Podstawy programowania

Tworzenie i analiza danych wektorowych w OGR

Metody filtrowania danych wektorowych

- Filtrowanie na podstawie atrybutów
- Filtrowanie na podstawie geometrii
- Filtrowanie przy pomocy zapytań SQL

Filtrowanie atrybutów w OGR

```
>>> ds = ogr.Open(r'C:/GIS')
>>> country_lyr = ds.GetLayer('world')
#ustawienie filtru
>>> country_lyr.SetAttributeFilter("continent = 'Asia'")
0
#obliczenie ilości obiektów po nałożeniu filtru
>>> country_lyr.GetFeatureCount()
53
```

Filtrowanie na podstawie geometrii obiektów

```
>>> ds = ogr.Open(r'C:/GIS')
>>> country_lyr = ds.GetLayer('world')
>>> country_lyr.SetAttributeFilter("name = 'Germany'")
>>> feat = country_lyr.GetNextFeature()
#pobranie geometrii filtrującej
>>> germany = feat.geometry().Clone()
>>> city_lyr=ds.GetLayer('world_cities')
#ilość wszystkich obiektów w warstwie filtrowanej
>>> city_lyr.GetFeatureCount()
1249
#zastosowanie filtru przestrzennego
>>> city_lyr.SetSpatialFilter(germany)
>>> city_lyr.GetFeatureCount()
5
```

Przesuwanie wskaźnika obiektów i klonowanie geometrii

```
>>> ds = ogr.Open(r'D:\data_store')
>>> lyr = ds.GetLayer('world')
>>> feat = lyr.GetNextFeature()
>>> geom = feat.geometry()
>>> geom_clone = feat.geometry().Clone()
>>> feat = lyr.GetNextFeature()
>>> print(geom_clone.GetArea())
0.012863109111786897
#instrukcja powodująca błąd interpretera – nie ma już takiej geometrii
>>> print(geom.GetArea())
```

Filtrowanie przy pomocy SQL

```
>>> ds = ogr.Open(r'C:/GIS')
sql = 'SELECT ogr_geom_area as area, name, pop_est FROM  world ORDER BY
    POP_EST DESC'
lyr = ds.ExecuteSQL(sql)
#złożone zapytanie
sql = 'SELECT pp.name AS city, pp.pop_min AS city_pop, c.name AS country,
    c.pop_est AS country_pop FROM world_cities pp LEFT JOIN world c ON
    pp.adm0_a3 = c.adm0_a3 WHERE pp.adm0cap = 1'
lyr = ds.ExecuteSQL(sql)
```

Filtrowanie i zapis wyników do nowej warstwy na podstawie atrybutów

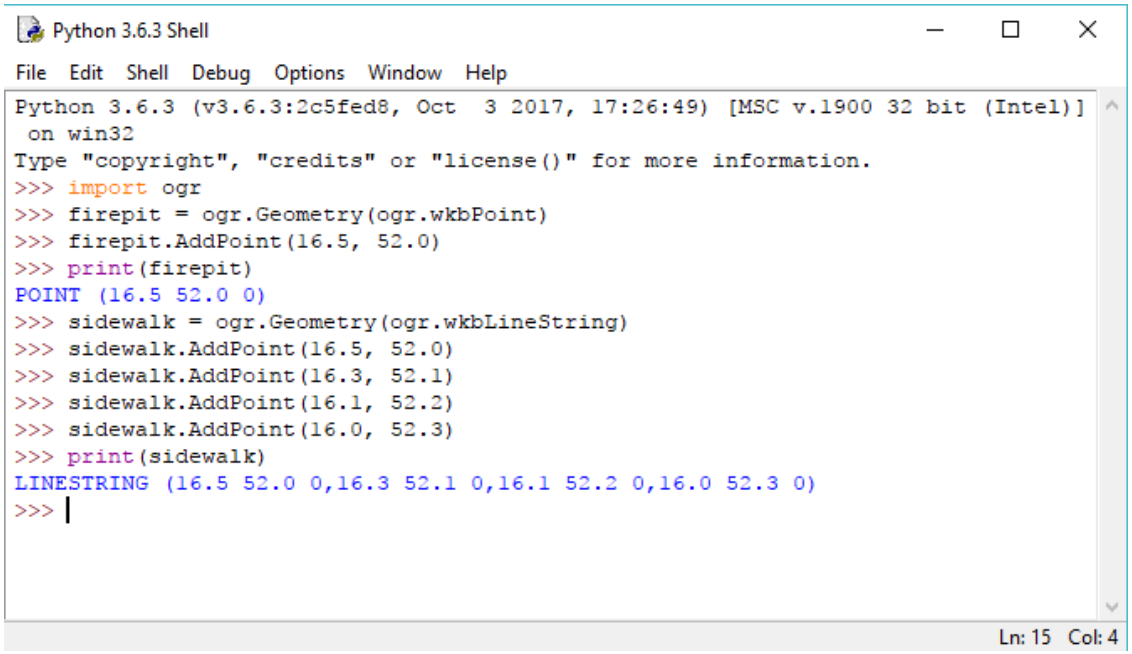
```
ds = ogr.Open(r'C:/GIS',1)
#otwarcie pliku world_cities
in_lyr = ds.GetLayer('world_cities')

#nałożenie filtru atrybutowego ograniczającego tylko do stolic
in_lyr.SetAttributeFilter("FEATURECLA = 'Admin-0 capital'")

#skopiowanie stolic do nowego pliku w obiekcie out_lyr
out_lyr = ds.CopyLayer(in_lyr, 'capital_cities')
```

