Podstawy programowania

Wstęp do geoprocesingu w Pythonie

Moduły Pythona związane z geoproscessingiem

http://pypi.python.org/pypi

gdal /ogr– biblioteka odczytu i zapisu danych przestrzennych

shapely – biblioteka do przetwarzania i analizowania obiektów geometrycznych

pyshp- obsługa plików Esri Shapefile

geojson – obsługa formatu GeoJSON

PyQGIS – biblioteka Pythona w QGIS

ArcPy –biblioteka Pythona w ArcGIS

Biblioteka GDAL/OGR

- GDAL - Geospatial Data Abstraction Library
  biblioteka służąca do odczytu i zapisu rastrowych danych przestrzennych. GDAL używa OGR do odczytu i zapisu wektorowych danych oraz OSR do definiowania układów współrzędnych. Projekcje układów współrzędnych są wspierane przez bibliotekę PROJ.4.

Literatura

- https://pcjericks.github.io/py-gdalogr-cookbook
- Garrard C. 2016. Geoprocessing with Python. Meap Edition.
- Westra E. 2015. Python Geospatial Analysis Essentials. Packt Publishing.

Dodanie biblioteki GDAL i Shapely

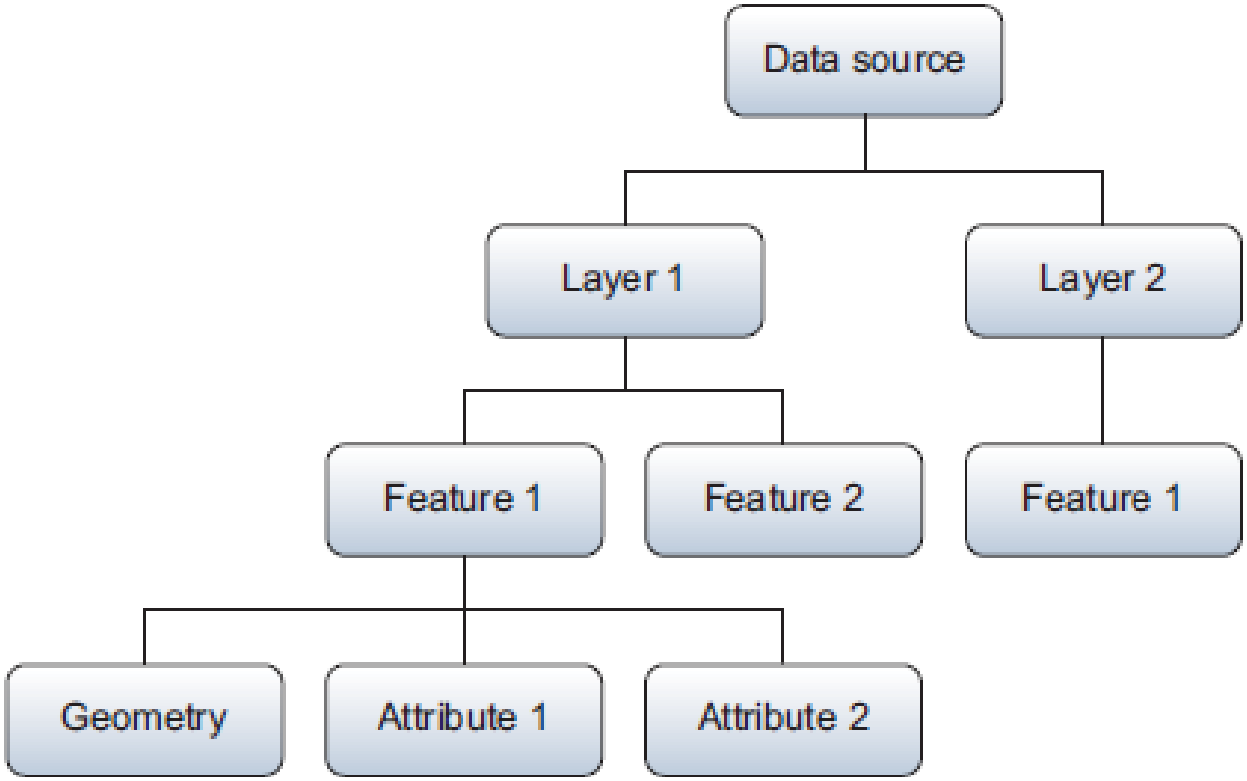
https://pythongisandstuff.wordpress.com/2016/04/13/installing-gdal-ogr-for-python-on-windows

