```
###############################################################################
#  $Id: osr.py 13451 2007-12-25 21:12:22Z mloskot $
#
#  Project:  OSR (OGRSpatialReference/CoordinateTransform) Python Interface
#  Purpose:  OSR Shadow Class Implementations
#  Author:   Frank Warmerdam, warmerdam@pobox.com
#
###############################################################################
#  Copyright (c) 2000, Frank Warmerdam <warmerdam@pobox.com>
#
#  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.
###############################################################################


SRS_PT_ALBERS_CONIC_EQUAL_AREA  = "Albers_Conic_Equal_Area"
SRS_PT_AZIMUTHAL_EQUIDISTANT  = "Azimuthal_Equidistant"
SRS_PT_CASSINI_SOLDNER    = "Cassini_Soldner"
SRS_PT_CYLINDRICAL_EQUAL_AREA = "Cylindrical_Equal_Area"
SRS_PT_ECKERT_IV    = "Eckert_IV"
SRS_PT_ECKERT_VI    = "Eckert_VI"
SRS_PT_EQUIDISTANT_CONIC  = "Equidistant_Conic"
SRS_PT_EQUIRECTANGULAR    = "Equirectangular"
SRS_PT_GALL_STEREOGRAPHIC = "Gall_Stereographic"
SRS_PT_GNOMONIC     = "Gnomonic"
SRS_PT_GOODE_HOMOLOSINE         = "Goode_Homolosine"
SRS_PT_HOTINE_OBLIQUE_MERCATOR  = "Hotine_Oblique_Mercator"
SRS_PT_HOTINE_OBLIQUE_MERCATOR_TWO_POINT_NATURAL_ORIGIN = \
    "Hotine_Oblique_Mercator_Two_Point_Natural_Origin"
SRS_PT_LABORDE_OBLIQUE_MERCATOR = "Laborde_Oblique_Mercator"
SRS_PT_LAMBERT_CONFORMAL_CONIC_1SP = "Lambert_Conformal_Conic_1SP"
SRS_PT_LAMBERT_CONFORMAL_CONIC_2SP = "Lambert_Conformal_Conic_2SP"
SRS_PT_LAMBERT_CONFORMAL_CONIC_2SP_BELGIUM = "Lambert_Conformal_Conic_2SP_Belgium)"
SRS_PT_LAMBERT_AZIMUTHAL_EQUAL_AREA = "Lambert_Azimuthal_Equal_Area"
SRS_PT_MERCATOR_1SP   = "Mercator_1SP"
SRS_PT_MERCATOR_2SP   = "Mercator_2SP"
SRS_PT_MILLER_CYLINDRICAL = "Miller_Cylindrical"
SRS_PT_MOLLWEIDE    = "Mollweide"
SRS_PT_NEW_ZEALAND_MAP_GRID     = "New_Zealand_Map_Grid"
SRS_PT_OBLIQUE_STEREOGRAPHIC    = "Oblique_Stereographic"
SRS_PT_ORTHOGRAPHIC   = "Orthographic"
SRS_PT_POLAR_STEREOGRAPHIC      = "Polar_Stereographic"
SRS_PT_POLYCONIC    = "Polyconic"
SRS_PT_ROBINSON     = "Robinson"
SRS_PT_SINUSOIDAL   = "Sinusoidal"
SRS_PT_STEREOGRAPHIC    = "Stereographic"
SRS_PT_SWISS_OBLIQUE_CYLINDRICAL= "Swiss_Oblique_Cylindrical"
SRS_PT_TRANSVERSE_MERCATOR      = "Transverse_Mercator"
SRS_PT_TRANSVERSE_MERCATOR_SOUTH_ORIENTED = "Transverse_Mercator_South_Orientated"
SRS_PT_TRANSVERSE_MERCATOR_MI_22= "Transverse_Mercator_MapInfo_22"