Filtrowanie i zapis wyników do nowej warstwy na postawie zapytania sql

```
ds = ogr.Open(r'C:/GIS',1)
#zapytanie sql odfiltrowujące

sql = "SELECT NAME, ADM0NAME FROM world_cities WHERE FEATURECLA =
    'Admin-0 capital'"

#wykonanie zapytania na poziomie źródła danych
in_lyr= ds.ExecuteSQL(sql)

#sprawdzenie zawartości warstwy
feat=in_lyr.GetNextFeature()
print(feat.name)

#skopiowanie odfiltrowanej warstwy do nowego pliku
out_lyr = ds.CopyLayer(in_lyr, 'capital_cities')
```

Tworzenie geometrii: punkty

```
>>> firepit = ogr.Geometry(ogr.wkbPoint)
>>> firepit.AddPoint(16.5, 52.0)
>>> print(firepit)
POINT (16.5 52.0 0)
```



Tworzenie geometrii: linie

```
>>> sidewalk = ogr.Geometry(ogr.wkbLineString)
>>> sidewalk.AddPoint(16.5, 52.0)
>>> sidewalk.AddPoint(16.3, 52.1)
>>> sidewalk.AddPoint(16.1, 52.2)
>>> sidewalk.AddPoint(16.0, 52.3)
>>> print(sidewalk)

LINESTRING (16.5 52.0 0,16.3 52.1 0,16.1 52.2 0,16.0 52.3 0)
```

Tworzenie geometrii: poligony

```
>>> ring = ogr.Geometry(ogr.wkbLinearRing)
>>> ring.AddPoint(16.5, 52.0)
>>> ring.AddPoint(16.4, 52.1)
>>> ring.AddPoint(16.3, 52.2)
>>> ring.AddPoint(16.2, 52.3)
>>> yard = ogr.Geometry(ogr.wkbPolygon)
>>> yard.AddGeometry(ring)
>>> yard.CloseRings()
>>> print(yard)
POLYGON ((16.5 52.0 0,16.4 52.1 0,16.3 52.2 0,16.2 52.3 0,16.5
    52.0 0))
```

Przykład: zapis geometrii do nowego pliku

```
import ogr, osr
#osr – biblioteka obsługi układów współrzędnych
 #otwarcie katalogu do edycji
ds = ogr.Open(r'C:/GIS',1)

#sprawdzenie czy istnieje warstwa i skasowanie jeżeli istnieje
if ds.GetLayer('points'):
        ds.DeleteLayer('points')

#definicja obiektu nowej projekcji
spatialRef = osr.SpatialReference()
#pobranie informacji o projekcji z kodu EPSG
spatialRef.ImportFromEPSG(4326)

#utworzenie warstwy docelowej
out_lyr = ds.CreateLayer('points', spatialRef, ogr.wkbPoint)

#dodanie pola do warstwy points
field_name = ogr.FieldDefn("ID", ogr.OFTString)
field_name.SetWidth(30)
out_lyr.CreateField(field_name)
```