- Pobrać ze strony wybraną bibliotekę zgodną z wersją Pythona np. cp27 i architekturą systemu np. win32 z linku:
- http://www.lfd.uci.edu/~gohlke/pythonlibs/#gdal
- W cmd wydać polecenie
  pip install GDAL-2.0.2-cp27-none-win32.whl
  podając ścieżkę do pliku
- Pobrać z linku zainstalować w systemie odpowiednią wersję
  https://pypi.python.org/pypi/Shapely/1.4.0
- W cmd wydać polecenie python, a następnie
  import gdal
  import ogr
  import shapely

Biblioteka OGR

- Biblioteka stanowiąca część GDAL zawierająca funkcje pozwalające odczytywać i zapisywać dowolne formaty wektorowe
- Pozwala tworzyć i przetwarzać obiekty geometryczne oraz ich atrybuty, filtrować i analizować dane na podstawie wartości atrybutów a także relacji przestrzennych

Struktura danych w OGR



Odczyt pliku wektorowego

```
import ogr
#otwarcie katalogu z warstwami
ds=ogr.Open(r'C:/Katalog_GIS')
#odczyt warstwy
layer = ds.GetLayer('landuse')
#odczyt obiektów w warstwie w pętli for
for feat in layer:
    feat_name = feat.GetField('name')
    geom = feat.geometry()
    geom_type = geometry.GetGeometryName()
    print ( feat_name, geom_type)
```

Sterowniki OGR

```
#ustawienie typu sterownika
driver = ogr.GetDriverByName('GeoJSON')
driver = ogr.GetDriverByame('ESRI Shapefile')
ds = driver.Open(r'C:\myshp.shp',0)
```

Lista obsługiwanych formatów:

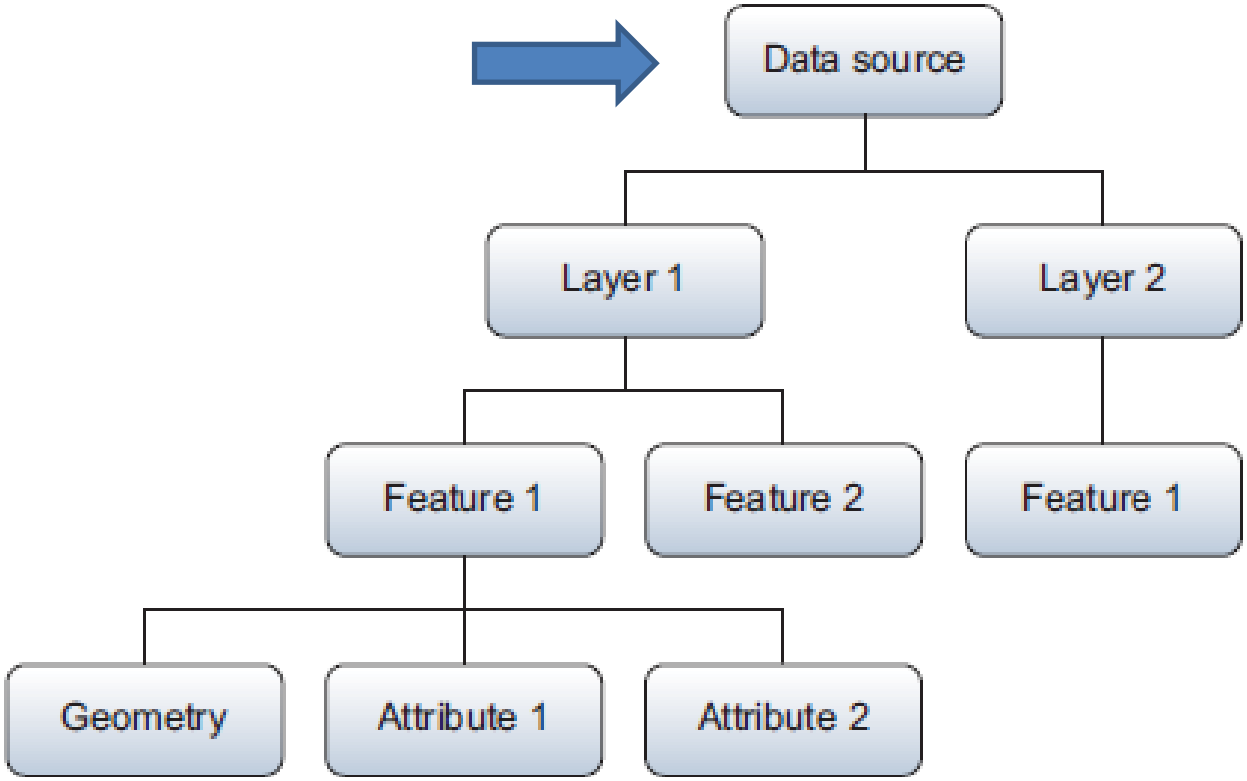
http://www.gdal.org/ogr_formats.html

```
#OGR samo wyszukuje odpowiedniego sterownika dla pliku
ds = ogr.Open(r'C:\myshp.shp',0)
```

