psn_factory=<class  
'gmisclib.mcmc.position_repeatable'>, adjust_prms=None)
```

This and prepares to restart a `mcmc.py` optimization from a log file.

#### Parameters

- arglist:** the list of arguments to the optimization routine (i.e. the linux command line).  
(*type*=`list(str)`)
- adjust\_prms:** a function to make arbitrary changes to the parameters before you restart.  
(*type*=`None` or *function*(`mcmc_indexclass.index_base`, `mcmc.problem_definition`, `mcmc_indexclass.index`) -> `mcmc_indexclass.index_base`. The parameters go into the first argument. The definition of the new problem is the second argument. The third argument is a random indexer for the new problem. (This is provided mostly so you can conveniently find out what parameters are expected.))

#### Notes:

- This assumes that an optimization does not require any fixed parameters.
- Don't mix `-restart` and `-Restart`.
- The `-Restart` flag will be treated as `-restart` if you attempt to adjust parameters.

### 84.2 Variables

Name	Description
<code>HUGE</code>	<b>Value:</b> <code>1e+38</code>
<code>__package__</code>	<b>Value:</b> <code>'gmisclib'</code>

## 85 Module `gmisclib.mcmc_socket`

This is a helper module to make use of `mcmc.py` and `mcmc_big.py`. It allows you to conveniently run a Monte-Carlo simulation of any kind until it converges (`stepper.run_to_bottom`) or until it has explored a large chunk of parameter space (`stepper.run_to_ergodic`).

It also helps you with logging the process.

When run in parallel, each processor does its thing more-or-less independently. However, every few steps, they exchange notes on their current progress. If one finds an improved vertex, it will be passed on to other processors via `mcmc.cooperate.py`.

### 85.1 Functions

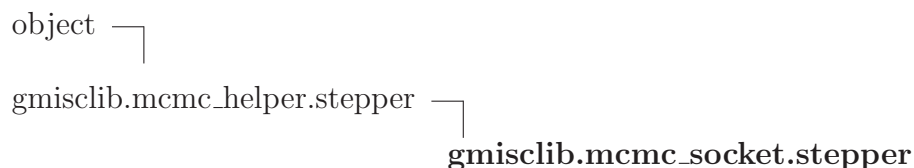
```
precompute_logp(lop)
```

```
test_opt(args)
```

### 85.2 Variables

Name	Description
Debug	Value: 0
__package__	Value: 'gmisclib'

### 85.3 Class `stepper`



#### 85.3.1 Methods

```
__init__(self, x, maxloops=-1, logger=None, share=None)
```

`x.__init__(...)` initializes `x`; see `help(type(x))` for signature

Overrides: `object.__init__` extit(inherited documentation)

**reset\_loops**(*self*, *maxloops*=-1)Overrides: *gmisclib.mcmc\_helper.stepper.reset\_loops***communicate\_hook**(*self*, *id*)Overrides: *gmisclib.mcmc\_helper.stepper.communicate\_hook***synchronize\_start**(*self*, *id*)Overrides: *gmisclib.mcmc\_helper.stepper.synchronize\_start***synchronize\_end**(*self*, *id*)Overrides: *gmisclib.mcmc\_helper.stepper.synchronize\_end***synchronize\_abort**(*self*, *id*)Overrides: *gmisclib.mcmc\_helper.stepper.synchronize\_abort***note**(*self*, *s*, *lvl*)Overrides: *gmisclib.mcmc\_helper.stepper.note***close**(*self*)

After calling `close()`, it is no longer legal to call `run_to_change()`, `run_to_ergodic()`, or `run_to_bottom()`. Logging is shut down, but the contents of the stepper are still available for inspection.

Overrides: *gmisclib.mcmc\_helper.stepper.close* `exitit`(inherited documentation)**size**(*self*)**rank**(*self*)

**run\_to\_bottom**(*self*, *ns*=3, *acceptable\_step*=None)

Run the *Markov Chain Monte-Carlo* until it converges near a minimum.

The general idea of the termination condition is that it keeps track of the angle between successive sequences of steps, where each sequence contains **ns** steps. If the angle is less than 90 degrees, it is evidence that the solution is still drifting in some consistent direction and therefore not yet at the maximum. Angles of greater than 90 degrees suggest that the optimization is wandering aimlessly near a minimum. It will run until a sufficient number of large angles are seen since the last BMCMC reset. A similar check is made on individual components of the parameter vector.

**Parameters**

**ns**: This controls how carefully it checks that it has really found a minimum. Large values will take longer but will assure more accurate convergence.

(*type*=**int**)

**acceptable\_step**: A callable object (i.e. class or function) that returns **True** or **False**, depending on whether a proposed step is acceptable. See `mcmc.T_acceptor` for an example.

**Raises**

`TooManyLoops` ?



---

**run\_to\_change**(*self*, *ncw*=1, *acceptable\_step*=None, *update\_T*=False)

---

Run the *Markov Chain Monte-Carlo* until it finds an acceptable step.

#### Parameters

**ncw:** How many steps should be accepted before it stops?  
(*type=int*)

**acceptable\_step:** A function that decides what steps are acceptable and what are not. This is passed to the MCMC stepper, `mcmc.BootStepper`.  
(*type=function(float) -> bool, often `mcmc.T_acceptor` or `step_acceptor`*)

**update\_T:** Obsolete. Must be False.  
(*type=boolean.*)

#### Raises

**TooManyLoops** if it takes a long time before enough steps are accepted.

---

**run\_to\_ergodic**(*self*, *ncw*=1, *T*=1.0)

---

Run the stepper until it has explored all of parameter space **ncw** times (as best as we can estimate). Note that this is a pretty careful routine. If the stepper finds a better region of parameter space so that log(P) improves part way through the process, the routine will reset itself and begin again.

NOTE: this sets **T** and **sortstrategy** for the **stepper**.

#### Parameters

**ncw:** how many times (or what fraction) of an ergodic exploration to make.  
(*type=int.*)

**T:** temperature at which to run the Monte-Carlo process. This is normally 1.0. If it is **None**, then the current temperature is not modified.  
(*type=float or None.*)

#### Return Value

a **position** at the end of the process.

### *Inherited from object*

`__delattr__()`, `__format__()`, `__getattr__()`, `__hash__()`, `__new__()`, `__reduce__()`, `__reduce_ex__()`, `__repr__()`, `__setattr__()`, `__sizeof__()`, `__str__()`, `__subclasshook__()`

**85.3.2 Properties**

Name	Description
<i>Inherited from object</i> __class__	

## 86 Module *gmisclib.multivariate\_classes*

Support module for *multivariate.py*

### 86.1 Functions

`vec_inv_variance(start)`

`diag_inv_variance(start)`

### 86.2 Variables

Name	Description
BAYES_PRIOR_SPACE	<b>Value:</b> 0
HUGE	<b>Value:</b> 1e+30
<code>__package__</code>	<b>Value:</b> 'gmisclib'

### 86.3 Class *QuadraticNotNormalizable*



#### 86.3.1 Methods

`__init__(self, s='')`

`x.__init__(...)` initializes `x`; see `help(type(x))` for signature

Overrides: `object.__init__` `extit`(inherited documentation)

*Inherited from `exceptions.ValueError`*

`--new--()`

***Inherited from exceptions.BaseException***

`--delattr--()`, `--getattr--()`, `--getitem--()`, `--getslice--()`, `--reduce--()`, `--repr--()`,  
`--setattr--()`, `--setstate--()`, `--str--()`, `--unicode--()`

***Inherited from object***

`--format--()`, `--hash--()`, `--reduce_ex--()`, `--sizeof--()`, `--subclasshook--()`

### 86.3.2 Properties

Name	Description
<i>Inherited from exceptions.BaseException</i>	
args, message	
<i>Inherited from object</i>	
<code>--class--</code>	

## 86.4 Class modeldesc

**Known Subclasses:** `gmsclib.multivariate_mm.multi_mu`, `gmsclib.multivariate_mm.multi_mu_diag`,  
`gmsclib.multivariate_q.quadratic`

Virtual base class for a description of a model of a particular size.

### 86.4.1 Methods

**`--init--`**(*self*, *ndim*)

**`ndim`**(*self*)  
 This returns the dimensionality of the data.

**`modeldim`**(*self*)  
 This gives the dimensionality of the model, i.e. the number of parameters required to specify the means and covariance matrix(*ces*).

**`unpack`**(*self*, *prms*)  
 This returns some subclass of model.

<b>new</b> ( <i>self</i> , <i>mu</i> , <i>invsigma</i> )
--

Creates a model that contains data.
-------------------------------------

<b>start</b> ( <i>self</i> , <i>dataset</i> )
---

Selects a random starting point from the dataset.
---

#### 86.4.2 Class Variables

Name	Description
<code>__doc__</code>	<b>Value:</b> ""Virtual base class for a description of a mo...
<code>LF</code>	<b>Value:</b> 0.111111111111

### 86.5 Class `model_with_numbers`

**Known Subclasses:** `gmisclib.multivariate_mm.multi_mu_diag_with_numbers`, `gmisclib.multivariate_mm.multi_mu_diag_without_numbers`, `gmisclib.multivariate_q.diag_quadratic_with_numbers`, `gmisclib.multivariate_q.quadratic_with_numbers`

Virtual base class for adding in the functions you need when you create a model with known parameters (like `mu` and `sigma`).

#### 86.5.1 Methods

<b>__init__</b> ( <i>self</i> , <i>details</i> , <i>bias</i> )
--

Bias is an overall shift of the log probability up or down. In a classifier, it is used to bias things toward one class or another.
---

<b>logp</b> ( <i>self</i> , <i>datum</i> )
--

<b>pack</b> ( <i>self</i> )
-----------------------------

Returns a vector of parameters.
---------------------------------

<b>ndim</b> ( <i>self</i> )
-----------------------------

<b>unpack</b> ( <i>self</i> , <i>prms</i> )
---

<b>new</b> ( <i>self</i> , <i>mu</i> , <i>invsigma</i> )
--

```
start(self, dataset)
```

```
offset(self)
```

```
addoff(self)
```

### 86.5.2 Class Variables

Name	Description
<code>__doc__</code>	<b>Value:</b> <code>"""Virtual base class for adding in the functi...</code>

## 87 Module `gmisclib.multivariance_mm`

This a helper module for `multivariance.py`

### 87.1 Functions

```
_multi_mu__init__(self, dataset=None, ndim=None, idmap=None,
details=None)
```

You either give it a complete dataset to look at, including class IDs, \*or\* the dimensionality of the data (`ndim`) and a map between classids and integers. This map can be obtained from `nice_hash`.

### 87.2 Variables

Name	Description
<code>__package__</code>	<b>Value:</b> <code>'gmisclib'</code>

### 87.3 Class `datum_c`

#### 87.3.1 Methods

```
__init__(self, vector, classid)
```

### 87.4 Class `multi_mu`

`gmisclib.multivariance_classes.modeldesc` — `gmisclib.multivariance_mm.multi_mu`

This describes a quadratic model of a known size, with multiple means (one for each different class of data).

## 87.4.1 Methods

**`__init__(self, dataset=None, ndim=None, idmap=None, details=None)`**

You either give it a complete dataset to look at, including class IDs, \*or\* the dimensionality of the data (*ndim*) and a map between classids and integers. This map can be obtained from *nice\_hash*.

Overrides: *gmisclib.multivariate\_classes.modeldesc.\_\_init\_\_*

**`modeldim(self)`**

This gives the dimensionality of the model, i.e. the number of parameters required to specify the means and covariance matrix(ces).

Overrides: *gmisclib.multivariate\_classes.modeldesc.modeldim* *exitit*(inherited documentation)

**`unpack(self, prms)`**

This returns some subclass of model.

Overrides: *gmisclib.multivariate\_classes.modeldesc.unpack* *exitit*(inherited documentation)

**`new(self, mu, invsigma, bias=0.0)`**

*Mu* is a mapping of classids to vectors. *invsigma* is a square matrix.

Overrides: *gmisclib.multivariate\_classes.modeldesc.new*

**`start(self, data)`**

Selects a random starting point from the dataset.

Overrides: *gmisclib.multivariate\_classes.modeldesc.start* *exitit*(inherited documentation)

**`ndim(self)`**

This returns the dimensionality of the data.

## 87.4.2 Class Variables

Name	Description
<code>__doc__</code>	<b>Value:</b> <code>""</code> "This describes a quadratic model of a known si..."

*continued on next page*



Name	Description
LF	Value: 0.111111111111

## 87.5 Class `multi_mu_with_numbers`

`gmisclib.multivariate_classes.model_with_numbers` — `gmisclib.multivariate_mm.multi_mu_with_numbers`

### 87.5.1 Methods

**`__init__`**(*self*, *mu*, *invsigma*, *details*, *bias*=0.0, *offset*=None)

`self.mu`, `self.invsigma`, and `self._offset` should be considered read-only for all users of this class.

Overrides: `gmisclib.multivariate_classes.model_with_numbers.__init__`

**`__str__`**(*self*)

**`__repr__`**(*self*)

**`addoff`**(*self*)

Overrides: `gmisclib.multivariate_classes.model_with_numbers.addoff`

**`pack`**(*self*)

Returns a vector of parameters.

Overrides: `gmisclib.multivariate_classes.model_with_numbers.pack`  
`extit`(inherited documentation)

**`logp`**(*self*, *datum*)

Overrides: `gmisclib.multivariate_classes.model_with_numbers.logp`

**`ndim`**(*self*)

**`new`**(*self*, *mu*, *invsigma*)

**`offset`**(*self*)

**start**(*self*, *dataset*)

**unpack**(*self*, *prms*)

### 87.5.2 Class Variables

Name	Description
<code>__doc__</code>	<b>Value:</b> <code>""</code> "Virtual base class for adding in the functi..."

## 87.6 Class `multi_mu_diag`

`gmisclib.multivariate_classes.modeldesc` — `gmisclib.multivariate_mm.multi_mu_diag`

This describes a quadratic model of a known size, with multiple means (one for each different class of data). The covariance matrix is shared and diagonal.

### 87.6.1 Methods

**\_\_init\_\_**(*self*, *dataset*=None, *ndim*=None, *idmap*=None, *details*=None)

You either give it a complete dataset to look at, including class IDs, \*or\* the dimensionality of the data (*ndim*) and a map between classids and integers. This map can be obtained from `nice_hash`.

Overrides: `gmisclib.multivariate_classes.modeldesc.__init__`

**modeldim**(*self*)

This gives the dimensionality of the model, i.e. the number of parameters required to specify the means and covariance matrix(ces).

Overrides: `gmisclib.multivariate_classes.modeldesc.modeldim` `extit`(inherited documentation)

**unpack**(*self*, *prms*)

This returns some subclass of `model`.

Overrides: `gmisclib.multivariate_classes.modeldesc.unpack` `extit`(inherited documentation)

**new**(*self*, *mu*, *invsigma*, *bias*=0.0)

Mu is a mapping of classids to vectors. Invsigma is a vector.

Overrides: `gmisclib.multivariate_classes.modeldesc.new`

**start**(*self*, *data*)

Selects a random starting point from the dataset.

Overrides: `gmisclib.multivariate_classes.modeldesc.start` `exitit`(inherited documentation)

**ndim**(*self*)

This returns the dimensionality of the data.

### 87.6.2 Class Variables

Name	Description
<code>__doc__</code>	<b>Value:</b> <code>""</code> "This describes a quadratic model of a known si...
<code>LF</code>	<b>Value:</b> 0.111111111111

## 87.7 Class `multi_mu_diag_with_numbers`

`gmisclib.multivariate_classes.model_with_numbers`

`gmisclib.multivariate_mm.multi_mu_diag_wi`

### 87.7.1 Methods

**\_\_init\_\_**(*self*, *mu*, *invsigma*, *details*, *bias*=0.0, *offset*=None)

`self.mu`, `self.invsigma`, and `self._offset` should be considered read-only for all users of this class.

Overrides: `gmisclib.multivariate_classes.model_with_numbers.__init__`

**\_\_str\_\_**(*self*)

**\_\_repr\_\_**(*self*)

**addoff**(*self*)Overrides: *gmisclib.multivariate\_classes.model\_with\_numbers.addoff***pack**(*self*)

Returns a vector of parameters.

Overrides: *gmisclib.multivariate\_classes.model\_with\_numbers.pack*  
exitit(inherited documentation)**logp**(*self*, *datum*)Overrides: *gmisclib.multivariate\_classes.model\_with\_numbers.logp***ndim**(*self*)**new**(*self*, *mu*, *invsigma*)**offset**(*self*)**start**(*self*, *dataset*)**unpack**(*self*, *prms*)

### 87.7.2 Class Variables

Name	Description
<code>__doc__</code>	<b>Value:</b> <code>"""Virtual base class for adding in the functi...</code>

## 88 Module `gmisclib.multivariate_q`

This a helper module for `multivariate.py`

### 88.1 Variables

Name	Description
<code>HUGE</code>	<b>Value:</b> <code>1e+30</code>
<code>__package__</code>	<b>Value:</b> <code>'gmisclib'</code>

### 88.2 Class `quadratic`

`gmisclib.multivariate_classes.modeldesc` — `gmisclib.multivariate_q.quadratic`

**Known Subclasses:** `gmisclib.multivariate_q.diag_quadratic`

This describes a quadratic model of a known size.

#### 88.2.1 Methods

**`__init__(self, dataset=None, ndim=None)`**

Overrides: `gmisclib.multivariate_classes.modeldesc.__init__`

**`modeldim(self)`**

This gives the dimensionality of the model, i.e. the number of parameters required to specify the means and covariance matrix(es).

Overrides: `gmisclib.multivariate_classes.modeldesc.modeldim` `exitit`(inherited documentation)

**`unpack(self, prms)`**

This returns some subclass of model.

Overrides: `gmisclib.multivariate_classes.modeldesc.unpack` `exitit`(inherited documentation)

**new**(*self*, *mu*, *invsigma*, *bias*=0.0)

Creates a model that contains data.

Overrides: `gmisclib.multivariate_classes.modeldesc.new` extit(inherited documentation)

**start**(*self*, *data*)

Returns (starting position vector, covariance matrix)

Overrides: `gmisclib.multivariate_classes.modeldesc.start`

**ndim**(*self*)

This returns the dimensionality of the data.

### 88.2.2 Class Variables

Name	Description
<code>__doc__</code>	<b>Value:</b> <code>"""This describes a quadratic model of a known size."""</code>
<code>LF</code>	<b>Value:</b> <code>0.111111111111</code>

## 88.3 Class `quadratic_with_numbers`

`gmisclib.multivariate_classes.model_with_numbers`

`gmisclib.multivariate_q.quadratic_with_num`

### 88.3.1 Methods

**\_\_init\_\_**(*self*, *mu*, *invsigma*, *details*, *bias*=0.0, *offset*=None)

`self.mu`, `self.invsigma`, and `self._offset` should be considered read-only for all users of this class.

Overrides: `gmisclib.multivariate_classes.model_with_numbers.__init__`

**\_\_str\_\_**(*self*)

**\_\_repr\_\_**(*self*)

**addoff**(*self*)Overrides: `gmisclib.multivariate_classes.model_with_numbers.addoff`**logp**(*self*, *datum*)Overrides: `gmisclib.multivariate_classes.model_with_numbers.logp`**ndim**(*self*)**new**(*self*, *mu*, *invsigma*)**offset**(*self*)**pack**(*self*)

Returns a vector of parameters.

**start**(*self*, *dataset*)**unpack**(*self*, *prms*)

### 88.3.2 Class Variables

Name	Description
<code>__doc__</code>	<b>Value:</b> <code>""</code> Virtual base class for adding in the functi...

## 88.4 Class *diag\_quadratic*

`gmisclib.multivariate_classes.modeldesc``gmisclib.multivariate_q.quadratic``gmisclib.multivariate_q.diag_quadratic`

### 88.4.1 Methods

**\_\_init\_\_**(*self*, *dataset*=None, *ndim*=None)Overrides: `gmisclib.multivariate_classes.modeldesc.__init__`

**modeldim**(*self*)

This gives the dimensionality of the model, i.e. the number of parameters required to specify the means and covariance matrix(es).

Overrides: *gmisclib.multivariate\_classes.modeldesc.modeldim* *exitit*(inherited documentation)

**unpack**(*self*, *prms*)

This returns some subclass of model.

Overrides: *gmisclib.multivariate\_classes.modeldesc.unpack* *exitit*(inherited documentation)

**new**(*self*, *mu*, *invsigma*, *bias*=0.0)

Creates a model that contains data.

Overrides: *gmisclib.multivariate\_classes.modeldesc.new* *exitit*(inherited documentation)

**start**(*self*, *data*)

Returns (starting position vector, covariance matrix)

Overrides: *gmisclib.multivariate\_classes.modeldesc.start* *exitit*(inherited documentation)

**ndim**(*self*)

This returns the dimensionality of the data.

**88.4.2 Class Variables**

Name	Description
LF	<b>Value:</b> 0.111111111111
__doc__	<b>Value:</b> """This describes a quadratic model of a known size."""



## 88.5 Class `diag_quadratic_with_numbers`

`gmisclib.multivariate_classes.model_with_numbers`

`gmisclib.multivariate_q.diag_quadratic_with_numbers`

### 88.5.1 Methods

**`__init__`**(*self*, *mu*, *invsigma*, *details*, *bias*=0.0, *offset*=None)

Bias is an overall shift of the log probability up or down. In a classifier, it is used to bias things toward one class or another.

Overrides: `gmisclib.multivariate_classes.model_with_numbers.__init__`  
`__init__`(inherited documentation)

**`__str__`**(*self*)

**`__repr__`**(*self*)

**`addoff`**(*self*)

Overrides: `gmisclib.multivariate_classes.model_with_numbers.addoff`

**`logp`**(*self*, *datum*)

Overrides: `gmisclib.multivariate_classes.model_with_numbers.logp`

**`ndim`**(*self*)

**`new`**(*self*, *mu*, *invsigma*)

**`offset`**(*self*)

**`pack`**(*self*)

Returns a vector of parameters.

**`start`**(*self*, *dataset*)

**`unpack`**(*self*, *prms*)

**88.5.2 Class Variables**

Name	Description
<code>__doc__</code>	<b>Value:</b> <code>"""Virtual base class for adding in the functi...</code>

## 89 Module *gmisc.lib.multivariate\_normal*

### 89.1 Functions

<code>test()</code>
---------------------

### 89.2 Variables

Name	Description
<code>--package--</code>	<b>Value:</b> <code>'gmisc.lib'</code>

### 89.3 Class *multivariate\_normal*

`multivariate_normal(mean, cov)` or `multivariate_normal(mean, cov, [m, n, ...])` returns an array containing multivariate normally distributed random numbers with specified mean and covariance.

`mean` must be a 1 dimensional array. `cov` must be a square two dimensional array with the same number of rows and columns as `mean` has elements.

The first form returns a single 1-D array containing a multivariate normal.

The second form returns an array of shape `(m, n, ..., cov.shape[0])`. In this case, `output[i,j,...,:]` is a 1-D array containing a multivariate normal.

#### 89.3.1 Methods

<code>--init--(self, mean, cov)</code>
--

<code>sample(self, shape=[])</code>
-------------------------------------

#### 89.3.2 Class Variables

Name	Description
<code>--doc--</code>	<b>Value:</b> <code>""multivariate_normal(mean, cov) or multivariate_normal(...</code>

## 90 Module `gmisclib.named_block_file`

Read and write files in the form [label] text text text [label2] text text ...

The format is imperfect to the extent that it does not allow the text to contain things that look like labels.

When used as a script, it selects regions of such a file and prints them on the standard output: Usage: `named_block_file.py key1 [key2] <input >output` The sections labelled `key1` and `key2` will be output in order, with no separators other than a newline in between.

### 90.1 Functions

<code>read(<i>fd</i>)</code>
------------------------------

<code>write_key(<i>fd</i>, <i>k</i>)</code>
---

<code>write_text(<i>fd</i>, <i>t</i>)</code>
--

<code>write_line(<i>fd</i>, <i>t</i>)</code>
--

<code>write_line_kv(<i>fd</i>, <i>k</i>, <i>v</i>)</code>
---

<code>write(<i>fd</i>, <i>d</i>)</code>
---

### 90.2 Variables

Name	Description
<code>--package--</code>	<b>Value:</b> <code>'gmisclib'</code>

## 91 Module gmisclib.nbest

Beam search through a graph.

### 91.1 Functions

**go**(*graph*, *nbeam*, *cbeam*)

Search the graph for the lowest cost routes. Returns [ (cost, route), ...] where cost is the cost of a route, and route is [node, node, node, ...] the list of nodes on the route. Routes are sorted in order of increasing cost.

**test**()

### 91.2 Variables

Name	Description
<code>--package--</code>	<b>Value:</b> None

### 91.3 Class node

This is a node of a graph, it includes links to other nodes.

#### 91.3.1 Methods

**--init--**(*self*, *cost*=0.0, *label*=None, *terminal*=0, *comment*=None)

Create a node, with a specified cost (used in the beam search), and a label (arbitrary information). Terminal nodes are marked, and terminate the search.

**add**(*self*, *nextnode*, *cost*=0.0, *label*=None)

Add a link from self to nextnode. Links can have a cost and label, too.

**--str--**(*self*)

**--repr--**(*self*)

#### 91.3.2 Class Variables

---

Name	Description
<code>--doc--</code>	<b>Value:</b> <code>""</code> "This is a node of a graph, it includes links to other ..."

## 92 Module gmisclib.nice\_hash

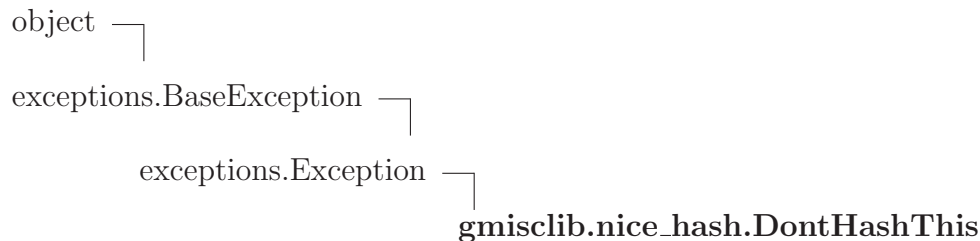
### 92.1 Functions

```
map(list, trimmer=<function <lambda> at 0x4ac7c08>)
```

### 92.2 Variables

Name	Description
h	<b>Value:</b> nice_hash(lambda x: x)
__package__	<b>Value:</b> None

### 92.3 Class DontHashThis



If your trimmer function raises this exception, then the input to nice\_hash.add() will be ignored.

#### 92.3.1 Methods

```
__init__(self)
x.__init__(...) initializes x; see help(type(x)) for signature
Overrides: object.__init__ extit(inherited documentation)
```

***Inherited from exceptions.Exception***

```
__new__()
```

***Inherited from exceptions.BaseException***

```
__delattr__(), __getattr__(), __getitem__(), __getslice__(), __reduce__(), __repr__(),
__setattr__(), __setstate__(), __str__(), __unicode__()
```

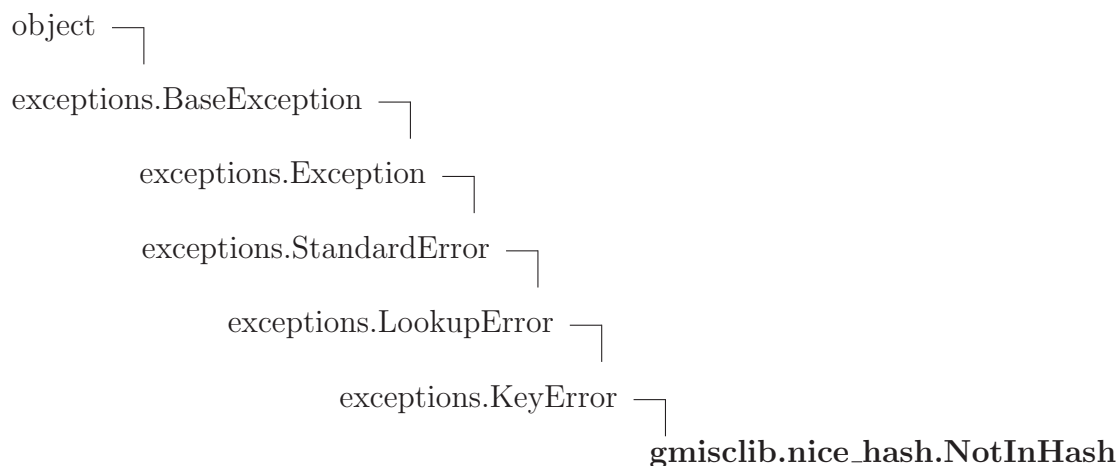
***Inherited from object***

`__format__()`, `__hash__()`, `__reduce_ex__()`, `__sizeof__()`, `__subclasshook__()`

### 92.3.2 Properties

Name	Description
<i>Inherited from <code>exceptions.BaseException</code></i>	
<code>args</code> , <code>message</code>	
<i>Inherited from <code>object</code></i>	
<code>__class__</code>	

## 92.4 Class *NotInHash*



### 92.4.1 Methods