Przykład: zapis geometrii do nowego pliku c.d.

```
#utworzenie pustego obiektu out_feat do zapisu
out_defn = out_lyr.GetLayerDefn()
out_feat = ogr.Feature(out_defn)

#utworzenie nowej geometrii i wstawienie do obiektu warstwy
geom = ogr.Geometry(ogr.wkbPoint)
geom.AddPoint(16.5,52.0)
out_feat.SetGeometry(geom)

#dodanie pola
out_feat.SetField('ID','city')

#zapis obiektu do warstwy
out_lyr.CreateFeature(out_feat)
```

Kasowanie obiektu z warstwy na podstawie atrybutów

```
#kasowanie obiektów w warstwie wykonuje się na podstawie ID pola (FID)

for feat in lyr:
    if feat.GetField('City_Name') == 'Seattle':
        lyr.DeleteFeature(feat.GetFID())
```

Edycja warstwy: dodanie nowego pola

```
#otwarcie katalogu do edycji
ds = ogr.Open(r'C:/GIS',1)
#pobranie warstwy landuse
landuse_lyr = ds.GetLayer('landuse')

#utworzenie nowego pola
landuse_lyr.CreateField(ogr.FieldDefn("area", ogr.OFTReal))

#wpis wartości area do nowego pola dla wszystkich obiektów
for feat in landuse_lyr:
    geom=feat.geometry()
    area=geom.GetArea() #obliczenie wartości powierzchni
    feat.SetField('area', area)
    landuse_lyr.SetFeature(feat)
```

Typy danych dla atrybutów

• Integer:	OFTInteger
• List of integers:	OFTIntegerList
• Floating point number:	OFTReal
• List of floating point numbers:	OFTRealList
• String:	OFTString
• List of strings:	OFTStringList
• Date:	OFTDate
• Time of day:	OFTTime
• Date and time:	OFTDateTime

Funkcja zmiany definicji pola warstwy

layer.AlterFieldDefn(iField, field_def, nFlags)

iField – index pola do zmiany

field_def – obiekt nowej definicji pola

nFlags – stała wartość typu zmiany

nFlags:

- Zmiana nazwy pola: ALTER_NAME_FLAG
- Zmiana typu pola: ALTER_TYPE_FLAG
- Zmiana długości i precyzji zmiennej pola ALTER_WIDTH_PRECISION_FLAG
- Zmiana wszystkich powyższych: ALTER_ALL_FLAG

Przykład zmiany definicji pola warstwy

```
#pobranie indeksu pola
i = lyr.GetLayerDefn().GetFieldIndex('Name')
#utworzenie nowej definicji pola
fld_defn = ogr.FieldDefn('City_Name', ogr.OFTString)
#zmiana pola
lyr.AlterFieldDefn(i, fld_defn, ogr.ALTER_NAME_FLAG)
```

Przykład: tworzenie danych wektorowych na podstawie danych z pliku CSV

```
import ogr, osr
#otwarcie katalogu do edycji
ds=ogr.Open(r'C:\GIS',1)
#kontrola warstwy w katalogu
if ds.GetLayer('rtk_points'):
        ds.DeleteLayer('rtk_points')
#definicja ścieżki do pliku csv
csv_fn=r'C:\GIS\GI22.csv'
#definicja nowej projekcji
spatialRef = osr.SpatialReference()
spatialRef.ImportFromEPSG(2177)

#utworzenie warstwy docelowej
out_lyr = ds.CreateLayer('rtk_points', spatialRef, ogr.wkbPoint)
#dodanie pól do warstwy
field_name = ogr.FieldDefn("ID", ogr.OFTString)
field_name.SetWidth(30)
out_lyr.CreateField(field_name)
out_lyr.CreateField(ogr.FieldDefn("Z", ogr.OFTReal))
```