OGRClass

- classGDALDataset
- classOGRLayer
- classOGRFeature
- classOGRGeometry

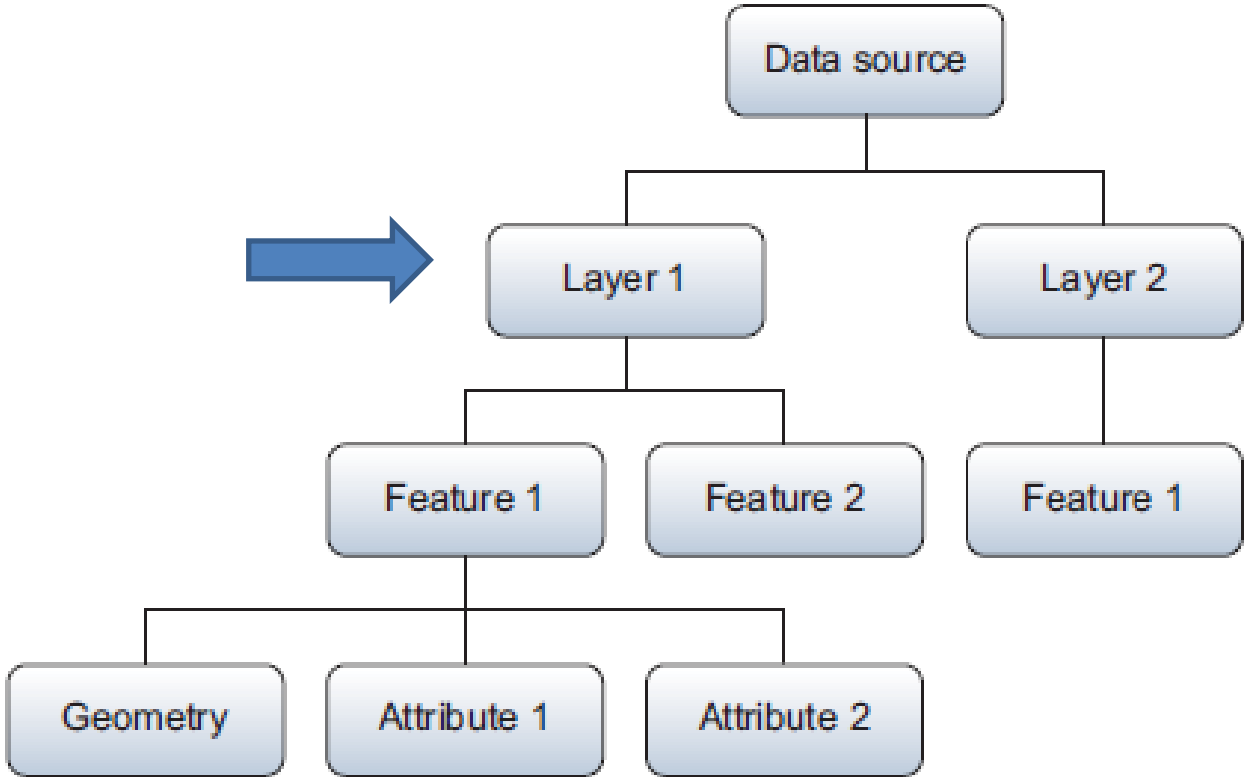
Obsługa źródła danych w OGR



Funkcje dla źródła danych

- ds.GetLayer()
- ds.CreateLayer()
- ds.DeleteLayer()
- ds.CopyLayer()