```

__init__(self, s)

x.__init__(...) initializes x; see help(type(x)) for signature
Overrides: object.__init__ extit(inherited documentation)

```

*Inherited from `exceptions.KeyError`*

`__new__()`, `__str__()`

*Inherited from `exceptions.BaseException`*

`__delattr__()`, `__getattribute__()`, `__getitem__()`, `__getslice__()`, `__reduce__()`, `__repr__()`,  
`__setattr__()`, `__setstate__()`, `__unicode__()`

*Inherited from `object`*



`__format__()`, `__hash__()`, `__reduce_ex__()`, `__sizeof__()`, `__subclasshook__()`

#### 92.4.2 Properties

Name	Description
<i>Inherited from <code>exceptions.BaseException</code></i>	
<code>args</code> , <code>message</code>	
<i>Inherited from <code>object</code></i>	
<code>__class__</code>	

### 92.5 Class simple

object └─ **gmisclib.nice\_hash.simple**

**Known Subclasses:** `gmisclib.nice_hash.hash`

This class generates a map from inputs to integers, where an equivalence class is defined by the trimmer function. It is just a stripped-down version of the `hash` class defined above. Faster and less memory consumption, if you don't need the `rmap` method.

#### 92.5.1 Methods

<b><code>__init__(self, trimmer=None)</code></b>
<code>x.__init__(...)</code> initializes <code>x</code> ; see <code>help(type(x))</code> for signature
<b>Parameters</b>
<b><code>trimmer</code>:</b> Defines equivalence classes on list; it can be some sort of projection operation. The default ( <code>None</code> ) makes the identity class equal to the inputs. It can also specify which inputs to ignore by raising the <code>DontHashThis</code> exception.
Overrides: <code>object.__init__</code>
<b><code>add(self, x)</code></b>
Returns an integer which is shared among all <code>x</code> in the equivalence class, but different from all other <code>x</code> . If the trimmer function raises the <code>DontHashThis</code> exception, it will ignore the input and return <code>None</code> .
<b>Return Value</b>
<code>int</code>

**add\_newonly**(*self*, *x*)

Similar to add, except it returns None if the equivalence class has already appeared.

**Return Value**

int or None

**get\_image**(*self*, *x*)

Returns an integer which is shared among all x in the equivalence class, but different from all other x.

**Raises**

NotInHash when x is not in the hash table.

**Note:** the return value is the same as **add**, except that this doesn't add anything: rather than increasing the size of the mapping, it raises NotInHash.

**map**(*self*)

Return the map from classes (as returned by the trimmer function) to integers.

**classes**(*self*)

Return a list over all the classes that have been seen.

**classmap**(*self*)

Return the map from integers to classes.

**Return Value**

[classname, classname, ...]

### *Inherited from object*

`__delattr__()`, `__format__()`, `__getattr__()`, `__hash__()`, `__new__()`, `__reduce__()`, `__reduce_ex__()`, `__repr__()`, `__setattr__()`, `__sizeof__()`, `__str__()`, `__subclasshook__()`

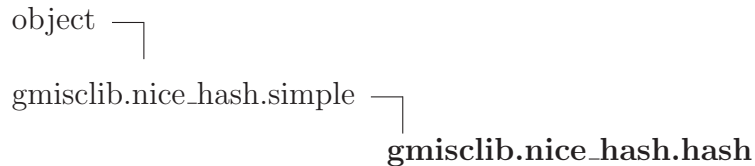
#### 92.5.2 Properties

Name	Description
<i>Inherited from object</i>	
<code>__class__</code>	

#### 92.5.3 Class Variables

Name	Description
<code>__doc__</code>	<b>Value:</b> <code>"""This class generates a map from inputs to intege...</code>

## 92.6 Class hash



### 92.6.1 Methods

<b><code>__init__(self, trimmer=None)</code></b> <hr/> This adds the <code>rmap</code> method to the <code>simple</code> class. <b>Parameters</b> <b><code>trimmer</code>:</b> Defines equivalence classes on list; it can be some sort of projection operation. The default ( <code>None</code> ) makes the identity class equal to the inputs. It can also specify which inputs to ignore by raising the <code>DontHashThis</code> exception. Overrides: <code>object.__init__</code>
<b><code>rmap(self)</code></b> <hr/> Returns the map from integers to inputs.
<b><code>test(self)</code></b>
<b><code>add(self, x)</code></b> <hr/> Returns an integer which is shared among all <code>x</code> in the equivalence class, but different from all other <code>x</code> . If the <code>trimmer</code> function raises the <code>DontHashThis</code> exception, it will ignore the input and return <code>None</code> . Note that if you add an item twice, it will appear twice in <code>rmap()</code> . <b>Return Value</b> <code>int</code> Overrides: <code>gmisclib.nice_hash.simple.add</code>

**add\_newonly**(*self*, *x*)

Similar to `add`, except it returns `None` if the equivalence class has already appeared.

The list of classes seen (`self.seen`) then contains only the first item that was added in each class, so `rmap()` will produce a dictionary containing single-item lists: `{int1:[item1],int2:[item2],int3:[item3], ...}`. Calls to `add()` and `add_newonly()` can be intermixed safely, though the result of `rmap()` will then not contain all added items.

**Return Value**

int or `None`

Overrides: `gmisclib.nice_hash.simple.add_newonly`

**classes**(*self*)

Return a list over all the classes that have been seen.

**classmap**(*self*)

Return the map from integers to classes.

**Return Value**

[classname, classname, ...]

**get\_image**(*self*, *x*)

Returns an integer which is shared among all *x* in the equivalence class, but different from all other *x*.

**Raises**

`NotInHash` when *x* is not in the hash table.

**Note:** the return value is the same as `add`, except that this doesn't add anything: rather than increasing the size of the mapping, it raises `NotInHash`.

**map**(*self*)

Return the map from classes (as returned by the trimmer function) to integers.

**Inherited from object**

`__delattr__()`, `__format__()`, `__getattr__()`, `__hash__()`, `__new__()`, `__reduce__()`, `__reduce_ex__()`, `__repr__()`, `__setattr__()`, `__sizeof__()`, `__str__()`, `__subclasshook__()`

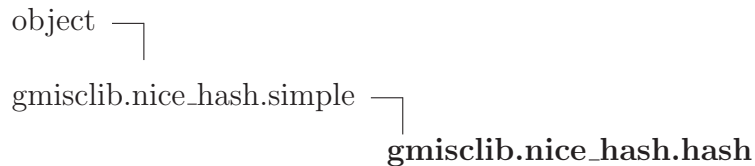
**92.6.2 Properties**

Name	Description
<i>Inherited from object</i>	
<code>--class--</code>	

### 92.6.3 Class Variables

Name	Description
<code>--doc--</code>	<b>Value:</b> <code>""</code> This class generates a map from inputs to intege...

## 92.7 Class hash



### 92.7.1 Methods

<b><code>--init__(self, trimmer=None)</code></b>
This adds the <code>rmap</code> method to the <code>simple</code> class.
<b>Parameters</b>
<b><code>trimmer</code>:</b> Defines equivalence classes on list; it can be some sort of projection operation. The default ( <code>None</code> ) makes the identity class equal to the inputs. It can also specify which inputs to ignore by raising the <code>DontHashThis</code> exception.
Overrides: <code>object.__init__</code>
<b><code>rmap(self)</code></b>
Returns the map from integers to inputs.
<b><code>test(self)</code></b>

**add**(*self*, *x*)

Returns an integer which is shared among all *x* in the equivalence class, but different from all other *x*. If the trimmer function raises the `DontHashThis` exception, it will ignore the input and return `None`. Note that if you add an item twice, it will appear twice in `rmap()`.

**Return Value**

int

Overrides: `gmisclib.nice_hash.simple.add`

**add\_newonly**(*self*, *x*)

Similar to `add`, except it returns `None` if the equivalence class has already appeared.

The list of classes seen (`self.seen`) then contains only the first item that was added in each class, so `rmap()` will produce a dictionary containing single-item lists: `{int1:[item1],int2:[item2],int3:[item3], ...}`. Calls to `add()` and `add_newonly()` can be intermixed safely, though the result of `rmap()` will then not contain all added items.

**Return Value**

int or `None`

Overrides: `gmisclib.nice_hash.simple.add_newonly`

**classes**(*self*)

Return a list over all the classes that have been seen.

**classmap**(*self*)

Return the map from integers to classes.

**Return Value**

[classname, classname, ...]

**get\_image**(*self*, *x*)

Returns an integer which is shared among all *x* in the equivalence class, but different from all other *x*.

**Raises**

`NotInHash` when *x* is not in the hash table.

**Note:** the return value is the same as `add`, except that this doesn't add anything: rather than increasing the size of the mapping, it raises `NotInHash`.

<b>map</b> ( <i>self</i> )
----------------------------

Return the map from classes (as returned by the trimmer function) to integers.
--

***Inherited from object***

`__delattr__()`, `__format__()`, `__getattr__()`, `__hash__()`, `__new__()`, `__reduce__()`, `__reduce_ex__()`, `__repr__()`, `__setattr__()`, `__sizeof__()`, `__str__()`, `__subclasshook__()`

**92.7.2 Properties**

Name	Description
<i>Inherited from object</i>	
<code>__class__</code>	

**92.7.3 Class Variables**

Name	Description
<code>__doc__</code>	<b>Value:</b> <code>"""This class generates a map from inputs to intege...</code>

## 93 Module *gmisclib.nicknames*

### 93.1 Functions

**nicknames**(*namelist*)

Takes a list of names and trims off junk from the beginning and ends of the names to produce a set of reasonable nicknames that are relatively compact and easy to read.

It returns a map from the names to the nicknames, along with the stuff trimmed off from the edges.

**test**()

### 93.2 Variables

Name	Description
<code>--package--</code>	<b>Value:</b> None



## 94 Module *gmisclib.nmf*

$V=WH$ , where  $W$  and  $H$  are non-negative.

From D. D. Lee and H. S. Seung, 'Learning the parts of objects by nonnegative matrix factorization.'

### 94.1 Functions

<b>nmf</b> ( <i>v</i> , <i>rank</i> )
---------------------------------------

### 94.2 Variables

Name	Description
EPS	<b>Value:</b> 1e-06
CC	<b>Value:</b> 2
IEPS	<b>Value:</b> 0.03
__package__	<b>Value:</b> 'gmisclib'

## 95 Module *gmisclib.opt*

This module is a Levenberg-Marquardt optimizer with stem-size control for numeric differentiation. It just needs a function that produces residuals. It is multi-threaded, so calculations of residuals can be farmed out to many processors.

UPDATED 12/2009 GPK: NOT TESTED!

### 95.1 Functions

**vec\_equal**(*a*, *b*)

Check that two Numeric vectors are exactly equal.

**sumsq**(*a*)

This is the overall error measure.

**maxabs**(*a*)

This is used as a measure of how much the function changed as a result of a parameter change.

**wtf**(*r\_dz*, *delta*, *quantum*)

This shows how important different measurements are to the final derivative estimate. The weight must be small when the change of a parameter is smaller than the quantum (i.e. when  $r\_delta < quantum$ ). It must also be small when the change in *z* is much larger than *T* ( $r\_dz >> 1$ ). It is largest when *r\_dz* is about 1.

**near\_duplicate**(*guess*, *tmp*, *quantum*)

**symmetry\_point**(*tmp*, *T*, *quantum*)

Given a list of (*delta*, *z*) in a differentiation, pick one more point that makes the differentiation more symmetric. This is done by picking a new *delta* that brings  $\sum\{delta\_i * wtf()\}$  closer to zero, where *wtf()* is the weight given to each point in the differentiation.

**scale\_est**(*tmp*, *T*)

**explored\_region**(*tmp*)

**clz\_point**(*qq, T, quantum*)

Given a list of (delta, z) tuples in a differentiation attempt, find another delta that (a) fills a gap in the sequence, (b) is preferably has a sign opposite most of the {delta} values, and (c) has a large wtf(). Points outside the sequence are also considered.

**deriv\_estimate**(*tmp, T, quantum*)

**need\_more\_diff\_pts**(*tmp, T, quantum*)

**diff\_one**(*x, i, sem*)

Differentiate z with respect to parameter i. We enter this function with the semaphore 'sem' already acquired.

**eval\_lambda**(*processor, a, b, startp, lamb, out, opt*)

**lamb\_correct**(*newscale, scale*)

**anneal\_guts**(*p, rv, T, lamb, sem, opt*)

**test1\_fcn**(*p, args, \*d*)

**test1**()

Linear.

**test1a\_fcn**(*p, args, \*d*)

**test1a**()

Linear.

**test2\_fcn**(*p, args, \*d*)

**test2**()

Nonlinear

**test3\_fcn**(*p, args, \*d*)

**test3()**

Linear, but correlated

**test4\_fcn**(*p*, *args*, \**d*)**test4()**

Linear, but with a constraint

**test5\_fcn**(*p*, *args*, \**d*)**test5()**

Very Nonlinear

**test6\_fcn**(*p*, *args*, \**d*)**test6()**

Curved valley.

**linconst\_min**(*nparam*, *param*, *min*)

Generate a linear constraint to be added onto a list and passed to `linear_constraint()`. This constraint expresses that `p[param]>=min`.

**linconst\_max**(*nparam*, *param*, *max*)

Generate a linear constraint to be added onto a list and passed to `linear_constraint()`. This constraint expresses that `p[param]<=max`.

**linear\_constraint**(*op*, *np*, *list\_of\_constraints*)

This constrains a step to lie inside a region bounded by a `list_of_constraints`.

**Parameters**

**op:** parameters at the beginning of the step.

**np:** target parameters.

**list\_of\_constraints:** [ (*vector*, *shift*), ... ] where the allowable region of each constraint is defined by `Num.dot(x, vector)+shift >= 0`. *Vector* is a numpy array.

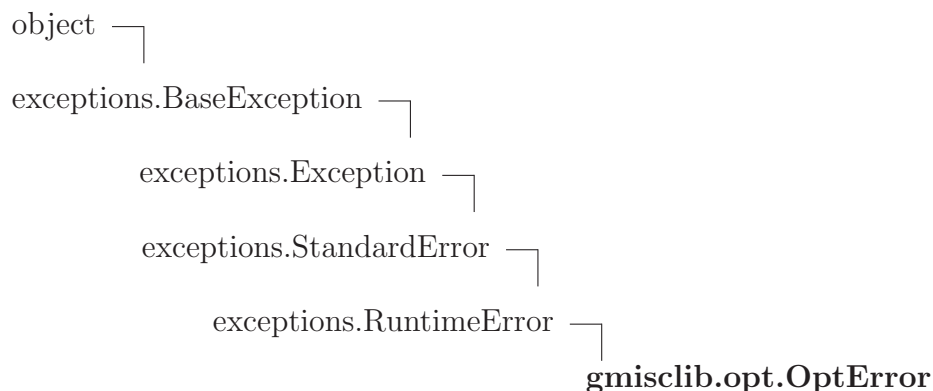
<b>test7()</b>
----------------

Constraint.
-------------

## 95.2 Variables

Name	Description
<code>--package--</code>	<b>Value:</b> <code>'gmislib'</code>

## 95.3 Class *OptError*



**Known Subclasses:** *gmislib.opt.BadResult*, *gmislib.opt.NoDerivative*, *gmislib.opt.NoDownhill*

### 95.3.1 Methods

<b><code>--init--(self, s=None)</code></b>  <code>x.--init--(...)</code> initializes x; see <code>help(type(x))</code> for signature Overrides: <code>object.--init--</code> extit(inherited documentation)
--

***Inherited from exceptions.RuntimeError***

`--new--()`

***Inherited from exceptions.BaseException***

`--delattr--()`, `--getattribute--()`, `--getitem--()`, `--getslice--()`, `--reduce--()`, `--repr--()`,  
`--setattr--()`, `--setstate--()`, `--str--()`, `--unicode--()`

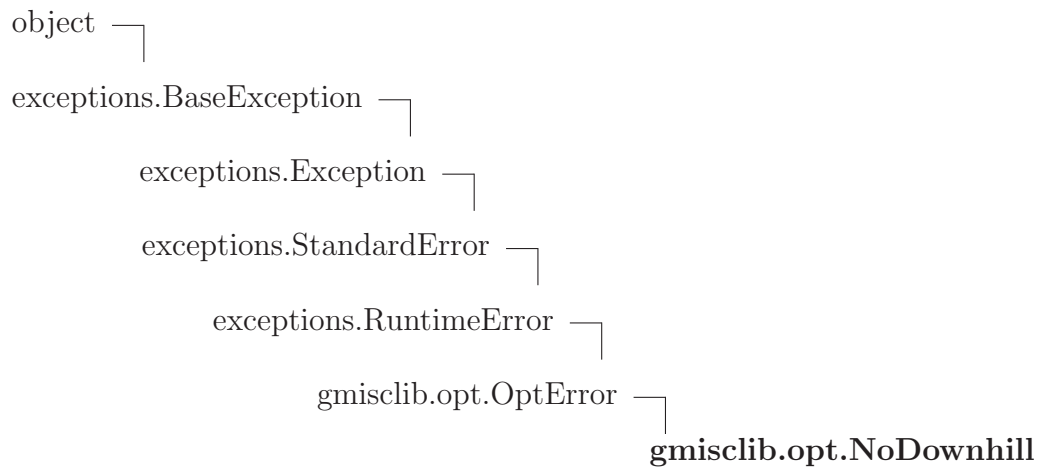
***Inherited from object***

`--format--()`, `--hash--()`, `--reduce_ex--()`, `--sizeof--()`, `--subclasshook--()`

### 95.3.2 Properties

Name	Description
<i>Inherited from exceptions.BaseException</i>	
args, message	
<i>Inherited from object</i>	
__class__	

## 95.4 Class NoDownhill



### 95.4.1 Methods

```

__init__(self, s=None)

x.__init__(...) initializes x; see help(type(x)) for signature
Overrides: object.__init__ extit(inherited documentation)

```

***Inherited from exceptions.RuntimeError***

\_\_new\_\_()

***Inherited from exceptions.BaseException***

\_\_delattr\_\_(), \_\_getattr\_\_(), \_\_getitem\_\_(), \_\_getslice\_\_(), \_\_reduce\_\_(), \_\_repr\_\_(),  
\_\_setattr\_\_(), \_\_setstate\_\_(), \_\_str\_\_(), \_\_unicode\_\_()

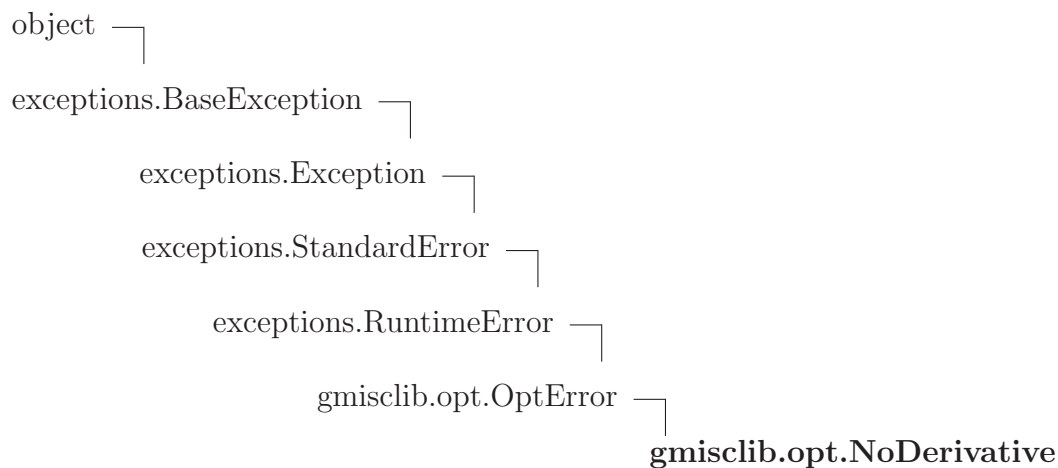
***Inherited from object***

\_\_format\_\_(), \_\_hash\_\_(), \_\_reduce\_ex\_\_(), \_\_sizeof\_\_(), \_\_subclasshook\_\_()

### 95.4.2 Properties

Name	Description
<i>Inherited from exceptions.BaseException</i>	
args, message	
<i>Inherited from object</i>	
__class__	

## 95.5 Class NoDerivative



### 95.5.1 Methods

```

__init__(self, s=None)

x.__init__(...) initializes x; see help(type(x)) for signature
Overrides: object.__init__ extit(inherited documentation)

```

***Inherited from exceptions.RuntimeError***

\_\_new\_\_()

***Inherited from exceptions.BaseException***

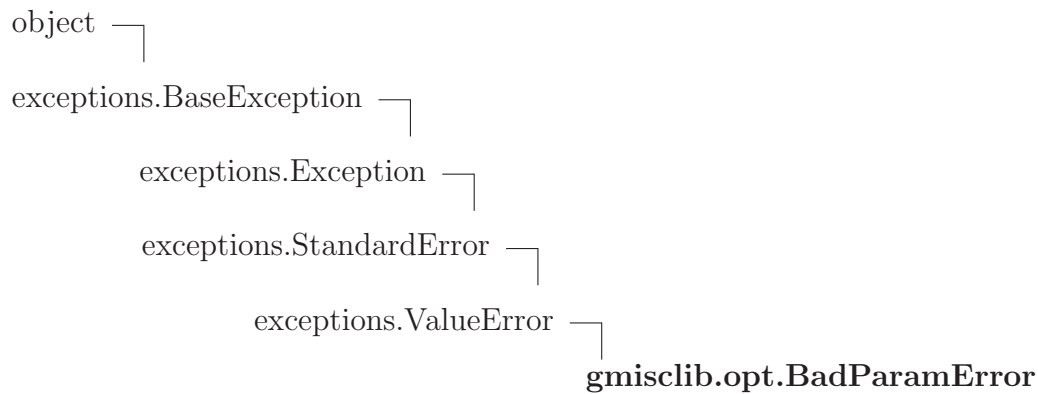
\_\_delattr\_\_(), \_\_getattr\_\_(), \_\_getitem\_\_(), \_\_getslice\_\_(), \_\_reduce\_\_(), \_\_repr\_\_(),  
 \_\_setattr\_\_(), \_\_setstate\_\_(), \_\_str\_\_(), \_\_unicode\_\_()

***Inherited from object***

\_\_format\_\_(), \_\_hash\_\_(), \_\_reduce\_ex\_\_(), \_\_sizeof\_\_(), \_\_subclasshook\_\_()

**95.5.2 Properties**

Name	Description
<i>Inherited from <code>exceptions.BaseException</code></i>	
<code>args</code> , <code>message</code>	
<i>Inherited from <code>object</code></i>	
<code>__class__</code>	

**95.6 Class `BadParamError`**

Raised when you give and opt instance some invalid control parameter.

**95.6.1 Methods**

```

__init__(self, s=None)
x.__init__(...) initializes x; see help(type(x)) for signature
Overrides: object.__init__ extit(inherited documentation)

```

*Inherited from `exceptions.ValueError`*

```
__new__()
```

*Inherited from `exceptions.BaseException`*

```

__delattr__(), __getattr__(), __getitem__(), __getslice__(), __reduce__(), __repr__(),
__setattr__(), __setstate__(), __str__(), __unicode__()

```

*Inherited from `object`*

```
__format__(), __hash__(), __reduce_ex__(), __sizeof__(), __subclasshook__()
```

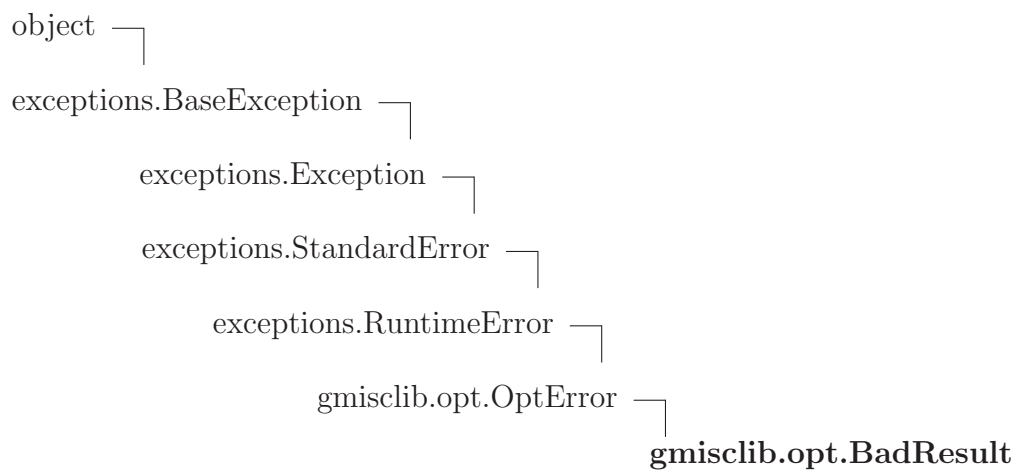


**95.6.2 Properties**

Name	Description
<i>Inherited from exceptions.BaseException</i>	
args, message	
<i>Inherited from object</i>	
__class__	

**95.6.3 Class Variables**

Name	Description
__doc__	<b>Value:</b> <code>""</code> "Raised when you give and opt instance some invalid con..."

**95.7 Class *BadResult***

The function to be optimized returns some illegal result. For instance, an array of the wrong size or type, or it returns `None` for the initial evaluation.

**95.7.1 Methods**

```

__init__(self, s=None)

x.__init__(...) initializes x; see help(type(x)) for signature
Overrides: object.__init__ extit(inherited documentation)

```

*Inherited from exceptions.RuntimeError*

`__new__()`

***Inherited from exceptions.BaseException***

`__delattr__()`, `__getattr__()`, `__getitem__()`, `__getslice__()`, `__reduce__()`, `__repr__()`,  
`__setattr__()`, `__setstate__()`, `__str__()`, `__unicode__()`

***Inherited from object***

`__format__()`, `__hash__()`, `__reduce_ex__()`, `__sizeof__()`, `__subclasshook__()`

### 95.7.2 Properties

Name	Description
<i>Inherited from exceptions.BaseException</i>	
args, message	
<i>Inherited from object</i>	
<code>__class__</code>	

### 95.7.3 Class Variables

Name	Description
<code>__doc__</code>	<b>Value:</b> <code>""</code> "The function to be optimized returns some illegal resu..."

## 95.8 Class mysem

This is a semaphore that is automatically released when it is de-allocated. It also contains a processor ID.

### 95.8.1 Methods

`__init__(self, id, sem)`

`release(self)`

`__del__(self)`

### 95.8.2 Class Variables

Name	Description
<code>--doc--</code>	<b>Value:</b> <code>"""This is a semaphore that is automatically released w...</code>

## 95.9 Class semclass

This is a semaphore to control the number of simultaneous computations. It hands out a mysem class which is used to release the semaphore.

### 95.9.1 Methods

<code>--init--(<i>self</i>, <i>n</i>)</code>
<code>acquire(<i>self</i>)</code>

### 95.9.2 Class Variables

Name	Description
<code>--doc--</code>	<b>Value:</b> <code>"""This is a semaphore to control the number...</code>

## 95.10 Class prms

This class records parameters and caches function evaluations. All function evaluations happen here.

### 95.10.1 Methods

<code>--init--(<i>self</i>, <i>p</i>, <i>fn</i>, <i>args</i>, <i>processor</i>, <i>diffparam</i>=None, <i>diffctr</i>=None)</code>
<code>zraw(<i>self</i>)</code>
<code>z(<i>self</i>)</code>
<code>sumsq(<i>self</i>)</code>

`__len__(self)``__str__(self)``__repr__(self)`

### 95.10.2 Class Variables

Name	Description
<code>__doc__</code>	<b>Value:</b> <code>"""This class records parameters and caches function eval..."""</code>

## 95.11 Class *LockedList*

### 95.11.1 Methods

`__init__(self, start=None)``append(self, x)``__len__(self)``pop(self, i=-1)``items(self)`

## 95.12 Class *opt*

A class that implements a optimizer.

## 95.12.1 Methods

---

**`__init__(self, p, T, fcn, args=None)`**


---

Create an optimizer object. You can also set the following variables between calls to `self.step()`: `scale`, `deriv_quantum`, `T`, `active`, `sem`, `constrain`

**Parameters**

- `p`: initial parameter vector.
- `T`: temperature for simulated annealing and numeric differentiation.
- `fcn`: `fcn(p, args, diffparam, diffctr, processor)`, where `p` = parameters, `args` = arbitrary reference passed to function, `diffparam` = `None` (unless differentiating), `diffctr` = `None` (unless differentiating), `processor` = `(proc_id, i_steps)`, where `proc_id` is in `range(self.nthreads)` and specifies which processor ought to be assigned to the task, and `i_steps` is the number of steps the optimization has gone through. The function is assumed to return either a Num Python array or a sequence of arrays and floats, mixed.

---

**`set_nthreads(self, n)`**


---

Set the number of simultaneous threads to be allowed. Calculations of the residuals are farmed out to threads.

---

**`z(self)`**


---

What is the current residual?

---

**`sumsq(self)`**


---

What is the current error?

---

**`print_ev_diag(self, misc={}, vectors=0)`**


---



---

**`Td(self)`**


---

This allows the differentiation temperature to be dynamically set.

---

**`dE(self)`**


---



---

**`quick_lambda(self, a, b, startp, lamb, ntries=10)`**


---



---

**`search_with_update(self, lamb, newp)`**


---

**calc\_curv**(*self*)

**measure**(*self*)

If you decrease the number of active parameters, it might behoove you to set `self.diff=None`, before you call this function, so that old derivatives get flushed.

**step**(*self*)

A minimization step.

**predicted\_sumsq\_delta**(*self*, *last\_p*)

The energy change of a step, assuming you're at the minimum.

**one\_anneal**(*self*, *T*)

Take one simulated annealing step, as fast as possible. We send all the processors in random directions, and the one that reports an acceptable step first is returned. Returns 1 on success, 0 if no good step could be found in a reasonable number of tries. The temperature can also be a function of one argument (as an alternative to a float), in which case, the temperature is calculated as `T(self)`.

**terminate**(*self*)

Returns number between 0 and 1 if the optimization seems finished. Returns -1 if it's clearly not done yet. This is designed to be called after `step()`, so that both `self.p` and `self.last_p` refer to the results of downhill steps.

**covariance\_under**(*self*)

This is an underestimate of covariance. Adding in the `self.lamb` term means that it attempts to correct for the nonlinearity of the problem, at least along the direction of search.

**covariance**(*self*)

Covariance estimate.

**run**(*self*, *maxsteps*=None, *misc*={}, *Ta*=None)

This is the normal top-level routine. You should feel free to make your own, though.

**print\_errbar**(*self*)

<b>sim_anneal</b> ( <i>self</i> , <i>nsteps</i> , <i>Ta</i> =None, <i>misc</i> ={})
---

A top-level simulated annealing routine.
--

<b>print_at</b> ( <i>self</i> , <i>misc</i> ={})
--

Prints a file to match the old C++ optimizer output.
--

### 95.12.2 Class Variables

Name	Description
<code>__doc__</code>	<b>Value:</b> ""A class that implements a optimiz...
<code>LOG_NAME</code>	<b>Value:</b> 'a.tmp'

### 95.12.3 Instance Variables

Name	Description
<code>T</code>	simulated annealing temperature.
<code>active</code>	vector of which parameters to optimize. Only the active parameters change their value.
<code>constrain</code>	a function that constrains the step. You can use it to do a constrained fit by having it project the proposed step back into the legal volume. Called as <code>self.constrain(old_prms, proposed_prms, self.args)</code> . It returns the constrained step, or None if there isn't any.
<code>deriv_quantum</code>	minimum step size used in differentiation.
<code>scale</code>	estimate of the step size to use in differentiation.
<code>sem</code>	a semaphore to control the number of threads to use (if you want to set the # of threads do <code>self.sem = semclass(nnn)</code> ).

## 96 Module *gmisclib.ortho\_poly*

### 96.1 Functions

<b>test_a_name</b> ( <i>name</i> )
------------------------------------

<b>F</b> ( <i>name</i> , <i>n</i> =None, <i>x</i> =None)
--

This is a factory function. Get any kind of ortho poly you want, as selected by the first argument.
---

<b>test</b> ()
----------------

### 96.2 Variables

Name	Description
<code>--package--</code>	<b>Value:</b> 'gmisclib'

### 96.3 Class *ortho*

**Known Subclasses:** *gmisclib.ortho\_poly.ortho\_poly*, *gmisclib.ortho\_poly.SLTB*, *gmisclib.ortho\_poly.SinCo*

#### 96.3.1 Methods

<b>--init--</b> ( <i>self</i> , <i>n</i> =None, <i>x</i> =None)
---

Argument <i>n</i> is how many points you want the polynomials evaluated at. The ordinates of the points are placed in <i>self.x</i> . You may alternatively specify the points as argument <i>x</i> , in which case the <i>x_()</i> function will never be called.
--

<b>wt</b> ( <i>self</i> )
---------------------------

Weighting function to get orthonormality. This sets <i>self.numwt</i> .
---

<b>x_</b> ( <i>self</i> , <i>m</i> )
--------------------------------------

Calculates the points at which the function is evaluated, if you want <i>m</i> evenly spaced points. By default, the function is assumed to range over the open interval (-1, 1). You can override this function in a subclass to get a different range.
--



**P**(*self*, *i*)

Self.P(*i*) is the *i*th orthogonal polynomial. The result is normalized so that `numpy.sum(self.wt() * self.P(i)**2) == 1`. It returns an un-shared array, so the array can be modified without affecting future calls to P(*i*).

**expand**(*self*, *c*)**compute**(*self*, *n*)**wt**\_*(self)*

### 96.3.2 Class Variables

Name	Description
registry	<b>Value:</b> {}

## 96.4 Class `ortho_poly`

gmisclib.ortho\_poly.ortho



**gmisclib.ortho\_poly.ortho\_poly**

**Known Subclasses:** gmisclib.ortho\_poly.Chebyshev, gmisclib.ortho\_poly.Chebyshev2, gmisclib.ortho\_poly.

Virtual base class. This is a superclass of all orthorgonal polynomials. This does the recurrence [Abramowitz and Stegun p.782], and ensures that the generated polynomials are all orthonormal to each other. The derived classed need to specify two functions: `recurse()` and `wt_()`. `Recurse()` is the recursion relation from P(*i*) and P(*i*-1) to P(*i*+1), and `wt_()` is the weighting function for the polynomial. `Wt_()` is needed to check orthogonality, and really defines the polynomial. These polynomials are guarenteed to be orthogonal to eachother when summed over the supplied set of *x* points. Thus, if you change `x_()` or `wt_()`, you get a different set of functions.

**96.4.1 Methods**

**`__init__(self, n=None, x=None)`**

Argument *n* is how many points you want the polynomials evaluated at. The ordinates of the points are placed in *self.x*. You may alternatively specify the points as argument *x*, in which case the *x\_()* function will never be called.

Overrides: *gmisclib.ortho\_poly.ortho.\_\_init\_\_* *exitit*(inherited documentation)

**`P(self, i)`**

*Self.P(i)* is the *i*th orthogonal polynomial. The result is normalized so that `numpy.sum(self.P(i)**2) == 1`.

Overrides: *gmisclib.ortho\_poly.ortho.P*

**`recurse(self, n, fn, fnm1)`**

Does the recurrence relation, evaluating *f<sub>·</sub>(n+1)* as a function of *n*, *f<sub>·</sub>n*, and *f<sub>·</sub>(n-1)*. For *n=0* and *n=1*, *fn* and *fnm1* may be *None*.

**`compute(self, n)`**

Overrides: *gmisclib.ortho\_poly.ortho.compute*

**`expand(self, c)`**

**`wt(self)`**

Weighting function to get orthonormality. This sets *self.\_numwt*.

**`wt_(self)`**

**`x_(self, m)`**

Calculates the points at which the function is evaluated, if you want *m* evenly spaced points. By default, the function is assumed to range over the open interval *(-1, 1)*. You can override this function in a subclass to get a different range.

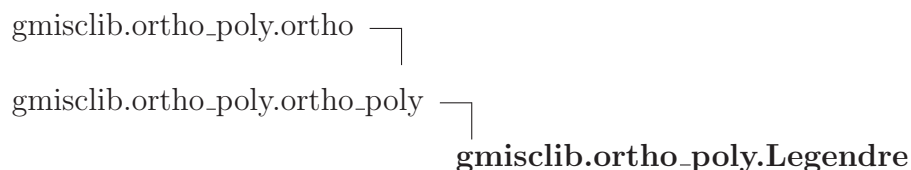
**96.4.2 Class Variables**

Name	Description
<code>__doc__</code>	<b>Value:</b> <code>""</code> "Virtual base cla..."

*continued on next page*

Name	Description
registry	Value: {}

## 96.5 Class Legendre



Legendre polynomials, orthonormal over  $(-1, 1)$  with weight 1. Called  $P_n(x)$  in Abramowitz and Stegun.

### 96.5.1 Methods

**`__init__(self, n=None, x=None)`**

Argument `n` is how many points you want the polynomials evaluated at. The ordinates of the points are placed in `self.x`. You may alternatively specify the points as argument `x`, in which case the `x_()` function will never be called.

Overrides: `gmisclib.ortho_poly.ortho.__init__` `exitit`(inherited documentation)

**`recurse(self, n, fn, fnm1)`**

Does the recurrence relation, evaluating  $f_{n+1}$  as a function of  $n$ ,  $f_n$ , and  $f_{n-1}$ . For  $n=0$  and  $n=1$ , `fn` and `fnm1` may be `None`.

Overrides: `gmisclib.ortho_poly.ortho_poly.recurse` `exitit`(inherited documentation)

**`wt_(self)`**

Overrides: `gmisclib.ortho_poly.ortho.wt_`

**`P(self, i)`**

`Self.P(i)` is the  $i$ th orthogonal polynomial. The result is normalized so that `numpy.sum(self.P(i)**2) == 1`.

Overrides: `gmisclib.ortho_poly.ortho.P`

**compute**(*self*, *n*)

Overrides: gmisclib.ortho\_poly.ortho.compute

**expand**(*self*, *c*)

**wt**(*self*)

Weighting function to get orthonormality. This sets self.\_numwt.

**x\_**(*self*, *m*)

Calculates the points at which the function is evaluated, if you want *m* evenly spaced points. By default, the function is assumed to range over the open interval (-1, 1). You can override this function in a subclass to get a different range.

### 96.5.2 Class Variables

Name	Description
<code>__doc__</code>	<b>Value:</b> <code>""</code> "Legendre polynomia..."
<code>name</code>	<b>Value:</b> <code>'Legendre'</code>
<code>registry</code>	<b>Value:</b> <code>{}</code>

## 96.6 Class Chebyshev

gmisclib.ortho\_poly.ortho

gmisclib.ortho\_poly.ortho\_poly

**gmisclib.ortho\_poly.Chebyshev**

Chebyshev polynomials of the first kind, orthonormal over (-1, 1) with weight  $(1-x^2)^{-1/2}$ . These are the equi-ripple polynomials. Called  $T_n(x)$  in Abramowitz and Stegun.

**96.6.1 Methods**

**`__init__(self, n=None, x=None)`**

Argument `n` is how many points you want the polynomials evaluated at. The ordinates of the points are placed in `self.x`. You may alternatively specify the points as argument `x`, in which case the `x_()` function will never be called.

Overrides: `gmisclib.ortho_poly.ortho.__init__` `exitit`(inherited documentation)

**`recurse(self, n, fn, fnm1)`**

Does the recurrence relation, evaluating  $f_{(n+1)}$  as a function of  $n$ ,  $f_n$ , and  $f_{(n-1)}$ . For  $n=0$  and  $n=1$ , `fn` and `fnm1` may be `None`.

Overrides: `gmisclib.ortho_poly.ortho_poly.recurse` `exitit`(inherited documentation)

**`wt_(self)`**

Overrides: `gmisclib.ortho_poly.ortho.wt_`

**`P(self, i)`**

`Self.P(i)` is the  $i$ th orthogonal polynomial. The result is normalized so that `numpy.sum(self.P(i)**2) == 1`.

Overrides: `gmisclib.ortho_poly.ortho.P`

**`compute(self, n)`**

Overrides: `gmisclib.ortho_poly.ortho.compute`

**`expand(self, c)`**

**`wt(self)`**

Weighting function to get orthonormality. This sets `self._numwt`.

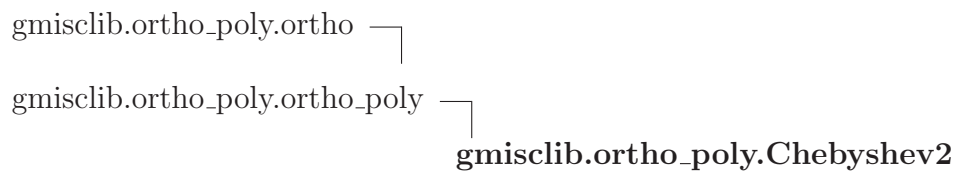
**`x_(self, m)`**

Calculates the points at which the function is evaluated, if you want  $m$  evenly spaced points. By default, the function is assumed to range over the open interval  $(-1, 1)$ . You can override this function in a subclass to get a different range.

**96.6.2 Class Variables**

Name	Description
<code>__doc__</code>	<b>Value:</b> <code>"""Chebyshev polynomials of the first ki...</code>
<code>name</code>	<b>Value:</b> <code>'Chebyshev'</code>
<code>registry</code>	<b>Value:</b> <code>{}</code>

## 96.7 Class *Chebyshev2*



Chebyshev polynomials of the second kind, orthonormal over  $(-1, 1)$  with weight  $(1-x^2)^{1/2}$ . Called  $Un(x)$  in Abramowitz and Stegun.

### 96.7.1 Methods

**`__init__(self, n=None, x=None)`**

Argument `n` is how many points you want the polynomials evaluated at. The ordinates of the points are placed in `self.x`. You may alternatively specify the points as argument `x`, in which case the `x_()` function will never be called.

Overrides: `gmisclib.ortho_poly.ortho.__init__` `exitit`(inherited documentation)

**`recurse(self, n, fn, fnm1)`**

Does the recurrence relation, evaluating `f_(n+1)` as a function of `n`, `f_n`, and `f_(n-1)`. For `n=0` and `n=1`, `fn` and `fnm1` may be `None`.

Overrides: `gmisclib.ortho_poly.ortho_poly.recurse` `exitit`(inherited documentation)

**`wt_(self)`**

Overrides: `gmisclib.ortho_poly.ortho.wt_`

**P**(*self*, *i*)

Self.P(i) is the ith orthogonal polynomial. The result is normalized so that `numpy.sum(self.P(i)**2) == 1`.

Overrides: `gmisclib.ortho_poly.ortho.P`

**compute**(*self*, *n*)

Overrides: `gmisclib.ortho_poly.ortho.compute`

**expand**(*self*, *c*)

**wt**(*self*)

Weighting function to get orthonormality. This sets `self._numwt`.

**x\_**(*self*, *m*)

Calculates the points at which the function is evaluated, if you want *m* evenly spaced points. By default, the function is assumed to range over the open interval  $(-1, 1)$ . You can override this function in a subclass to get a different range.

### 96.7.2 Class Variables

Name	Description
<code>__doc__</code>	<b>Value:</b> <code>""</code> "Chebyshev polynomials of the second ki...
<code>name</code>	<b>Value:</b> <code>'Chebyshev2'</code>
<code>registry</code>	<b>Value:</b> <code>{}</code>

## 96.8 Class SinCos

`gmisclib.ortho_poly.ortho` —

**`gmisclib.ortho_poly.SinCos`**

1,  $\sin(\pi x)$ ,  $\cos(\pi x)$ ,  $\sin(2\pi x)$ ,  $\cos(2\pi x)$ ... orthonormal over  $(-1, 1)$  with weight 1.

## 96.8.1 Methods

**`__init__(self, n=None, x=None)`**

Argument `n` is how many points you want the polynomials evaluated at. The ordinates of the points are placed in `self.x`. You may alternatively specify the points as argument `x`, in which case the `x_()` function will never be called.

Overrides: `gmisclib.ortho_poly.ortho.__init__` `exitit`(inherited documentation)

**`compute(self, n)`**

Overrides: `gmisclib.ortho_poly.ortho.compute`

**`wt_(self)`**

Overrides: `gmisclib.ortho_poly.ortho.wt_`

**`P(self, i)`**

`Self.P(i)` is the `i`th orthogonal polynomial. The result is normalized so that `numpy.sum(self.wt() * self.P(i)**2) == 1`. It returns an un-shared array, so the array can be modified without affecting future calls to `P(i)`.

**`expand(self, c)`**

**`wt(self)`**

Weighting function to get orthonormality. This sets `self.numwt`.

**`x_(self, m)`**

Calculates the points at which the function is evaluated, if you want `m` evenly spaced points. By default, the function is assumed to range over the open interval `(-1, 1)`. You can override this function in a subclass to get a different range.

## 96.8.2 Class Variables

Name	Description
<code>__doc__</code>	Value: <code>"""1, sin(pi*x), cos(pi*x), sin(2*pi*x), cos(2*pi*x)...</code>
<code>name</code>	Value: <code>'SinCos'</code>
<code>registry</code>	Value: <code>{}</code>



## 96.9 Class SLTB

gmsclib.ortho\_poly.ortho —  
**gmsclib.ortho\_poly.SLTB**

Smooth Local Trigonometric Basis. From Björn Jawerth, Yi Liu, Wim Sweldens, "Signal Compression with Smooth Local Basis Functions".

### 96.9.1 Methods

**\_\_init\_\_**(*self*, *n*=None, *x*=None, *eps*=0.5)

Argument *n* is how many points you want the polynomials evaluated at. The ordinates of the points are placed in *self.x*. You may alternatively specify the points as argument *x*, in which case the *x\_()* function will never be called.

Overrides: gmsclib.ortho\_poly.ortho.\_\_init\_\_ extit(inherited documentation)

**compute**(*self*, *n*)

Overrides: gmsclib.ortho\_poly.ortho.compute

**r**(*self*, *x*)

This must have  $l^{*2}+r^{*2}=1$

**b**(*self*, *x*)

**wt\_**(*self*)

Overrides: gmsclib.ortho\_poly.ortho.wt\_

**x\_**(*self*, *m*)

Calculates the points at which the function is evaluated, if you want *m* evenly spaced points. The function is assumed to range over the open interval  $(-eps, 1+eps)$ . You can override this function in a subclass to get a different range.

Overrides: gmsclib.ortho\_poly.ortho.x\_

**test**()

**P**(*self*, *i*)

Self.P(i) is the ith orthogonal polynomial. The result is normalized so that `numpy.sum(self.wt() * self.P(i)**2) == 1`. It returns an un-shared array, so the array can be modified without affecting future calls to P(i).

**expand**(*self*, *c*)

**wt**(*self*)

Weighting function to get orthonormality. This sets self.\_numwt.

### 96.9.2 Class Variables

Name	Description
<code>__doc__</code>	<b>Value:</b> <code>""</code> "Smooth Local Trigonometric Bas..."
<code>name</code>	<b>Value:</b> <code>'SLTB'</code>
<code>registry</code>	<b>Value:</b> <code>{}</code>

## 97 Module `gmisclib.parse_tree_number`

### 97.1 Functions

**`is_mul_of(a, b, indent='')`**

Returns the ratio `a/b` if `a` contains `b` as a factor, returns `None` otherwise. The routine does not promise to find all possible factors.

**`abs(x)`**

**`cos(x)`**

**`sin(x)`**

**`exp(x)`**

**`log(x)`**

**`sqrt(x)`**

**`coerce_into(x)`**

**`operatorN(operator, operands)`**

### 97.2 Variables

Name	Description
<code>__package__</code>	<b>Value:</b> <code>'gmisclib'</code>
<code>__warningregistry__</code>	<b>Value:</b> <code>{('the sets module is deprecated', &lt;type 'exceptions.Depr...</code>

### 97.3 Class Array

object —  
     `gmisclib.parse_tree_number.Array`

## 97.3.1 Methods

<b><code>__init__(self, name)</code></b>
--

This creates an array. Arrays are basic elements of expressions.
--

Overrides: <code>object.__init__</code>
---

<b><code>__getitem__(self, key)</code></b>
--

<b><code>__str__(self, env={})</code></b>
---

<code>str(x)</code>
---------------------

Overrides: <code>object.__str__</code> <code>exitit</code> (inherited documentation)
--

<b><code>__repr__(self, env={})</code></b>
--

<code>str(x)</code>
---------------------

Overrides: <code>object.__repr__</code> <code>exitit</code> (inherited documentation)
---

<b><code>indices(self, env={})</code></b>
---

<b><code>eval(self, env={})</code></b>
--

<b><code>variables(self, env={})</code></b>
---

<b><code>debug(self)</code></b>
---------------------------------

*Inherited from object*

`__delattr__()`, `__format__()`, `__getattr__()`, `__hash__()`, `__new__()`, `__reduce__()`, `__reduce_ex__()`, `__setattr__()`, `__sizeof__()`, `__subclasshook__()`

## 97.3.2 Properties

Name	Description
<i>Inherited from object</i>	
<code>__class__</code>	

## 97.4 Class `output_mixin`

object —  
     `gmisclib.parse_tree_number.output_mixin`

**Known Subclasses:** `gmisclib.parse_tree_number.Expression`, `gmisclib.parse_tree_number.Float`, `gmisclib.parse_tree_number.Name`, `gmisclib.parse_tree_number._operator`

### 97.4.1 Methods

**`__init__(self)`**

`x.__init__(...)` initializes `x`; see `help(type(x))` for signature

Overrides: `object.__init__` `exitit`(inherited documentation)

**`eval(self, env={})`**

This returns an `Expression`.

**`variables(self, env={})`**

Any definitions in `env` are used to resolve otherwise undefined names. This returns a `Set` containing all undefined variable names.

**`indices(self, variable, env={})`**

**`debug(self)`**

**`__str__(self, env={})`**

`Env` is optional. Anything defined in `env` is used to resolve references, but the outcome may still contain undefined variables, which are printed as names.

Overrides: `object.__str__`

### *Inherited from `object`*

`__delattr__()`, `__format__()`, `__getattr__()`, `__hash__()`, `__new__()`, `__reduce__()`, `__reduce_ex__()`, `__repr__()`, `__setattr__()`, `__sizeof__()`, `__subclasshook__()`

### 97.4.2 Properties

Name	Description
<i>Inherited from object</i>	
<code>__class__</code>	

## 97.5 Class `abstract_number`

object └─ `gmisclib.parse_tree_number.abstract_number`

**Known Subclasses:** `gmisclib.parse_tree_number.Expression`, `gmisclib.parse_tree_number.Float`, `gmisclib.parse_tree_number.Name`, `gmisclib.parse_tree_number._operator`

### 97.5.1 Methods

`__init__(self)`

`x.__init__(...)` initializes `x`; see `help(type(x))` for signature

Overrides: `object.__init__` extit(inherited documentation)

`__add__(self, other)`

`__radd__(self, other)`

`__sub__(self, other)`

`__rsub__(self, other)`

`__pow__(self, other)`

`__rpow__(self, other)`

`__mul__(self, other)`

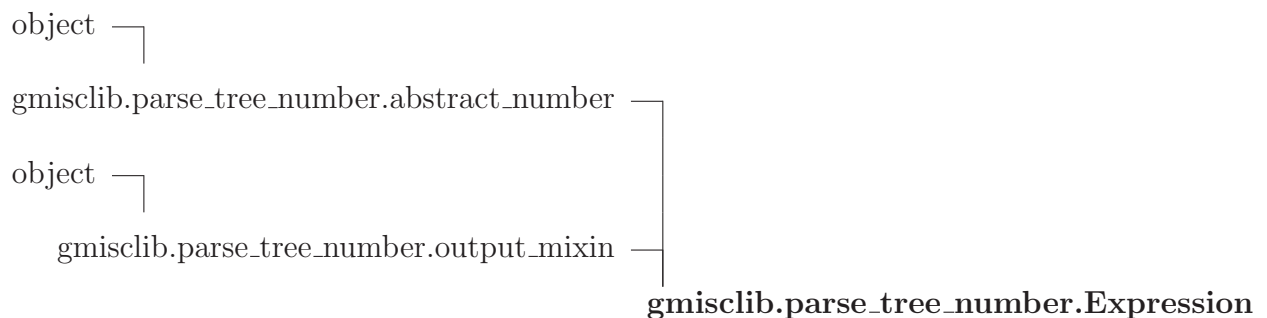
`__rmul__(self, other)`

`__div__(self, other)`

`__rdiv__(self, other)`

`--truediv--(self, other)``--rtruediv--(self, other)``--neg--(self)``--pos--(self)``--abs--(self)`***Inherited from object***`--delattr--()`, `--format--()`, `--getattr--()`, `--hash--()`, `--new--()`, `--reduce--()`, `--reduce_ex--()`,  
`--repr--()`, `--setattr--()`, `--sizeof--()`, `--str--()`, `--subclasshook--()`**97.5.2 Properties**

Name	Description
<i>Inherited from object</i>	
<code>--class--</code>	

**97.6 Class Expression****97.6.1 Methods**`--init--(self, value)``x.__init__(...)` initializes `x`; see `help(type(x))` for signatureOverrides: `object.__init__` `extit`(inherited documentation)