Przykład: tworzenie danych wektorowych na podstawie danych z pliku CSV c.d.

```
#utworzenie pustego obiektu
out_defn = out_lyr.GetLayerDefn()
out_feat = ogr.Feature(out_defn)

#otwarcie pliku csv do odczytu
csv_ds = ogr.Open(csv_fn)
csv_lyr = csv_ds.GetLayer()

#odczyt iwstawienie danych do pliku shapefile
for csv_row in csv_lyr:
        x = csv_row.GetFieldAsDouble('x')
        y = csv_row.GetFieldAsDouble('y')
        geom = ogr.Geometry(ogr.wkbPoint)
        geom.AddPoint(x, y)
        z=csv_row.GetField('z')
        id=csv_row.GetField('id')
        out_feat.SetGeometry(geom)
        out_feat.SetField('Z',z)
        out_feat.SetField('ID',id)
        out_lyr.CreateFeature(out_feat)
```

Reprojekcja danych

```
>>> import ogr, osr
>>> source = osr.SpatialReference()
>>> source.ImportFromEPSG(4326)
>>> target = osr.SpatialReference()
>>> target.ImportFromEPSG(2180)

>>> transform = osr.CoordinateTransformation(source, target)
>>> point = ogr.CreateGeometryFromWkt("POINT (16.5 52.0)")
>>> point.Transform(transform)

>>> print (point.ExportToWkt())
POINT (328438.280810746 462259.477835927)
```

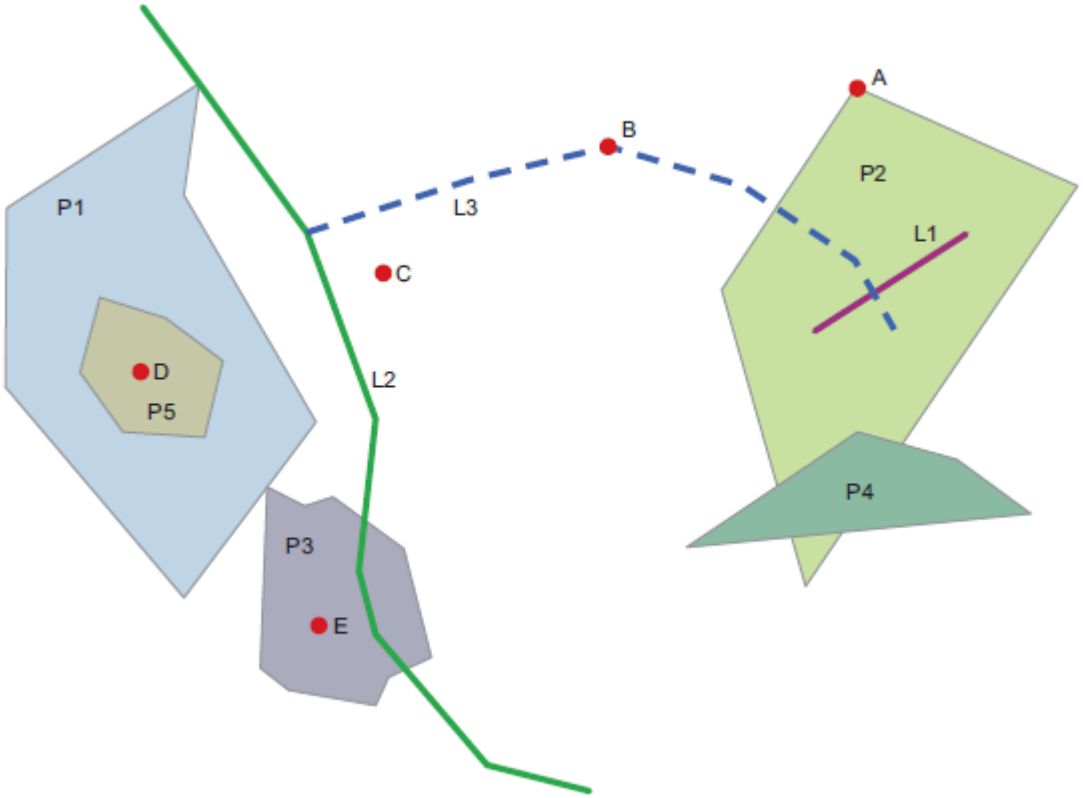
Przykład: transformacja geometrii do nowego układu współrzędnych: osr

```
import ogr, osr
#zmienna przechowująca informacje o definicji układu współrzędnych
sr = osr.SpatialReference()
#definicja nowego układu współrzędnych
sr.ImportFromEPSG(4326)

ds = ogr.Open(r'C:/GIS/osm_landuse',1)
in_lyr = ds.GetLayer('lasy')
if ds.GetLayer('lasy4326'):
   ds.DeleteLayer('lasy4326')

out_lyr = ds.CreateLayer('lasy4326', sr, ogr.wkbPolygon)
out_lyr.CreateFields(in_lyr.schema)
out_feat = ogr.Feature(out_lyr.GetLayerDefn())

for in_feat in in_lyr:
        geom = in_feat.geometry().Clone()
        geom.TransformTo(sr) #uruchomienie funkcji transformującej do nowego układu współrzędnych
        out_feat.SetGeometry(geom)
        for i in range(in_feat.GetFieldCount()):
                out_feat.SetField(i, in_feat.GetField(i))
                out_lyr.CreateFeature(out_feat)
```