SRS_PT_TRANSVERSE_MERCATOR_MI_23= "Transverse_Mercator_MapInfo_23"
SRS_PT_TRANSVERSE_MERCATOR_MI_24= "Transverse_Mercator_MapInfo_24"
SRS_PT_TRANSVERSE_MERCATOR_MI_25= "Transverse_Mercator_MapInfo_25"
SRS_PT_TUNISIA_MINING_GRID  = "Tunisia_Mining_Grid"
SRS_PT_VANDERGRINTEN    = "VanDerGrinten"
SRS_PT_KROVAK     = "Krovak"

SRS_PP_CENTRAL_MERIDIAN         = "central_meridian"
SRS_PP_SCALE_FACTOR             = "scale_factor"
SRS_PP_STANDARD_PARALLEL_1      = "standard_parallel_1"
SRS_PP_STANDARD_PARALLEL_2      = "standard_parallel_2"
SRS_PP_PSEUDO_STD_PARALLEL_1    = "pseudo_standard_parallel_1"
SRS_PP_LONGITUDE_OF_CENTER      = "longitude_of_center"
SRS_PP_LATITUDE_OF_CENTER       = "latitude_of_center"
SRS_PP_LONGITUDE_OF_ORIGIN      = "longitude_of_origin"
SRS_PP_LATITUDE_OF_ORIGIN       = "latitude_of_origin"
SRS_PP_FALSE_EASTING            = "false_easting"
SRS_PP_FALSE_NORTHING           = "false_northing"
SRS_PP_AZIMUTH                  = "azimuth"
```

```
SRS_PP_LONGITUDE_OF_POINT_1     = "longitude_of_point_1"
SRS_PP_LATITUDE_OF_POINT_1      = "latitude_of_point_1"
SRS_PP_LONGITUDE_OF_POINT_2     = "longitude_of_point_2"
SRS_PP_LATITUDE_OF_POINT_2      = "latitude_of_point_2"
SRS_PP_LONGITUDE_OF_POINT_3     = "longitude_of_point_3"
SRS_PP_LATITUDE_OF_POINT_3      = "latitude_of_point_3"
SRS_PP_RECTIFIED_GRID_ANGLE     = "rectified_grid_angle"
SRS_PP_LANDSAT_NUMBER           = "landsat_number"
SRS_PP_PATH_NUMBER              = "path_number"
SRS_PP_PERSPECTIVE_POINT_HEIGHT = "perspective_point_height"
SRS_PP_FIPSZONE                 = "fipszone"
SRS_PP_ZONE                     = "zone"

SRS_UL_METER      = "Meter"
SRS_UL_FOOT     = "Foot (International)"
SRS_UL_FOOT_CONV                = "0.3048"
SRS_UL_US_FOOT      = "U.S. Foot"
SRS_UL_US_FOOT_CONV             = "0.3048006"
SRS_UL_NAUTICAL_MILE    = "Nautical Mile"
SRS_UL_NAUTICAL_MILE_CONV       = "1852.0"
SRS_UL_LINK     = "Link"
SRS_UL_LINK_CONV                = "0.20116684023368047"
SRS_UL_CHAIN      = "Chain"
SRS_UL_CHAIN_CONV               = "2.0116684023368047"
SRS_UL_ROD      = "Rod"
SRS_UL_ROD_CONV                 = "5.02921005842012"

SRS_DN_NAD27      = "North_American_Datum_1927"
SRS_DN_NAD83      = "North_American_Datum_1983"
SRS_DN_WGS72      = "WGS_1972"
SRS_DN_WGS84      = "WGS_1984"

SRS_WGS84_SEMIMAJOR             = 6378137.0
SRS_WGS84_INVFLATTENING         = 298.257223563