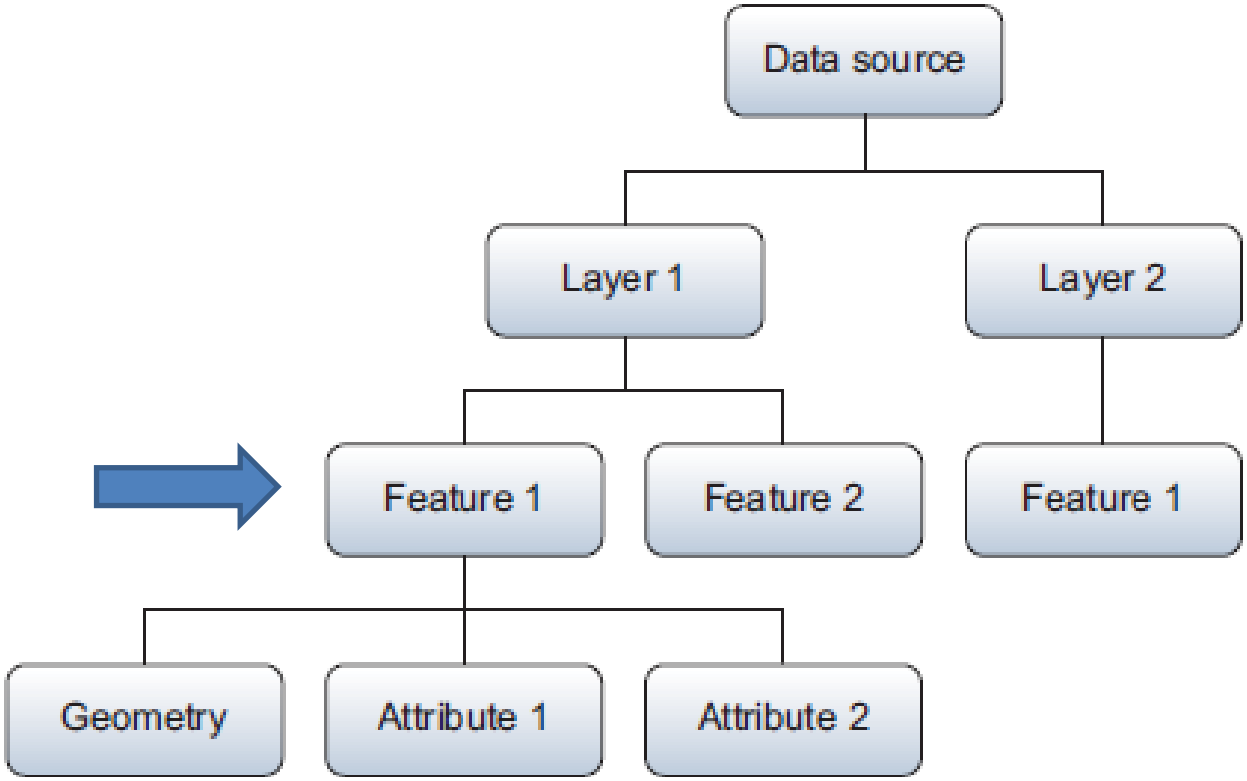
Obsługa warstw w OGR



Funkcje dla warstwy

- layer.GetFeatureCount()
- layer.GetExtent()
- layer.GetSpatialRef()
- layer.GetGeomType()
- layer.CreateField()
- layer.GetLayerDefn()
- layer.GetFeature()
- layer.CreateFeature()
- layer.ResetReading()