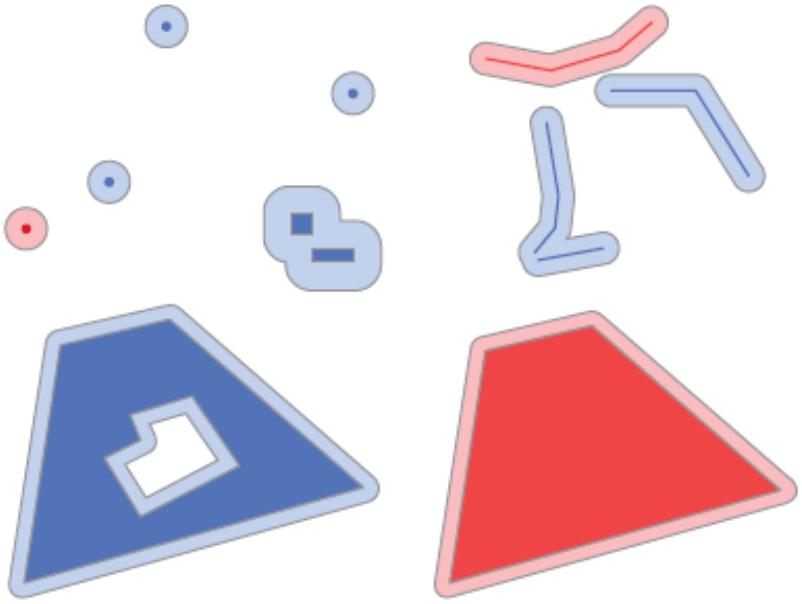
Analizy danych wektorowych w OGR

- Analiza relacji przecinania
- Analiza buforowa

Analizy OGR: analiza relacji przecinania



Analizy z OGR: analiza buforowa



Przykład I: analiza przecinania

Obliczyć sumę powierzchni obiektów danego typu (np. lasów) w ramach innego obiektu (np. granice miasta)

```
#otwarcie katalogu do odczytu danych wektorowych
ds=ogr.Open(r"C:/GIS")

#otwarcie warstwy landuse
land_lyr=ds.GetLayer('landuse')

#ustawienie filtru atrybutowego
land_lyr.SetAttributeFilter("fclass='forest'")

#otwarcie warstwy gminy_wielkopolska
poz_lyr= ds.GetLayer('gminy_wielkopolska')

#ustawienie filtru atrybutowego
poz_lyr.SetAttributeFilter("jpt_nazwa_='Poznań'")

#uruchomienie wskaźnika obiektów warstwy
poz_feat=poz_lyr.GetNextFeature()

#pobranie geometrii dla obiektu Poznan
poz_geom=poz_feat.geometry().Clone()

#nałożenie filtra przestrzennego na warstwę landuse
land_lyr.SetSpatialFilter(poz_geom)
```