############################################################################
# Various free standing functions.

def GetProjectionMethods():
def GetWellKnownGeogCSAsWKT( name ):
def GetUserInputAsWKT( user_def ):


############################################################################
# SpatialReference

class SpatialReference:
    def __init__(self,obj=None, wkt=None):
    def __del__(self):
    def Reference( self ):
    def Dereference( self ):
    def ImportFromWkt( self, wkt ):
    def ImportFromProj4( self, proj4 ):
    def ImportFromESRI( self, prj_lines ):
    def ImportFromPCI( self, proj, units = "METRE", proj_parms = None ):
    def ImportFromUSGS( self, proj_code, zone=0, proj_parms=(), datum_code=0 ):
    def ImportFromXML( self, xml ):
    def ExportToWkt(self):
    def ExportToPrettyWkt(self,simplify=0):
    def ExportToProj4(self):
    def ExportToPCI(self):
    def ExportToUSGS(self):
    def ExportToXML( self, dialect = '' ):
    def CloneGeogCS(self):
    def Clone(self):
    def Validate(self):
    def StripCTParms(self):
    def FixupOrdering(self):
    def Fixup(self):
    def MorphToESRI(self):
    def MorphFromESRI(self):
    def ImportFromEPSG(self,code):
    def IsGeographic(self):
    def IsProjected(self):
    def IsLocal(self):
    def GetAttrValue(self, name, child = 0):
    def SetAttrValue(self, name, value):
    def SetWellKnownGeogCS(self, name):
    def SetFromUserInput(self, name):
    def CopyGeogCSFrom( self, src_srs ):
    def SetTOWGS84( self, p1, p2, p3, p4=0.0, p5=0.0, p6=0.0, p7=0.0 ):
    def GetTOWGS84( self ):
    def SetGeogCS( self, geog_name, datum_name, ellipsoid_name, semi_major, inv_flattening,
```

```
                    pm_name = 'Greenwich', pm_offset = 0.0,
                    units = 'degree', conv_to_radian = 0.0174532925199433 ):
    def SetProjCS(self, name = "unnamed" ):
    def IsSameGeogCS(self, other):
    def IsSame(self, other):
    def GetSemiMajor(self):
    def GetSemiMinor(self):
    def GetInvFlattening(self):
    def SetAngularUnits(self, units_name, to_radians ):
    def GetAngularUnits( self ):
    def SetLinearUnits(self, units_name, to_meters ):
    def GetLinearUnits( self ):
    def GetLinearUnitsName( self ):
    def SetAuthority( self, target_key, authority_name, authority_code ):
    def GetAuthorityCode( self, target_key ):
    def GetAuthorityName( self, target_key ):
    def SetUTM(self, zone, is_north = 1):
    def SetStatePlane(self, zone, is_nad83 = 1, overrideunitsname='', overrideunits = 0.0 ):
    def AutoIdentifyEPSG( self ):
    def SetAttrValue( self, node_path, value ):
    def SetProjection( self, name ):
    def SetProjParm( self, name, value ):
    def GetProjParm( self, name, default_val = 0.0 ):
    def SetNormProjParm( self, name, value ):
    def GetNormProjParm( self, name, default_val = 0.0 ):
    def __str__( self ):
    def SetACEA( self, stdp1, stdp2, clat, clong, fe, fn ):
    def SetAE( self, clat, clong, fe, fn ):
    def SetCEA( self, stdp1, cm, fe, fn ):
    def SetCS( self, clat, clong, fe, fn ):
    def SetBonne( self, clat, clong, fe, fn ):
    def SetEC( self, stdp1, stdp2, clat, clong, fe, fn ):
    def SetEckertIV( self, cm, fe, fn ):
    def SetEckertVI( self, cm, fe, fn ):
    def SetEquirectangular( self, clat, clong, fe, fn ):
    def SetGS( self, cm, fe, fn ):
    def SetGH( self, cm, fe, fn ):
    def SetGnomonic( self, clat, clong, fe, fn ):
    def SetHOM( self, clat, clong, azi, recttoskew, scale, fe, fn ):
    def SetHOM2PNO( self, clat, lat1, long1, lat2, long2, scale, fe, fn ):
    def SetKrovak( self, clat, clong, azi, pstdparlat, scale, fe, fn ):
    def SetLAEA( self, clat, clong, fe, fn ):
    def SetLCC( self, stdp1, stdp2, clat, clong, fe, fn ):
    def SetLCCB( self, stdp1, stdp2, clat, clong, fe, fn ):
    def SetLCC1SP( self, clat, clong, scale, fe, fn ):
    def SetMC( self, clat, clong, fe, fn ):
    def SetMercator( self, clat, clong, scale, fe, fn ):
    def SetMollweide( self, cm, fe, fn ):
    def SetNZMG( self, clat, clong, fe, fn ):
    def SetOS( self, olat, cm, fe, fn ):
    def SetOrthographic( self, clat, clong, fe, fn ):
    def SetPolyconic( self, clat, clong, fe, fn ):
    def SetPS( self, clat, clong, scale, fe, fn ):
    def SetRobinson( self, clong, fe, fn ):
    def SetSinusoidal( self, clong, fe, fn ):
    def SetStereographic( self, clat, clong, scale, fe, fn ):
    def SetSOC( self, lato, cm, fe, fn ):
    def SetTM( self, clat, clong, scale, fe, fn ):
    def SetTMSO( self, clat, clong, scale, fe, fn ):
    def SetTMG( self, clat, clong, fe, fn ):
    def SetVDG( self, clong, fe, fn ):


############################################################################
# CoordinateTransformation

class CoordinateTransformation:
    def __init__(self,source,target = None):
    def __del__(self):
    def TransformPoint(self, x, y, z = 0):
    def TransformPoints(self, points):

```