```
--iadd--(self, other)
```

```
--isub--(self, other)
```

```
--imul--(self, other)
```

```
--idiv--(self, other)
```

```
eval(self, env={})
```

This returns an Expression.

Overrides: gmisclib.parse\_tree\_number.output\_mixin.eval extit(inherited documentation)

```
variables(self, env={})
```

Any definitions in env are used to resolve otherwise undefined names. This returns a Set containing all undefined variable names.

Overrides: gmisclib.parse\_tree\_number.output\_mixin.variables extit(inherited documentation)

```
indices(self, variable, env={})
```

Overrides: gmisclib.parse\_tree\_number.output\_mixin.indices

```
--str--(self, env={})
```

Env is optional. Anything defined in env is used to resolve references, but the outcome may still contain undefined variables, which are printed as names.

Overrides: object.\_\_str\_\_ extit(inherited documentation)

```
--repr--(self, env={})
```

Env is optional. Anything defined in env is used to resolve references, but the outcome may still contain undefined variables, which are printed as names.

Overrides: object.\_\_repr\_\_ extit(inherited documentation)

```
debug(self)
```

Overrides: gmisclib.parse\_tree\_number.output\_mixin.debug

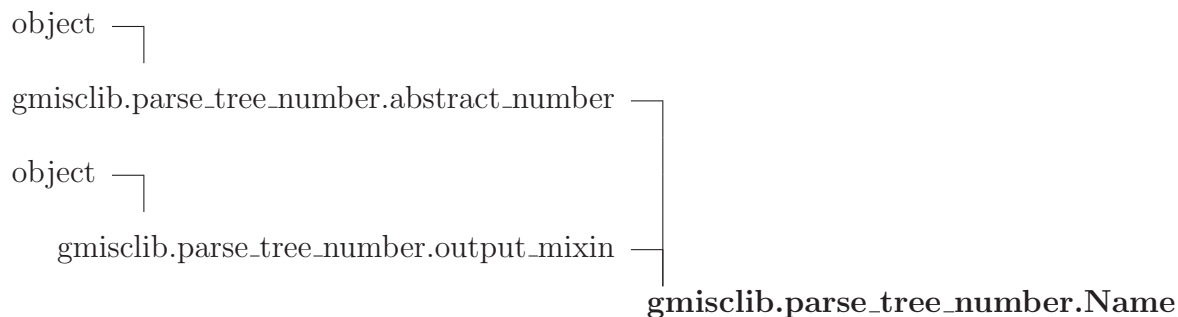
```
--float--(self)
```



`priority(self)``__abs__(self)``__add__(self, other)``__div__(self, other)``__mul__(self, other)``__neg__(self)``__pos__(self)``__pow__(self, other)``__radd__(self, other)``__rdiv__(self, other)``__rmul__(self, other)``__rpow__(self, other)``__rsub__(self, other)``__rtruediv__(self, other)``__sub__(self, other)``__truediv__(self, other)`***Inherited from object***`__delattr__(), __format__(), __getattr__(), __hash__(), __new__(), __reduce__(), __reduce_ex__(),  
__setattr__(), __sizeof__(), __subclasshook__()`**97.6.2 Properties**

Name	Description
<i>Inherited from object</i>	
<code>--class--</code>	

## 97.7 Class Name



### 97.7.1 Methods

<b><code>--init--</code></b> ( <i>self</i> , <i>name</i> )
This creates a named variable. Names are basic parts of expressions. Overrides: <code>object.__init__</code>
<b><code>--str--</code></b> ( <i>self</i> , <i>env</i> ={})
Env is optional. Anything defined in <i>env</i> is used to resolve references, but the outcome may still contain undefined variables, which are printed as names. Overrides: <code>object.__str__</code> <code>exitit</code> (inherited documentation)
<b><code>--repr--</code></b> ( <i>self</i> , <i>env</i> ={})
Env is optional. Anything defined in <i>env</i> is used to resolve references, but the outcome may still contain undefined variables, which are printed as names. Overrides: <code>object.__repr__</code> <code>exitit</code> (inherited documentation)
<b><code>eval</code></b> ( <i>self</i> , <i>env</i> )
This returns an Expression. Overrides: <code>gmisclib.parse_tree_number.output_mixin.eval</code> <code>exitit</code> (inherited documentation)

**variables**(*self*, *env*={})

Any definitions in *env* are used to resolve otherwise undefined names. This returns a Set containing all undefined variable names.

Overrides: `gmisclib.parse_tree_number.output_mixin.variables` `exitit`(inherited documentation)

**indices**(*self*, *variable*, *env*={})

Overrides: `gmisclib.parse_tree_number.output_mixin.indices`

**debug**(*self*)

Overrides: `gmisclib.parse_tree_number.output_mixin.debug`

**\_\_abs\_\_**(*self*)**\_\_add\_\_**(*self*, *other*)**\_\_div\_\_**(*self*, *other*)**\_\_mul\_\_**(*self*, *other*)**\_\_neg\_\_**(*self*)**\_\_pos\_\_**(*self*)**\_\_pow\_\_**(*self*, *other*)**\_\_radd\_\_**(*self*, *other*)**\_\_rdiv\_\_**(*self*, *other*)**\_\_rmul\_\_**(*self*, *other*)**\_\_rpow\_\_**(*self*, *other*)**\_\_rsub\_\_**(*self*, *other*)**\_\_rtruediv\_\_**(*self*, *other*)

```
--sub--(self, other)
```

```
--truediv--(self, other)
```

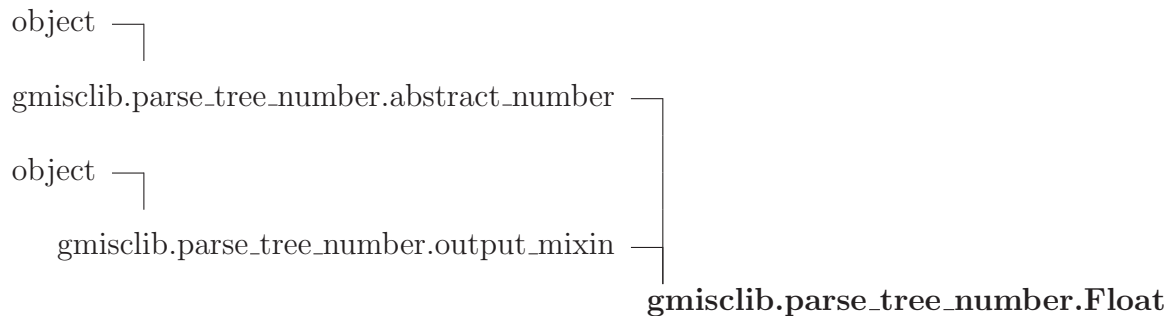
### *Inherited from object*

```
--delattr--(), --format--(), --getattr__(), --hash--(), --new--(), --reduce--(), --reduce_ex--(),
--setattr--(), --sizeof--(), --subclasshook--()
```

#### 97.7.2 Properties

Name	Description
<i>Inherited from object</i>	
<code>--class--</code>	

## 97.8 Class Float



#### 97.8.1 Methods

```
--init--(self, v)
```

`x.__init__(...)` initializes `x`; see `help(type(x))` for signature

Overrides: `object.__init__` extit(inherited documentation)

```
eval(self, env={})
```

This returns an Expression.

Overrides: `gmisclib.parse_tree_number.output_mixin.eval` extit(inherited documentation)

**variables**(*self*, *env*={})

Any definitions in *env* are used to resolve otherwise undefined names. This returns a Set containing all undefined variable names.

Overrides: gmisclib.parse\_tree\_number.output\_mixin.variables [exitit\(inherited documentation\)](#)

**indices**(*self*, *variable*, *env*={})

Overrides: gmisclib.parse\_tree\_number.output\_mixin.indices

**debug**(*self*, *env*={})

Overrides: gmisclib.parse\_tree\_number.output\_mixin.debug

**\_\_str\_\_**(*self*, *env*={})

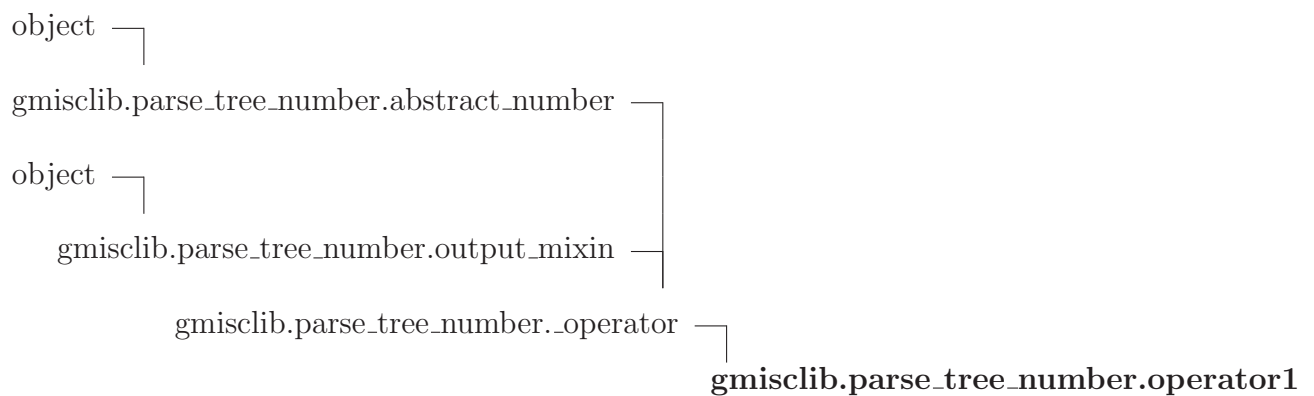
Env is optional. Anything defined in *env* is used to resolve references, but the outcome may still contain undefined variables, which are printed as names.

Overrides: object.\_\_str\_\_ [exitit\(inherited documentation\)](#)

**\_\_float\_\_**(*self*)**\_\_abs\_\_**(*self*)**\_\_add\_\_**(*self*, *other*)**\_\_div\_\_**(*self*, *other*)**\_\_mul\_\_**(*self*, *other*)**\_\_neg\_\_**(*self*)**\_\_pos\_\_**(*self*)**\_\_pow\_\_**(*self*, *other*)**\_\_radd\_\_**(*self*, *other*)**\_\_rdiv\_\_**(*self*, *other*)

`--rmul--(self, other)``--rpow--(self, other)``--rsub--(self, other)``--rtruediv--(self, other)``--sub--(self, other)``--truediv--(self, other)`***Inherited from object***`--delattr--()`, `--format--()`, `--getattr--()`, `--hash--()`, `--new--()`, `--reduce--()`, `--reduce_ex--()`, `--repr--()`, `--setattr--()`, `--sizeof--()`, `--subclasshook--()`**97.8.2 Properties**

Name	Description
<i>Inherited from object</i>	
<code>--class--</code>	

**97.9 Class `operator1`**

**97.9.1 Methods**

**`__init__(self, operator, operand)`**

`x.__init__(...)` initializes `x`; see `help(type(x))` for signature

Overrides: `object.__init__` extit(inherited documentation)

**`__str__(self, env={})`**

Env is optional. Anything defined in `env` is used to resolve references, but the outcome may still contain undefined variables, which are printed as names.

Overrides: `object.__str__` extit(inherited documentation)

**`__repr__(self, env={})`**

Env is optional. Anything defined in `env` is used to resolve references, but the outcome may still contain undefined variables, which are printed as names.

Overrides: `object.__repr__` extit(inherited documentation)

**`eval(self, env={})`**

This returns an Expression.

Overrides: `gmisclib.parse_tree_number.output_mixin.eval` extit(inherited documentation)

**`variables(self, env={})`**

Any definitions in `env` are used to resolve otherwise undefined names. This returns a Set containing all undefined variable names.

Overrides: `gmisclib.parse_tree_number.output_mixin.variables` extit(inherited documentation)

**`indices(self, variable, env={})`**

Overrides: `gmisclib.parse_tree_number.output_mixin.indices`

**`debug(self)`**

Overrides: `gmisclib.parse_tree_number.output_mixin.debug`

**`__abs__(self)`**

`--add__(self, other)``--div__(self, other)``--mul__(self, other)``--neg__(self)``--pos__(self)``--pow__(self, other)``--radd__(self, other)``--rdiv__(self, other)``--rmul__(self, other)``--rpow__(self, other)``--rsub__(self, other)``--rtruediv__(self, other)``--sub__(self, other)``--truediv__(self, other)`*Inherited from `gmisclib.parse_tree_number.operator`*`priority()`*Inherited from object*`--delattr__(), --format__(), --getattr__(), --hash__(), --new__(), --reduce__(), --reduce_ex__(),  
--setattr__(), --sizeof__(), --subclasshook__()`

### 97.9.2 Properties

Name	Description
<i>Inherited from object</i>	

*continued on next page*

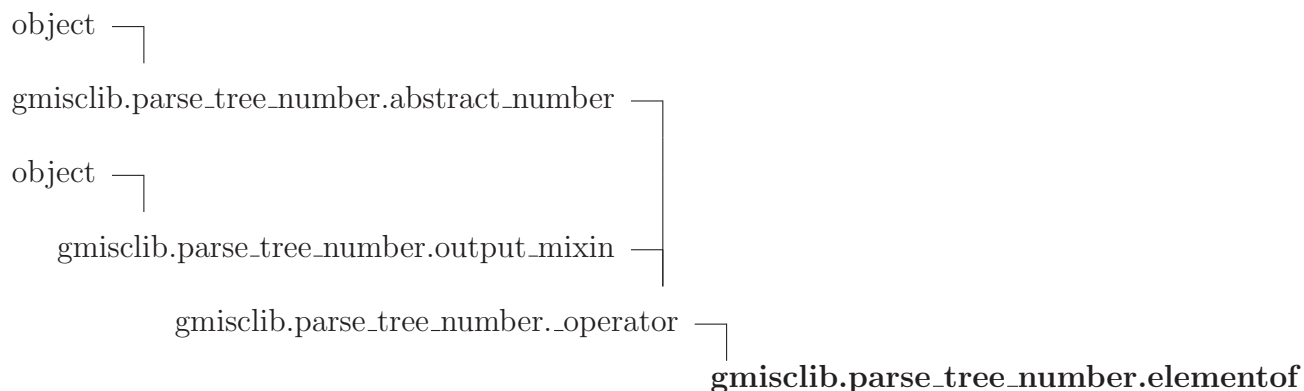


Name	Description
<code>--class--</code>	

### 97.9.3 Class Variables

Name	Description
Printable	<b>Value:</b> { <code>'U-'</code> : <code>'-'</code> , <code>'abs'</code> : <code>'abs'</code> , <code>'cos'</code> : <code>'math.cos'</code> , <code>'exp'</code> : <code>'math...</code>
<i>Inherited from gmisclib.parse_tree_number._operator</i>	
Priority	

## 97.10 Class elementof



### 97.10.1 Methods

**--init--**(*self*, *array*, *index*)

*x*.**--init--**(...) initializes *x*; see `help(type(x))` for signature

Overrides: `object.__init__` `extit`(inherited documentation)

**--str--**(*self*, *env*={})

*Env* is optional. Anything defined in *env* is used to resolve references, but the outcome may still contain undefined variables, which are printed as names.

Overrides: `object.__str__` `extit`(inherited documentation)

```
--repr--(self, env={})
```

Env is optional. Anything defined in env is used to resolve references, but the outcome may still contain undefined variables, which are printed as names.

Overrides: object.\_\_repr\_\_ exitit(inherited documentation)

```
variables(self, env={})
```

Any definitions in env are used to resolve otherwise undefined names. This returns a Set containing all undefined variable names.

Overrides: gmisclib.parse\_tree\_number.output\_mixin.variables exitit(inherited documentation)

```
indices(self, variable, env={})
```

Overrides: gmisclib.parse\_tree\_number.output\_mixin.indices

```
debug(self)
```

Overrides: gmisclib.parse\_tree\_number.output\_mixin.debug

```
--abs--(self)
```

```
--add--(self, other)
```

```
--div--(self, other)
```

```
--mul--(self, other)
```

```
--neg--(self)
```

```
--pos--(self)
```

```
--pow--(self, other)
```

```
--radd--(self, other)
```

```
--rdiv--(self, other)
```

```
--rmul--(self, other)
```

<code>--rpow--(<i>self</i>, <i>other</i>)</code>
--

<code>--rsub--(<i>self</i>, <i>other</i>)</code>
--

<code>--rtruediv--(<i>self</i>, <i>other</i>)</code>
--

<code>--sub--(<i>self</i>, <i>other</i>)</code>
---

<code>--truediv--(<i>self</i>, <i>other</i>)</code>
---

<code>eval(<i>self</i>, <i>env</i>={})</code>
---

This returns an Expression.
-----------------------------

*Inherited from `gmisclib.parse_tree_number._operator`*

`priority()`

*Inherited from object*

`--delattr--()`, `--format--()`, `--getattr--()`, `--hash--()`, `--new--()`, `--reduce--()`, `--reduce_ex--()`,  
`--setattr--()`, `--sizeof--()`, `--subclasshook--()`

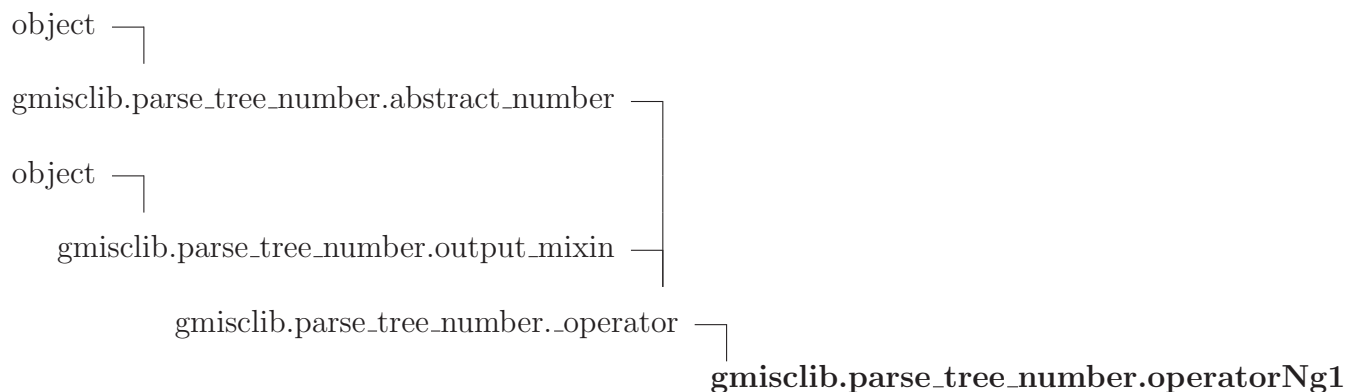
### 97.10.2 Properties

Name	Description
<i>Inherited from object</i>	
<code>--class--</code>	

### 97.10.3 Class Variables

Name	Description
<i>Inherited from <code>gmisclib.parse_tree_number._operator</code></i>	
Priority	

## 97.11 Class `operatorNg1`



### 97.11.1 Methods

**`__init__(self, operator, operands)`**

`x.__init__(...)` initializes `x`; see `help(type(x))` for signature

Overrides: `object.__init__` `exitit`(inherited documentation)

**`more(self, items)`**

**`__str__(self, env={})`**

`Env` is optional. Anything defined in `env` is used to resolve references, but the outcome may still contain undefined variables, which are printed as names.

Overrides: `object.__str__` `exitit`(inherited documentation)

**`__repr__(self, env={})`**

`Env` is optional. Anything defined in `env` is used to resolve references, but the outcome may still contain undefined variables, which are printed as names.

Overrides: `object.__repr__` `exitit`(inherited documentation)

**`variables(self, env={})`**

Any definitions in `env` are used to resolve otherwise undefined names. This returns a `Set` containing all undefined variable names.

Overrides: `gmisclib.parse_tree_number.output_mixin.variables` `exitit`(inherited documentation)

**indices**(*self*, *variable*, *env*={})

Overrides: gmisclib.parse\_tree.number.output\_mixin.indices

**debug**(*self*)

Overrides: gmisclib.parse\_tree.number.output\_mixin.debug

**\_\_abs\_\_**(*self*)**\_\_add\_\_**(*self*, *other*)**\_\_div\_\_**(*self*, *other*)**\_\_mul\_\_**(*self*, *other*)**\_\_neg\_\_**(*self*)**\_\_pos\_\_**(*self*)**\_\_pow\_\_**(*self*, *other*)**\_\_radd\_\_**(*self*, *other*)**\_\_rdiv\_\_**(*self*, *other*)**\_\_rmul\_\_**(*self*, *other*)**\_\_rpow\_\_**(*self*, *other*)**\_\_rsub\_\_**(*self*, *other*)**\_\_rtruediv\_\_**(*self*, *other*)**\_\_sub\_\_**(*self*, *other*)**\_\_truediv\_\_**(*self*, *other*)**eval**(*self*, *env*={})

This returns an Expression.

*Inherited from `gmisclib.parse_tree_number._operator`*

`priority()`

*Inherited from object*

`__delattr__()`, `__format__()`, `__getattribute__()`, `__hash__()`, `__new__()`, `__reduce__()`, `__reduce_ex__()`,  
`__setattr__()`, `__sizeof__()`, `__subclasshook__()`

#### 97.11.2 Properties

Name	Description
<i>Inherited from object</i> <code>__class__</code>	

#### 97.11.3 Class Variables

Name	Description
<i>Inherited from <code>gmisclib.parse_tree_number._operator</code></i> Priority	

## 98 Module *gmisclib.permute*

Find different permutations of an array.

### 98.1 Functions

<b>next</b> ( <i>a</i> , <i>n</i> )
-------------------------------------

Finds the first valid permutation of <i>n</i> items after <i>a</i> . A valid permutation is an array of length <i>n</i> which contains values from 0 to <i>n</i> -1, once each. This function returns the lexicographically next permutation; it's input need not be a valid permutation, but all the entries of <i>a</i> should be in the range 0 to <i>n</i> -1, inclusive.
---

<b>factorial</b> ( <i>n</i> )
-------------------------------

<b>permutations</b> ( <i>n</i> )
----------------------------------

Returns an array of all the <i>n</i> ! permutations of <i>n</i> objects.
--

### 98.2 Variables

Name	Description
<code>__package__</code>	<b>Value:</b> <code>'gmisclib'</code>

## 99 Module `gmisclib.pylab_oneaxis`

You can control the axis tick and grid properties

### 99.1 Functions

<b>plot_oneaxis</b> ( <i>dlist</i> )
--------------------------------------

<i>dlist</i> is a list of (label, (ctr, uerr, derr)).
---

### 99.2 Variables

Name	Description
<code>--package--</code>	<b>Value:</b> <code>'gmisclib'</code>



## 100 Module `gmisclib.pylab.starplot`

This makes a little 'star' of error bars, using `pylab/matplotlib`.

### 100.1 Functions

`grey_out(c)`

`star(cx, cy, lch, lcolormap=None, mcolormap=None, dimmer=<function grey_out at 0x5f315f0>, **kwargs)`

This makes a star of error bars.

#### Parameters

**cx:** x-coordinate of the star center

**cy:** y-coordinate of the star center

**lch:** list of error bars to create about the center.

(*type=list(errorbar).*)

`plot(xlch, **kwargs)`

### 100.2 Variables

Name	Description
Grey	Value: (0.7, 0.7, 0.7)
<code>--package--</code>	Value: 'gmisclib'

### 100.3 Class `errorbar_maker`

object  `gmisclib.pylab.starplot.errorbar_maker`

This is a class that makes error bars, tilted at various angles.

## 100.3.1 Methods

```
__init__(self, cx, cy, n, **kwargs)
```

x.\_\_init\_\_(...) initializes x; see help(type(x)) for signature

Overrides: object.\_\_init\_\_ extit(inherited documentation)

```
make(self, i, xl, xc, xh, **kwargs)
```

*Inherited from object*

\_\_delattr\_\_(), \_\_format\_\_(), \_\_getattr\_\_(), \_\_hash\_\_(), \_\_new\_\_(), \_\_reduce\_\_(), \_\_reduce\_ex\_\_(),  
\_\_repr\_\_(), \_\_setattr\_\_(), \_\_sizeof\_\_(), \_\_str\_\_(), \_\_subclasshook\_\_()

## 100.3.2 Properties

Name	Description
<i>Inherited from object</i>	
__class__	

## 100.4 Class errorbar

```
object └─ gmisclib.pylab.starplot.errorbar
```

## 100.4.1 Methods

```
__init__(self, l, c, h, dim=False, sortorder=None, **kwargs)
```

x.\_\_init\_\_(...) initializes x; see help(type(x)) for signature

Overrides: object.\_\_init\_\_ extit(inherited documentation)

```
__mul__(self, other)
```

```
__cmp__(self, other)
```

*Inherited from object*

\_\_delattr\_\_(), \_\_format\_\_(), \_\_getattr\_\_(), \_\_hash\_\_(), \_\_new\_\_(), \_\_reduce\_\_(), \_\_reduce\_ex\_\_(),  
\_\_repr\_\_(), \_\_setattr\_\_(), \_\_sizeof\_\_(), \_\_str\_\_(), \_\_subclasshook\_\_()

#### 100.4.2 Properties

Name	Description
<i>Inherited from object</i> __class__	

## 101 Module `gmisclib.read_dicom`

### 101.1 Functions

<code>read_header(f)</code>
-----------------------------

<code>read_body(f)</code>
---------------------------

<code>read_imgs(f)</code>
---------------------------

### 101.2 Variables

Name	Description
<code>__package__</code>	Value: <code>'gmisclib'</code>

### 101.3 Class `img_with_mx`

object —  
     `gmisclib.read_dicom.img_with_mx`

This is a way of reading in a 2-D image when you don't know the final size. It keeps a larger image around and keeps track of the area that has been set.

#### 101.3.1 Methods

<code>__init__(self, i, j)</code>
-----------------------------------

Create an image that contains pixel (i,j).
--

Overrides: <code>object.__init__</code>
---

<code>resize(self, inxt, jnxt)</code>
---------------------------------------

<code>set(self, i, j, val)</code>
-----------------------------------

Set a pixel, expanding the allocated space as needed.
---

<code>get(self)</code>
------------------------

Get the image, trimming it to show the area that has been set.
--

***Inherited from object***

`__delattr__()`, `__format__()`, `__getattr__()`, `__hash__()`, `__new__()`, `__reduce__()`, `__reduce_ex__()`,  
`__repr__()`, `__setattr__()`, `__sizeof__()`, `__str__()`, `__subclasshook__()`

**101.3.2 Properties**

Name	Description
<i>Inherited from object</i>	
<code>__class__</code>	

## 102 Module *gmisclib.robust\_multivariate*

Does a robust estimate of covariance. It doesn't converge, unfortunately, for vectors of dimension higher than 1.

### 102.1 Functions

<b>cov_wt</b> ( <i>veclist</i> , <i>weights</i> )
---

<b>integrate_guts</b> ( <i>xlow</i> , <i>xhigh</i> , <i>fcn</i> , <i>c</i> , <i>n</i> )
---

<b>integrate</b> ( <i>xlow</i> , <i>xhigh</i> , <i>fcn</i> , <i>c=None</i> , <i>nstart=5</i> , <i>tol=0.0001</i> )
--

A generally useful function, does the integral of <i>fcn</i> ( <i>x</i> , <i>c</i> ) from <i>xlow</i> to <i>xhigh</i> .
---

<b>wtfunc</b> ( <i>q</i> , <i>E</i> )
---------------------------------------

<b>wtdim</b> ( <i>q</i> , <i>Edim</i> )
---

<b>covariance</b> ( <i>veclist</i> , <i>emax=1.0</i> )
--

<i>Veclist</i> [ <i>i</i> ] is a vector.
--

### 102.2 Variables

Name	Description
<code>--package--</code>	<b>Value:</b> 'gmisclib'

## 103 Module gmisclib.root

Solve a 1-dimensional equation to find the roots. This does alternating secant and bisection steps. It will work for any function, discontinuous or not. For a nasty function, it might be a factor of 2 slower than bisection, but for a nearly-linear function, it can be much faster than bisection.

### 103.1 Functions

**root**(*f, xl, xh, p, epsx, epsf*)

Find a root of an equation.

**Parameters**

**f**: is the function to solve, called as  $f(x, p)$ .

**p**: is arbitrary data for **f**. This function assumes there is known to be a root in  $[xl, xh]$  and that  $f(xl, p)$  has a different sign from  $f(xh, p)$ . It finds it within a region of width *epsx*, or at least finds an  $x$  such that  $-epsf < f(x, p) < epsf$ .

**Return Value**

a root in the interval

(*type=float*)

**test**()

**iroot**(*y, xl, xh, epsy=0.0*)

Find a zero in an array *y*. This function assumes there is known to be a root in  $[xl, xh]$ . It returns a real-number index into the array which linearly interpolates to zero.

**testi**()

### 103.2 Variables

Name	Description
<code>--package--</code>	<b>Value:</b> None

## 104 Module `gmisclib.rubber_array`

This acts like a dictionary, but indexed by integers and stored rather more efficiently, at least in terms of memory. It is designed for very sparse arrays.

### 104.1 Variables

Name	Description
<code>__package__</code>	<b>Value:</b> <code>'gmisclib'</code>

### 104.2 Class `ext_block`

object └─ `gmisclib.rubber_array.ext_block`

#### 104.2.1 Methods

```
__init__(self, n, vtype=<type 'numpy.float64'>)  

x.__init__(...) initializes x; see help(type(x)) for signature  

Overrides: object.__init__ extit(inherited documentation)
```

```
set(self, k, v)
```

```
get(self, k, IdxErrArg)
```

```
contains(self, value)
```

#### *Inherited from `object`*

```
__delattr__(), __format__(), __getattr__(), __hash__(), __new__(), __reduce__(), __reduce_ex__(),  

__repr__(), __setattr__(), __sizeof__(), __str__(), __subclasshook__()
```

#### 104.2.2 Properties

Name	Description
<i>Inherited from <code>object</code></i>	
<code>__class__</code>	



### 104.3 Class `extensible_array`



#### 104.3.1 Methods

```
--init__(self, bsz=300)
```

`x.__init__(...)` initializes `x`; see `help(type(x))` for signature

Overrides: `object.__init__` `extit`(inherited documentation)

```
--len__(self)
```

```
--setitem__(self, k, v)
```

Doesn't do slice objects.