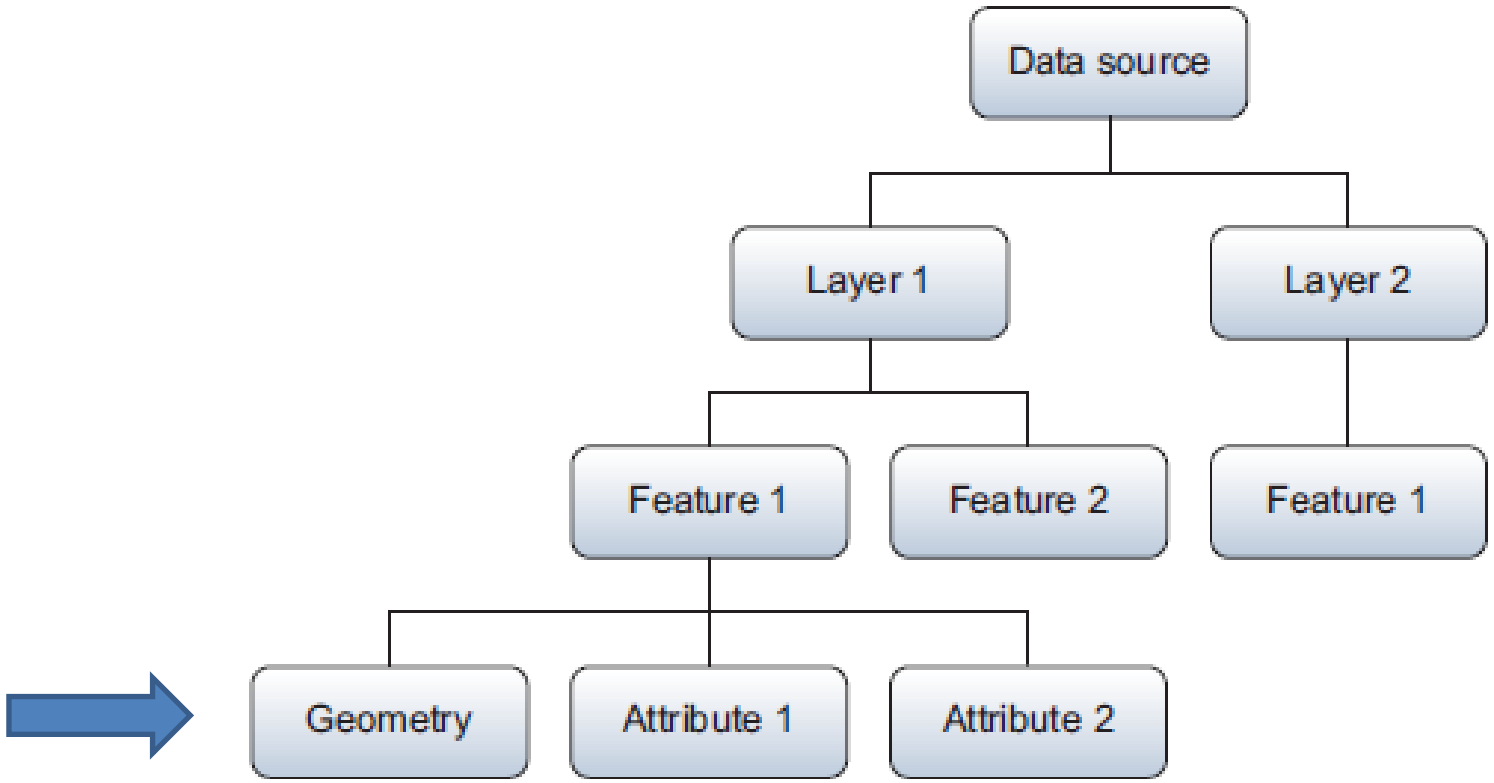
Obsługa obiektów w OGR



Funkcje dla obiektów

- feature.GetField()
- feature.SetField()
- feature.GetFieldCount
- feature.GetGeometryRef()
- feature.geometry()
- feature.SetGeometry(geom)

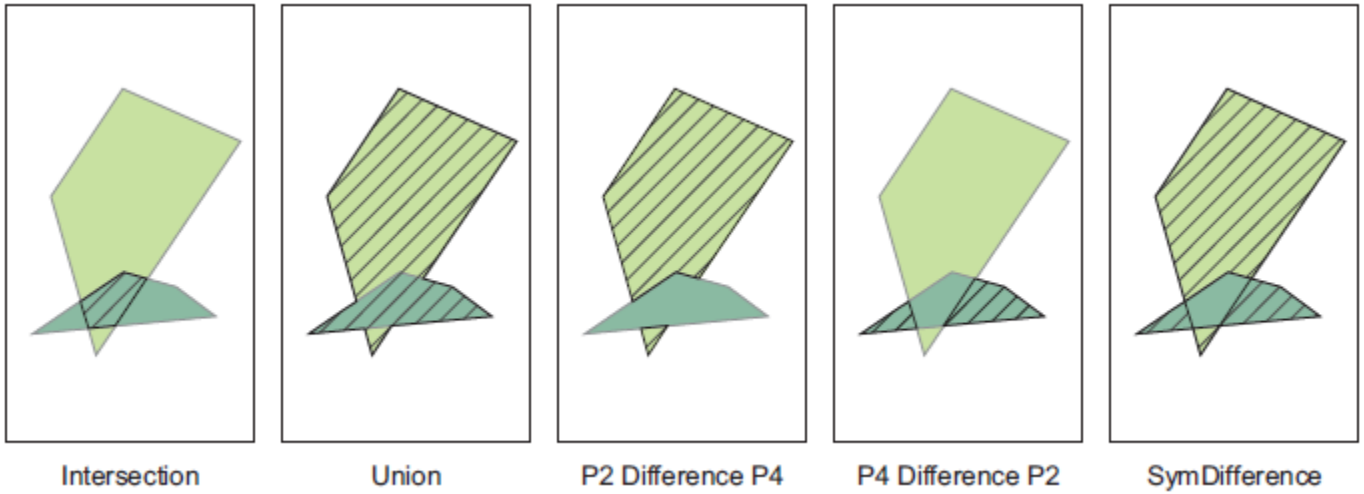
Obsługa geometrii w OGR



Funkcje dla geometrii

- geometry.GetGeometryName()
- geometry.ExportToWkt()
- geometry.TransformTo()
- geometry.GetArea()
- geometry.Length()
- geometryA.intersection(geometryB)

Rodzaje funkcji przetwarzających dane geometryczne



Rodzaje funkcji zwracających wartość logiczną

- Intersects
- Touches
- Crosses
- Within
- Contains
- Overlaps
- Disjoint

Obliczenia ilości obiektów

```
>>> layer = ds.GetLayer('world')
>>> num_features = layer.GetFeatureCount()
>>> last_feature = layer.GetFeature(num_features - 1)
>>> last_feature.name
'Zimbabwe'
```

