```
#******************************************************************************
#  $Id: gdal.py 7976 2005-08-04 19:42:09Z fwarmerdam $
#
#  Name:     gdal.py
#  Project:  GDAL Python Interface
#  Purpose:  GDAL Shadow Class Implementations
#  Author:   Frank Warmerdam, warmerdam@pobox.com
#
#******************************************************************************
#  Copyright (c) 2000, Frank Warmerdam
#
#  Permission is hereby granted, free of charge, to any person obtaining a
#  copy of this software and associated documentation files (the "Software"),
#  to deal in the Software without restriction, including without limitation
#  the rights to use, copy, modify, merge, publish, distribute, sublicense,
#  and/or sell copies of the Software, and to permit persons to whom the
#  Software is furnished to do so, subject to the following conditions:
#
#  The above copyright notice and this permission notice shall be included
#  in all copies or substantial portions of the Software.
#
#  THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS
#  OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
#  FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL
#  THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
#  LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING
#  FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER
#  DEALINGS IN THE SOFTWARE.
#******************************************************************************


def ToNULLableString(x):
def FreeNULLableString(x):

###############################################################################
# CPL Level Services

def Debug(msg_class, message):
def Error(err_class = CE_Failure, err_code = CPLE_AppDefined, msg = 'error' ):
def ErrorReset():
def GetLastErrorNo():
def GetLastErrorType():
def GetLastErrorMsg():
def PushErrorHandler( handler = "CPLQuietErrorHandler" ):
def PopErrorHandler():
def PushFinderLocation( x ):
def PopFinderLocation():
def FinderClean():
def FindFile( classname, basename ):
def SetConfigOption( name, value ):
def GetConfigOption( name, default ):
def TestBoolean( value ):
def ParseXMLString( text ):
def SerializeXMLTree( tree ):
def EscapeString( in_string, length = -1, scheme = CPLES_BackslashQuotable ):
def UnescapeString( in_string, scheme = CPLES_BackslashQuotable ):


###############################################################################
# GDAL Services not related to objects.

def GeneralCmdLineProcessor( args, options = 0 ):
def GetCacheMax():
def SetCacheMax( new_max ):
def GetCacheUsed():
def GetDataTypeSize(type):
def DataTypeIsComplex(type):
def GetDataTypeName(type):
def GetDataTypeByName(name):
def GetColorInterpretationName(type):
def GetPaletteInterpretationName(type):
def DecToDMS(angle, axis, precision = 2):
def PackedDMSToDec(packed_angle):
def DecToPackedDMS(angle):
def TermProgress( ratio, msg = '', ptr = None ):
def AllRegister():
def GetDriverList():
def GetDriverByName(name):
def Open(file,access=GA_ReadOnly):
def OpenShared(file,access=GA_ReadOnly):


###############################################################################
# Some GDAL algorithms.
```

```
 85:
 86: def ComputeMedianCutPCT( red, green, blue, color_count, ct, callback = None, callback_data = None ):
 87: def DitherRGB2PCT( red, green, blue, target, ct, callback = None, callback_data = None ):
 88: def RGBFile2PCTFile( src_filename, dst_filename ):
 89: def AutoCreateWarpedVRT( src_ds, src_wkt = None, dst_wkt = None,
 90:                          eResampleAlg = GRA_NearestNeighbour,
 91:                          maxerror = 0.0 ):
 92: def ReprojectImage( src_ds, dst_ds, src_wkt = None, dst_wkt = None,
 93:                     eResampleAlg = GRA_NearestNeighbour, warp_memory = 0.0,
 94:                     maxerror = 0.0 ):
 95: def CreateAndReprojectImage( src_ds, dst_filename, src_wkt = None, dst_wkt = None,
 96:                              dst_driver = None, create_options = [],
 97:                              eResampleAlg = GRA_NearestNeighbour,
 98:                              warp_memory = 0.0, maxerror = 0.0 ):
 99:
100:
101: ##############################################################################
102: # GCP
103:
104: class GCP:
105:     self.GCPX
106:     self.GCPY
107:     self.GCPZ
108:     self.GCPPixel
109:     self.GCPLine
110:     self.Info
111:     self.Id
112:     def __init__(self):
113:     def __str__(self):
114:     def serialize(self,with_Z=0):
115: def GCPsToGeoTransform( gcp_list, approx_ok = 1 ):
116:
117:
118: ##############################################################################
119: # MajorObject
120:
121: class MajorObject:
122:     def GetMetadata( self, domain = '' ):
123:     def SetMetadata(self, metadata, domain = ''):
124:     def GetDescription(self):