```
--getitem__(self, k)
```

```
unset(self, key)
```

```
--contains__(self, item)
```

```
--iter__(self)
```

#### *Inherited from `object`*

`--delattr__()`, `--format__()`, `--getattr__()`, `--hash__()`, `--new__()`, `--reduce__()`, `--reduce_ex__()`,  
`--repr__()`, `--setattr__()`, `--sizeof__()`, `--str__()`, `--subclasshook__()`

#### 104.3.2 Properties

Name	Description
<i>Inherited from <code>object</code></i>	
<code>--class__</code>	

## 105 Module gmisclib.s\_lin\_fit

Fit a plane to some data.

### 105.1 Functions

**plane**(*data*, *wt*=None)

Fit a plane to some data. where the fitting function is  $f = c[0]*1 + c[1]*dep[0] + c[2]*dep[1] + \dots$  where *c* is the array of coefficients. The length of *c* is equal to the number of dependent variables in each datum.

**Parameters**

**data:** [ (independent, dependent, dependent...), ...]

**Return Value**

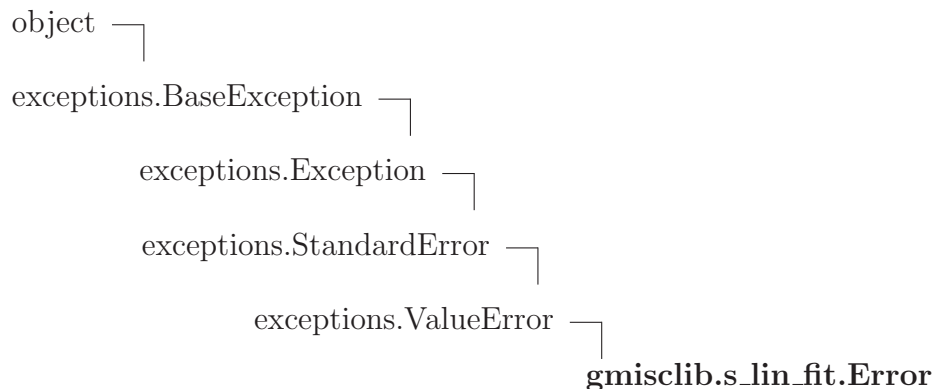
(*c*, *opt*, *resid*, *rank*, *sv*), where *bestfit* is the best fit to the data (i.e., the values of *f*), *resid* is the residual (a single float number), *rank* the rank of the fit (int), and *sv* is the array of the singular values.

**test**()

### 105.2 Variables

Name	Description
<code>__package__</code>	<b>Value:</b> 'gmisclib'

### 105.3 Class Error



**105.3.1 Methods**

**`__init__`**(*self*, *s*)

`x.__init__(...)` initializes `x`; see `help(type(x))` for signature

Overrides: `object.__init__` `__init__`(inherited documentation)

***Inherited from `exceptions.ValueError`***`__new__()`***Inherited from `exceptions.BaseException`***

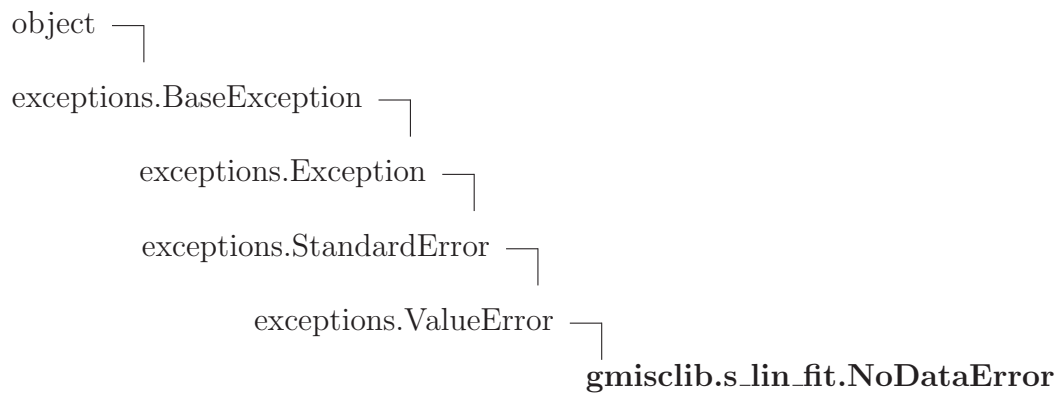
`__delattr__()`, `__getattr__()`, `__getitem__()`, `__getslice__()`, `__reduce__()`, `__repr__()`,  
`__setattr__()`, `__setstate__()`, `__str__()`, `__unicode__()`

***Inherited from `object`***

`__format__()`, `__hash__()`, `__reduce_ex__()`, `__sizeof__()`, `__subclasshook__()`

**105.3.2 Properties**

Name	Description
<i>Inherited from <code>exceptions.BaseException</code></i>	
<code>args</code> , <code>message</code>	
<i>Inherited from <code>object</code></i>	
<code>__class__</code>	

**105.4 Class `NoDataError`**

**105.4.1 Methods**

**`--init--`**(*self*, *s*)

*x*.`--init--`(...) initializes *x*; see `help(type(x))` for signature

Overrides: `object.--init--` `exitit`(inherited documentation)

*Inherited from exceptions.ValueError*

`--new--`()

*Inherited from exceptions.BaseException*

`--delattr--`(), `--getattr--`(), `--getitem--`(), `--getslice--`(), `--reduce--`(), `--repr--`(),  
`--setattr--`(), `--setstate--`(), `--str--`(), `--unicode--`()

*Inherited from object*

`--format--`(), `--hash--`(), `--reduce_ex--`(), `--sizeof--`(), `--subclasshook--`()

**105.4.2 Properties**

Name	Description
<i>Inherited from exceptions.BaseException</i>	
<code>args</code> , <code>message</code>	
<i>Inherited from object</i>	
<code>--class--</code>	

## 106 Module *gmisclib.sbd\_array*

### 106.1 Functions

<b><code>symmetrize</code></b> ( <i>matrix</i> , <i>direction</i> )
---

<b><code>solve</code></b> ( <i>a</i> , <i>b0</i> )
--

solves $a*x = b$ , where <i>a</i> is class <i>sbd</i> : symmetric, positive definite, and band-diagonal. Note that this destroys the contents of <i>a</i> .
---

<b><code>multiply</code></b> ( <i>a</i> , <i>x</i> )
--

Calculates $a*x$ , where <i>a</i> is class <i>sbd</i> : symmetric, positive definite, and band-diagonal.
--

<b><code>test1</code></b> ()
------------------------------

<b><code>err</code></b> ( <i>a</i> , <i>b</i> )
---

<b><code>test2</code></b> ()
------------------------------

<b><code>testm</code></b> ()
------------------------------

<b><code>testa</code></b> ()
------------------------------

<b><code>testbdi</code></b> ()
--------------------------------

### 106.2 Variables

Name	Description
<code>--package--</code>	<b>Value:</b> 'gmisclib'

## 106.3 Class *sbd*

### 106.3.1 Methods

**`--init--`**(*self*, *sz*, *bw*, *\_data=None*)

Constructs a symmetric band-diagonal matrix of  $sz*sz$ , with a nonzero bandwidth of *bw*.  $bw==0$  corresponds to a diagonal matrix.

**`--copy--`**(*self*)

Copy the data, not just the data description.

**`--getitem--`**(*self*, *key*)

Pulls an element of the array. Key is a 2-tuple that specifies the 'virtual' position in the array (i.e., as if the array were a full square matrix, rather than a band diagonal symmetric matrix).

**`--setitem--`**(*self*, *key*, *value*)

**`increment`**(*self*, *key*, *delta*)

Increment a single value (and its symmetric partner, if off the diagonal).

**`bd_increment`**(*self*, *key*, *delta*)

Block diagonal increment.

**`--str--`**(*self*)

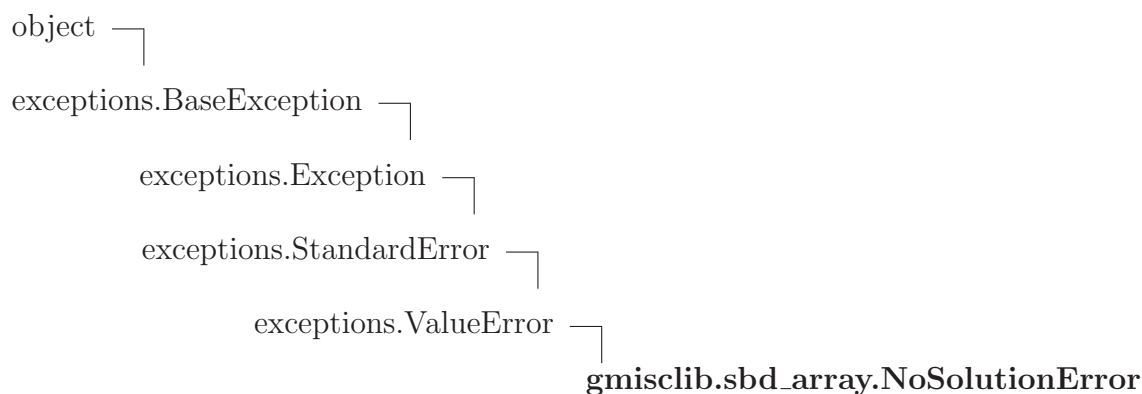
**`--repr--`**(*self*)

**`--pow--`**(*self*, *other*)

**`--getslice--`**(*self*, *i*, *j*)

This is copy semantics, not shared reference.

## 106.4 Class *NoSolutionError*



### 106.4.1 Methods

**`--init--(self, s)`**  
`x.--init--(...)` initializes `x`; see `help(type(x))` for signature  
 Overrides: `object.--init--` `exitit` (inherited documentation)

#### *Inherited from exceptions.ValueError*

`--new--()`

#### *Inherited from exceptions.BaseException*

`--delattr--()`, `--getattr--()`, `--getitem--()`, `--getslice--()`, `--reduce--()`, `--repr--()`,  
`--setattr--()`, `--setstate--()`, `--str--()`, `--unicode--()`

#### *Inherited from object*

`--format--()`, `--hash--()`, `--reduce.ex--()`, `--sizeof--()`, `--subclasshook--()`

### 106.4.2 Properties

Name	Description
<i>Inherited from exceptions.BaseException</i>	
<code>args</code> , <code>message</code>	
<i>Inherited from object</i>	
<code>--class--</code>	

## 107 Module gmisclib.segmentfile

This parses a '.in' file from xwaves. The '.in' files show segmentation of an utterance. `segmentfile.parse(s)` takes a list of lines from such a file, and returns three things: \* the title of the file (typically the utterance) \* a list of groups (a group is typically a word) \* a list of segments.

Groups include a list of the segments of which they are made, along with a name and an index. Segments contain a phoneme, a start/end time, an index, and the group to which they belong.

### 107.1 Functions

**parse(*l*)**

Parses a list of lines from an ESPS Xmark file (.in), and returns a tuple (title, list of groups, list of segments).

**read(*f*)**

Read a ESPS Xmark (.in) file, and return information. See `parse()`.

**test()**

### 107.2 Variables

Name	Description
<code>__package__</code>	<b>Value:</b> None

### 107.3 Class segment

#### 107.3.1 Methods

**`__init__(self, st, en, phn, si)`**

**`setgroup(self, g)`**

**`__str__(self)`**



```
--repr--(self)
```

## 107.4 Class group

### 107.4.1 Methods

```
--init--(self, gi, gn)
```

```
add(self, seg)
```

```
--str--(self)
```

```
--repr--(self)
```

## 108 Module `gmisclib.sharp.energy`

### 108.1 Functions

<b><code>complex_median</code></b> ( <i>x</i> )
---

<b><code>one_median</code></b> ( <i>binw</i> , <i>center</i> , <i>v</i> , <i>nbins</i> =None)
---

Local power in a signal. <i>dt</i> and <i>binw</i> are measured in bins.
--

<b><code>stepped_median</code></b> ( <i>dt</i> , <i>binw</i> , <i>v</i> , <i>nbins</i> =None)
---

Local power in a signal. <i>dt</i> and <i>binw</i> are measured in bins.
--

<b><code>pwr</code></b> ( <i>dt</i> , <i>binw</i> , <i>v</i> , <i>nbins</i> =3)
---

## 109 Module *gmisclib.solve\_sum\_abs*

Solves various equations involving minimizing the sum of absolute values of things.

### 109.1 Functions

**`solve1`**(*y0*, *y1*)

We minimize the equation  $\text{sum\_over\_i}(\text{abs}(y0\_i \cdot (1-x) + y1\_i \cdot x))$  to find the best *x*. Returns *x*. I don't know if this is strictly correct.

**`solve_fit`**(*x*, *y*, *epsx*=1e-07, *epsm*=1e-07)

Solves for the line  $\hat{y} = m \cdot x + b$  that minimizes  $\text{sum}(\text{abs}(y - \hat{y}))$ . In other words, it's a fit to a straight line, but a highly robust fit that will not be disturbed by outliers. The algorithm is from Numerical Recipes, Volume 2.

#### Parameters

**`epsx`**: a tolerance used in the iterative solution. Eps is roughly the required accuracy of the x-intercept of the solution, expressed as a fraction of the x-range of the data.

(*type*=*float*)

**`epsm`**: a tolerance used in the iterative solution. Roughly, it is the accuracy of *m* in the solution. Note the `tangent()` call!

(*type*=*float*)

#### Return Value

(*mhat*, *bhat*), where *mhat*\**x*+*bhat* is the best fit to the data.

(*type*=*tuple(float, float)*)

**solve\_fit\_wt**(*x*, *y*, *wt*, *epsx*=1e-07, *epsm*=1e-07)

Solves for the line  $yhat = m*x + b$  that minimizes  $sum(abs(y - yhat))$ . In other words, it's a fit to a straight line, but a highly robust fit that will not be disturbed by outliers. The algorithm is from Numerical Recipes, Volume 2.

#### Parameters

**epsx**: a tolerance used in the iterative solution. Eps is roughly the required accuracy of the x-intercept of the solution, expressed as a fraction of the x-range of the data.

(*type=float*)

**epsm**: a tolerance used in the iterative solution. Roughly, it is the accuracy of *m* in the solution. Note the `tangent()` call!

(*type=float*)

#### Return Value

(*mhat*, *bhat*), where  $mhat*x + bhat$  is the best fit to the data.

(*type=tuple(float, float)*)

**test1**()

**test2**()

## 109.2 Variables

Name	Description
<code>--package--</code>	<b>Value:</b> <code>'gmisclib'</code>

## 110 Module `gmislib.spread_jobs`

A module that starts a bunch of subprocesses and distributes work amongst them, then collects the results.

Subprocesses must follow a protocol: they must listen for commands on the standard input (commands are encoded with `cPickle`), and they must produce `cPickled` tuples on their standard output. NOTE: THEY CANNOT PRINT ANYTHING ELSE! (But it's OK for subprocesses to produce stuff on the standard error output.)

### PROTOCOL:

1. Execcing state: `spread_jobs` execs a group of subprocesses. You have full control of the argument list.
2. Preparation state: `spread_jobs` will send a list of `cPickled` things, one at a time to the subprocess. No explicit terminator is added, so the subprocess must either know how many things are coming or the list should contain some terminator. (E.g. the last item of the list could be `None`, and the subprocess would wait for it.) In this state, the subprocess must not produce anything on the standard output.
3. Running state: `spread_jobs` will send one `cPickled` thing to the subprocess and then wait for a `cPickled` tuple to come back.

The request to the subprocess is a `tuple(int, arbitrary)`. The `int` is a task-ID number which must be returned with the answer. The `arbitrary` is whatever information the subprocess needs to do its job.

The subprocess responds with a 3-tuple. The first element of the tuple is either:

- An instance of `TooBusy`. This causes the main process to put the task back on the queue and ignore this subprocess for a while. The second element is printed; the third is ignored.
- An instance of `RemoteException`. This leads to termination of the job and causes an exception to be raised on the main thread. The other two elements of the tuple are printed.
- Anything else. In that case, the first element is returned on the output list and the other two elements are printed.

The subprocess loops in the running state. Normally, it should terminate when its standard input is closed. (It can terminate itself if it wishes by simply closing the standard output and exiting.)

4. Shutdown state: The subprocess can produce anything it wants. This will be collected up and returned to the caller of `spread_jobs.main`.

You can use this to run certain normal linux commands by not sending anything in the preparatory state or the running state. You will then be handed the standard output as a list of strings.

Normally, the action happens in the running state.

Normally, the subprocess looks much like this:

```
import cPickle
while True:
    try:
        control = cPickle.load(stdin)
    except EOFError:
        break
    d, so, se = compute(control)
    cPickle.dump((d, so, se), stdout, CP.HIGHEST_PROTOCOL)
    stdout.flush()
stdout.close()
```

## 110.1 Functions

```
main(todo, list_of_args, connection_factory=<class
'gmisclib.spread_jobs.Connection_subprocess'>, stdin=None,
verbose=False, timing_callback=None, tail_callback=None,
past_performance={})
```

Pass a bunch of work to other processes.

### Parameters

**stdin:** a list of stuff to send to the other processes before the computation is properly commenced.  
(*type=list(whatever)*)

**todo:** a sequence of tasks to do  
(*type=sequence(whatever)*)

**list\_of\_args:** (*type=list(list(str))*)

**past\_performance:** a `PastPerformance` object if you want the system to remember which machines were more/less successful last time and to start jobs on the more successful machines first. `None` if you don't want any memory. The default is to have memory.  
(*type=None or PastPerformance.*)

### Return Value

A 2-tuple. The first item is a list of the results produced by the other processes. Items in the returned list correspond to items in the `todo` list. These are the stuff that comes out, pickled, on the standard output after each chunk of data is fed into the standard input. The second item is a list of the remaining outputs, as read by `file.readlines()`; these are one per process.

(*type=tuple(list(whatever), list(list(str)))*)

```
replace(list_of_lists, *fr)
```

```
Replace(list_of_lists, pat, length, replacement)
```

```
append(list_of_lists, *a)
```

```
delay_sanitise(x)
```

```
test_worker(x)
```

<code>test_(<i>script</i>)</code>
-----------------------------------

<code>one_shot_test(<i>input</i>)</code>
--

## 110.2 Variables

Name	Description
<code>--package--</code>	<b>Value:</b> 'gmisclib'

## 110.3 Class *notComputed*

object └─ `gmisclib.spread_jobs.notComputed`

A singleton marker for values that haven't been computed.

### 110.3.1 Methods

#### *Inherited from object*

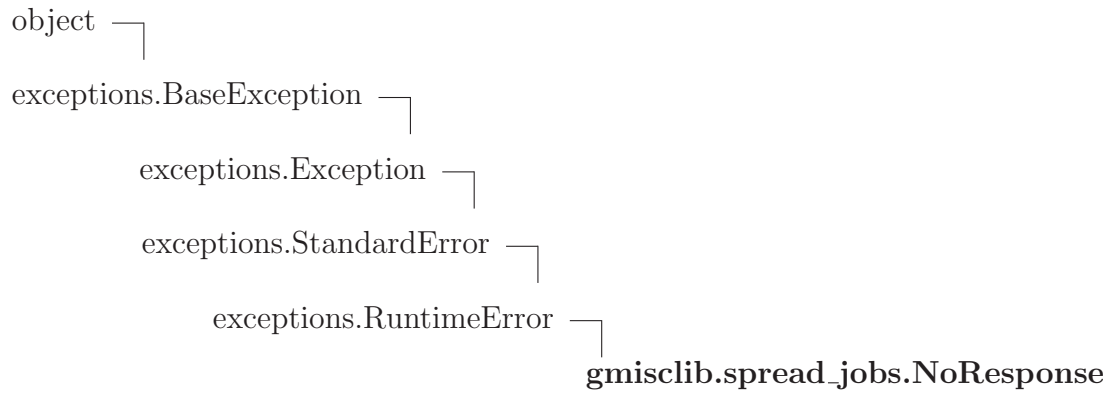
`--delattr--()`, `--format--()`, `--getattr--()`, `--hash--()`, `--init--()`, `--new--()`, `--reduce--()`,  
`--reduce_ex--()`, `--repr--()`, `--setattr--()`, `--sizeof--()`, `--str--()`, `--subclasshook--()`

### 110.3.2 Properties

Name	Description
<i>Inherited from object</i>	
<code>--class--</code>	



## 110.4 Class NoResponse



### 110.4.1 Methods

`__init__(self, *s)`  
`x.__init__(...)` initializes `x`; see `help(type(x))` for signature  
 Overrides: `object.__init__` `exitit`(inherited documentation)

#### *Inherited from exceptions.RuntimeError*

`__new__()`

#### *Inherited from exceptions.BaseException*

`__delattr__()`, `__getattr__()`, `__getitem__()`, `__getslice__()`, `__reduce__()`, `__repr__()`,  
`__setattr__()`, `__setstate__()`, `__str__()`, `__unicode__()`

#### *Inherited from object*

`__format__()`, `__hash__()`, `__reduce_ex__()`, `__sizeof__()`, `__subclasshook__()`

### 110.4.2 Properties

Name	Description
<i>Inherited from exceptions.BaseException</i>	
<code>args</code> , <code>message</code>	
<i>Inherited from object</i>	
<code>__class__</code>	

## 110.5 Class RemoteException



An exception that corresponds to one raised by a subprocess. This is raised in the parent process.

### 110.5.1 Methods

```
__init__(self, *s)
```

*x*.**\_\_init\_\_**(...) initializes *x*; see `help(type(x))` for signature  
 Overrides: `object.__init__` `exitit`(inherited documentation)

```
__repr__(self)
```

`repr(x)`  
 Overrides: `object.__repr__` `exitit`(inherited documentation)

```
raise_me(self)
```

*Inherited from exceptions.Exception*

```
__new__()
```

*Inherited from exceptions.BaseException*

```
__delattr__(), __getattr__(), __getitem__(), __getslice__(), __reduce__(), __setattr__(),  

__setstate__(), __str__(), __unicode__()
```

*Inherited from object*

```
__format__(), __hash__(), __reduce_ex__(), __sizeof__(), __subclasshook__()
```

### 110.5.2 Properties

Name	Description
<i>Inherited from exceptions.BaseException</i>	

*continued on next page*

Name	Description
args, message	
<i>Inherited from object</i>	
__class__	

## 110.6 Class *TooBusy*



### 110.6.1 Methods

<b>__init__(self, delay)</b>  x.__init__(...) initializes x; see help(type(x)) for signature Overrides: object.__init__ extit(inherited documentation)
---

#### *Inherited from object*

\_\_delattr\_\_(), \_\_format\_\_(), \_\_getattr\_\_(), \_\_hash\_\_(), \_\_new\_\_(), \_\_reduce\_\_(), \_\_reduce\_ex\_\_(),  
 \_\_repr\_\_(), \_\_setattr\_\_(), \_\_sizeof\_\_(), \_\_str\_\_(), \_\_subclasshook\_\_()

### 110.6.2 Properties

Name	Description
<i>Inherited from object</i>	
__class__	

## 110.7 Class *PastPerformance*



This class keeps track of which machines are more and less successful. It is normally used

as a key, to sort machines into order.

### 110.7.1 Methods

**`__init__(self)`**

`x.__init__(...)` initializes `x`; see `help(type(x))` for signature

**Return Value**

new empty dictionary

Overrides: `object.__init__` extit(inherited documentation)

**`add_many(self, kvpairs)`**

**`__call__(self, x)`**

**`add(self, key, value, weight=1.0)`**

Add value to the value indexed by key.

**`copy(self)`**

**Return Value**

a shallow copy of `D`

Overrides: `dict.copy` extit(inherited documentation)

**`get_avg(self, key)`**

**`get_avgs(self)`**

**`get_both(self, key)`**

**`merge(self, other)`**

#### *Inherited from dict*

`__cmp__()`, `__contains__()`, `__delitem__()`, `__eq__()`, `__ge__()`, `__getattr__()`, `__getitem__()`, `__gt__()`, `__iter__()`, `__le__()`, `__len__()`, `__lt__()`, `__ne__()`, `__new__()`, `__repr__()`, `__setitem__()`, `__sizeof__()`, `clear()`, `fromkeys()`, `get()`, `has_key()`, `items()`, `iteritems()`, `iterkeys()`, `itervalues()`, `keys()`, `pop()`, `popitem()`, `setdefault()`, `update()`, `values()`, `viewitems()`, `viewkeys()`, `viewvalues()`

#### *Inherited from object*

`--delattr--()`, `--format--()`, `--reduce--()`, `--reduce_ex--()`, `--setattr--()`, `--str--()`, `--subclasshook--()`

### 110.7.2 Properties

Name	Description
<i>Inherited from object</i>	
<code>--class--</code>	

### 110.7.3 Class Variables

Name	Description
<i>Inherited from dict</i>	
<code>--hash--</code>	

## 110.8 Class CannotCreateConnection



### 110.8.1 Methods

<code>--init--(<i>self</i>, *<i>s</i>)</code>
<code>x.--init--(...)</code> initializes x; see <code>help(type(x))</code> for signature
Overrides: <code>object.--init--</code> extit(inherited documentation)

*Inherited from exceptions.Exception*

`--new--()`

*Inherited from exceptions.BaseException*

`--delattr--()`, `--getattribute--()`, `--getitem--()`, `--getslice--()`, `--reduce--()`, `--repr--()`,  
`--setattr--()`, `--setstate--()`, `--str--()`, `--unicode--()`

*Inherited from object*

`__format__()`, `__hash__()`, `__reduce_ex__()`, `__sizeof__()`, `__subclasshook__()`

### 110.8.2 Properties

Name	Description
<i>Inherited from <code>exceptions.BaseException</code></i>	
<code>args</code> , <code>message</code>	
<i>Inherited from <code>object</code></i>	
<code>__class__</code>	

## 110.9 Class Connection



**Known Subclasses:** `gmislib.spread_jobs.Connection_subprocess`

This class represents a connection from the master process down to one of the slaves. It also keeps track of how often the slave reports that it is too busy to work.

### 110.9.1 Methods

<b><code>__init__(self, arglist)</code></b>
<code>x.__init__(...)</code> initializes <code>x</code> ; see <code>help(type(x))</code> for signature
<b>Parameters</b>
<b><code>arglist</code>:</b> an argument list to execute to start a subprocess. <i>(type=a sequence of anything that can be converted to strings.)</i>
<b>Raises</b>
<b><code>OSError</code></b> when connection cannot be set up.
Overrides: <code>object.__init__</code>
<b>Note:</b> This is where the <code>arglist</code> is finally converted to a list of strings.

<b><code>send(self, todo)</code></b>
<b>Raises</b>
<b><code>IOError</code></b> Trouble sending.

<code>get(self)</code>
------------------------

<b>Return Value</b>
---------------------

(answer, standard_output, standard_error) or None.
--

<b>Raises</b>
---------------

<code>EOFError</code> No data available from slave.
---

<code>close(self)</code>
--------------------------

Closes the channel to the slave.
----------------------------------

<code>wait(self)</code>
-------------------------

Waits for the slave to terminate and closes the channel from the slave.
---

<b>Return Value</b>
---------------------

any final output.
-------------------

( <i>type=list(str)</i> )
---------------------------

<code>argstring(self)</code>
------------------------------

<code>usefulness(self)</code>
-------------------------------

<code>I_am_working(self, now)</code>
--------------------------------------

<code>mystate(self, state)</code>
-----------------------------------

<code>performance(self)</code>
--------------------------------

### *Inherited from object*

`__delattr__()`, `__format__()`, `__getattr__()`, `__hash__()`, `__new__()`, `__reduce__()`, `__reduce_ex__()`, `__repr__()`, `__setattr__()`, `__sizeof__()`, `__str__()`, `__subclasshook__()`

#### 110.9.2 Properties

Name	Description
<i>Inherited from object</i>	
<code>__class__</code>	

#### 110.9.3 Class Variables

Name	Description
BUSYFAC1	<b>Value:</b> 0.85
BUSYFAC2	<b>Value:</b> 0.15
ConnectError	<b>Value:</b> ()
SendError	<b>Value:</b> (<type 'exceptions.IOError'>, <type 'exceptions.ValueErro...>)
GetError	<b>Value:</b> (<type 'exceptions.EOFError'>)

#### 110.9.4 Instance Variables

Name	Description
uness	How useful is this thread?

### 110.10 Class `Connection_subprocess`



This is a `Connection` via `stdin/stdout` to a subprocess.

#### 110.10.1 Methods

**`__init__(self, arglist)`**

`x.__init__(...)` initializes `x`; see `help(type(x))` for signature

**Parameters**

**arglist:** an argument list to execute to start a subprocess.

**Raises**

`OSError` when connection cannot be set up.

Overrides: `object.__init__` `exitit`(inherited documentation)

**`send(self, todo)`**

**Raises**

`IOError` Trouble sending.

Overrides: `gmisclib.spread_jobs.Connection.send` `exitit`(inherited documentation)



**get**(*self*)

**Return Value**

(answer, standard\_output, standard\_error) or None.

**Raises**

EOFError No data available from slave.

Overrides: *gmisclib.spread\_jobs.Connection.get* *exitit*(inherited documentation)

**close1**(*self*)

**close2**(*self*)

**I\_am\_working**(*self*, *now*)

**argstring**(*self*)

**close**(*self*)

Closes the channel to the slave.

**mystate**(*self*, *state*)

**performance**(*self*)

**usefulness**(*self*)

**wait**(*self*)

Waits for the slave to terminate and closes the channel from the slave.

**Return Value**

any final output.

(*type=list(str)*)

**Inherited from object**

`__delattr__()`, `__format__()`, `__getattr__()`, `__hash__()`, `__new__()`, `__reduce__()`, `__reduce_ex__()`,  
`__repr__()`, `__setattr__()`, `__sizeof__()`, `__str__()`, `__subclasshook__()`

**110.10.2 Properties**

Name	Description
<i>Inherited from object</i>	

*continued on next page*

Name	Description
<code>--class--</code>	

### 110.10.3 Class Variables

Name	Description
<code>GetError</code>	<b>Value:</b> (<type 'exceptions.EOFError'>, <class 'cPickle.Unpickling...>)
<code>ConnectError</code>	<b>Value:</b> (<type 'exceptions OSError'>)
<code>BUSYFAC1</code>	<b>Value:</b> 0.85
<code>BUSYFAC2</code>	<b>Value:</b> 0.15
<code>SendError</code>	<b>Value:</b> (<type 'exceptions IOError'>, <type 'exceptions.ValueErro...>)

### 110.10.4 Instance Variables

Name	Description
<code>uness</code>	How useful is this thread?

## 110.11 Class *workers\_c*

object └─ `gmisclib.spread_jobs.workers_c`

This creates a group of worker threads that take tasks from the iqueue and put the answers on the oqueue.

### 110.11.1 Methods

```
--init__(self, connection_factory, list_of_args, iqueue, oqueue, stdin, solock,
verbose=False, tail_callback=None, past_performance=None)
```

`x.__init__(...)` initializes `x`; see `help(type(x))` for signature

Overrides: `object.__init__` `exitit` (inherited documentation)

```
join(self)
```

```
pass_performance(self, x)
```

<code>--len--(self)</code>
----------------------------

<code>num_active(self)</code>
-------------------------------

**Return Value**

total usefulness of all workers and the number of live workers

*(type=(float, int))***Inherited from *object***

`--delattr--()`, `--format--()`, `--getattr--()`, `--hash--()`, `--new--()`, `--reduce--()`, `--reduce_ex--()`,  
`--repr--()`, `--setattr--()`, `--sizeof--()`, `--str--()`, `--subclasshook--()`

**110.11.2 Properties**

Name	Description
<i>Inherited from object</i>	
<code>--class--</code>	

**110.12 Class `unpickled_pseudofile`**

StringIO.StringIO └─ **`gmisclib.spread_jobs.unpickled_pseudofile`**

For testing.

**110.12.1 Methods**

<code>--init--(self)</code>
-----------------------------

Overrides: StringIO.StringIO.`--init--`

<code>close(self)</code>
--------------------------

Free the memory buffer.

Overrides: StringIO.StringIO.`close` `exitit`(inherited documentation)**Inherited from *StringIO.StringIO***

`--iter--()`, `flush()`, `getvalue()`, `isatty()`, `next()`, `read()`, `readline()`, `readlines()`, `seek()`,  
`tell()`, `truncate()`, `write()`, `writelines()`

## 111 Module *gmisclib.sqlbase*

This is a module for using a SQLite database as a collection of python objects.

For each SQL table, you derive a class from `DBx`. Each instance of that class corresponds to one row of the table.

### 111.1 Functions

**`multiselect(cll, db, where, *args)`**

This is used for SQL select operations on a join of several tables.

**Parameters**

**`cll`:** This a sequence of (`class`, `name`) pairs, where `class` matches a table in the database, and `name` is used to refer to that table in the SQL select statement.

*(type=a sequence of (class, str) where class is derived from DBx)*

**`where`:** This is the body of the select statement. Everything after `select * from tablename`.

*(type=str)*

**`args`:** All the arguments for the `select` statement.

**`db`:** *(type=sqlite3.Connection)*

**Note:** As an example, if you have a table `X` in the database and a class `x`, and `x.SQL_name=="X"`, and each row in `X` has an `id` number, and also the ID number of the `next` item, then you can find pairs of adjacent items with this call:

```
multiselect((x, "x1"), (x, "x2")), db, "x1.next = x2.id")
```

---

**get\_version**(*db\_connection*, *list\_of\_classes*)
 

---

This finds which of several variants of a table actually exists in the database. The intent is to let you upgrade from one version of table format to another.

**Parameters**

**db\_connection:** A `sqlite3.Connection` to a database.

**list\_of\_classes:** Possible classes to try. Each will be checked for consistency with the database.

*(type=sequence of some class derived from `sqlbase.DBx`. These are the classes themselves, not class instances.)*

**Return Value**

The first class in the list that is consistent with the structure of the database.

*(type=One of the classes in `list_of_classes`.)*

---



---

**multiclass\_get\_by\_id**(*possible\_classes*, *db*, *idx*)
 

---

This reads in a row from the database when you don't know exactly what information is available. The idea is that there is a one of several possible types of objects stored in the database, and you will be happy to take whichever one is available. This makes most sense when they are different versions of the same class. You access the objects by their ID number.

**Parameters**

**possible\_classes:** possible classes to match a table in the database. Classes must be derived from `DBx`, must have a name that matches a table in the database, must have the right number of columns, and the raw data must be convertible to an object of that class.

*(type=a sequence of classes, where each class is derived from `DBx`)*

**idx:** the id of a row in the database.

*(type=int)*

**db:** *(type=a `sqlite3.Connection` to a database.)*

**Note:** Ideally, there should be a 1:1 relationship between the names of tables in the database and the `SQL_name` attributes of classes. So, when you have a new version of a class, you really should have a new name for the corresponding table in the database. However, if you wish to have several classes that have the same `SQL_name` attribute, a class can raise `ColumnMismatchError` if the class and the data row are incompatible.

---

test()
--------

## 111.2 Variables

Name	Description
DEBUG	Value: 0
__package__	Value: 'gmisclib'

## 111.3 Class DBMetaClass



This is used by all the SQL classes, and initializes the class. Specifically, it makes sure that each class has a `_cache` dictionary to cache instances of the class, and a `idx` counter to be the integer index into the SQL database.

### 111.3.1 Methods

<b>__new__</b> ( <i>meta, classname, bases, cdict</i> ) <b>Return Value</b> a new object with type S, a subtype of T Overrides: object.__new__ extit(inherited documentation)
--

#### *Inherited from type*

\_\_call\_\_(), \_\_delattr\_\_(), \_\_eq\_\_(), \_\_ge\_\_(), \_\_getattr\_\_(), \_\_gt\_\_(), \_\_hash\_\_(), \_\_init\_\_(),  
 \_\_instancecheck\_\_(), \_\_le\_\_(), \_\_lt\_\_(), \_\_ne\_\_(), \_\_repr\_\_(), \_\_setattr\_\_(), \_\_subclasscheck\_\_(),  
 \_\_subclasses\_\_(), mro()

#### *Inherited from object*

\_\_format\_\_(), \_\_reduce\_\_(), \_\_reduce\_ex\_\_(), \_\_sizeof\_\_(), \_\_str\_\_(), \_\_subclasshook\_\_()

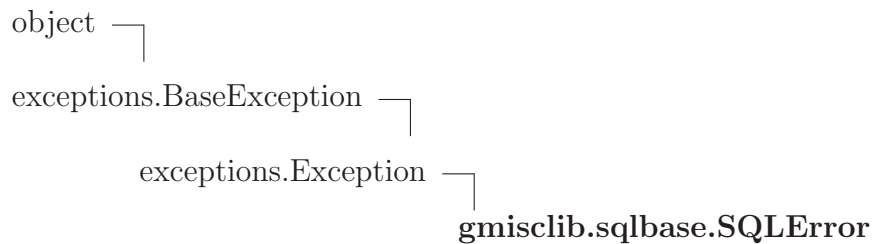
### 111.3.2 Properties

Name	Description
<i>Inherited from type</i>	

*continued on next page*

Name	Description
<code>__abstractmethods__</code> , <code>__base__</code> , <code>__bases__</code> , <code>__basicsize__</code> , <code>__dictoffset__</code> , <code>__flags__</code> , <code>__itemsize__</code> , <code>__mro__</code> , <code>__name__</code> , <code>__weakrefoffset__</code>	
<i>Inherited from object</i>	
<code>__class__</code>	

## 111.4 Class *SQLiteError*



**Known Subclasses:** *gmslib.sqlitebase.ColumnMismatchError*, *gmslib.sqlitebase.NoSuchTable*

Any errors raised by this module. This does not include errors raised by *sqlite* itself.

### 111.4.1 Methods