Odczyt metadanych

```
#odczyt zasięgu warstwy
>>> extent = layer.GetExtent()
>>> extent
(-180.0, 180.00000000000003, -89.99892578124998, 83.59960937500006)
#odczyt typu geometrii
>>> print(layer.GetGeomType())
3
>>> print(layer.GetGeomType() == ogr.wkbPoint)
False
>>> print(layer.GetGeomType() == ogr.wkbPolygon)
True

>>> feat = layer.GetFeature(0)
>>> print(feat.geometry().GetGeometryName())
POLYGON
```

Typy geometrii w OGR

Geometry type	OGR constant
Point	`wkbPoint`
Mulitpoint	`wkbMultiPoint`
Line	`wkbLineString`
Multiline	`wkbMultiLineString`
Polygon	`wkbPolygon`
Multipolygon	`wkbMultiPolygon`
Unknown geometry type	`wkbUnknown`
No geometry	`wkbNone`

Odczyt definicji układu współrzędnych

```
>>>print(layer.GetSpatialRef())

PROJCS["ETRS89_Poland_CS92",
    GEOGCS["ETRS89",
        DATUM["European_Terrestrial_Reference_System_1989",
            SPHEROID["GRS 1980",6378137,298.257222101,
                AUTHORITY["EPSG","7019"]],
            TOWGS84[0,0,0,0,0,0,0],
            AUTHORITY["EPSG","6258"]],
        PRIMEM["Greenwich",0,
            AUTHORITY["EPSG","8901"]],
        UNIT["degree",0.0174532925199433,
            AUTHORITY["EPSG","9122"]],
        AUTHORITY["EPSG","4258"]],
    PROJECTION["Transverse_Mercator"],
    PARAMETER["latitude_of_origin",0],
    PARAMETER["central_meridian",19],
    PARAMETER["scale_factor",0.9993],
    PARAMETER["false_easting",500000],
    PARAMETER["false_northing",-5300000],
    UNIT["Meter",1]]
```

Zapis danych wektorowych do nowego pliku shapefile

```
import ogr, os
#otwarcie katalogu do zapisu i odczytu 1 lub 0- tylko odczyt
ds=ogr.Open(r'C:\Katalog_GIS',1)
#wczytanie warstwy źródłowej
in_lyr=ds.GetLayer('world')
#utworzenie  nowej warstwy docelowej
out_lyr = ds.CreateLayer('panstwo', in_lyr.GetSpatialRef(), ogr.wkbPolygon)
#skopiowanie układu pól
out_lyr.CreateFields(in_lyr.schema)
#utworzenie definicji obiektu
out_defn = out_lyr.GetLayerDefn()
#utworzenie pustego obiektu
out_feat = ogr.Feature(out_defn)
```

Funkcja tworząca warstwę

#Schemat funkcji tworzącej nową warstwę

ds.CreateLayer(nazwa_warstwy, srs, geom_type, options)

#przykład definicji nowej warstwy

```
out_lyr=ds.CreateLayer('panstwo', in_lyr.GetSpatialRef(),
    ogr.wkbPolygon)
```