Przykład I: analiza przecinania

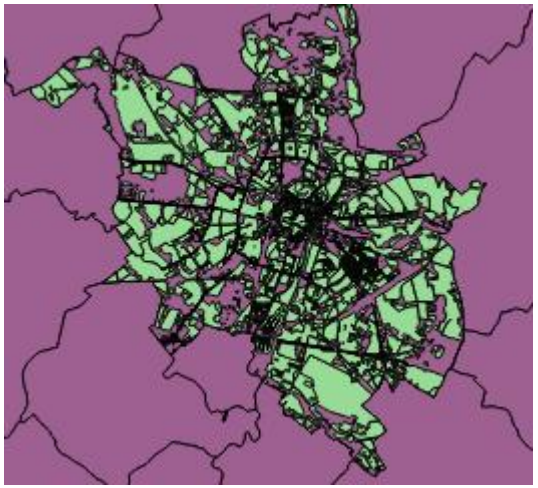
```
#utworzenie zmiennej do sumowania wartości pola
calc_area = 0
#wykonanie w pętli obliczeń powierzchni przecinającej się z obszarem miasta
for feat in land_lyr:
    intersect=feat.geometry().Intersection(poz_geom)
    calc_area = calc_area+ intersect.GetArea()

#przeliczenie wyników na procenty
pcnt = calc_area /poz_geom.GetArea()

#wyświetlenie wyników
print('{:.1%} powierzchni miasta to lasy'.format(pcnt))
```

Przykład II: przycinanie warstw

```
#skrypt wykonujący przycinanie warstwy budynki do obszaru miasta
ds=ogr.Open(r'C:/GIS',1)
clip_lyr=ds.GetLayer('gminy_wielkopolska')
#wybranie geometrii granic miasta
clip_lyr.SetAttributeFilter("jpt_nazwa_ = 'Poznań'")
clip_feat=clip_lyr.GetNextFeature()
clip_geom=clip_feat.geometry().Clone()

in_lyr=ds.GetLayer('landuse')
in_lyr.SetSpatialFilter(clip_geom)

if ds.GetLayer('landuse_poznan'):
        ds.DeleteLayer('landuse_poznan')

#definicja nowej warstwy
out_lyr = ds.CreateLayer('landuse_poznan', in_lyr.GetSpatialRef(), ogr.wkbMultiPolygon)
```

Przykład II: przycinanie warstw

```
#przepisanie struktury pól z warstwy źródłowej
out_lyr.CreateFields(in_lyr.schema)

#utworzenie pustego obiektu do wpisywania danych
out_defn = out_lyr.GetLayerDefn()
out_feat = ogr.Feature(out_defn)

#przycinanie i tworzenie nowej warstwy
for feat in in_lyr:
    geom=feat.geometry().Intersection(clip_geom)
    out_feat.SetGeometry(geom)
    for i in range(feat.GetFieldCount()):
        value = feat.GetField(i)
        out_feat.SetField(i, value)
    out_lyr.CreateFeature(out_feat)
```



Przykład III: analiza buforowa

```
#Obliczyć ilość obiektów wiosek znajdujących się  w strefie bufora cieków
      w zasięgu 1000 m

ds=ogr.Open(r"C:/GIS")
rzeki_lyr=ds.GetLayer('rzeki_wp')
points_lyr=ds.GetLayer('osm_points_poznan')
points_lyr.SetAttributeFilter("place='village'")

memory_driver = ogr.GetDriverByName('memory')
#utworzenie miejsca w pamięci na dane
memory_ds = memory_driver.CreateDataSource('temp')

#utworzenie w pamięci warstwy 'buffer'
buff_lyr = memory_ds.CreateLayer('buffer')
buff_feat = ogr.Feature(buff_lyr.GetLayerDefn())
```

Przykład III: analiza buforowa

```
#tworzenie geometrii buforu 1000 m dla rzek
for feat in rzeki_lyr:
    buff_geom = feat.geometry().Buffer(1000)
    tmp = buff_feat.SetGeometry(buff_geom)
    tmp = buff_lyr.CreateFeature(buff_feat)

result_lyr = memory_ds.CreateLayer('result')
#przecinanie obszaru buforu z warstwą punktową
buff_lyr.Intersection (points_lyr, result_lyr)

print('Wioski: {} w granicach buforu'.format(result_lyr.GetFeatureCount()))
```