```

__init__(self, *s)

x.__init__(...) initializes x; see help(type(x)) for signature
Overrides: object.__init__ extit(inherited documentation)

```

*Inherited from exceptions.Exception*

```
__new__()
```

*Inherited from exceptions.BaseException*

```

__delattr__(), __getattr__(), __getitem__(), __getslice__(), __reduce__(), __repr__(),
__setattr__(), __setstate__(), __str__(), __unicode__()

```

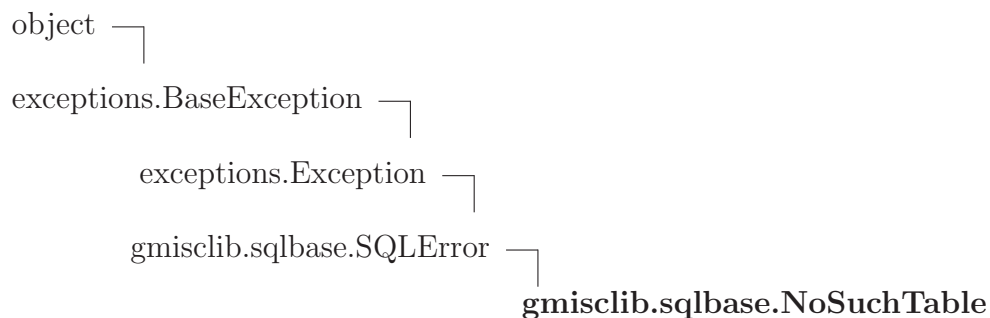
*Inherited from object*

```
__format__(), __hash__(), __reduce_ex__(), __sizeof__(), __subclasshook__()
```

### 111.4.2 Properties

Name	Description
<i>Inherited from exceptions.BaseException</i>	
args, message	
<i>Inherited from object</i>	
__class__	

## 111.5 Class NoSuchTable



### 111.5.1 Methods

```

__init__(self, *s)

x.__init__(...) initializes x; see help(type(x)) for signature
Overrides: object.__init__ extit(inherited documentation)

```

*Inherited from exceptions.Exception*

```
__new__()
```

*Inherited from exceptions.BaseException*

```

__delattr__(), __getattr__(), __getitem__(), __getslice__(), __reduce__(), __repr__(),
__setattr__(), __setstate__(), __str__(), __unicode__()

```

*Inherited from object*

```
__format__(), __hash__(), __reduce_ex__(), __sizeof__(), __subclasshook__()
```

### 111.5.2 Properties

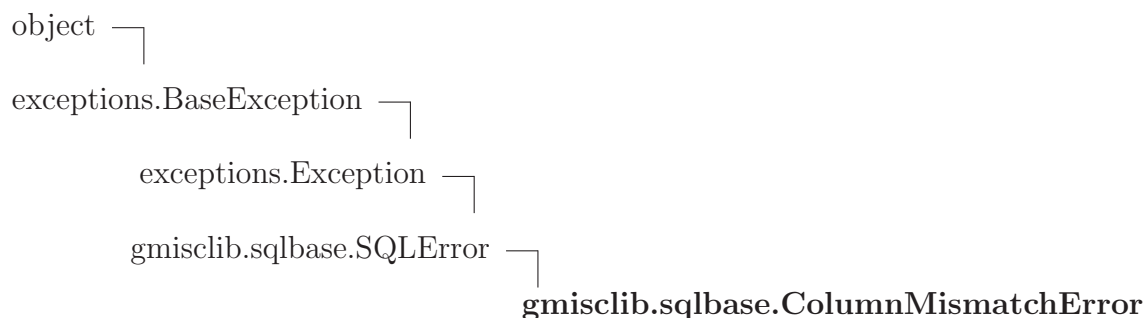
Name	Description
<i>Inherited from exceptions.BaseException</i>	

*continued on next page*



Name	Description
args, message	
<i>Inherited from object</i>	
__class__	

## 111.6 Class ColumnMismatchError



### 111.6.1 Methods

```

__init__(self, *s)

x.__init__(...) initializes x; see help(type(x)) for signature
Overrides: object.__init__ extit(inherited documentation)

```

#### *Inherited from exceptions.Exception*

```
__new__()
```

#### *Inherited from exceptions.BaseException*

```

__delattr__(), __getattr__(), __getitem__(), __getslice__(), __reduce__(), __repr__(),
__setattr__(), __setstate__(), __str__(), __unicode__()

```

#### *Inherited from object*

```
__format__(), __hash__(), __reduce_ex__(), __sizeof__(), __subclasshook__()
```

### 111.6.2 Properties

Name	Description
<i>Inherited from exceptions.BaseException</i>	
args, message	

*continued on next page*

Name	Description
<i>Inherited from object</i>	
<code>--class--</code>	

## 111.7 Class DB



**Known Subclasses:** gmisclib.sqlbase.DBx

Base class for all persistent objects in a sqlite database. Not the normal API. Derived classes need to define a `write()` method.

### 111.7.1 Methods

<b><code>--init--</code></b> ( <i>self</i> , <i>connection</i> )
<code>x.__init__(...)</code> initializes x; see <code>help(type(x))</code> for signature
<b>Parameters</b>
<b>connection:</b> ( <i>type=sqlite3.Connection</i> )
Overrides: <code>object.__init__</code>

<b><code>bump_id</code></b> ( <i>cls</i> , <i>delta</i> )
You can call this manually if you want to introduce a gap between ID numbers.

**write(*self*)**

This method writes an instance out to its database. This method should assemble its attributes to form an argument list for `_write`, and then call `_write` with that argument list. It should be used like this:

```
def write(self):  
    return self._write(self.something, self.other, self.stuff, self.misc)
```

**Return Value**

the return value of `_write`.

(*type=int*)

**Notes:**

- `write()` forces an immediate write or an update of the database, whether or not you have called `make_dirty()`. `make_dirty()` simply arranges for a write to occur when the object is eventually destroyed.
- The instance remembers the database from which it came.
- Do not pass `self.id` to `self._write()`. That's handled automatically.

**make\_dirty(*self*)**

Changes to an instance are not written to the database unless you call `make_dirty()` or `write()`. `write()` forces an immediate write, whereas `make_dirty()` simply marks the instance so that it will eventually be written out when the destructor is called.

**Note:** Global variables are destroyed very late, possibly after other necessary objects have already been destroyed, so you may not want to trust that `make_Dirty()` will behave well on global variables.

**is\_dirty(*self*)****\_\_del\_\_(*self*)**

**confirm**(*cls*, *connection*)

This can be called to confirm that the column names in the database matches what the class expects.

**Parameters**

**connection:** A connection to a database.  
(*type=sqlite3.Connection*)

**Return Value**

None  
(*type=None*)

**Raises**

**ColumnMismatchError** When something is wrong.  
**NoSuchTable** when the table doesn't exist.

**create**(*cls*, *connection*, *if\_not\_exists=True*)

This creates the table in the database. It is harmless if the table already exists.

**Parameters**

**connection:** A connection to a database.  
(*type=sqlite3.Connection*)

**Return Value**

None  
(*type=None*)

**set\_ID**(*cls*, *connection*)

This should be called when you first open an existing database when you have the intent to write. It sets the initial ID number to be larger than the largest ID in the database.

**index\_unique**(*cls*, *connection*, \**columns*)

**index\_nonunique**(*cls*, *connection*, \**columns*)

**flush\_cache**(*cls*)

**Inherited from object**

`--delattr--()`, `--format--()`, `--getattr--()`, `--hash--()`, `--new--()`, `--reduce--()`, `--reduce_ex--()`,  
`--repr--()`, `--setattr--()`, `--sizeof--()`, `--str--()`, `--subclasshook--()`

### 111.7.2 Properties

Name	Description
<i>Inherited from object</i>	
<code>--class--</code>	

### 111.7.3 Class Variables

Name	Description
<code>S_notinDB</code>	<b>Value:</b> 0
<code>S_inDB</code>	<b>Value:</b> 1
<code>S_modified</code>	<b>Value:</b> 2
<code>SQL_name</code>	<b>Value:</b> 'DB'
<code>idx</code>	<b>Value:</b> 0

## 111.8 Class DBx



**Known Subclasses:** gmisclib.sqlbase.\_InconsistentTable, gmisclib.sqlbase.\_SampleTable

This class is the main interface. Normally, you derive a class from this to represent a table in the database. Each of your derived classes **MUST** contain an attribute `COL_type` which specifies what columns are in that table. Each derived class will have a unique ID number called `id`; this will be a column in the database, and it will be stored in each instance object. The `DBx` class manages that ID number for you.

Beyond that, a derived class **MUST** redefine `_fromtuple`.

## 111.8.1 Methods

**`__init__(self, connection)`**

`x.__init__(...)` initializes `x`; see `help(type(x))` for signature

**Parameters**

**connection:** a database connection.

(*type=sqlite3.Connection*)

Overrides: `object.__init__`

**`get_id(self)`**

**`select(cls, db, where, *args)`**

**`get_by_id(cls, db, idx)`**

**`__repr__(self)`**

`repr(x)`

Overrides: `object.__repr__` `exitit`(inherited documentation)

**`__del__(self)`**

**`bump_id(cls, delta)`**

You can call this manually if you want to introduce a gap between ID numbers.

**`confirm(cls, connection)`**

This can be called to confirm that the column names in the database matches what the class expects.

**Parameters**

**connection:** A connection to a database.

(*type=sqlite3.Connection*)

**Return Value**

None

(*type=None*)

**Raises**

**ColumnMismatchError** When something is wrong.

**NoSuchTable** when the table doesn't exist.

---

**create**(*cls*, *connection*, *if\_not\_exists*=True)

This creates the table in the database. It is harmless if the table already exists.

**Parameters**

**connection:** A connection to a database.

(*type*=*sqlite3.Connection*)

**Return Value**

None

(*type*=*None*)

---

**flush\_cache**(*cls*)

---

**index\_nonunique**(*cls*, *connection*, \**columns*)

---

**index\_unique**(*cls*, *connection*, \**columns*)

---

**is\_dirty**(*self*)

---

**make\_dirty**(*self*)

Changes to an instance are not written to the database unless you call **make\_dirty()** or **write()**. **write()** forces an immediate write, whereas **make\_dirty()** simply marks the instance so that it will eventually be written out when the destructor is called.

**Note:** Global variables are destroyed very late, possibly after other necessary objects have already been destroyed, so you may not want to trust that **make\_Dirty()** will behave well on global variables.

---

**set\_ID**(*cls*, *connection*)

This should be called when you first open an existing database when you have the intent to write. It sets the initial ID number to be larger than the largest ID in the database.

**write(*self*)**

This method writes an instance out to its database. This method should assemble its attributes to form an argument list for `_write`, and then call `_write` with that argument list. It should be used like this:

```
def write(self):
    return self._write(self.something, self.other, self.stuff, self.misc)
```

**Return Value**

the return value of `_write`.

(*type=int*)

**Notes:**

- `write()` forces an immediate write or an update of the database, whether or not you have called `make_dirty()`. `make_dirty()` simply arranges for a write to occur when the object is eventually destroyed.
- The instance remembers the database from which it came.
- Do not pass `self.id` to `self._write()`. That's handled automatically.

***Inherited from object***

`__delattr__()`, `__format__()`, `__getattr__()`, `__hash__()`, `__new__()`, `__reduce__()`, `__reduce_ex__()`, `__setattr__()`, `__sizeof__()`, `__str__()`, `__subclasshook__()`

**111.8.2 Properties**

Name	Description
<i>Inherited from object</i>	
<code>__class__</code>	

**111.8.3 Class Variables**



Name	Description
COL_type	a sequence of ( <code>column_name</code> , <code>data_type_name</code> ) pairs. Those name the columns of the table and tell what type of data is in each. <code>data_type_name</code> must be the name of a legal sqlite3 data type OR, it must be "FKEY <code>tablename</code> <code>column_name</code> " where <i>tablename</i> is the name of the table into which the foreign key points and <i>column_name</i> is the name of the column in that table. Typical tuples in COL_type are ("Number_of_dogs", "INTEGER") or ("dog_id", "FKEY dog.info id"). Note that COL_type should not include the ID number, which is always column id; it is added automatically.
SQL_name	the name of the corresponding table in the database. This defaults to the name of the leaf class. <b>Value:</b> 'DBx'
S_inDB	<b>Value:</b> 1
S_modified	<b>Value:</b> 2
S_notinDB	<b>Value:</b> 0
idx	the lowest unused ID number.

#### 111.8.4 Instance Variables

Name	Description
id	the instance's ID number. Note that an ID number is allocated for every instance you create. Instances you look up with <code>select</code> won't necessarily be new: they may just be a pointer to an existing, perhaps cached instance. The ID number of an instance is persistent and is written into the database.

## 112 Module *gmisclib.stats*

Some of these functions, specifically `f_value()`, `fprob()`, `betai()`, and `betacf()`, are taken from `stats.py` "A collection of basic statistical functions for python." by Gary Strangman. They are copyright 1999-2007 Gary Strangman; All Rights Reserved and released under the MIT license.

The rest is copyright Greg Kochanski 2009.

### 112.1 Functions

**f\_value**(*ER*, *EF*, *dfR*, *dfF*)

Returns an F-statistic given the following:

**Parameters**

- ER**: error associated with the null hypothesis (the Restricted model)
- EF**: error associated with the alternate hypothesis (the Full model)
- dfR**: degrees of freedom the Restricted model (null hypothesis)
- dfF**: degrees of freedom associated with the Full model

**Return Value**

f-statistic (not the probability)  
(*type=float*)

**fprob**(*dfnum*, *dfden*, *F*)

Returns the (1-tailed) significance level (p-value) of an F statistic given the degrees of freedom for the numerator (*dfR-dfF*) and the degrees of freedom for the denominator (*dfF*).

Usage: `lfprob(dfnum, dfden, F)` where usually *dfnum=dfbn*, *dfden=dfwn*

**betai**(*a*, *b*, *x*)

Returns the incomplete beta function:

$$I\text{-sub-x}(a,b) = 1/B(a,b) * (\text{Integral}(0,x) \text{ of } t^{a-1}(1-t)^{b-1} dt)$$

where  $a \geq 0, b > 0$  and  $B(a,b) = G(a)G(b)/(G(a+b))$  where  $G(a)$  is the gamma function of  $a$ . The continued fraction formulation is implemented here, using the `betacf` function. (Adapted from: Numerical Recipes in C.)

Usage: `betai(a,b,x)`

**betacf**(*a*, *b*, *x*)

This function evaluates the continued fraction form of the incomplete Beta function, betai. (Adapted from: Numerical Recipes in C.)

Usage: betacf(*a*,*b*,*x*)

**gamma**ln(*x*)

Returns the gamma function of *xx*.  $\Gamma(z) = \int_0^{\infty} t^{z-1} \exp(-t) dt$ . (Adapted from: Numerical Recipes in C. via code Copyright (c) 1999-2000 Gary Strangman and released under the LGPL.)

**test\_fprob**(*dfnum*, *dfden*)**t\_value**(*ndof*, *p2sided*=0.99)

**ltqnorm(*p*)**

Lower tail quantile for standard normal distribution function.

This function returns an approximation of the inverse cumulative standard normal distribution function. I.e., given *P*, it returns an approximation to the *X* satisfying  $P = \Pr\{Z \leq X\}$  where *Z* is a random variable from the standard normal distribution.

The algorithm uses a minimax approximation by rational functions and the result has a relative error whose absolute value is less than 1.15e-9.

**Parameters**

*p*: (*type=**float* (*0,1*))

**Return Value**

*float*

**Raises**

**ValueError** if the argument is out of range.

**Author:** Peter John Acklam

**Notes:**

- Time-stamp: 2000-07-19 18:26:14
- Downloaded from  
<http://home.online.no/~pjacklam/notes/invnorm/impl/field/ltqnorm.txt>  
 GPK 4/22/2011. Documentation at  
<http://home.online.no/~pjacklam/notes/invnorm/index.html>,  
 which is part of this package at  
[.../references/stats\\_invnorm\\_pjacklam\\_2011.html](http://home.online.no/~pjacklam/notes/invnorm/references/stats_invnorm_pjacklam_2011.html).
- Modified from the author's original perl code (original comments follow below) by dfield@yahoo-inc.com. May 3, 2004.

**Contacts:** pjacklam@online.no, Greg Kochanski <gpk@kochanski.org>

**See Also:** <http://home.online.no/~pjacklam>

**112.2 Variables**

Name	Description
t_Table	<b>Value:</b> [(1, 0.5, 1.0), (1, 0.95, 12.71), (1, 0.98, 31.82), (1, 0...
__package__	<b>Value:</b> 'gmisclib'

## 113 Module `gmisclib.system_load`

### 113.1 Functions

<code>load_avg()</code>
-------------------------

<code>mem_pressure(verbose=False)</code>
--

**Return Value**

Big ( $>1$ ) when the system is really hurting for memory. Small ( $<0.5$ ) when things are going well. Also, by way of the amount of dirty memory, it indirectly looks at the rate of writing to the disk.  
(*type=float*)

<code>ncpu()</code>
---------------------

<code>get_ncpu()</code>
-------------------------

<code>get_loadavg()</code>
----------------------------

<code>load_now(my_weight=1, other_weight=2, d_weight=0.5, ignore=None, verbose=False)</code>
--

<code>niceness()</code>
-------------------------

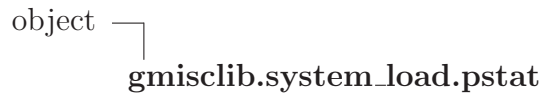
**Return Value**

positive numbers if the process is niced; negative numbers if not.  
(*type=float in the range [-0.5, 0.5]*)

### 113.2 Variables

Name	Description
<code>PROC</code>	<b>Value:</b> <code>'/proc'</code>
<code>__package__</code>	<b>Value:</b> <code>'gmisclib'</code>

### 113.3 Class *pstat*



#### 113.3.1 Methods

**`--init--`**(*self*, *pid*)

*x*.**`--init--`**(...) initializes *x*; see `help(type(x))` for signature

Overrides: `object.--init--` `exitit`(inherited documentation)

**`--getattr--`**(*self*, *name*)

#### *Inherited from object*

`--delattr--`(), `--format--`(), `--getattribute--`(), `--hash--`(), `--new--`(), `--reduce--`(), `--reduce_ex--`(),  
`--repr--`(), `--setattr--`(), `--sizeof--`(), `--str--`(), `--subclasshook--`()

#### 113.3.2 Properties

Name	Description
<i>Inherited from object</i>	
<code>--class--</code>	

## 114 Module `gmisclib.threaded_io`

This is designed to let you do asynchronous I/O conveniently.

### 114.1 Functions

**`to_be_joined(tlist)`**

Pick a ripe thread for joining.

**Parameters**

`tlist`: a list of threads.

(*type*=`list(threading.Thread)`)

**Return Value**

a thread that is either finished and ready to be joined, or (if none are ready yet) the oldest thread.

(*type*=`threading.Thread`)

**`check_n_raise(x)`**

**`pairmap(fn, inlist, *args, **kwargs)`**

**`testmap()`**

### 114.2 Variables

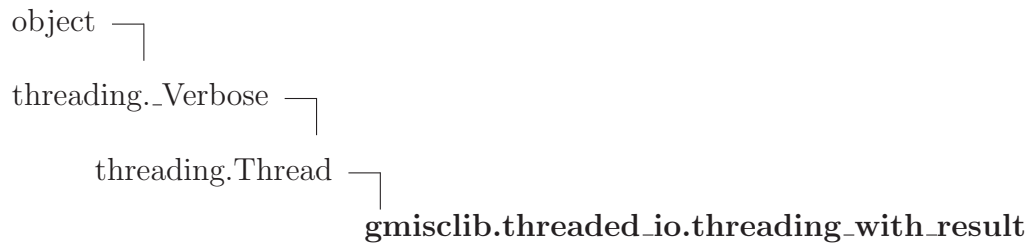
Name	Description
<code>__package__</code>	<b>Value:</b> <code>'gmisclib'</code>

### 114.3 Class `CaughtException`

object └─ `gmisclib.threaded_io.CaughtException`

**114.3.1 Methods****`__init__`**(*self*, *x*)*x*.`__init__`(...) initializes *x*; see `help(type(x))` for signatureOverrides: `object.__init__` `exitit`(inherited documentation)**`__repr__`**(*self*)`repr`(*x*)Overrides: `object.__repr__` `exitit`(inherited documentation)**`reraise`**(*self*)**`get`**(*self*)***Inherited from object***`__delattr__`(), `__format__`(), `__getattr__`(), `__hash__`(), `__new__`(), `__reduce__`(), `__reduce_ex__`(), `__setattr__`(), `__sizeof__`(), `__str__`(), `__subclasshook__`()**114.3.2 Properties**

Name	Description
<i>Inherited from object</i>	
<code>__class__</code>	

**114.4 Class `threading_with_result`**



**114.4.1 Methods**

```
__init__(self, tag=None, group=None, target=None, name=None, args=(),
kwargs={})
```

`x.__init__(...)` initializes `x`; see `help(type(x))` for signature

Overrides: `object.__init__` `exitit`(inherited documentation)

```
run(self)
```

Overrides: `threading.Thread.run`

```
get(self)
```

***Inherited from `threading.Thread`***

`__repr__()`, `getName()`, `isAlive()`, `isDaemon()`, `is_alive()`, `join()`, `setDaemon()`, `setName()`, `start()`

***Inherited from `object`***

`__delattr__()`, `__format__()`, `__getattr__()`, `__hash__()`, `__new__()`, `__reduce__()`, `__reduce_ex__()`, `__setattr__()`, `__sizeof__()`, `__str__()`, `__subclasshook__()`

**114.4.2 Properties**

Name	Description
<i>Inherited from <code>threading.Thread</code></i>	
<code>daemon</code> , <code>ident</code> , <code>name</code>	
<i>Inherited from <code>object</code></i>	
<code>__class__</code>	

**114.5 Class `Thread_pool`**

```
object └─ gmisclib.threaded_io.Thread_pool
```

**Known Subclasses:** `gmisclib.threaded_io.Thread_poolW`, `gmisclib.threaded_io.Thread_poolR`

## 114.5.1 Methods

```
__init__(self, n=None)
```

x.\_\_init\_\_(...) initializes x; see help(type(x)) for signature

Overrides: object.\_\_init\_\_ extit(inherited documentation)

```
__len__(self)
```

```
startRV(self, target=None, args=None, kwargs=None)
```

```
startNR(self, target=None, args=None, kwargs=None)
```

```
startQ(self, queue, tag, target=None, args=None, kwargs=None)
```

```
join(self)
```

```
__del__(self)
```

*Inherited from object*

```
__delattr__(), __format__(), __getattr__(), __hash__(), __new__(), __reduce__(), __reduce_ex__(),
__repr__(), __setattr__(), __sizeof__(), __str__(), __subclasshook__()
```

## 114.5.2 Properties

Name	Description
<i>Inherited from object</i>	
<code>__class__</code>	

## 114.6 Class Thread\_poolW

object └

gmisclib.threaded\_io.Thread\_pool └

**gmisclib.threaded\_io.Thread\_poolW**

Here, you have a pool of n threads. You can call `x.start(function, args, kwargs)` to start something running (perhaps a write to a file), and then go off and compute something else while the I/O completes. If you need to wait for completion, call `x.join()`.

This class is not designed to return values from the function.

#### 114.6.1 Methods

```
__init__(self, n=None)
```

*x.\_\_init\_\_*(...) initializes *x*; see `help(type(x))` for signature

Overrides: `object.__init__` `exitit`(inherited documentation)

```
start(self, fn, args=None, kwargs=None)
```

```
__del__(self)
```

```
__len__(self)
```

```
join(self)
```

```
startNR(self, target=None, args=None, kwargs=None)
```

```
startQ(self, queue, tag, target=None, args=None, kwargs=None)
```

```
startRV(self, target=None, args=None, kwargs=None)
```

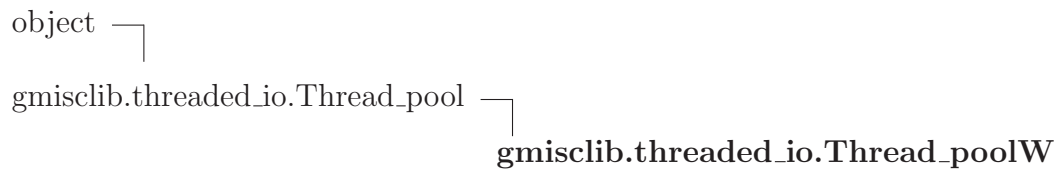
#### *Inherited from object*

`__delattr__()`, `__format__()`, `__getattr__()`, `__hash__()`, `__new__()`, `__reduce__()`, `__reduce_ex__()`, `__repr__()`, `__setattr__()`, `__sizeof__()`, `__str__()`, `__subclasshook__()`

#### 114.6.2 Properties

Name	Description
<i>Inherited from object</i>	
<code>__class__</code>	

## 114.7 Class `Thread_poolW`



Here, you have a pool of `n` threads. You can call `x.start(function, args, kwargs)` to start something running (perhaps a write to a file), and then go off and compute something else while the I/O completes. If you need to wait for completion, call `x.join()`.

This class is not designed to return values from the function.

### 114.7.1 Methods

```
__init__(self, n=None)
```

`x.__init__(...)` initializes `x`; see `help(type(x))` for signature

Overrides: `object.__init__` `exitit`(inherited documentation)

```
start(self, fn, args=None, kwargs=None)
```

```
__del__(self)
```

```
__len__(self)
```

```
join(self)
```

```
startNR(self, target=None, args=None, kwargs=None)
```

```
startQ(self, queue, tag, target=None, args=None, kwargs=None)
```

```
startRV(self, target=None, args=None, kwargs=None)
```

#### *Inherited from `object`*

```
__delattr__(), __format__(), __getattr__(), __hash__(), __new__(), __reduce__(), __reduce_ex__(),  

__repr__(), __setattr__(), __sizeof__(), __str__(), __subclasshook__()
```

### 114.7.2 Properties

Name	Description
<i>Inherited from object</i>	
<code>--class--</code>	

## 114.8 Class `Thread_poolR`

object └

`gmisclib.threaded_io.Thread_pool` └

**`gmisclib.threaded_io.Thread_poolR`**

Here, you have a pool of `n` threads. You can call `x.start(function, args, kwargs)` to start something running (perhaps a write to a file), and then go off and compute something else while the I/O completes. If you need to wait for completion, call `x.join()`.

This class is designed to return values from the function via `get()` or `getany()`.

### 114.8.1 Methods

**`--init--`**(*self*, *n=None*)

`x.__init__(...)` initializes `x`; see `help(type(x))` for signature

Overrides: `object.__init__` `extit`(inherited documentation)

**`start`**(*self*, *tag*, *fn*, *args=None*, *kwargs=None*)

**`getany`**(*self*)

#### Return Value

the first available answer, as a (tag, value) pair. The tag was set in the call to **`start`** and the value is the result of the computation. If the computation raised an exception, **`getany`** will return a `CaughtException` object instead.

(*type*=a tuple(*whatever*, *whatever*) pair from a computation or a *CaughtException*)

**`get`**(*self*, *tag*)

#### Return Value

whatever or a `CaughtException`

<b>has_answer</b> ( <i>self</i> )
-----------------------------------

Is there an answer ready?
---------------------------

<b>__del__</b> ( <i>self</i> )
--------------------------------

<b>__len__</b> ( <i>self</i> )
--------------------------------

<b>join</b> ( <i>self</i> )
-----------------------------

<b>startNR</b> ( <i>self</i> , <i>target</i> =None, <i>args</i> =None, <i>kwargs</i> =None)
---

<b>startQ</b> ( <i>self</i> , <i>queue</i> , <i>tag</i> , <i>target</i> =None, <i>args</i> =None, <i>kwargs</i> =None)
--

<b>startRV</b> ( <i>self</i> , <i>target</i> =None, <i>args</i> =None, <i>kwargs</i> =None)
---

### *Inherited from object*

`__delattr__()`, `__format__()`, `__getattr__()`, `__hash__()`, `__new__()`, `__reduce__()`, `__reduce_ex__()`,  
`__repr__()`, `__setattr__()`, `__sizeof__()`, `__str__()`, `__subclasshook__()`

### 114.8.2 Properties

Name	Description
<i>Inherited from object</i>	
<code>__class__</code>	

## 115 Module *gmisclib.tsops*

Do mathematical operations on time series, where the two operands don't necessarily have the same sampling times. It finds a common sampling time and sampling interval, then interpolates as necessary to bring the data onto the common time axis.

### 115.1 Functions

```
time(datasets, start=None, end=None)
```

```
time2(a, b)
```

```
test_fig()
```

```
interp_fill(a, t, fill)
```

```
interpN_fill(a, t, fill)
```

```
interp(a, t)
```

Interpolate to a specified time axis. This does a linear interpolation.

#### Parameters

**a**: data to be interpolated (a time series)

(*type*=*gpkimgclass.gpk\_img*)

**t**: (*type*=*an array of times.*)

#### Return Value

data interpolated onto the specified time values.

(*type*=*numpy array.*)

```
test_interp1()
```

```
test_interp2()
```

```
test_interp3()
```

**interpN**(*a*, *t*)

Interpolate to a specified time axis via nearest-neighbor interpolation. *A* is a *gpkimgclass*, and *t* is an array of times. Returns a Numeric array, not a *gpkimgclass*.

**common**(*data\_sets*, *start*=None, *end*=None)

Computes a common time axis for several datasets. Linearly interpolate as needed. The data sets need not be the same width, and need not have the same sampling interval or a common starting time.

**Parameters**

**data\_sets:** this is a list of the data to be put on a common time axis.

*(type=list(gpkimgclass.gpk\_img))*

**start:** this allows you to restrict the output data to a smaller region.

*(type=float or None)*

**end:** this allows you to restrict the output data to a smaller region.

*(type=float or None)*

**Return Value**

the *time\_axis* (as a 1-D numpy array of time values), followed by a 2-D numpy array for each of the input data sets.

*(type=list(numpy.ndarray) where the first ndarray is 1-D and the rest are two dimensional.)*

**commonN**(*data\_sets*, *start*=None, *end*=None)

Put several data sets on a common time axis. Interpolate by choosing nearest neighbor.

**mul**(*a*, *b*, *hdr\_op*=None)**copy\_interval**(*a*, *t0*, *t1*, *hdr\_op*=None, *mode*='rr')

This copies the part of the time-series in *a* where  $t_0 < t < t_1$ . '*a*' is a *gpk\_img* object.

**apply**(*fcn*, *a*, *hdrfcn*=<function <lambda> at 0x629f938>)

Apply a function, point-by-point to the data in *a*.



```
resample(a, dt)
```

```
test()
```

## 115.2 Variables

Name	Description
<code>__package__</code>	<b>Value:</b> <code>'gmisclib'</code>

## 115.3 Class *axis*

This class represents a time axis for a time series. Indices of the underlying array are assumed to be zero-based.

### 115.3.1 Methods

```
__init__(self, start=None, dt=None, n=None, end=None, crpix=0)
```

```
N(self)
```

```
coord(self, index)
```

What is the  $i^{\text{th}}$  coordinate if the  $i^{\text{th}}$  `index==index`? More plainly, this function gives you the coordinate that corresponds to the index.

```
coords(self)
```

Generate an array of all the time values.