#definicja pól na podstawie schematu warstwy wejściowej

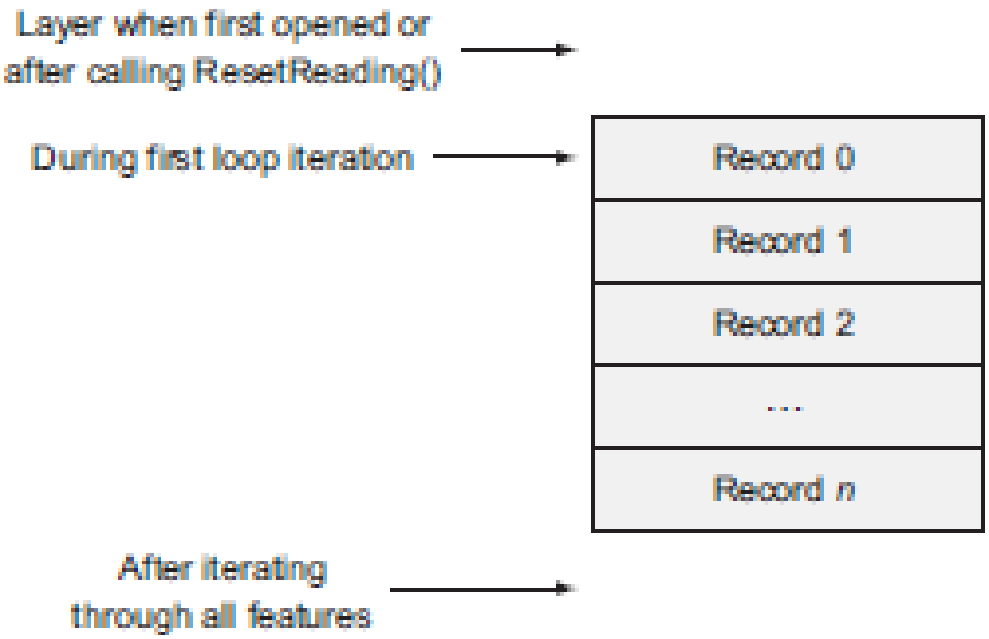
```
out_lyr.CreateFields(in_lyr.schema)
```

Zapis danych wektorowych do pliku shapefile

```
#kopiowanie wybranego obiektu
for in_feat in in_lyr:
    if in_feat.GetField('name')=='Poland':
        geom = in_feat.geometry()
        out_feat.SetGeometry(geom)
     #pobranie poszczególnych pól
        for i in range(in_feat.GetFieldCount()):
            value = in_feat.GetField(i)
            out_feat.SetField(i, value)
     #zapis obiektu do warstwy
        out_lyr.CreateFeature(out_feat)
```

Wielokrotny odczyt warstwy

layer.ResetReading()



Przemieszczanie się wskaźnika

```
>>>lyr = ds.GetLayer('world')
>>>feat = lyr.GetNextFeature()
>>>print(feat.admin)
Afghanistan
feat = lyr.GetNextFeature()
print(feat.admin)
Angola
>>>feat = lyr.GetNextFeature()
>>>print(feat.admin)
Anguilla

>>>lyr.ResetReading()
>>>lyr.GetNextFeature()
<osgeo.ogr.Feature; proxy of <Swig Object of type 'OGRFeatureShadow *' at 0x000002842B539BD0> >
>>>feat = lyr.GetNextFeature()
>>>print(feat.admin)
Afghanistan
```

Metody filtrowania danych

- 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')
>>> lyr.SetAttributeFilter("continent = 'Asia'")
0
>>> lyr.GetFeatureCount()
53
```

Filtrowanie 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'''
```

Filtrowanie i zapis wyników do nowej warstwy na podstawie atrybutów

```
ds = ogr.Open(r'C:/GIS',1)
in_lyr = ds.GetLayer('world_cities')
in_lyr.SetAttributeFilter("FEATURECLA = 'Admin-0 capital'")
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
in_lyr= ds.ExecuteSQL(sql)

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

Rysowanie danych wektorowych

```
import matplotlib.pyplot as plt
import ogr

ds = ogr.Open(r'C:/GIS')
lyr = ds.GetLayer('wojewodztwa')

for row in lyr:
    geom = row.geometry()
    ring = geom.GetGeometryRef(0)
    coords = ring.GetPoints()
    x, y = zip(*coords)
plt.plot(x, y, 'k')
plt.axis('equal')
plt.show()
```