Inne Biblioteki GeoPythona

- Shapely
- Psypog2
- Pandas i Geopandas

Biblioteka shapely

Biblioteka służąca do przetwarzania i analizowania obiektów geometrycznych w formacie well-known text

https://toblerity.org/shapely/manual.html

```
from shapely.geometry import Polygon
from shapely.geometry import Point
from shapely.geometry import LineString

polygon = Polygon([(0, 0), (0, 1), (1, 1),(1, 0)])
polygon.area #obliczenie powierzchni
polygon.length #długość obwiedni
polygon.bounds #granice obwiedni
polygon.geom_type
Point(0,0).distance(Point(1,1)) #odległość
line = LineString([(0, 0), (1, 1)])
line.length
```

Shapely: analiza geometrii

```
import ogr, shapely.wkt

ds=ogr.Open(r'C:/GIS',1)
lyr=ds.GetLayer('gminy_wielkopolska')
lyr.SetAttributeFilter("jpt_nazwa_ = 'Poznań'")
feat = lyr.GetNextFeature()

geom = feat.GetGeometryRef()

wkt = geom.ExportToWkt()
#załadowanie do obiektu shapely
outline = shapely.wkt.loads(wkt)
#odczyt właściwości
print (outline.area, outline.length)
print(outline.bounds, outline.geom_type)
print (outline.centroid.x, outline.centroid.y)
```

Shapely – funkcja interpolate

```
>>> from shapely.geometry import Point, LineString
>>> line = LineString([(0, 0), (1, 1)])
>>> print line
LINESTRING (0 0, 1 1)
>>> point=Point(0,1)
>>> print point
POINT (0 1)
>>> pt_interpolate=line.interpolate(line.project(point))
>>> print pt_interpolate
POINT (0.5 0.5)
>>> point.distance(pt_interpolate)
0.7071067811865476
>>> pt_025=line.interpolate(0.25)
>>> print pt_025
POINT (0.1767766952966369 0.1767766952966369)
```

Biblioteka Psycopg2-komunikacja z bazą danych: import shapefile do bazy danych

```
import ogr, psycopg2
#psycopg2 –biblioteka przeznaczona do komunikacji z bazą danych PostgreSQL
#nawiązanie połączenia z bazą danych
connection = psycopg2.connect(database="wielkopolska", user="postgres", password="haslo")
#utworzenie obiektu cursor służącego do wykonywania zapytań sql
cursor = connection.cursor()
#wysłanie zapytania kasowania tabeli
cursor.execute("DROP TABLE IF EXISTS gis.lasy")
#wysłanie zapytania utworzenie nowej tabeli
cursor.execute("CREATE TABLE gis.lasy (" +
      "gid SERIAL PRIMARY KEY," +
      "typ VARCHAR," +
      "nazwa VARCHAR," +
      "geom geometry(multipolygon,2180))")
#wysłanie zapytania nałożenie indeksu przestrzennego
cursor.execute("CREATE INDEX lasy_index ON gis.lasy USING GIST(geom)")
```

Biblioteka Psycopg2-komunikacja z bazą danych: Import shapefile do PostGIS

```
#odczyt warstwy shapefile
ds = ogr.Open(r'C:/GIS/osm_landuse',1)
layer = ds.GetLayer('landuse')
layer.SetAttributeFilter("fclass='forest'")

for feat in layer:
    fclass = feat.GetField("fclass")
    name = feat.GetField("name")
    geometry = feat.GetGeometryRef()
    wkt = geometry.ExportToWktgeometrii do postaci wkt
    #wysłanie zapytania wstawianie danych do ()#przepisanie tabeli
    cursor.execute("INSERT INTO gis.lasy (typ, nazwa, geom) VALUES (%s, %s,
    ST_Multi(ST_GeomFromText(%s,2180)))", (fclass, name, wkt))

#zatwierdzenie zmian w bazie danych
connection.commit()
```

Biblioteka Psycopg2-odczyt z bazy danych PostGIS

```
import ogr, psycopg2

#nawiązanie połączenia
connection = psycopg2.connect(database="wielkopolska", user="postgres",
    password="haslo")
cursor = connection.cursor()

#wykonanie zapytania
cursor.execute("SELECT gid, name, ST_Area(geom) as area FROM gis.landuse
    ORDER BY name LIMIT 10")
#wyświetlenie wyników
for gid, name, area in cursor:
    print(gid, name, area)
```