```
index(self, t, limit=False, error=True)
```

```
indices(self, t, limit=False, error=True)
```

```
dt(self)
```

```
start(self)
```

```
end(self)
```

```
--str--(self)
```

```
--repr--(self)
```

## 116 Module `gmislib.wavesurfer_lab`

When used as a script, this reads label files produced by `wavesurfer` and prints the result.

Also contains functions that can be used for reading and writing the label format preferred by `Wavesurfer`.

### 116.1 Functions

**read**(*filename*, *loose*=0)

Read in label (transcription) files produced by `wavesurfer`.

Note that leading or trailing spaces in the label are removed.

**Parameters**

**filename:** name of label file or '-' to mean `sys.stdin`

(*type*=*str*)

**loose:** how many minor deviations from the ideal format are allowed

(*type*=*int*)

**Return Value**

(header, data). Data = [(starttime, endtime, label), ...].

**write**(*fd*, *header*, *data*)

Write label information to a file. Note: Expects data in [(t0, t1, label), ...] form.

**Parameters**

**fd:** where to write

(*type*=*file or file-like object*)

**header:** header information.

(*type*=*dict*)

**data:** a listing of the segments to write to the file.

(*type*=[(*segment\_start\_time*, *segment\_end\_time*, *segment\_label*), ...])

### 116.2 Variables

Name	Description
--package--	<b>Value:</b> 'gmisclib'

## 117 Module `gmisclib.wavio`

When run as a script: `python ~/lib/wavio.py [-g gain] -wavin|-wavout infile outfile` Reads or writes .wav files from any format supported by `gpkimgclass.py` .

As a library, allows reading of class `gpk_img` data from a .WAV file, and writing class `gpk_img` to a .WAV file.

### 117.1 Functions

#### **`read_hdr(fn)`**

Read header information for a .WAV file. The header contains information formatted as per `gpkimgclass.gpk_img`, i.e. FITS standard information.

##### **Parameters**

**fn:** filename  
(*type=*`str`)

##### **Return Value**

`dict(str: str or float or int)`

#### **`read(fn, tstart=None, t_end=None, t_error=0)`**

Reads in a .WAV file and returns a `gpkimgclass.gpk_img` instance that contains the header and data information. If `tstart` and/or `t_end` are specified, it seeks and only reads in the necessary data between `tstart` and `t_end`.

##### **Parameters**

**fn:** filename  
(*type=*`str`)

**tstart:** time at which to start reading audio data  
(*type=*`float`)

**t\_end:** time at which to stop reading audio data  
(*type=*`float`)

**t\_error:** What to do if the requested `t_start` or `t_end` are out of range? Either raise a `ValueError` (`T_ERROR`) or adjust the limits (`T_LIMIT`).  
(*type=*`int`, either `T_ERROR` or `T_LIMIT`.)

##### **Return Value**

audio data  
(*type=*`gpkimgclass` instance)

```
write(data, fname, scalefac=1, allow_overflow=0)
```

#### Parameters

- data:** is a class `gpk_img` object containing data to be written (note that the header information is ignored except for the sampling rate (`CDELTA2`) and bits per pixel (`BITPIX`)). The length and number of channels is taken from `data.d.shape`.
- fname:** is the name of a file to write it to (or a file object), (*type=*`str` or *file*,)
- scalefac:** is a factor to multiply the data
- allow\_overflow:** can be either
- `OV_ERROR` (default, means raise a `ValueError` exception if the `data*scalefac` overflows),
  - `OV_LIMIT` (means limit the `data*scalefac` to prevent overflows – this clips the audio), or
  - `OV_IGNORE` (means let the overflows happen and don't worry.)

#### Raises

**ValueError** Missing information in `data`.

**Overflow** The output format is clipping the data. Strictly speaking, on 16-bit data, 32767 is considered clipping even though it possibly might be OK. However such extreme values are very likely to be generated by clipping at an earlier stage of the processing, so it's probably an error even if the error is not being made here.

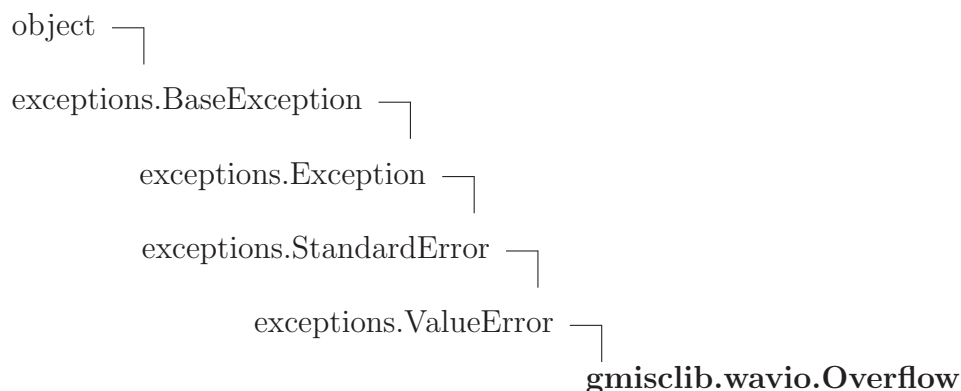
## 117.2 Variables

Name	Description
<code>OV_ERROR</code>	Cause <code>write</code> to raise an error on overflow. <b>Value:</b> 0
<code>OV_LIMIT</code>	Cause <code>write</code> to truncate overflows. <b>Value:</b> 1
<code>OV_IGNORE</code>	Cause <code>write</code> to silently ignore overflows. <b>Value:</b> 2
<code>T_ERROR</code>	Cause <code>read</code> to raise an error when timing is out of range. <b>Value:</b> 0

*continued on next page*

Name	Description
<code>T_LIMIT</code>	Cause <code>read</code> to quietly supply as much of the requested data as possible. <b>Value:</b> 1
<code>__package__</code>	<b>Value:</b> <code>'gmisclib'</code>

### 117.3 Class Overflow



#### 117.3.1 Methods

```

__init__(self, *s)

x.__init__(...) initializes x; see help(type(x)) for signature
Overrides: object.__init__ extit(inherited documentation)

```

*Inherited from `exceptions.ValueError`*

```
__new__()
```

*Inherited from `exceptions.BaseException`*

```

__delattr__(), __getattr__(), __getitem__(), __getslice__(), __reduce__(), __repr__(),
__setattr__(), __setstate__(), __str__(), __unicode__()

```

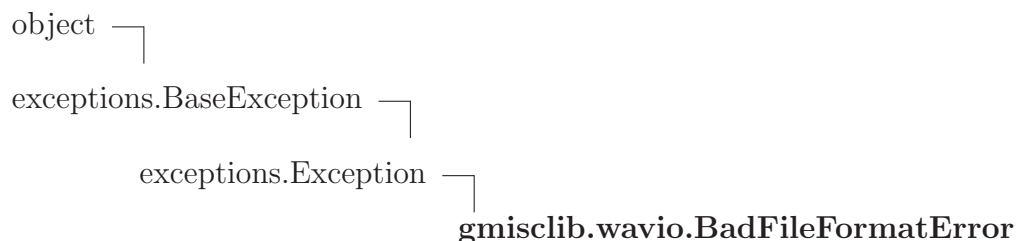
*Inherited from `object`*

```
__format__(), __hash__(), __reduce_ex__(), __sizeof__(), __subclasshook__()
```

#### 117.3.2 Properties

Name	Description
<i>Inherited from exceptions.BaseException</i>	
args, message	
<i>Inherited from object</i>	
__class__	

## 117.4 Class *BadFileFormatError*



### 117.4.1 Methods

**\_\_init\_\_(self, \*s)**  
 x.\_\_init\_\_(...) initializes x; see help(type(x)) for signature  
 Overrides: object.\_\_init\_\_ extit(inherited documentation)

#### *Inherited from exceptions.Exception*

\_\_new\_\_()

#### *Inherited from exceptions.BaseException*

\_\_delattr\_\_(), \_\_getattr\_\_(), \_\_getitem\_\_(), \_\_getslice\_\_(), \_\_reduce\_\_(), \_\_repr\_\_(),  
 \_\_setattr\_\_(), \_\_setstate\_\_(), \_\_str\_\_(), \_\_unicode\_\_()

#### *Inherited from object*

\_\_format\_\_(), \_\_hash\_\_(), \_\_reduce\_ex\_\_(), \_\_sizeof\_\_(), \_\_subclasshook\_\_()

### 117.4.2 Properties

Name	Description
<i>Inherited from exceptions.BaseException</i>	
args, message	
<i>Inherited from object</i>	
__class__	



## 118 Module `gmisclib.weighted_percentile`

This computes order statistics on data with weights.

### 118.1 Functions

**wp**(*data*, *wt*, *percentiles*)

Compute weighted percentiles. If the weights are equal, this is the same as normal percentiles. Elements of the **data** and **wt** arrays correspond to each other and must have equal length (unless **wt** is **None**).

#### Parameters

- data:** The data.  
(*type=A **numpy.ndarray** array or a **list** of numbers.*)
- wt:** How important is a given piece of data.  
(*type=**None** or a **numpy.ndarray** array or a **list** of numbers. All the weights must be non-negative and the sum must be greater than zero.*)
- percentiles:** what percentiles to use. (Not really percentiles, as the range is 0-1 rather than 0-100.)  
(*type=a **list** of numbers between 0 and 1.*)

#### Return Value

the weighted percentiles of the data.  
(*type=[ **float**, ... ]*)

**wtd\_median**(*data*, *wt*)

The weighted median is the point where half the weight is above and half the weight is below. If the weights are equal, this is the same as the median. Elements of the **data** and **wt** arrays correspond to each other and must have equal length (unless **wt** is `None`).

**Parameters**

**data**: The data.

*(type=A `numpy.ndarray` array or a `list` of numbers.)*

**wt**: How important is a given piece of data.

*(type=`None` or a `numpy.ndarray` array or a `list` of numbers.  
All the weights must be non-negative and the sum must be greater than zero.)*

**Return Value**

the weighted median of the data.

*(type=`float`)*

**wtd\_median\_across**(*list\_of\_vectors*, *wt*)

Takes a weighted component-by-component median of a sequence of vectors.

**Parameters**

**list\_of\_vectors**: the data to be combined

*(type=any sequence of lists or `numpy.ndarray`.  
All the inside lists must be of the same length.)*

**wt**: sequence of numbers or `None`

*(type=a vector of weights (one weight for each input vector) or `None`.)*

**Return Value**

the median vector.

*(type=`numpy.ndarray`)*

**test\_wp**()

**test\_median**()

**test**()

## 118.2 Variables

Name	Description
--package--	<b>Value:</b> 'gmisclib'

## 119 Module `gmisclib.xmlmisc`

This contains helper functions and classes for processing XML, based on the `ElementTree` module.

### 119.1 Functions

#### **`treestructures(elt)`**

This gives you a view of what the XML hierarchy looks like. It's intended to help you deduce the correct DTD, or just to understand how the data is arranged. It shows you the kinds of leaf tags you have and the kinds of paths through the XML tree necessary to get to each leaf.

#### **Parameters**

**elt**: an XML tree structure

(*type=an `xml.etree.cElementTree Element`*)

#### **Return Value**

a sequence of embedding lists. Each embedding list contains the tags you go through en route to a leaf node. So, `<a> <b> <c>x</c></b> </a>` would yield a list `['a', 'b', 'c']` and `<a> <b>x</b> <c> y </c> </a>` would yield `['a', 'b']` and `['a', 'c']`.

(*type=iterator(list(str))*)

#### **`test_loc_finder()`**

#### **`test()`**

### 119.2 Variables

Name	Description
<code>__package__</code>	<b>Value:</b> <code>'gmisclib'</code>

### 119.3 Class `loc_finder`

object — `gmisclib.xmlmisc.loc_finder`

A class for specifying a path through an XML tree to a particular node. Basically, this is a way of referring to a particular element that will survive serialization of the XML tree.

### 119.3.1 Methods

**`__init__(self, root_element)`**

Create an instance of the `loc_finder` class and initialize it from a portion of an XML tree.

**Parameters**

**`root_element`:** the top element of the tree (or subtree). All positions will be computed relative to this element.  
(*type=elTree.Element.*)

Overrides: `object.__init__`

**Notes:**

- This walks the entire XML file and builds a cache, so it is expensive to initialize, but fairly cheap to use.
- this works on `elTree.Element` object, not on an `elTree.ElementTree` object. If you have `x` which is a `elTree.ElementTree`, then use `loc_finder(x.get_root())`.

**`find_up(self, el, tag)`**

Search upwards (towards the document root) to find the first tag of the specified type.

**Parameters**

**`tag`:** the tag of an XML element.  
(*type=str*)

**Return Value**

The smallest enclosing element that has the specified tag.  
(*type=elTree.Element*)

**path**(*self*, *el*)

Describe the path through an XML file to a specific element.

**Parameters**

*el*: The target element.

(*type=elTree.Element*)

**Return Value**

A tuple, intended to be handed to `walkto`.

(*type=tuple(int)*)

**walkto**(*elem*, *path*)

If you have an element in an elementTree and a path obtained from `loc_finder(elem).path(X)`, this will walk you back to element X.

**Parameters**

*elem*: an element in an XML tree. Normally, this is the root element.

(*type=elTree.Element*)

*path*: a location in the specified XML (sub)tree.

(*type=tuple(int)*)

**Return Value**

The element specified by the *path*, in the (sub)tree.

(*type=elTree.Element*)

**Inherited from object**

`__delattr__()`, `__format__()`, `__getattr__()`, `__hash__()`, `__new__()`, `__reduce__()`, `__reduce_ex__()`, `__repr__()`, `__setattr__()`, `__sizeof__()`, `__str__()`, `__subclasshook__()`

**119.3.2 Properties**

Name	Description
<i>Inherited from object</i>	
<code>__class__</code>	

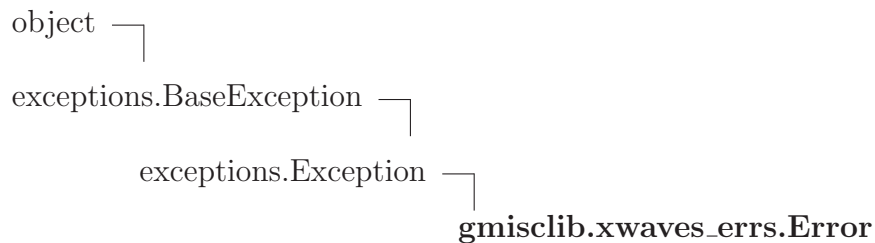
## 120 Module gmisclib.xwaves\_errs

Errors for reading label (typically speech transcription) files.

### 120.1 Variables

Name	Description
<code>--package--</code>	<b>Value:</b> None

### 120.2 Class Error



**Known Subclasses:** gmisclib.HTK\_MLF\_io.ReferencedFileNotFound, gmisclib.xwaves\_errs.BadFileFormat, gmisclib.xwaves\_errs.DataError, gmisclib.xwaves\_errs.NoSuchFileError

#### 120.2.1 Methods

```

__init__(self, *x)

x.__init__(...) initializes x; see help(type(x)) for signature
Overrides: object.__init__ extit(inherited documentation)
  
```

*Inherited from exceptions.Exception*

```
__new__()
```

*Inherited from exceptions.BaseException*

```

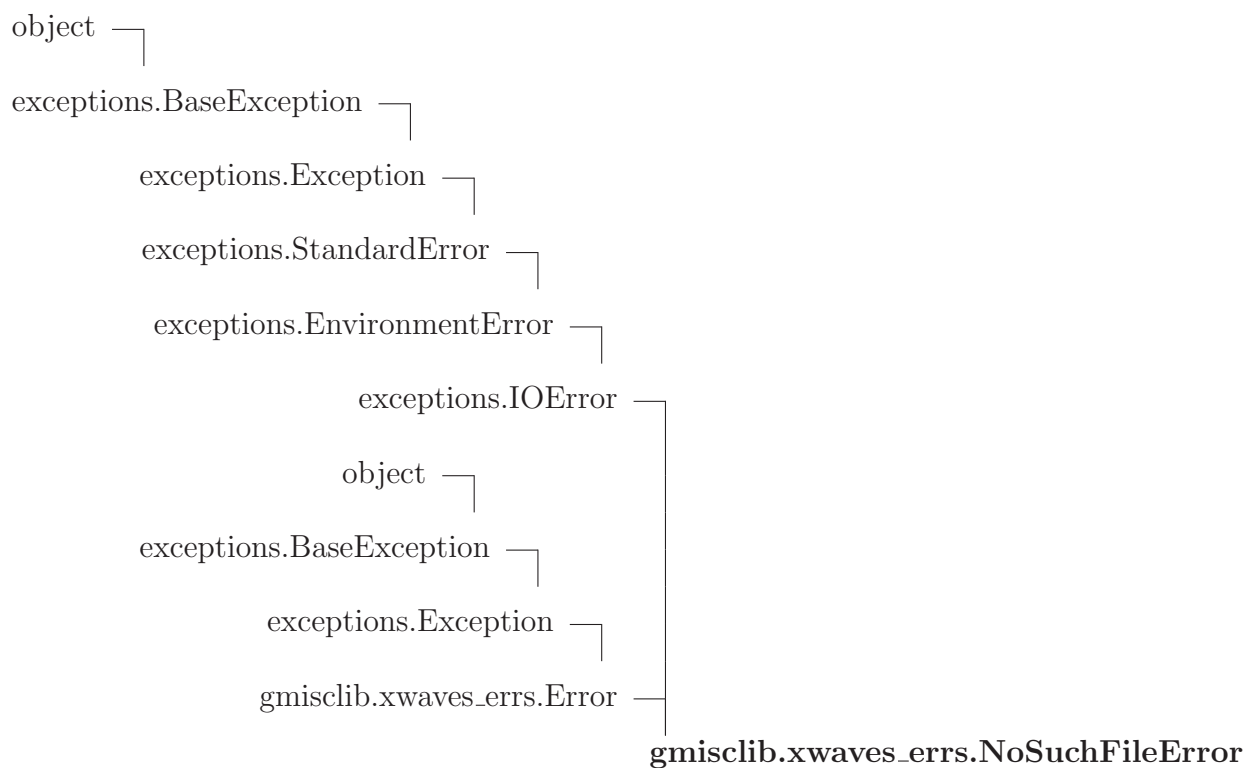
__delattr__(), __getattr__(), __getitem__(), __getslice__(), __reduce__(), __repr__(),
__setattr__(), __setstate__(), __str__(), __unicode__()
  
```

*Inherited from object*

```
__format__(), __hash__(), __reduce_ex__(), __sizeof__(), __subclasshook__()
```

**120.2.2 Properties**

Name	Description
<i>Inherited from <code>exceptions.BaseException</code></i>	
<code>args</code> , <code>message</code>	
<i>Inherited from <code>object</code></i>	
<code>__class__</code>	

**120.3 Class `NoSuchFileError`****120.3.1 Methods**

```

__init__(self, *x)

x.__init__(...) initializes x; see help(type(x)) for signature
Overrides: object.__init__ extit(inherited documentation)

```

*Inherited from `exceptions.IOError`*

```
__new__()
```



*Inherited from `exceptions.EnvironmentError`*

`__reduce__()`, `__str__()`

*Inherited from `exceptions.BaseException`*

`__delattr__()`, `__getattr__()`, `__getitem__()`, `__getslice__()`, `__repr__()`, `__setattr__()`,  
`__setstate__()`, `__unicode__()`

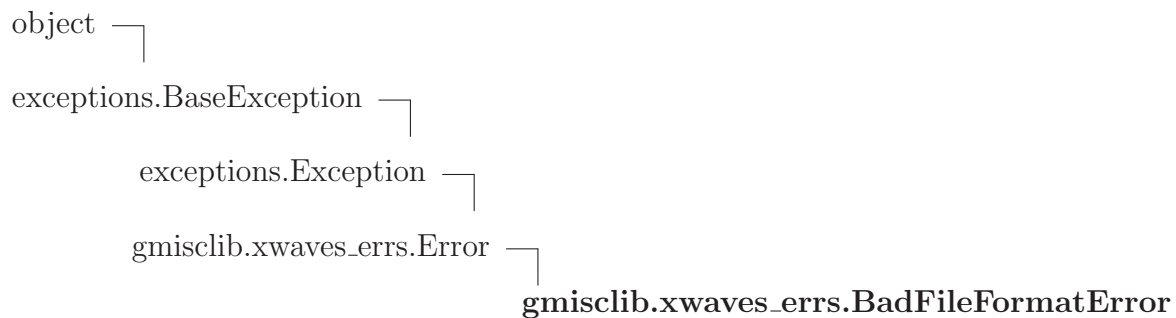
*Inherited from `object`*

`__format__()`, `__hash__()`, `__reduce_ex__()`, `__sizeof__()`, `__subclasshook__()`

### 120.3.2 Properties

Name	Description
<i>Inherited from <code>exceptions.EnvironmentError</code></i>	<code>errno</code> , <code>filename</code> , <code>strerror</code>
<i>Inherited from <code>exceptions.BaseException</code></i>	<code>args</code> , <code>message</code>
<i>Inherited from <code>object</code></i>	<code>__class__</code>

## 120.4 Class `BadFileFormatError`



### 120.4.1 Methods

`__init__(self, *x)`  
`x.__init__(...)` initializes `x`; see `help(type(x))` for signature  
 Overrides: `object.__init__` `exitit`(inherited documentation)

*Inherited from `exceptions.Exception`*

`__new__()`

**Inherited from `exceptions.BaseException`**

`__delattr__()`, `__getattr__()`, `__getitem__()`, `__getslice__()`, `__reduce__()`, `__repr__()`,  
`__setattr__()`, `__setstate__()`, `__str__()`, `__unicode__()`

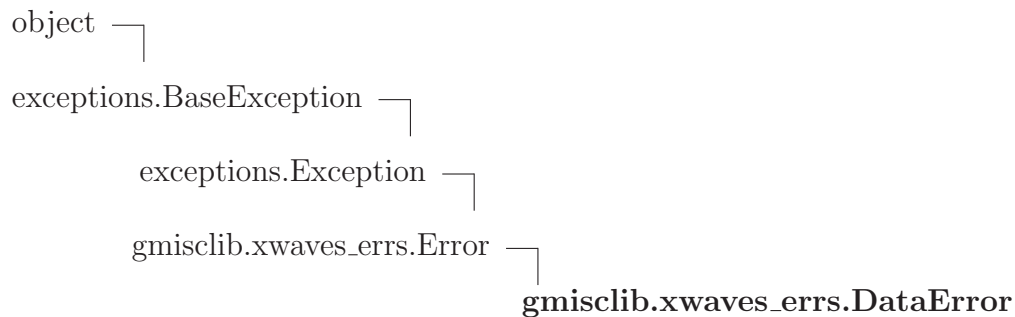
**Inherited from `object`**

`__format__()`, `__hash__()`, `__reduce_ex__()`, `__sizeof__()`, `__subclasshook__()`

#### 120.4.2 Properties

Name	Description
<i>Inherited from <code>exceptions.BaseException</code></i>	
<code>args</code> , <code>message</code>	
<i>Inherited from <code>object</code></i>	
<code>__class__</code>	

### 120.5 Class `DataError`



**Known Subclasses:** `gmislib.xwaves.errs.DataOutOfOrderError`

#### 120.5.1 Methods

**`__init__`**(*self*, \**x*)  
*x*.`__init__`(...) initializes *x*; see `help(type(x))` for signature  
 Overrides: `object.__init__` extit(inherited documentation)

**Inherited from `exceptions.Exception`**

`__new__()`

***Inherited from `exceptions.BaseException`***

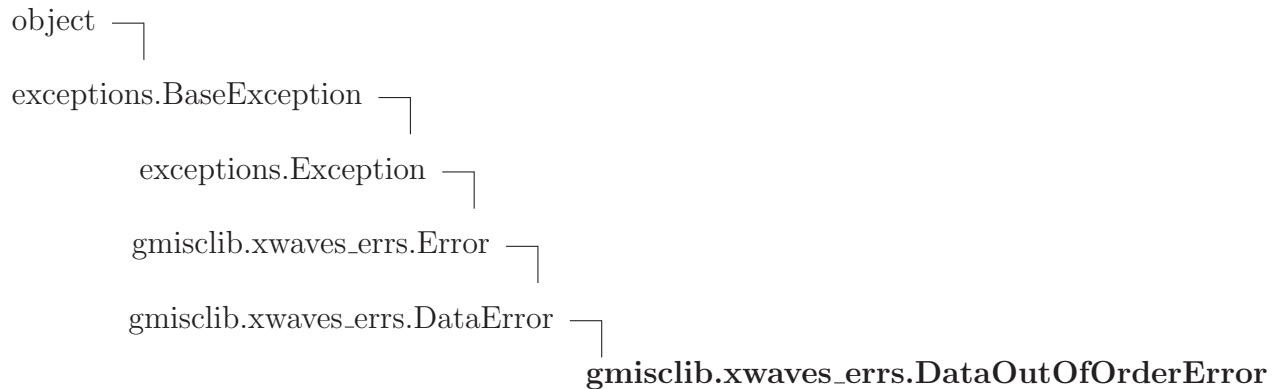
`__delattr__()`, `__getattr__()`, `__getitem__()`, `__getslice__()`, `__reduce__()`, `__repr__()`,  
`__setattr__()`, `__setstate__()`, `__str__()`, `__unicode__()`

***Inherited from `object`***

`__format__()`, `__hash__()`, `__reduce_ex__()`, `__sizeof__()`, `__subclasshook__()`

**120.5.2 Properties**

Name	Description
<i>Inherited from <code>exceptions.BaseException</code></i>	
<code>args</code> , <code>message</code>	
<i>Inherited from <code>object</code></i>	
<code>__class__</code>	

**120.6 Class `DataOutOfOrderError`****120.6.1 Methods**

`__init__(self, *s)`  
`x.__init__(...)` initializes `x`; see `help(type(x))` for signature  
 Overrides: `object.__init__` `exitit`(inherited documentation)

***Inherited from `exceptions.Exception`***

`__new__()`

***Inherited from `exceptions.BaseException`***

`__delattr__()`, `__getattr__()`, `__getitem__()`, `__getslice__()`, `__reduce__()`, `__repr__()`,  
`__setattr__()`, `__setstate__()`, `__str__()`, `__unicode__()`

***Inherited from object***

`__format__()`, `__hash__()`, `__reduce_ex__()`, `__sizeof__()`, `__subclasshook__()`

**120.6.2 Properties**

Name	Description
<i>Inherited from exceptions.BaseException</i>	
args, message	
<i>Inherited from object</i>	
<code>__class__</code>	

## 121 Module *gmisclib.xwaves\_lab*

Reads label files produced by ESPS xwaves. Writes labels files that ESPS xwaves can read.

### 121.1 Functions

<b>read</b> ( <i>filename</i> , <i>loose</i> =0)
--

read in .lab files produced by ESPS xlabel. returns (header, data). Data = [(time, label), ...]. Note that leading or trailing spaces in the label are removed.
---

<b>start_stop</b> ( <i>d</i> , <i>dropfirst</i> =False)
---

Converts data in (end_time, label) format to (t_start, t_stop, label)
---

<b>end_marks</b> ( <i>d</i> , <i>beglabel</i> , <i>delta</i> =0.001, <i>interlabel</i> =None)
---

Converts data in a (start, stop, label) format to (end_time, label). It introduces extra labels if neccessary. A beginning label is required, to mark the start of the first segment.
---

<b>write</b> ( <i>fd</i> , <i>header</i> , <i>data</i> )
--

Expects data in [(t, label), ...] form.
---

### 121.2 Variables

Name	Description
<code>__package__</code>	<b>Value:</b> 'gmisclib'

## 122 Module *gmisclib.xwaves\_mark*

Read in .in files produced by ESPS xmark. Returns (header, data), where data is (time, word\_or\_phoneme, type), where type is 0 for words, 1 for phonemes. Times are guaranteed to be increasing inside the set of all words, and also inside the set of all phonemes. Word marks precede the corresponding phoneme marks.

### 122.1 Functions

**write**(*fd, hdr, data, SortData=1*)

**read**(*filename*)

Read in .in files produced by ESPS xmark. Returns (header, data), where data is (time, word\_or\_phoneme, type), where type is 0 for words, 1 for phonemes. Times are guaranteed to be increasing inside the set of all words, and also inside the set of all phonemes. Word marks precede the corresponding phoneme marks.

**mark\_to\_lab**(*data, ty*)

This function lets you take a mixed list of word and phone labels, such as provided by *xwaves\_mark.read()*, and will select out one or the other type. Type is PHONE or WORD.

**combine\_2labs**(*whdr, wd, phdr, pd, utterance, TOL=0.001*)

Combine two XLAB files into one XMARK file. It forces (within rounding errors) the words to enclose the phones.

### 122.2 Variables

Name	Description
DecrTol	<b>Value:</b> 0.004999
PHONE	<b>Value:</b> 1
WORD	<b>Value:</b> 0
__doc__	<b>Value:</b> read.__doc__
__package__	<b>Value:</b> 'gmisclib'

## 123 Module `gpk_wavio`

When run as a script: `python ~/lib/wavio.py [-g gain] -wavin|-wavout infile outfile` Reads or writes .wav files from any format supported by `gpkimgclass.py` .

As a library, allows reading of class `gpk_img` data from a .WAV file, and writing class `gpk_img` to a .WAV file.

### 123.1 Variables

Name	Description
<code>--package--</code>	<b>Value:</b> None

## 124 Module *mcmc*

Adaptive Markov-Chain Monte-Carlo algorithm. This can be used to generate samples from a probability distribution, or also as a simulated annealing algorithm for maximization. This can be used as a script, from the command line, or (more likely) it can be imported and its functions and classes can be used.

The central interfaces are the `BootStepper` class, and within it, the `step()` method is used iteratively to take a Markov step.

To use it as a script, you create your own module that implements a few functions, and run `mcmc.py`. It will import your module and call your functions repeatedly.

The algorithm is described in Kochanski and Rosner 2010, and earlier versions have been used in ZZZ. It evolved originally from `amoeba_anneal` (in Numerical Recipes, Press et al.). The essential feature is that it keeps a large archive of previous positions (possibly many times more than  $N$  of them). It samples two positions from the archive and subtracts them to generate candidate steps. This has the nice property that when sampling from a multivariate Gaussian distribution, the candidate steps match the distribution nicely.

It can be operated in two modes (or set to automatically switch). One is optimization mode where it heads for the maximum of the probability distribution. The other mode is sampling mode, where it asymptotically follows a Markov sampling procedure and has the proper statistical properties.

As a script, it takes the following arguments:

- `-o fff` Specifies a log file for some basic logging.
- `{C -py ddd/mm}` Specifies the module to load and to optimize. If the name contains a slash, it searches `ddd` instead of the python search path, looking for a module named `mm`. If no slash, then it looks up the module in `PYTHONPATH`.
- `-NI NN` Sets the number of iterations. Required, unless the module defines `yourmod.NI`.
- `--` Stops interpretation of the argument list. The remainder is passed to `yourmod.init()`, if it is defined, then to `yourmod.start()`, if that is defined.

It uses the following functions and variables:

- `yourmod.c` (required). This is an object that is passed into `yourmod.resid()` or `yourmod.logp()`. You can make it be anything you want, as it is only touched by your functions. Often, it contains data or parameters for these functions, but it can be `None` if you have no need for it.
- `yourmod.start(arglist, c)` (Not required). This is where you initialize the Monte-Carlo iteration. If you define `yourmod.start()`, it is passed the `arglist`, after the standard `mcmc.py` flags and arguments are removed from it. `Yourmod.start()` must return a list of one or more starting vectors, (either a list of numpy arrays or a list of sequences of floats). These starting vectors are used to seed the iteration.



If it is not defined, `mcmc.py` will read a list of starting vectors in ascii format from the standard input using `_read()`.

- `yourmod.init(ndim, ni, c, arglist)` (Not required). This is primarily there to open a log file. `Init` is passed the remainder of the scripts argument list, after `yourmod.start()` looks at it. It can do anything it wishes with the argument list. It can return an object or `None`.

If it returns something other than `None`, it will be used this way:

```
x = yourmod.init(ndim, ni, yourmod.c, arglist) x.add(parameters, iteration) # add is
there to accumulate averages of parameters # or to log parameter vectors. x.finish(sys.stdout)
# finish can be used to print a summary or close a log file.
```

`Ndim` is the length of the parameter vector, `ni` is the number of iterations, `c` is `yourmod.c`.

- `yourmod.resid(x, c)` (Either `yourmod.logp()` or `yourmod.resid()` is required.)
- `yourmod.logp(x, c)` (Either `yourmod.logp` or `yourmod.resid` is required.) These functions are the thing which is to be optimized. `Yourmod.log` returns the logarithm (base e) of the probability density at position `x`. (It does not need to be normalized, so it really only needs something proportional to the probability density.) `X` is a numpy array (a 1-dimensional vector), and `c` is an arbitrary object that you define.

`Yourmod.logp` should return `None` if `x` is a silly value. Either that, or it can raise a `NotGoodPosition` exception. Note that the optimizer will tend to increase values of `logp`. It's a maximizer, not a minimizer!

If you give `yourmod.resid()` instead, `logp()` will be calculated as -0.5 times the sum of the squares of the residuals. It should return a numpy vector. It may return `None`, or raise `NotGoodPosition`, which will cause the optimizer to treat the position as extremely bad.

- `yourmod.log(result_of_init, i, p, w)` (Not required). This is basically a function to log your data. No return value. It is called at every step and passed the result of `yourmod.init()` (which might be a file descriptor to write into). It is also passed the iteration count, `i`, the current parameter vector, `p`, and `w`, which is reserved for possible future use.
- `yourmod.NI` (required unless the `-NI` argument is given on the command line) Specifies how many iterations. This is an integer.
- `yourmod.STARTLOW` (not required). Integer 0 or 1. If true, the iteration is started from the largest value of `logp` that is known.
- `yourmod.V` (Not required) This specifies the covariance matrix that is used to hop from one test point to another. It is a function that takes the result of `yourmod.start()` and returns a Numeric/numarray 2-dimensional square matrix,

If it is not specified, `mcmc` will use some crude approximation and will probably start more slowly, but should work OK. If it is not specified and `yourmod.start()` returns only one starting position, then it will just use an identity matrix as the covariance matrix, and it might still work, but don't expect too much.

## 124.1 Functions

`logp_from_resid(x, c, resid_fcn)`

`go(argv, theStepper)`

Run the optimization, controlled by command line flags.

## 124.2 Variables

Name	Description
Debug	<b>Value:</b> 0
MEMORYS_WORTH_OF_PARAMETERS	<b>Value:</b> 100000000.0
__package__	<b>Value:</b> None

## 124.3 Class `def_logger`

### 124.3.1 Methods

`__init__(self, ndim, ni, c, arglist)`

`add(self, p, i)`

`finish(self, stdout)`

## 125 Module q3html

This program takes files named \*.q in the current directory, and converts them to HTML. To do that, it looks for a python script called 'headfoot.py' or '../headfoot.py' or ... . This script needs to define one variable ( server ) and two functions (header and footer).

The .q files have the following format:

```
TITLE title string
OTHER_HEADER_KEYWORD header info

P
    Text is 3>5 ?
    A ref="foo.html" No!
        More text inside the link.
    IMG ref="foo.gif"
    UL
        LI List item
        LI Another List item
```

The header info is separated from the body with a blank line. HTML tags are always the first thing on a line. Line continuations and enclosures are indicated with indentation. Tags automatically close when the indentation gets smaller.

The PRE tag is the only exception. You have to close it with a line beginning '/PRE'. No indenting is needed inside a PRE /PRE pair.

The user needs to define one thing: the address of the web server. This address is prepended to all hyperlinks and image references that aren't absolute.

All the header information gets put into a dictionary, and is passed into the header() and footer() functions. That dictionary has 'filename' and '\_server\_root' entries added.

The user can also define three things:

DEFAULT\_HEADER, a dictionary passed to the header() and footer functions(). Useful things to put in DEFAULT\_HEADER are:

- 'lang': 'en' # appears in the <html lang="en"> tag
- 'stylesheet': url becomes a link to a stylesheet: <link rel="stylesheet" type="text/css" href="url">
- Anything in the form M.x:y becomes a <meta name="x" content="y">
- M.description and M.keywords are useful.
- Anything in the form HTTP.x:y becomes a <meta http-equiv="x" content="y">

Header(thd, hinfo) takes two arguments, a file descriptor to which it should write the HTML, and a dictionary of header information.

Footer(thd, hinfo) takes the same arguments.  
See \_header() and \_footer() below, for examples.

## 125.1 Functions

**qfiles**(*dir*)

**av**(*k*, *v*)

**measure\_indent**(*s*)

**prepare\_text**(*l*)

**starts\_with\_a\_tag**(*l*)

**tokenize**(*l*)

**format\_dict**(*d*, *drop*=None)

**dequote**(*a*)

**urlq**(*s*)

**prepend**(*a*, *b*)

Process a URL. Pathnames are prepended with the server root. Complete URLs are untouched.

**process\_tag**(*d*, *od*, *hinfo*, *eol*)

**escape**(*s*)

**get\_logical\_line**(*lines*)

**process**(*lines*, *od*, *hinfo*)

**swapend**(*a*, *e*)

**aget**(*dic*, *alist*)

```
sw(s, prefix)
```

```
dpre(s, prefix)
```

```
agp(dic, prefix)
```

```
get(dic, key, dfl)
```

```
readheader(fd, t)
```

```
needcopy(a, b, force=None)
```

Do we need to reconstruct file b from file a? Also reconstruct b if it is older than the date in force.

```
file_only(url)
```

Given a URL, just return the file part, not the target specifier inside the file.