125:     def SetDescription(self, description ):
126:
127:
128: ##############################################################################
129: # Driver
130:
131: class Driver(MajorObject):
132:     self.ShortName
133:     self.LongName
134:     self.HelpTopic
135:     def __init__(self, _obj):
136:     def Create(self, filename, xsize, ysize, bands=1, datatype=GDT_Byte, options = []):
137:     def CreateCopy(self, filename, source_ds, strict=1, options=[],
138:                    callback = None, callback_data = None ):
139:     def Delete(self, filename):
140:     def Register(self):
141:     def Deregister(self):
142:
143:
144: ##############################################################################
145: # Dataset
146:
147: class Dataset(MajorObject):
148:     self.RasterXSize
149:     self.RasterYSize
150:     self.RasterCount
151:     def __init__(self, _obj):
152:     def RefreshBandInfo( self ):
153:     def __del__(self):
154:     def GetDriver(self):
155:     def GetRasterBand(self, i):
156:     def GetGeoTransform(self):
157:     def SetGeoTransform(self,transform):
158:     def SetProjection(self,projection):
159:     def GetProjection(self):
160:     def GetProjectionRef(self):
161:     def GetSubDatasets(self):
162:     def GetGCPCount(self):
163:     def GetGCPProjection(self):
164:     def GetGCPs(self):
165:     def SetGCPs(self, gcp_list, projection = '' ):
166:     def BuildOverviews(self, resampling="NEAREST", overviewlist = None,
167:                        callback = None, callback_data = None):
168:     def ReadAsArray(self, xoff=0, yoff=0, xsize=None, ysize=None):
```

```
169:     def FlushCache(self):
170:     def AddBand(self, datatype = GDT_Byte, options = [] ):
171:     def AdviseRead( self, nXOff, nYOff, nXSize, nYSize, nBufXSize = None, nBufYSize = None,
172:                     eDT = None, BandMap = None, options = [] ):
173:     def ReadRaster(self, xoff, yoff, xsize, ysize,
174:                    buf_xsize = None, buf_ysize = None, buf_type = None,
175:                    band_list = None ):
176:     def WriteRaster(self, xoff, yoff, xsize, ysize,
177:                     buf_string,
178:                     buf_xsize = None, buf_ysize = None, buf_type = None,
179:                     band_list = None ):
180:
181:
182: ##############################################################################
183: # Band
184:
185: class Band(MajorObject):
186:     self.DataType
187:     self.XSize
188:     self.YSize
189:     def __init__(self, _obj):
190:     def ReadRaster(self, xoff, yoff, xsize, ysize,
191:                    buf_xsize = None, buf_ysize = None, buf_type = None):
192:     def WriteRaster(self, xoff, yoff, xsize, ysize,
193:                     buf_string,
194:                     buf_xsize = None, buf_ysize = None, buf_type = None ):
195:     def ReadAsArray(self, xoff=0, yoff=0, win_xsize=None, win_ysize=None,
196:                     buf_xsize=None, buf_ysize=None, buf_obj=None):
197:     def WriteArray(self, array, xoff=0, yoff=0):
198:     def GetRasterColorInterpretation(self):
199:     def SetRasterColorInterpretation(self, interp):
200:     def GetRasterColorTable(self):
201:     def SetRasterColorTable(self, ct):
202:     def FlushCache(self):
203:     def GetHistogram(self, min=-0.5, max=255.5, buckets=256, include_out_of_range=0, approx_ok = 0 ):
204:     def GetDefaultHistogram( self, force = 1 ):
205:     def SetDefaultHistogram( self, min, max, histogram ):
206:     def ComputeRasterMinMax(self, approx_ok = 0):
207:     def GetStatistics( self, approx_ok = 0, force = 1 ):
208:     def GetNoDataValue(self):
209:     def SetNoDataValue(self,value):
210:     def GetMinimum(self):
211:     def GetMaximum(self):
212:     def GetOffset(self):
213:     def GetScale(self):
214:     def GetOverviewCount(self):
215:     def GetOverview(self, ov_index ):
216:     def Checksum( self, xoff=0, yoff=0, xsize=None, ysize=None ):
217:     def Fill( self, real_fill, imag_fill = 0.0 ):
218:     def ComputeBandStats( self, samplestep = 1, progress_cb = None, progress_data = None ):
219:     def AdviseRead( self, nXOff, nYOff, nXSize, nYSize, nBufXSize = None, nBufYSize = None,
220:                     eDT = None, options = [] ):
221:
222:
223: ##############################################################################
224: # ColorTable
225:
226: class ColorTable:
227:     def __init__(self, _obj = None, mode = GPI_RGB ):
228:     def __del__(self):
229:     def Clone(self):
230:     def GetPaletteInterpretation( self ):
231:     def GetCount( self ):
232:     def GetColorEntry( self, i ):
233:     def GetColorEntryAsRGB( self, i ):
234:     def SetColorEntry( self, i, color ):
235:     def __str__(self):
236:     def serialize(self):
237:
```