```
del_fin_sl(s)
```

```
go(env, force=None)
```

```
easygo()
```

## 125.2 Variables

Name	Description
XML	<b>Value:</b> 0
POINTTAG	<b>Value:</b> 1
LL	<b>Value:</b> 60
TABW	<b>Value:</b> 8
DEFAULT_TAG	<b>Value:</b> 'P'
DEFAULT_INDENT	<b>Value:</b> -1
MTIME	<b>Value:</b> 8
DEFF	<b>Value:</b> 'headfoot.py'
__package__	<b>Value:</b> None
nm	<b>Value:</b> 'lang'
q	<b>Value:</b> 9001

## 125.3 Class *lineC*

### 125.3.1 Methods

<code>__init__(self, l)</code>
--------------------------------

<code>getarg(self, k, defv)</code>
------------------------------------

## 126 Module *run\_several*

This script runs a bunch of processes in parallel. You tell it what to run on the standard input. It spawns off a specified number of subprocesses, and hands the jobs to the subprocesses.

Usage: *run\_several.py* [-v] Ncpu Oom\_adjust <list\_of\_commands> >log\_of\_outputs

Flags: -v means "be more verbose".

Ncpu is the maximum number of cores to devote to running these tasks (you can enter zero to have it use all the cores of the local machine), or +N or -N to have it use N more (or less) cores than the machine has. Further x and \* allow you to multiply the number of CPUs by a factor. (NB: The +, -, x, \* forms always leave at least one core running, no matter what N is or how few cores you have.) All this can be useful if you want to avoid loading the machine too heavily, or if you want to boost the loading because your jobs are limited by disk seek time.

Oom\_adjust is a number given to Linux's OOM killer. When the system starts running out of memory, the OOM killer kills processes. Positive numbers (e.g. 3) make it more likely that a process will be killed. So, if your subtasks are likely to use too much memory and are not critical, set Oom\_adjust positive.

The commands in list\_of\_commands are just piped into Bash's standard input one at a time. Each line is (typically) processed by a different instance of bash, so don't try any multi-line commands. The resulting standard outputs and standard errors are kept separate and each processes' outputs are printed as a lump when it completes. Stderr from a subprocess comes out as stdout from *run\_several.py*; various blocks of output are separated by lines with hash marks and strings of "—" (see *write\_msgs()*).

Note that the processes may finish in any arbitrary order. The integer on the stdout and stderr separator lines gives you the (zero-based) line number in the input file. *Run\_several.py* returns failure if any of its subprocesses fail, and it will terminate on the first failure.

Example 1:

```
$ echo "pwd" | run_several.py 0 0
# command 0: pwd
# stdout 0 -----(exited with 0)
/home/gpk/speechresearch/gmisc/lib/bin
# stderr 0 -----
$
```

Example 2:

```
$ { echo uname -m; echo uname -p; echo uname -o; } | run_several.py 0 0
# command 0: uname -m
# stdout 0 -----(exited with 0)
```

```

x86_64
# stderr 0 -----
# command 1: uname -p
# stdout 1 -----(exited with 0)
unknown
# stderr 1 -----
# command 2: uname -o
# stdout 2 -----(exited with 0)
GNU/Linux
# stderr 2 -----
$

```

## 126.1 Functions

<b>write_msgs</b> ( <i>p</i> , <i>exitstatus</i> )
--

This produces the output.
---------------------------

<b>get_ncpu</b> ()
--------------------

<b>wait_until_unloaded</b> ()
-------------------------------

Wait a while until the system becomes less loaded. It won't wait forever, though.
---

<b>set_oom</b> ( <i>pid</i> , <i>ooma</i> )
---

<b>run_processes</b> ( <i>fd</i> , <i>np</i> , <i>ooma</i> , <i>verbose</i> )
---

<b>parse_NP</b> ( <i>s</i> )
------------------------------

## 126.2 Variables

Name	Description
DT	Basic interval to sleep when waiting for the system to become less heavily loaded. <b>Value:</b> 1.0
--package--	<b>Value:</b> None



## 127 Module *select\_fiat\_entries*

This program reads a FIAT format data file and displays some of the information. You can select some of the lines with the `-select` flag, and the display the selected lines in several different ways.

Flags:

- `-select expr` Where `expr` is a python expression that returns True or False. All the normal builtins are available. The data in the line is presented as a set of local variables, so that if the value in the column labelled "filename" is "f005.wav", then you could test for `'filename=="f005.wav"'`. Note that everything is a string, so you may need to convert to int or float before some tests: `'int(nspeakers)<5'`.
- `-showav` Displays the entire line in `a=v; format`.
- `-showfiat` Displays the resulting file in FIAT format, preserving the header information.
- `-show format` This displays the file as a text file, using evaluating format (which is a snippet of python code that should return a string): `-show "%s # %s'%(fname,loudness)'` will display the `fname` and `loudness` columns of the selected lines, separated by a hash mark.
- `-rav` Reads the input file in `a=v; format` rather than `fiat`.
- `-senv` Exec some python code to set a global variable that is visible in the `-select` expression. (e.g. `"x=3"`).
- `-penv` Exec some python code to set a global variable that is visible in the `-show` expression. (e.g. `"x=3"`).
- `-strap` exception value

### 127.1 Functions

```
process(f, read_fcn, o, select)
```

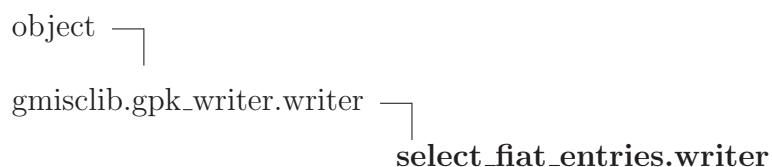
```
fiat_read(fn)
```

```
avio_read(fn)
```

### 127.2 Variables

Name	Description
SHOWAV	<b>Value:</b> 1
SHOWFIAT	<b>Value:</b> 2
__package__	<b>Value:</b> None

## 127.3 Class writer



### 127.3.1 Methods

**comment**(*self*, *comment*)

Add a comment to the data file.

Overrides: gmisclib.gpk\_writer.writer.comment

**header**(*self*, *k*, *v*)

Overrides: gmisclib.gpk\_writer.writer.header

**\_\_init\_\_**(*self*, *fd*, *s*)

x.\_\_init\_\_(...) initializes x; see help(type(x)) for signature

Overrides: object.\_\_init\_\_ extit(inherited documentation)

**datum**(*self*, *data\_item*)

Overrides: gmisclib.gpk\_writer.writer.datum

**globals**(*self*, *code*)

**append**(*self*, *d*)

**close**(*self*)

**comments**(*self*, *comments*)

Add comments to the data file. Comments can appear anywhere.

**data**(*self*, *dataset*)

Write a series of lines to the output file.

**extend**(*self*, *d*)

<code>flush(<i>self</i>)</code>
---------------------------------

<code>headers(<i>self</i>, <i>h</i>)</code>
---

***Inherited from object***

`--delattr--()`, `--format--()`, `--getattr--()`, `--hash--()`, `--new--()`, `--reduce--()`, `--reduce_ex--()`,  
`--repr--()`, `--setattr--()`, `--sizeof--()`, `--str--()`, `--subclasshook--()`

**127.3.2 Properties**

Name	Description
<i>Inherited from object</i>	
<code>--class--</code>	

## Index

ds9, 187

gmisclib (*package*), 27–31

gmisclib.accent\_spec (*module*), 64–66

gmisclib.accent\_spec.BadMatchError (*class*),  
65–66

gmisclib.accent\_spec.prefix (*function*), 64

gmisclib.accent\_spec.preshow (*function*),  
64

gmisclib.accent\_spec.suffix (*function*), 64

gmisclib.accent\_spec.sufshow (*function*),  
65

gmisclib.accent\_spec.test (*function*), 65

gmisclib.array\_window (*module*), 67–68

gmisclib.array\_window.array\_window (*class*),  
67–68

gmisclib.array\_window.test (*function*), 67

gmisclib.avio (*module*), 69–72

gmisclib.avio.BadFormatError (*class*), 70–  
71

gmisclib.avio.concoct (*function*), 69

gmisclib.avio.parse (*function*), 69

gmisclib.avio.read (*function*), 69

gmisclib.avio.read\_hdc (*function*), 70

gmisclib.avio.test1 (*function*), 69

gmisclib.avio.test2 (*function*), 70

gmisclib.avio.writer (*class*), 71–72

gmisclib.bark\_scale (*module*), 73

gmisclib.bark\_scale.acosh (*function*), 73

gmisclib.bark\_scale.asinh (*function*), 73

gmisclib.bark\_scale.bark\_to\_f (*function*),  
73

gmisclib.bark\_scale.cbw (*function*), 73

gmisclib.bark\_scale.f\_to\_bark (*function*),  
73

gmisclib.beamsearch (*module*), 74

gmisclib.beamsearch.beamsearch (*func-*  
*tion*), 74

gmisclib.blue\_data\_attributes (*module*), 75–  
76

gmisclib.blue\_data\_attributes.blue\_attributes  
(*class*), 75–76

gmisclib.blue\_data\_attributes.test (*func-*  
*tion*), 75

gmisclib.blue\_data\_selector (*module*), 77–  
80

gmisclib.blue\_data\_selector.bluedata (*class*),  
77–80

gmisclib.blue\_data\_selector.test (*function*),  
77

gmisclib.cache (*module*), 81–89

gmisclib.cache.BadFileFormat (*class*), 83–  
84

gmisclib.cache.cache\_info (*class*), 84–89

gmisclib.cache.cachepath (*function*), 81

gmisclib.cache.CannotGetStat (*class*), 82–  
83

gmisclib.cache.fileinfo (*function*), 81

gmisclib.cache.modFileInfo (*function*), 81

gmisclib.cache.modinfo (*function*), 81

gmisclib.cache.modinfo\_guts (*function*),  
81

gmisclib.cache.namedModInfo (*function*),  
81

gmisclib.cache.test (*function*), 82

gmisclib.cache.test\_errs (*function*), 82

gmisclib.cache.test\_normal (*function*), 82

gmisclib.cache.walkcache (*function*), 81

gmisclib.chunkio (*module*), 90–100

gmisclib.chunkio.BadFileFormat (*class*),  
90–91

gmisclib.chunkio.chunk (*class*), 91–92

gmisclib.chunkio.chunk\_w (*class*), 96–97

gmisclib.chunkio.chunkstring\_w (*class*),  
98–100

gmisclib.chunkio.datachunk (*class*), 92–  
94

gmisclib.chunkio.datachunk\_w (*class*), 97–  
98

gmisclib.chunkio.stringchunk (*class*), 94–  
96

gmisclib.chunkio.test (*function*), 90

gmisclib.chunkio.test\_e (*function*), 90

gmisclib.convex\_hull2d (*module*), 101–102

- gmislib.convex\_hull2d.convexHull (*function*), 101
- gmislib.convex\_hull2d.DuplicatePoints (*class*), 101–102
- gmislib.convex\_hull2d.saveAsEps (*function*), 101
- gmislib.convex\_hull2d.test (*function*), 101
- gmislib.dict\_vector (*module*), 103–108
  - gmislib.dict\_vector.dict\_vector (*class*), 104–108
  - gmislib.dict\_vector.log (*function*), 103
  - gmislib.dict\_vector.max\_between (*function*), 103
  - gmislib.dict\_vector.max\_within (*function*), 103
  - gmislib.dict\_vector.min\_between (*function*), 103
  - gmislib.dict\_vector.min\_within (*function*), 103
  - gmislib.dict\_vector.outer (*function*), 104
  - gmislib.dict\_vector.round (*function*), 103
  - gmislib.dict\_vector.to\_float (*function*), 103
  - gmislib.dict\_vector.to\_int (*function*), 104
  - gmislib.dict\_vector.to\_numpy\_float\_1 (*function*), 103
  - gmislib.dict\_vector.to\_numpy\_float\_2 (*function*), 103
- gmislib.dictops (*module*), 109–121
  - gmislib.dictops.add\_doc (*function*), 111
  - gmislib.dictops.add\_dol (*function*), 111
  - gmislib.dictops.argmax (*function*), 111
  - gmislib.dictops.BadFileFormatError (*class*), 112
  - gmislib.dictops.compose (*function*), 110
  - gmislib.dictops.dict\_of\_accums (*class*), 115–116
  - gmislib.dictops.dict\_of\_averages (*class*), 116–117
  - gmislib.dictops.dict\_of\_lists (*class*), 112–114
  - gmislib.dictops.dict\_of\_maxes (*class*), 117–119
  - gmislib.dictops.dict\_of\_sets (*class*), 114–115
  - gmislib.dictops.dict\_of\_X (*class*), 119–120
  - gmislib.dictops.filter (*function*), 109
  - gmislib.dictops.intersection (*function*), 109
  - gmislib.dictops.list\_of\_dicts (*class*), 120–121
  - gmislib.dictops.read2c (*function*), 110
  - gmislib.dictops.remove (*function*), 109
  - gmislib.dictops.rev1to1 (*function*), 110
  - gmislib.dictops.test (*function*), 111
  - gmislib.dictops.test\_intersection (*function*), 110
- gmislib.die (*module*), 122–125
  - gmislib.die.catch (*function*), 122
  - gmislib.die.catchexit (*function*), 122
  - gmislib.die.counter (*class*), 124–125
  - gmislib.die.dbg (*function*), 122
  - gmislib.die.die (*function*), 122
  - gmislib.die.exit (*function*), 123
  - gmislib.die.flush (*function*), 123
  - gmislib.die.get (*function*), 123
  - gmislib.die.info (*function*), 122
  - gmislib.die.note (*function*), 123
  - gmislib.die.se\_write (*function*), 124
  - gmislib.die.so\_write (*function*), 124
  - gmislib.die.uprint (*function*), 124
  - gmislib.die.warn (*function*), 122
- gmislib.die2 (*module*), 126–127
  - gmislib.die2.catch (*function*), 126
  - gmislib.die2.catchexit (*function*), 126
  - gmislib.die2.dbg (*function*), 126
  - gmislib.die2.die (*function*), 126
  - gmislib.die2.exit (*function*), 126
  - gmislib.die2.get (*function*), 126
  - gmislib.die2.info (*function*), 126
  - gmislib.die2.note (*function*), 126
  - gmislib.die2.show (*function*), 126
  - gmislib.die2.warn (*function*), 126
- gmislib.do\_exec (*module*), 128
  - gmislib.do\_exec.get (*function*), 128
  - gmislib.do\_exec.getall (*function*), 128
- gmislib.ds9\_region (*module*), 129

- gmisclib.ds9\_region.writer (*class*), 129
- gmisclib.dtw4 (*module*), 130–136
  - gmisclib.dtw4.check\_names (*function*), 130
  - gmisclib.dtw4.dtw (*function*), 130
  - gmisclib.dtw4.dtw\_ti (*function*), 130
  - gmisclib.dtw4.dtwjump (*function*), 131
  - gmisclib.dtw4.map2dist (*function*), 131
  - gmisclib.dtw4.map2index (*function*), 131
  - gmisclib.dtw4.map2shared (*function*), 131
  - gmisclib.dtw4.namelist (*function*), 132
  - gmisclib.dtw4.NoRoute (*class*), 133–134
  - gmisclib.dtw4.parse\_args (*function*), 132
  - gmisclib.dtw4.scale\_tmap (*function*), 132
  - gmisclib.dtw4.shift\_idxmap (*function*), 132
  - gmisclib.dtw4.slice\_c (*class*), 134
  - gmisclib.dtw4.state\_c (*class*), 134–135
  - gmisclib.dtw4.state\_jump\_c (*class*), 135–136
  - gmisclib.dtw4.test (*function*), 132
  - gmisclib.dtw4.test\_partial (*function*), 132
  - gmisclib.dtw4.test\_same (*function*), 132
  - gmisclib.dtw4.test\_scaled (*function*), 132
  - gmisclib.dtw4.test\_stretched (*function*), 132
  - gmisclib.dtw4.time\_it (*function*), 133
- gmisclib.dyn\_prog (*module*), 137
  - gmisclib.dyn\_prog.path (*function*), 137
  - gmisclib.dyn\_prog.test1 (*function*), 137
  - gmisclib.dyn\_prog.test2 (*function*), 137
  - gmisclib.dyn\_prog.test3 (*function*), 137
- gmisclib.edit\_distance (*module*), 138–142
  - gmisclib.edit\_distance.def\_cost (*function*), 139
  - gmisclib.edit\_distance.dist (*function*), 138
  - gmisclib.edit\_distance.distf1 (*function*), 138
  - gmisclib.edit\_distance.distf2 (*function*), 138, 139
  - gmisclib.edit\_distance.free\_del (*function*), 140
  - gmisclib.edit\_distance.free\_sub (*function*), 139
  - gmisclib.edit\_distance.test (*function*), 140
  - gmisclib.edit\_distance.test2 (*function*), 140
  - gmisclib.edit\_distance.text\_cost (*class*), 140–142
- gmisclib.entropy (*module*), 143
  - gmisclib.entropy.information\_gained (*function*), 143
  - gmisclib.entropy.multinomial\_fixer (*function*), 143
  - gmisclib.entropy.multinomial\_logp (*function*), 143
  - gmisclib.entropy.P (*function*), 143
- gmisclib.erb\_scale (*module*), 144
  - gmisclib.erb\_scale.ebw (*function*), 144
  - gmisclib.erb\_scale.erb\_to\_f (*function*), 144
  - gmisclib.erb\_scale.f\_to\_erb (*function*), 144
- gmisclib.evolution (*module*), 145
- gmisclib.fake\_file (*module*), 146
  - gmisclib.fake\_file.file (*class*), 146
  - gmisclib.fake\_file.open (*function*), 146
- gmisclib.fiat\_merge (*module*), 147
  - gmisclib.fiat\_merge.merge (*function*), 147
- gmisclib.fiatio (*module*), 148–160
  - gmisclib.fiatio.col\_order (*function*), 149
  - gmisclib.fiatio.ConflictingColumnSpecification (*class*), 159–160
  - gmisclib.fiatio.FiatioWarning (*class*), 154–155
  - gmisclib.fiatio.merged\_writer (*class*), 157–159
  - gmisclib.fiatio.read (*function*), 150
  - gmisclib.fiatio.read\_as\_float\_array (*function*), 153
  - gmisclib.fiatio.read\_merged (*function*), 151
  - gmisclib.fiatio.readiter (*function*), 152
  - gmisclib.fiatio.shared\_data\_values (*function*), 150
  - gmisclib.fiatio.test (*function*), 154
  - gmisclib.fiatio.test1 (*function*), 153
  - gmisclib.fiatio.test2 (*function*), 154
  - gmisclib.fiatio.test3 (*function*), 154
  - gmisclib.fiatio.test4 (*function*), 154
  - gmisclib.fiatio.write (*function*), 150
  - gmisclib.fiatio.write\_array (*function*), 149
  - gmisclib.fiatio.writer (*class*), 155–157

- gmisclib.find\_home (*module*), 161
  - gmisclib.find\_home.data (*function*), 161
  - gmisclib.find\_home.directory (*function*), 161
  - gmisclib.find\_home.executable (*function*), 161
  - gmisclib.find\_home.module (*function*), 161
  - gmisclib.find\_home.os\_prgm (*function*), 161
- gmisclib.find\_ngram (*module*), 162–163
  - gmisclib.find\_ngram.find (*function*), 162
  - gmisclib.find\_ngram.find\_lab (*function*), 162
  - gmisclib.find\_ngram.find\_mark (*function*), 162
  - gmisclib.find\_ngram.matches (*function*), 162
- gmisclib.findleak (*module*), 164
  - gmisclib.findleak.alloc1k (*function*), 164
  - gmisclib.findleak.get\_refcounts (*function*), 164
  - gmisclib.findleak.lookvmstat (*function*), 164
  - gmisclib.findleak.print\_top\_N (*function*), 164
  - gmisclib.findleak.readvm (*function*), 164
  - gmisclib.findleak.vm (*function*), 164
- gmisclib.ftest (*module*), 165–166
  - gmisclib.ftest.agammln (*function*), 165
  - gmisclib.ftest.betacf (*function*), 165
  - gmisclib.ftest.betai (*function*), 165
  - gmisclib.ftest.fprob (*function*), 165
  - gmisclib.ftest.gammln (*function*), 165
- gmisclib.fuzzygraph (*module*), 167
  - gmisclib.fuzzygraph.addcurve (*function*), 167
- gmisclib.g2\_select (*module*), 168–170
  - gmisclib.g2\_select.accept (*function*), 168
  - gmisclib.g2\_select.evaluate (*function*), 168
  - gmisclib.g2\_select.filter\_iter (*function*), 168
  - gmisclib.g2\_select.filterlist (*function*), 168
  - gmisclib.g2\_select.selector\_c (*class*), 169–170
  - gmisclib.g2\_select.test (*function*), 169
- gmisclib.g2\_select.why (*function*), 168
- gmisclib.g2\_select.whynot (*function*), 168
- gmisclib.g\_closure (*module*), 171–173
  - gmisclib.g\_closure.ArgUnspecifiedError (*class*), 171–172
  - gmisclib.g\_closure.Closure (*class*), 172–173
  - gmisclib.g\_closure.NotYet (*class*), 171
- gmisclib.g\_datetime (*module*), 174
  - gmisclib.g\_datetime.ISO2datetime (*function*), 174
  - gmisclib.g\_datetime.timedelta2float (*function*), 174
- gmisclib.g\_ds9 (*module*), 175–189
  - gmisclib.g\_ds9.BadTagError (*class*), 179–180
  - gmisclib.g\_ds9.circle (*class*), 182–183
  - gmisclib.g\_ds9.copy (*function*), 175
  - gmisclib.g\_ds9.dequote (*function*), 175
  - gmisclib.g\_ds9.ds9 (*class*), 186–188
  - gmisclib.g\_ds9.ds9\_file (*class*), 188–189
  - gmisclib.g\_ds9.ds9\_io (*class*), 178–179
  - gmisclib.g\_ds9.execwait (*class*), 177–178
  - gmisclib.g\_ds9.line (*class*), 181–182
  - gmisclib.g\_ds9.make\_chunks (*function*), 175
  - gmisclib.g\_ds9.point (*class*), 183–184
  - gmisclib.g\_ds9.property\_parser (*function*), 175
  - gmisclib.g\_ds9.r\_list\_factory (*function*), 175
  - gmisclib.g\_ds9.read\_regions (*function*), 175
  - gmisclib.g\_ds9.read\_regions\_iter (*function*), 175
  - gmisclib.g\_ds9.Region (*class*), 180–181
  - gmisclib.g\_ds9.region\_parser (*function*), 175
  - gmisclib.g\_ds9.test\_io (*function*), 175
  - gmisclib.g\_ds9.test\_plot (*function*), 176
  - gmisclib.g\_ds9.test\_property\_parser (*function*), 175
  - gmisclib.g\_ds9.test\_region\_parser (*function*), 175
  - gmisclib.g\_ds9.test\_tags (*function*), 176



- gmisclib.g\_ds9.test\_text\_kluge (*function*), 175
- gmisclib.g\_ds9.text (*class*), 184–186
- gmisclib.g\_ds9.test\_kluge (*function*), 175
- gmisclib.g\_ds9.UnknownProperty (*class*), 176–177
- gmisclib.g\_encode (*module*), 190–191
  - gmisclib.g\_encode.BadFormatError (*class*), 190–191
  - gmisclib.g\_encode.encoder (*class*), 191
  - gmisclib.g\_encode.test (*function*), 190
- gmisclib.g\_entropy (*module*), 192
  - gmisclib.g\_entropy.entropy\_probs (*function*), 192
  - gmisclib.g\_entropy.entropy\_vec (*function*), 192
- gmisclib.g\_exec (*module*), 193–198
  - gmisclib.g\_exec.ExecError (*class*), 197–198
  - gmisclib.g\_exec.get (*function*), 195
  - gmisclib.g\_exec.get\_raw (*function*), 196
  - gmisclib.g\_exec.getall (*function*), 196
  - gmisclib.g\_exec.getiter (*function*), 194
  - gmisclib.g\_exec.getiter\_raw (*function*), 194
  - gmisclib.g\_exec.getlast (*function*), 196
  - gmisclib.g\_exec.test (*function*), 197
- gmisclib.g\_implements (*module*), 199–202
  - gmisclib.g\_implements.check (*function*), 200
  - gmisclib.g\_implements.GITTypeError (*class*), 201–202
  - gmisclib.g\_implements.impl (*function*), 199
  - gmisclib.g\_implements.make\_optional (*function*), 200
  - gmisclib.g\_implements.make\_strict (*function*), 200
  - gmisclib.g\_implements.make\_varargs (*function*), 200
  - gmisclib.g\_implements.make\_vartype (*function*), 200
  - gmisclib.g\_implements.Strict (*function*), 199
  - gmisclib.g\_implements.test (*function*), 200
  - gmisclib.g\_implements.Vartype (*function*), 199
  - gmisclib.g\_implements.why (*function*), 199
- gmisclib.g\_keyed\_accum (*module*), 203
  - gmisclib.g\_keyed\_accum.keyed\_avg (*class*), 203
  - gmisclib.g\_keyed\_accum.test (*function*), 203
- gmisclib.g\_lin\_fit (*module*), 204–205
  - gmisclib.g\_lin\_fit.fit (*function*), 204
  - gmisclib.g\_lin\_fit.test (*function*), 204
  - gmisclib.g\_lin\_fit.test1 (*function*), 204
  - gmisclib.g\_lin\_fit.test2 (*function*), 204
  - gmisclib.g\_lin\_fit.test3 (*function*), 204
  - gmisclib.g\_lin\_fit.test4 (*function*), 204
  - gmisclib.g\_lin\_fit.test5 (*function*), 204
  - gmisclib.g\_lin\_fit.test6 (*function*), 204
- gmisclib.g\_localfit (*module*), 206–211
  - gmisclib.g\_localfit.err\_before\_fit (*function*), 206
  - gmisclib.g\_localfit.fit\_giving\_sigmas (*function*), 209
  - gmisclib.g\_localfit.leaktest (*function*), 210
  - gmisclib.g\_localfit.localfit (*function*), 207
  - gmisclib.g\_localfit.pack (*function*), 206
  - gmisclib.g\_localfit.reg\_localfit (*function*), 208
  - gmisclib.g\_localfit.robust\_localfit (*function*), 209
  - gmisclib.g\_localfit.test0 (*function*), 210
  - gmisclib.g\_localfit.test\_fgs1 (*function*), 210
  - gmisclib.g\_localfit.test\_localfit11 (*function*), 210
  - gmisclib.g\_localfit.test\_localfit11e (*function*), 210
  - gmisclib.g\_localfit.test\_localfit21 (*function*), 210
  - gmisclib.g\_localfit.test\_localfit21u (*function*), 210
  - gmisclib.g\_localfit.test\_localfit22 (*function*), 210
  - gmisclib.g\_localfit.test\_wt (*function*), 211
- gmisclib.g\_pipe (*module*), 212–213
  - gmisclib.g\_pipe.pfd (*class*), 212–213



- gmisclib.g\_pipe.popen2 (*function*), 212
- gmisclib.g\_pipe.test (*function*), 212
- gmisclib.g\_pipe\_old (*module*), 214–215
  - gmisclib.g\_pipe\_old.pfd (*class*), 214–215
  - gmisclib.g\_pipe\_old.popen2 (*function*), 214
  - gmisclib.g\_pipe\_old.test (*function*), 214
- gmisclib.g\_place\_label (*module*), 216–219
  - gmisclib.g\_place\_label.boxc (*class*), 216–217
  - gmisclib.g\_place\_label.Gauss (*function*), 216
  - gmisclib.g\_place\_label.gauss (*function*), 216
  - gmisclib.g\_place\_label.q\_gauss (*function*), 216
  - gmisclib.g\_place\_label.text\_template (*class*), 217–218
  - gmisclib.g\_place\_label.viewport (*class*), 218–219
- gmisclib.g\_pylab (*module*), 220
- gmisclib.g\_selector (*module*), 221–225
  - gmisclib.g\_selector.false (*class*), 222–223
  - gmisclib.g\_selector.one\_selector (*class*), 224–225
  - gmisclib.g\_selector.selector (*class*), 221–222
  - gmisclib.g\_selector.selector\_not (*function*), 221
  - gmisclib.g\_selector.selector\_op (*class*), 223–224
  - gmisclib.g\_selector.test (*function*), 221
  - gmisclib.g\_selector.true (*class*), 222
- gmisclib.g\_ucode (*module*), 226
  - gmisclib.g\_ucode.e (*function*), 226
  - gmisclib.g\_ucode.test (*function*), 226
  - gmisclib.g\_ucode.u (*function*), 226
- gmisclib.gpk\_getopt (*module*), 227
  - gmisclib.gpk\_getopt.parse (*function*), 227
- gmisclib.gpk\_hdr (*module*), 228
  - gmisclib.gpk\_hdr.hdr (*class*), 228
  - gmisclib.gpk\_hdr.uq (*function*), 228
- gmisclib.gpk\_lapack (*module*), 229–230
  - gmisclib.gpk\_lapack.err (*function*), 229
  - gmisclib.gpk\_lapack.NoSolutionError (*class*), 229–230
  - gmisclib.gpk\_lapack.solve (*function*), 229
  - gmisclib.gpk\_lapack.test1 (*function*), 229
  - gmisclib.gpk\_lapack.test2 (*function*), 229
  - gmisclib.gpk\_lapack.testa (*function*), 229
  - gmisclib.gpk\_lapack.testbdi (*function*), 229
- gmisclib.gpk\_lsqr (*module*), 231–239
  - gmisclib.gpk\_lsqr.all\_between (*function*), 231
  - gmisclib.gpk\_lsqr.linear\_least\_squares (*class*), 233–236
  - gmisclib.gpk\_lsqr.lls\_base (*class*), 231–233
  - gmisclib.gpk\_lsqr.reg\_linear\_least\_squares (*class*), 236–239
  - gmisclib.gpk\_lsqr.test0 (*function*), 231
  - gmisclib.gpk\_lsqr.test\_hat (*function*), 231
  - gmisclib.gpk\_lsqr.test\_m1 (*function*), 231
  - gmisclib.gpk\_lsqr.test\_m2 (*function*), 231
  - gmisclib.gpk\_lsqr.test\_m2r (*function*), 231
  - gmisclib.gpk\_lsqr.test\_m2rR (*function*), 231
  - gmisclib.gpk\_lsqr.test\_svd (*function*), 231
  - gmisclib.gpk\_lsqr.test\_vec (*function*), 231
  - gmisclib.gpk\_lsqr.test\_vec2 (*function*), 231
- gmisclib.gpk\_rlsqr (*module*), 240–247
  - gmisclib.gpk\_rlsqr.NotEnoughData (*class*), 240–241
  - gmisclib.gpk\_rlsqr.robust\_linear\_fit (*class*), 244–247
  - gmisclib.gpk\_rlsqr.test\_r1 (*function*), 240
  - gmisclib.gpk\_rlsqr.test\_r1hat (*function*), 240
  - gmisclib.gpk\_rlsqr.test\_w1 (*function*), 240
  - gmisclib.gpk\_rlsqr.test\_w1b (*function*), 240
  - gmisclib.gpk\_rlsqr.w\_linear\_least\_squares (*class*), 241–244
- gmisclib.gpk\_writer (*module*), 248–250
  - gmisclib.gpk\_writer.null\_writer (*class*), 249–250
  - gmisclib.gpk\_writer.writer (*class*), 248–249
- gmisclib.gpkmisc (*module*), 251–260
  - gmisclib.gpkmisc.a\_factor (*function*), 254
  - gmisclib.gpkmisc.asinh (*function*), 256

- gmisc.lib.gpkmisc.avg (*function*), 251
- gmisc.lib.gpkmisc.chooseP (*function*), 257
- gmisc.lib.gpkmisc.ComplexMedian (*function*), 252
- gmisc.lib.gpkmisc.dir\_lock (*class*), 258–259
- gmisc.lib.gpkmisc.distrib (*function*), 257
- gmisc.lib.gpkmisc.dropfront (*function*), 254
- gmisc.lib.gpkmisc.entropy (*function*), 252
- gmisc.lib.gpkmisc.erf (*function*), 256
- gmisc.lib.gpkmisc.factor (*function*), 254
- gmisc.lib.gpkmisc.find\_in\_PATH (*function*), 255
- gmisc.lib.gpkmisc.gammaln (*function*), 254
- gmisc.lib.gpkmisc.gcd (*function*), 255
- gmisc.lib.gpkmisc.geo\_mean (*function*), 251
- gmisc.lib.gpkmisc.get\_mtime (*function*), 255
- gmisc.lib.gpkmisc.jackknife (*function*), 252
- gmisc.lib.gpkmisc.log\_Combinations (*function*), 252
- gmisc.lib.gpkmisc.log\_factorial (*function*), 252
- gmisc.lib.gpkmisc.log\_Student\_t\_dens (*function*), 252
- gmisc.lib.gpkmisc.makedirs (*function*), 252
- gmisc.lib.gpkmisc.mean (*function*), 251
- gmisc.lib.gpkmisc.mean\_ad (*function*), 251
- gmisc.lib.gpkmisc.mean\_stddev (*function*), 251
- gmisc.lib.gpkmisc.median (*function*), 251
- gmisc.lib.gpkmisc.median\_across (*function*), 251
- gmisc.lib.gpkmisc.median\_ad (*function*), 251
- gmisc.lib.gpkmisc.misc\_mode (*function*), 257
- gmisc.lib.gpkmisc.need\_to\_recompute (*function*), 256
- gmisc.lib.gpkmisc.open\_compressed (*function*), 254
- gmisc.lib.gpkmisc.open\_nowipe (*function*), 253
- gmisc.lib.gpkmisc.prereq\_mtime (*function*), 256
- gmisc.lib.gpkmisc.PrereqError (*class*), 259–260
- gmisc.lib.gpkmisc.primes (*function*), 254
- gmisc.lib.gpkmisc.resample (*function*), 252
- gmisc.lib.gpkmisc.shuffle\_Nrep (*function*), 253
- gmisc.lib.gpkmisc.Student\_t\_dens (*function*), 252
- gmisc.lib.gpkmisc.test\_primes (*function*), 255
- gmisc.lib.gpkmisc.testCM (*function*), 252
- gmisc.lib.gpkmisc.testSNR (*function*), 253
- gmisc.lib.gpkmisc.thr\_iter\_read (*function*), 252
- gmisc.lib.gpkmisc.threaded\_readable\_file (*class*), 257–258
- gmisc.lib.gpkmisc.truncate (*function*), 256
- gmisc.lib.hilbert\_xform (*module*), 261
- gmisc.lib.hilbert\_xform.hilbert (*function*), 261
- gmisc.lib.hilbert\_xform.nextpow2 (*function*), 261
- gmisc.lib.HList (*module*), 32
- gmisc.lib.HList.read (*function*), 32
- gmisc.lib.HTK\_HMM\_io (*module*), 33–42
- gmisc.lib.HTK\_HMM\_io.BadFormatError (*class*), 33–34
- gmisc.lib.HTK\_HMM\_io.deref (*function*), 33
- gmisc.lib.HTK\_HMM\_io.dict\_format (*function*), 33
- gmisc.lib.HTK\_HMM\_io.hmm (*class*), 38–39
- gmisc.lib.HTK\_HMM\_io.hmmset (*class*), 39–40
- gmisc.lib.HTK\_HMM\_io.HTKformat (*function*), 33
- gmisc.lib.HTK\_HMM\_io.mixture (*class*), 36–37
- gmisc.lib.HTK\_HMM\_io.NoPhonemeToFormat (*class*), 41–42
- gmisc.lib.HTK\_HMM\_io.read (*function*), 33

- gmisclib.HTK\_HMM\_io.read\_floats\_b (*function*), 33
- gmisclib.HTK\_HMM\_io.read\_floats\_t (*function*), 33
- gmisclib.HTK\_HMM\_io.read\_vec (*function*), 33
- gmisclib.HTK\_HMM\_io.ref (*class*), 37
- gmisclib.HTK\_HMM\_io.state (*class*), 37–38
- gmisclib.HTK\_HMM\_io.test (*function*), 33
- gmisclib.HTK\_HMM\_io.tokstream (*class*), 40–41
- gmisclib.HTK\_HMM\_io.vec (*class*), 34–36
- gmisclib.HTK\_MLF\_io (*module*), 43–47
  - gmisclib.HTK\_MLF\_io.parse\_label\_line (*function*), 43
  - gmisclib.HTK\_MLF\_io.read (*function*), 44
  - gmisclib.HTK\_MLF\_io.readiter (*function*), 43
  - gmisclib.HTK\_MLF\_io.readone (*function*), 43
  - gmisclib.HTK\_MLF\_io.ReferencedFileNotFound (*class*), 45–46
  - gmisclib.HTK\_MLF\_io.writer (*class*), 46–47
- gmisclib.kl\_dist (*module*), 262–265
  - gmisclib.kl\_dist.cross (*function*), 263
  - gmisclib.kl\_dist.estimate\_tr\_probs (*function*), 262
  - gmisclib.kl\_dist.kl\_nonzero\_prob\_m (*function*), 262
  - gmisclib.kl\_dist.kl\_nonzero\_probs (*function*), 262
  - gmisclib.kl\_dist.kl\_nonzero\_tr\_prob\_m (*function*), 263
  - gmisclib.kl\_dist.kl\_nonzero\_tr\_probs (*function*), 263
  - gmisclib.kl\_dist.kldist\_Markov (*function*), 263
  - gmisclib.kl\_dist.kldist\_Markov\_m (*function*), 263
  - gmisclib.kl\_dist.kldist\_Markov\_mm (*function*), 263
  - gmisclib.kl\_dist.kldist\_vec (*function*), 262
  - gmisclib.kl\_dist.multinomial\_fixer (*function*), 262
  - gmisclib.kl\_dist.multinomial\_logp (*function*), 262
  - gmisclib.kl\_dist.NoConvergenceError (*class*), 263–264
  - gmisclib.kl\_dist.NotMarkovError (*class*), 264–265
  - gmisclib.kl\_dist.P (*function*), 262
  - gmisclib.kl\_dist.solve\_for\_pi (*function*), 263
  - gmisclib.kl\_dist.tr\_from\_obs (*function*), 262
- gmisclib.load\_mod (*module*), 266–268
  - gmisclib.load\_mod.load (*function*), 266
  - gmisclib.load\_mod.load\_named (*function*), 266, 267
  - gmisclib.load\_mod.split\_name (*function*), 266
- gmisclib.lreg\_fill (*module*), 269
  - gmisclib.lreg\_fill.fill (*function*), 269
- gmisclib.makemake (*module*), 270–272
  - gmisclib.makemake.blank (*function*), 270
  - gmisclib.makemake.date (*function*), 270
  - gmisclib.makemake.FileNotFound (*class*), 271–272
  - gmisclib.makemake.finish (*function*), 270
  - gmisclib.makemake.log (*function*), 270
  - gmisclib.makemake.maketemp (*function*), 270
  - gmisclib.makemake.ncpu (*function*), 270
  - gmisclib.makemake.path\_to (*function*), 270
  - gmisclib.makemake.quote (*function*), 270
  - gmisclib.makemake.read (*function*), 271
  - gmisclib.makemake.rule (*function*), 270
  - gmisclib.makemake.set\_debug (*function*), 270
  - gmisclib.makemake.set\_make\_prog (*function*), 270
  - gmisclib.makemake.setlog (*function*), 270
  - gmisclib.makemake.var (*function*), 270

- gmisclib.matrix\_arrange (*module*), 273
  - gmisclib.matrix\_arrange.diagonalize (*function*), 273
  - gmisclib.matrix\_arrange.map\_array (*function*), 273
  - gmisclib.matrix\_arrange.test (*function*), 273
- gmisclib.matrix\_arrange\_entropy (*module*), 274
  - gmisclib.matrix\_arrange\_entropy.make\_min\_entropy (*function*), 274
  - gmisclib.matrix\_arrange\_entropy.test2 (*function*), 274
  - gmisclib.matrix\_arrange\_entropy.test3 (*function*), 274
- gmisclib.matrix\_rearrange\_labels (*module*), 275–282
  - gmisclib.matrix\_rearrange\_labels.blockfom (*class*), 282
  - gmisclib.matrix\_rearrange\_labels.diagfom (*class*), 280–281
  - gmisclib.matrix\_rearrange\_labels.diagfom2 (*class*), 281–282
  - gmisclib.matrix\_rearrange\_labels.neg\_near\_diag2 (*function*), 278
  - gmisclib.matrix\_rearrange\_labels.pos\_near\_diag2 (*function*), 278
  - gmisclib.matrix\_rearrange\_labels.swap\_toward\_blocks (*function*), 279
  - gmisclib.matrix\_rearrange\_labels.swap\_toward\_blocks2 (*function*), 278
  - gmisclib.matrix\_rearrange\_labels.swap\_toward\_diag (*function*), 275
  - gmisclib.matrix\_rearrange\_labels.swap\_toward\_diag2 (*function*), 277
  - gmisclib.matrix\_rearrange\_labels.swap\_toward\_gmisclib (*function*), 275
  - gmisclib.matrix\_rearrange\_labels.swap\_toward\_gmisclib2 (*function*), 276
  - gmisclib.matrix\_rearrange\_labels.symm\_swap\_toward (*function*), 275
  - gmisclib.matrix\_rearrange\_labels.test (*function*), 280
  - gmisclib.matrix\_rearrange\_labels.test1 (*function*), 279
- gmisclib.mcmc (*module*), 283–322
  - gmisclib.mcmc.acceptor\_base (*class*), 297–299
  - gmisclib.mcmc.adjuster (*class*), 308–309
  - gmisclib.mcmc.Archive (*class*), 313–315
  - gmisclib.mcmc.BootStepper (*class*), 318–322
  - gmisclib.mcmc.bootstepper (*function*), 284
  - gmisclib.mcmc.ContPrmArchive (*class*), 315–318
  - gmisclib.mcmc.diag\_variance (*function*), 284
  - gmisclib.mcmc.hashcounter\_c (*class*), 312–313
  - gmisclib.mcmc.make\_list\_of\_positions (*function*), 284
  - gmisclib.mcmc.NoBoot (*class*), 309–310
  - gmisclib.mcmc.NotGoodPosition (*class*), 310–312
  - gmisclib.mcmc.position\_base (*class*), 290–292
  - gmisclib.mcmc.position\_nonrepeatable (*class*), 294–297
  - gmisclib.mcmc.position\_repeatable (*class*), 292–294
  - gmisclib.mcmc.problem\_definition (*class*), 284–287
  - gmisclib.mcmc.problem\_definition\_F (*class*), 287–290
  - gmisclib.mcmc.rough\_acceptor\_base (*class*), 301–303
  - gmisclib.mcmc.rough\_T\_acceptor (*class*), 303–306
  - gmisclib.mcmc.start\_is\_list\_a (*function*), 284
  - gmisclib.mcmc.start\_is\_list\_p (*function*), 284
  - gmisclib.mcmc.minimal\_stepper (*class*), 306–308
  - gmisclib.mcmc.T\_acceptor (*class*), 299–301
  - gmisclib.mcmc.test (*function*), 284
  - gmisclib.mcmc.test\_ (*function*), 284

- gmislib.mcmc.big (*module*), 324–328
  - gmislib.mcmc.big.BootStepper (*class*), 324–328
  - gmislib.mcmc.big.bootstepper (*function*), 324
  - gmislib.mcmc.big.find\_closest\_p (*function*), 324
  - gmislib.mcmc.big.N\_maximum (*function*), 324
  - gmislib.mcmc.big.test (*function*), 324
  - gmislib.mcmc.big.test2d (*function*), 324
- gmislib.mcmc.cooperate (*module*), 329–334
  - gmislib.mcmc.cooperate.Barrier (*class*), 329–330
  - gmislib.mcmc.cooperate.connection (*class*), 332–334
  - gmislib.mcmc.cooperate.LateToBarrier (*class*), 331–332
  - gmislib.mcmc.cooperate.Oops (*class*), 330–331
  - gmislib.mcmc.cooperate.op\_float\_median (*function*), 329
  - gmislib.mcmc.cooperate.op\_string\_median (*function*), 329
  - gmislib.mcmc.cooperate.test0 (*function*), 329
  - gmislib.mcmc.cooperate.test0s (*function*), 329
  - gmislib.mcmc.cooperate.test1 (*function*), 329
  - gmislib.mcmc.cooperate.test\_many (*function*), 329
- gmislib.mcmc.helper (*module*), 335–346
  - gmislib.mcmc.helper.logger\_template (*class*), 337–338
  - gmislib.mcmc.helper.make\_stepper\_from\_lop (*function*), 335
  - gmislib.mcmc.helper.make\_stepper\_from\_lov (*function*), 335
  - gmislib.mcmc.helper.step\_acceptor (*class*), 342–346
  - gmislib.mcmc.helper.stepper (*class*), 338–342
  - gmislib.mcmc.helper.test (*function*), 336
  - gmislib.mcmc.helper.test1 (*function*), 335
  - gmislib.mcmc.helper.test\_probe (*function*), 335
  - gmislib.mcmc.helper.TooManyLoops (*class*), 336–337
  - gmislib.mcmc.helper.warnevery (*class*), 337
- gmislib.mcmc.idx (*module*), 347–360
  - gmislib.mcmc.idx.Expo (*class*), 351–352
  - gmislib.mcmc.idx.Fixed (*class*), 348–349
  - gmislib.mcmc.idx.LogNormal (*class*), 355–357
  - gmislib.mcmc.idx.logp\_prior\_normalized (*function*), 347
  - gmislib.mcmc.idx.Normal (*class*), 352–354
  - gmislib.mcmc.idx.probdist\_c (*class*), 347–348
  - gmislib.mcmc.idx.problem\_definition (*class*), 357–360
  - gmislib.mcmc.idx.Uniform (*class*), 354–355
  - gmislib.mcmc.idx.Weibull (*class*), 349–351
- gmislib.mcmc.indexclass (*module*), 361–381
  - gmislib.mcmc.indexclass.default\_mapfp (*function*), 361
  - gmislib.mcmc.indexclass.guess (*class*), 368–372
  - gmislib.mcmc.indexclass.guess\_to\_indexer (*function*), 361
  - gmislib.mcmc.indexclass.index (*class*), 372–377
  - gmislib.mcmc.indexclass.index\_base (*class*), 364–366
  - gmislib.mcmc.indexclass.index\_counted (*class*), 377–381
  - gmislib.mcmc.indexclass.IndexKeyError (*class*), 363–364



- gmislib.mcmc\_indexclass.MissingParameterError(*function*), 391
- (*class*), 366–367
- gmislib.mcmc\_indexclass.PriorProbDist
- (*class*), 361–363
- gmislib.mcmc\_indexclass.reindex (*function*), 361
- gmislib.mcmc\_indexclass.sampler (*class*), 367–368
- gmislib.mcmc\_logger (*module*), 382–390
- gmislib.mcmc\_logger.logger\_A (*class*), 385–386
- gmislib.mcmc\_logger.logger\_c (*class*), 384–385
- gmislib.mcmc\_logger.logger\_N (*class*), 386–388
- gmislib.mcmc\_logger.lti\_c (*class*), 388–389
- gmislib.mcmc\_logger.NoDataError (*class*), 389–390
- gmislib.mcmc\_logger.read\_currentlist (*function*), 382
- gmislib.mcmc\_logger.read\_multisample (*function*), 382
- gmislib.mcmc\_logger.WildNumber (*class*), 383–384
- gmislib.mcmc\_logtools (*module*), 391–394
- gmislib.mcmc\_logtools.all (*function*), 391
- gmislib.mcmc\_logtools.ascii\_cmp (*function*), 392
- gmislib.mcmc\_logtools.drop\_files (*function*), 392
- gmislib.mcmc\_logtools.each\_best (*function*), 392
- gmislib.mcmc\_logtools.get\_pmap (*function*), 391
- gmislib.mcmc\_logtools.indexer\_covar (*function*), 391
- gmislib.mcmc\_logtools.indexer\_stdev (*function*), 392
- gmislib.mcmc\_logtools.key\_cmp (*function*), 393
- gmislib.mcmc\_logtools.last (*function*), 392
- gmislib.mcmc\_logtools.list\_prm\_samples
- gmislib.mcmc\_logtools.logp\_stdev (*function*), 391
- gmislib.mcmc\_logtools.near\_each\_max (*function*), 392
- gmislib.mcmc\_logtools.onelog (*class*), 394
- gmislib.mcmc\_logtools.overall\_best (*function*), 392
- gmislib.mcmc\_logtools.P\_bayes (*function*), 393
- gmislib.mcmc\_logtools.P\_bayes\_list (*function*), 393
- gmislib.mcmc\_logtools.print\_index\_error (*function*), 392
- gmislib.mcmc\_logtools.read\_many\_files (*function*), 391
- gmislib.mcmc\_logtools.read\_uid\_many\_files (*function*), 391
- gmislib.mcmc\_logtools.some\_after\_convergence (*function*), 391
- gmislib.mcmc\_m4p (*module*), 395–399
- gmislib.mcmc\_m4p.is\_root (*function*), 395
- gmislib.mcmc\_m4p.precompute\_logp (*function*), 395
- gmislib.mcmc\_m4p.rank (*function*), 395
- gmislib.mcmc\_m4p.size (*function*), 395
- gmislib.mcmc\_m4p.stepper (*class*), 395–399
- gmislib.mcmc\_m4p.test\_ (*function*), 395
- gmislib.mcmc\_mpi (*module*), 400–404
- gmislib.mcmc\_mpi.precompute\_logp (*function*), 400
- gmislib.mcmc\_mpi.stepper (*class*), 400–404
- gmislib.mcmc\_mpi.test\_ (*function*), 400
- gmislib.mcmc\_newlogger (*module*), 405–416
- gmislib.mcmc\_newlogger.DBGlogger (*class*), 410–412
- gmislib.mcmc\_newlogger.load\_module (*function*), 408
- gmislib.mcmc\_newlogger.logger (*class*), 412–413
- gmislib.mcmc\_newlogger.logger\_base (*class*),

- 409–410
- gmisclib.mcmc\_newlogger.logline (*class*), 414–416
- gmisclib.mcmc\_newlogger.NoDataError (*class*), 413–414
- gmisclib.mcmc\_newlogger.ok (*function*), 405
- gmisclib.mcmc\_newlogger.print\_log (*function*), 408
- gmisclib.mcmc\_newlogger.read\_human\_fmt (*function*), 408
- gmisclib.mcmc\_newlogger.read\_log (*function*), 405
- gmisclib.mcmc\_newlogger.read\_multi\_uid (*function*), 407
- gmisclib.mcmc\_newlogger.read\_multilog (*function*), 406
- gmisclib.mcmc\_newlogger.read\_raw (*function*), 405
- gmisclib.mcmc\_newlogger.read\_tail\_uid (*function*), 407
- gmisclib.mcmc\_newlogger.readiter (*function*), 405
- gmisclib.mcmc\_newlogger.WildNumber (*class*), 408–409
- gmisclib.mcmc\_pypar (*module*), 417–420
  - gmisclib.mcmc\_pypar.is\_root (*function*), 417
  - gmisclib.mcmc\_pypar.precompute\_logp (*function*), 417
  - gmisclib.mcmc\_pypar.Reduce (*function*), 417
  - gmisclib.mcmc\_pypar.size (*function*), 417
  - gmisclib.mcmc\_pypar stepper (*class*), 417–420
  - gmisclib.mcmc\_pypar.test\_opt (*function*), 417
- gmisclib.mcmc\_restart (*module*), 421
  - gmisclib.mcmc\_restart.setup (*function*), 421
- gmisclib.mcmc\_socket (*module*), 422–426
  - gmisclib.mcmc\_socket.precompute\_logp (*function*), 422
  - gmisclib.mcmc\_socket stepper (*class*), 422–426
  - gmisclib.mcmc\_socket.test\_opt (*function*), 422
- gmisclib.mcmcS2 (*module*), 323
  - gmisclib.mcmcS2.go\_step (*function*), 323
- gmisclib.MFCCFile (*module*), 48
  - gmisclib.MFCCFile.MFCCFile (*class*), 48
- gmisclib.MLF\_file (*module*), 49–52
  - gmisclib.MLF\_file.BadFormatError (*class*), 49–50
  - gmisclib.MLF\_file.block\_MLF\_file (*class*), 51–52
  - gmisclib.MLF\_file.NotInMLFFile (*class*), 50–51
- gmisclib.multivariate\_classes (*module*), 427–430
  - gmisclib.multivariate\_classes.diag\_inv\_variance (*function*), 427
  - gmisclib.multivariate\_classes.model\_with\_numbers (*class*), 429–430
  - gmisclib.multivariate\_classes.modeldesc (*class*), 428–429
  - gmisclib.multivariate\_classes.QuadraticNotNormalized (*class*), 427–428
  - gmisclib.multivariate\_classes.vec\_inv\_variance (*function*), 427
- gmisclib.multivariate\_mm (*module*), 431–436
  - gmisclib.multivariate\_mm.multi\_mu\_init\_ (*function*), 431
  - gmisclib.multivariate\_mm.datum\_c (*class*), 431
  - gmisclib.multivariate\_mm.multi\_mu (*class*), 431–433
  - gmisclib.multivariate\_mm.multi\_mu\_diag (*class*), 434–435
  - gmisclib.multivariate\_mm.multi\_mu\_diag\_with\_numbers (*class*), 435–436
  - gmisclib.multivariate\_mm.multi\_mu\_with\_numbers (*class*), 433–434
- gmisclib.multivariate\_q (*module*), 437–442
  - gmisclib.multivariate\_q.diag\_quadratic

- (*class*), 439–440
- gmisclib.multivariate\_q.diag\_quadratic\_with\_numbers58
  - (*class*), 440–442
- gmisclib.multivariate\_q.quadratic (*class*), 437–438
- gmisclib.multivariate\_q.quadratic\_with\_numbers58
  - (*class*), 438–439
- gmisclib.multivariate\_normal (*module*), 443
  - gmisclib.multivariate\_normal.multivariate\_normal (*class*), 443
  - gmisclib.multivariate\_normal.test (*function*), 443
- gmisclib.named\_block\_file (*module*), 444
  - gmisclib.named\_block\_file.read (*function*), 444
  - gmisclib.named\_block\_file.write (*function*), 444
  - gmisclib.named\_block\_file.write\_key (*function*), 444
  - gmisclib.named\_block\_file.write\_line (*function*), 444
  - gmisclib.named\_block\_file.write\_line\_kv (*function*), 444
  - gmisclib.named\_block\_file.write\_text (*function*), 444
- gmisclib.nbest (*module*), 445–446
  - gmisclib.nbest.go (*function*), 445
  - gmisclib.nbest.node (*class*), 445–446
  - gmisclib.nbest.test (*function*), 445
- gmisclib.nice\_hash (*module*), 447–455
  - gmisclib.nice\_hash.DontHashThis (*class*), 447–448
  - gmisclib.nice\_hash.hash (*class*), 451–455
  - gmisclib.nice\_hash.map (*function*), 447
  - gmisclib.nice\_hash.NotInHash (*class*), 448–449
  - gmisclib.nice\_hash.simple (*class*), 449–451
- gmisclib.nicknames (*module*), 456
  - gmisclib.nicknames.nicknames (*function*), 456
  - gmisclib.nicknames.test (*function*), 456
- gmisclib.nmf (*module*), 457
  - gmisclib.nmf.nmf (*function*), 457
- gmisclib.Num (*module*), 53–57
  - gmisclib.Numeric\_gpk (*module*), 58–63
  - gmisclib.Numeric\_gpk.add\_overlap (*function*), 58
  - gmisclib.Numeric\_gpk.argmax (*function*), 58
  - gmisclib.Numeric\_gpk.bevel\_concat (*function*), 58
  - gmisclib.Numeric\_gpk.block\_stdev (*function*), 60
  - gmisclib.Numeric\_gpk.convolve (*function*), 60
  - gmisclib.Numeric\_gpk.edge\_window (*function*), 60
  - gmisclib.Numeric\_gpk.edge\_window\_t (*function*), 61
  - gmisclib.Numeric\_gpk.EdgesTooWide (*class*), 62–63
  - gmisclib.Numeric\_gpk.interp (*function*), 59
  - gmisclib.Numeric\_gpk.interpN (*function*), 59
  - gmisclib.Numeric\_gpk.KolmogorovSmirnov (*function*), 59
  - gmisclib.Numeric\_gpk.limit (*function*), 59
  - gmisclib.Numeric\_gpk.median\_across (*function*), 58
  - gmisclib.Numeric\_gpk.N\_frac\_rank (*function*), 58
  - gmisclib.Numeric\_gpk.N\_maximum (*function*), 58
  - gmisclib.Numeric\_gpk.N\_mean\_ad (*function*), 58
  - gmisclib.Numeric\_gpk.N\_median (*function*), 58
  - gmisclib.Numeric\_gpk.N\_median\_across (*function*), 59
  - gmisclib.Numeric\_gpk.N\_minimum (*function*), 58
  - gmisclib.Numeric\_gpk.Poisson (*function*), 58
  - gmisclib.Numeric\_gpk.qform (*function*), 59



- gmisclib.Numeric\_gpk.set\_diag (*function*), 59
- gmisclib.Numeric\_gpk.split\_into\_clumps (*function*), 59
- gmisclib.Numeric\_gpk.trimmed\_mean\_across (*function*), 59
- gmisclib.Numeric\_gpk.trimmed\_mean\_sigma\_across (*function*), 59
- gmisclib.Numeric\_gpk.trimmed\_stdev\_across (*function*), 59
- gmisclib.Numeric\_gpk.vec\_variance (*function*), 59
- gmisclib.Numeric\_gpk.zero\_pad\_end (*function*), 58
- gmisclib.opt (*module*), 458–471
  - gmisclib.opt.anneal\_guts (*function*), 459
  - gmisclib.opt.BadParamError (*class*), 464–465
  - gmisclib.opt.BadResult (*class*), 465–466
  - gmisclib.opt.clz\_point (*function*), 458
  - gmisclib.opt.deriv\_estimate (*function*), 459
  - gmisclib.opt.diff\_one (*function*), 459
  - gmisclib.opt.eval\_lambda (*function*), 459
  - gmisclib.opt.explored\_region (*function*), 458
  - gmisclib.opt.lamb\_correct (*function*), 459
  - gmisclib.opt.linconst\_max (*function*), 460
  - gmisclib.opt.linconst\_min (*function*), 460
  - gmisclib.opt.linear\_constraint (*function*), 460
  - gmisclib.opt.LockedList (*class*), 468
  - gmisclib.opt.maxabs (*function*), 458
  - gmisclib.opt.mysem (*class*), 466–467
  - gmisclib.opt.near\_duplicate (*function*), 458
  - gmisclib.opt.need\_more\_diff\_pts (*function*), 459
  - gmisclib.opt.NoDerivative (*class*), 463–464
  - gmisclib.opt.NoDownhill (*class*), 462–463
  - gmisclib.opt.opt (*class*), 468–471
  - gmisclib.opt.OptError (*class*), 461–462
  - gmisclib.opt.prms (*class*), 467–468
  - gmisclib.opt.scale\_est (*function*), 458
  - gmisclib.opt.semclass (*class*), 467
  - gmisclib.opt.sumsq (*function*), 458
  - gmisclib.opt.symmetry\_point (*function*), 458
  - gmisclib.opt.test1 (*function*), 459
  - gmisclib.opt.test1\_fcn (*function*), 459
  - gmisclib.opt.test1a (*function*), 459
  - gmisclib.opt.test1a\_fcn (*function*), 459
  - gmisclib.opt.test2 (*function*), 459
  - gmisclib.opt.test2\_fcn (*function*), 459
  - gmisclib.opt.test3 (*function*), 459
  - gmisclib.opt.test3\_fcn (*function*), 459
  - gmisclib.opt.test4 (*function*), 460
  - gmisclib.opt.test4\_fcn (*function*), 460
  - gmisclib.opt.test5 (*function*), 460
  - gmisclib.opt.test5\_fcn (*function*), 460
  - gmisclib.opt.test6 (*function*), 460
  - gmisclib.opt.test6\_fcn (*function*), 460
  - gmisclib.opt.test7 (*function*), 460
  - gmisclib.opt.vec\_equal (*function*), 458
  - gmisclib.opt.wtf (*function*), 458
- gmisclib.ortho\_poly (*module*), 472–482
  - gmisclib.ortho\_poly.Chebyshev (*class*), 476–478
  - gmisclib.ortho\_poly.Chebyshev2 (*class*), 478–479
  - gmisclib.ortho\_poly.F (*function*), 472
  - gmisclib.ortho\_poly.Legendre (*class*), 475–476
  - gmisclib.ortho\_poly.ortho (*class*), 472–473
  - gmisclib.ortho\_poly.ortho\_poly (*class*), 473–475
  - gmisclib.ortho\_poly.SinCos (*class*), 479–480
  - gmisclib.ortho\_poly.SLTB (*class*), 480–482
  - gmisclib.ortho\_poly.test (*function*), 472
  - gmisclib.ortho\_poly.test\_a\_name (*function*), 472
- gmisclib.parse\_tree\_number (*module*), 483–502
  - gmisclib.parse\_tree\_number.abs (*function*), 483

- gmisclib.parse\_tree\_number.abstract\_number  
(*class*), 486–487
- gmisclib.parse\_tree\_number.Array (*class*),  
483–484
- gmisclib.parse\_tree\_number.coerce\_into (*func-*  
*tion*), 483
- gmisclib.parse\_tree\_number.cos (*function*),  
483
- gmisclib.parse\_tree\_number.elementof (*class*),  
497–499
- gmisclib.parse\_tree\_number.exp (*function*),  
483
- gmisclib.parse\_tree\_number.Expression (*class*),  
487–490
- gmisclib.parse\_tree\_number.Float (*class*),  
492–494
- gmisclib.parse\_tree\_number.is\_mul\_of (*func-*  
*tion*), 483
- gmisclib.parse\_tree\_number.log (*function*),  
483
- gmisclib.parse\_tree\_number.Name (*class*),  
490–492
- gmisclib.parse\_tree\_number.operator1 (*class*),  
494–497
- gmisclib.parse\_tree\_number.operatorN (*func-*  
*tion*), 483
- gmisclib.parse\_tree\_number.operatorNg1  
(*class*), 499–502
- gmisclib.parse\_tree\_number.output\_mixin  
(*class*), 484–486
- gmisclib.parse\_tree\_number.sin (*function*),  
483
- gmisclib.parse\_tree\_number.sqrt (*function*),  
483
- gmisclib.permute (*module*), 503
  - gmisclib.permute.factorial (*function*), 503
  - gmisclib.permute.next (*function*), 503
  - gmisclib.permute.permutations (*function*),  
503
- gmisclib.pylab\_oneaxis (*module*), 504
  - gmisclib.pylab\_oneaxis.plot\_oneaxis (*func-*  
*tion*), 504
- gmisclib.pylab\_starplot (*module*), 505–507
  - gmisclib.pylab\_starplot.errorbar (*class*),  
506–507
  - gmisclib.pylab\_starplot.errorbar\_maker (*class*),  
505–506
  - gmisclib.pylab\_starplot.grey\_out (*func-*  
*tion*), 505
  - gmisclib.pylab\_starplot.plot (*function*),  
505
  - gmisclib.pylab\_starplot.star (*function*),  
505
- gmisclib.read\_dicom (*module*), 508–509
  - gmisclib.read\_dicom.img\_with\_mx (*class*),  
508–509
  - gmisclib.read\_dicom.read\_body (*function*),  
508
  - gmisclib.read\_dicom.read\_header (*func-*  
*tion*), 508
  - gmisclib.read\_dicom.read\_imgs (*function*),  
508
- gmisclib.robust\_multivariate (*module*), 510
  - gmisclib.robust\_multivariate.cov\_wt (*func-*  
*tion*), 510
  - gmisclib.robust\_multivariate.covariance  
(*function*), 510
  - gmisclib.robust\_multivariate.integrate (*func-*  
*tion*), 510
  - gmisclib.robust\_multivariate.integrate\_guts  
(*function*), 510
  - gmisclib.robust\_multivariate.wtdim (*func-*  
*tion*), 510
  - gmisclib.robust\_multivariate.wtfunc (*func-*  
*tion*), 510
- gmisclib.root (*module*), 511
  - gmisclib.root.iroot (*function*), 511
  - gmisclib.root.root (*function*), 511
  - gmisclib.root.test (*function*), 511
  - gmisclib.root.testi (*function*), 511
- gmisclib.rubber\_array (*module*), 512–513
  - gmisclib.rubber\_array.ext\_block (*class*),  
512
  - gmisclib.rubber\_array.extensible\_array (*class*),  
512–513
- gmisclib.s\_lin\_fit (*module*), 514–516
  - gmisclib.s\_lin\_fit.Error (*class*), 514–515
  - gmisclib.s\_lin\_fit.NoDataError (*class*), 515–

- 516
- gmislib.s\_lin\_fit.plane (*function*), 514
- gmislib.s\_lin\_fit.test (*function*), 514
- gmislib.sbd\_array (*module*), 517–519
  - gmislib.sbd\_array.err (*function*), 517
  - gmislib.sbd\_array.multiply (*function*), 517
  - gmislib.sbd\_array.NoSolutionError (*class*), 518–519
  - gmislib.sbd\_array.sbd (*class*), 517–518
  - gmislib.sbd\_array.solve (*function*), 517
  - gmislib.sbd\_array.symmetrize (*function*), 517
  - gmislib.sbd\_array.test1 (*function*), 517
  - gmislib.sbd\_array.test2 (*function*), 517
  - gmislib.sbd\_array.testa (*function*), 517
  - gmislib.sbd\_array.testbdi (*function*), 517
  - gmislib.sbd\_array.testm (*function*), 517
- gmislib.segmentfile (*module*), 520–521
  - gmislib.segmentfile.group (*class*), 521
  - gmislib.segmentfile.parse (*function*), 520
  - gmislib.segmentfile.read (*function*), 520
  - gmislib.segmentfile.segment (*class*), 520–521
  - gmislib.segmentfile.test (*function*), 520
- gmislib.sharp\_energy (*module*), 522
  - gmislib.sharp\_energy.complex\_median (*function*), 522
  - gmislib.sharp\_energy.one\_median (*function*), 522
  - gmislib.sharp\_energy.pwr (*function*), 522
  - gmislib.sharp\_energy.stepped\_median (*function*), 522
- gmislib.solve\_sum\_abs (*module*), 523–524
  - gmislib.solve\_sum\_abs.solve1 (*function*), 523
  - gmislib.solve\_sum\_abs.solve\_fit (*function*), 523
  - gmislib.solve\_sum\_abs.solve\_fit\_wt (*function*), 523
  - gmislib.solve\_sum\_abs.test1 (*function*), 524
  - gmislib.solve\_sum\_abs.test2 (*function*), 524
- gmislib.spread\_jobs (*module*), 525–539
  - gmislib.spread\_jobs.append (*function*), 527
  - gmislib.spread\_jobs.CannotCreateConnection (*class*), 533–534
  - gmislib.spread\_jobs.Connection (*class*), 534–536
  - gmislib.spread\_jobs.Connection\_subprocess (*class*), 536–538
  - gmislib.spread\_jobs.delay\_sanitise (*function*), 527
  - gmislib.spread\_jobs.main (*function*), 527
  - gmislib.spread\_jobs.NoResponse (*class*), 528–529
  - gmislib.spread\_jobs.notComputed (*class*), 528
  - gmislib.spread\_jobs.one\_shot\_test (*function*), 528
  - gmislib.spread\_jobs.PastPerformance (*class*), 531–533
  - gmislib.spread\_jobs.RemoteException (*class*), 529–531
  - gmislib.spread\_jobs.Replace (*function*), 527
  - gmislib.spread\_jobs.replace (*function*), 527
  - gmislib.spread\_jobs.test\_ (*function*), 527
  - gmislib.spread\_jobs.test\_worker (*function*), 527
  - gmislib.spread\_jobs.TooBusy (*class*), 531
  - gmislib.spread\_jobs.unpickled\_pseudofile (*class*), 539
  - gmislib.spread\_jobs.workers\_c (*class*), 538–539
- gmislib.sqlbase (*module*), 540–553
  - gmislib.sqlbase.ColumnMismatchError (*class*), 545–546
  - gmislib.sqlbase.DB (*class*), 546–549
  - gmislib.sqlbase.DBMetaClass (*class*), 542–543
  - gmislib.sqlbase.DBx (*class*), 549–553
  - gmislib.sqlbase.get\_version (*function*), 540
  - gmislib.sqlbase.multiclass\_get\_by\_id (*function*), 540

- tion*), 541
- gmisclib.sqlbase.multiselect (*function*), 540
- gmisclib.sqlbase.NoSuchTable (*class*), 544–545
- gmisclib.sqlbase.SQLError (*class*), 543–544
- gmisclib.sqlbase.test (*function*), 541
- gmisclib.stats (*module*), 554–556
  - gmisclib.stats.betacf (*function*), 554
  - gmisclib.stats.betai (*function*), 554
  - gmisclib.stats.f\_value (*function*), 554
  - gmisclib.stats.fprob (*function*), 554
  - gmisclib.stats.gammaln (*function*), 555
  - gmisclib.stats.ltnorm (*function*), 555
  - gmisclib.stats.t\_value (*function*), 555
  - gmisclib.stats.test\_fprob (*function*), 555
- gmisclib.system\_load (*module*), 557–558
  - gmisclib.system\_load.load\_avg (*function*), 557
  - gmisclib.system\_load.load\_now (*function*), 557
  - gmisclib.system\_load.mem\_pressure (*function*), 557
  - gmisclib.system\_load.ncpu (*function*), 557
  - gmisclib.system\_load.niceness (*function*), 557
  - gmisclib.system\_load.pstat (*class*), 557–558
- gmisclib.threaded\_io (*module*), 559–566
  - gmisclib.threaded\_io.CaughtException (*class*), 559–560
  - gmisclib.threaded\_io.check\_n\_raise (*function*), 559
  - gmisclib.threaded\_io.pairmap (*function*), 559
  - gmisclib.threaded\_io.testmap (*function*), 559
  - gmisclib.threaded\_io.Thread\_pool (*class*), 561–562
  - gmisclib.threaded\_io.Thread\_poolR (*class*), 565–566
  - gmisclib.threaded\_io.Thread\_poolW (*class*), 562–565
- gmisclib.threaded\_io.threading\_with\_result (*class*), 560–561
- gmisclib.threaded\_io.to\_be\_joined (*function*), 559
- gmisclib.tsops (*module*), 567–570
  - gmisclib.tsops.apply (*function*), 568
  - gmisclib.tsops.axis (*class*), 569–570
  - gmisclib.tsops.common (*function*), 568
  - gmisclib.tsops.commonN (*function*), 568
  - gmisclib.tsops.copy\_interval (*function*), 568
  - gmisclib.tsops.interp (*function*), 567
  - gmisclib.tsops.interp\_fill (*function*), 567
  - gmisclib.tsops.interpN (*function*), 567
  - gmisclib.tsops.interpN\_fill (*function*), 567
  - gmisclib.tsops.mul (*function*), 568
  - gmisclib.tsops.resample (*function*), 568
  - gmisclib.tsops.test (*function*), 569
  - gmisclib.tsops.test\_fig (*function*), 567
  - gmisclib.tsops.test\_interp1 (*function*), 567
  - gmisclib.tsops.test\_interp2 (*function*), 567
  - gmisclib.tsops.test\_interp3 (*function*), 567
  - gmisclib.tsops.time (*function*), 567
  - gmisclib.tsops.time2 (*function*), 567
- gmisclib.wavesurfer\_lab (*module*), 571–572
  - gmisclib.wavesurfer\_lab.read (*function*), 571
  - gmisclib.wavesurfer\_lab.write (*function*), 571
- gmisclib.wavio (*module*), 573–577
  - gmisclib.wavio.BadFileFormatError (*class*), 576–577
  - gmisclib.wavio.Overflow (*class*), 575–576
  - gmisclib.wavio.read (*function*), 573
  - gmisclib.wavio.read\_hdr (*function*), 573
  - gmisclib.wavio.write (*function*), 573
- gmisclib.weighted\_percentile (*module*), 577–579
  - gmisclib.weighted\_percentile.test (*function*), 578
  - gmisclib.weighted\_percentile.test\_median (*function*), 578
  - gmisclib.weighted\_percentile.test\_wp (*function*), 578

- gmisc.lib.weighted\_percentile.wp (*function*), 577
- gmisc.lib.weighted\_percentile.wtd\_median (*function*), 577
- gmisc.lib.weighted\_percentile.wtd\_median\_across (*function*), 578
- gmisc.lib.xmlmisc (*module*), 580–582
  - gmisc.lib.xmlmisc.loc\_finder (*class*), 580–582
  - gmisc.lib.xmlmisc.test (*function*), 580
  - gmisc.lib.xmlmisc.test\_loc\_finder (*function*), 580
  - gmisc.lib.xmlmisc.treestructures (*function*), 580
- gmisc.lib.xwaves\_errs (*module*), 583–588
  - gmisc.lib.xwaves\_errs.BadFileFormatError (*class*), 585–586
  - gmisc.lib.xwaves\_errs.DataError (*class*), 586–587
  - gmisc.lib.xwaves\_errs.DataOutOfOrderError (*class*), 587–588
  - gmisc.lib.xwaves\_errs.Error (*class*), 583–584
  - gmisc.lib.xwaves\_errs.NoSuchFileError (*class*), 584–585
- gmisc.lib.xwaves\_lab (*module*), 589
  - gmisc.lib.xwaves\_lab.end\_marks (*function*), 589
  - gmisc.lib.xwaves\_lab.read (*function*), 589
  - gmisc.lib.xwaves\_lab.start\_stop (*function*), 589
  - gmisc.lib.xwaves\_lab.write (*function*), 589
- gmisc.lib.xwaves\_mark (*module*), 590
  - gmisc.lib.xwaves\_mark.combine\_2labs (*function*), 590
  - gmisc.lib.xwaves\_mark.mark\_to\_lab (*function*), 590
  - gmisc.lib.xwaves\_mark.read (*function*), 590
  - gmisc.lib.xwaves\_mark.write (*function*), 590
- gpk\_wavio (*module*), 591
- Markov Chain Monte-Carlo, 339, 341, 397, 398, 402, 403, 418, 419, 424, 425
- mcmc (*module*), 592–594
  - mcmc.def\_logger (*class*), 594
  - mcmc.def\_logger.\_\_init\_\_ (*method*), 594
  - mcmc.def\_logger.add (*method*), 594
  - mcmc.def\_logger.finish (*method*), 594
  - mcmc.go (*function*), 594
  - mcmc.logp\_from\_resid (*function*), 594
- q3html (*module*), 595–598
  - q3html.aget (*function*), 596
  - q3html.agp (*function*), 597
  - q3html.av (*function*), 596
  - q3html.del\_fin\_sl (*function*), 597
  - q3html.dequote (*function*), 596
  - q3html.dpre (*function*), 597
  - q3html.easygo (*function*), 597
  - q3html.escape (*function*), 596
  - q3html.file\_only (*function*), 597
  - q3html.format\_dict (*function*), 596
  - q3html.get (*function*), 597
  - q3html.get\_logical\_line (*function*), 596
  - q3html.go (*function*), 597
  - q3html.lineC (*class*), 597–598
    - q3html.lineC.\_\_init\_\_ (*method*), 598
    - q3html.lineC.getarg (*method*), 598
  - q3html.measure\_indent (*function*), 596
  - q3html.needcopy (*function*), 597
  - q3html.prepare\_text (*function*), 596
  - q3html.prepend (*function*), 596
  - q3html.process (*function*), 596
  - q3html.process\_tag (*function*), 596
  - q3html.qfiles (*function*), 596
  - q3html.readheader (*function*), 597
  - q3html.starts\_with\_a\_tag (*function*), 596
  - q3html.sw (*function*), 596
  - q3html.swapend (*function*), 596
  - q3html.tokenize (*function*), 596
  - q3html.urlq (*function*), 596
- run\_several (*module*), 599–600
  - run\_several.get\_ncpu (*function*), 600
  - run\_several.parse\_NP (*function*), 600
  - run\_several.run\_processes (*function*), 600
  - run\_several.set\_oom (*function*), 600
  - run\_several.wait\_until\_unloaded (*function*), 600

---

run\_several.write\_msgs (*function*), 600

select\_fiat\_entries (*module*), 601–603

- select\_fiat\_entries.avio\_read (*function*), 601
- select\_fiat\_entries.fiat\_read (*function*), 601
- select\_fiat\_entries.process (*function*), 601
- select\_fiat\_entries.writer (*class*), 601–603
  - select\_fiat\_entries.writer.globals (*method*), 602

selector, 168

xpaset, 178, 188