Podstawy programowania

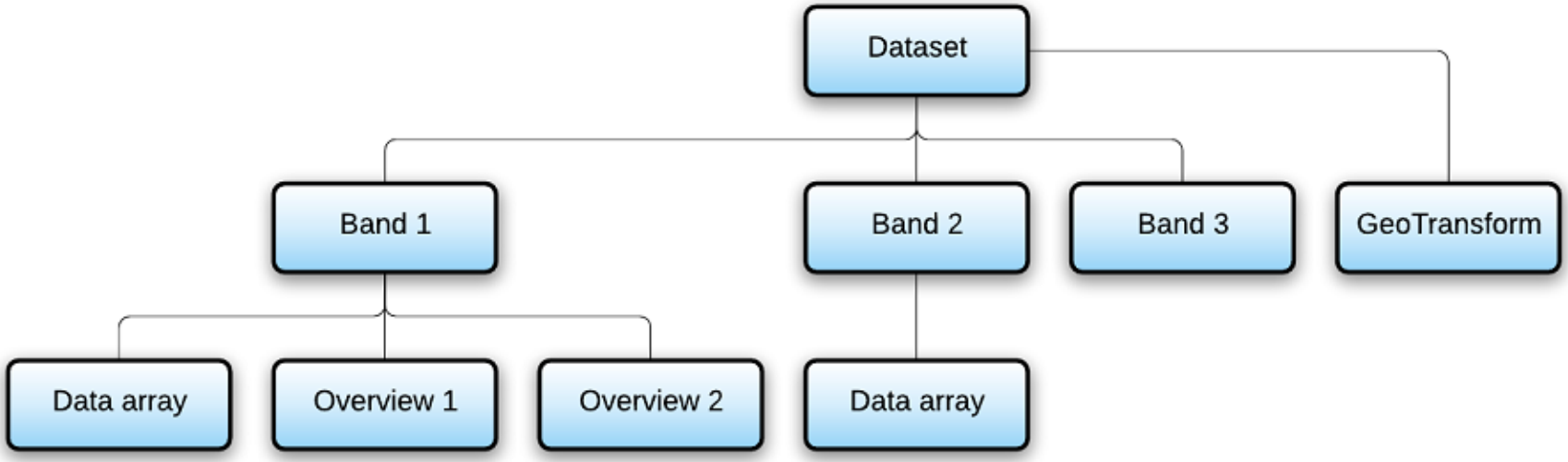
Przetwarzanie i analiza rastra w GDAL

https://pcjericks.github.io/py-gdalogr-cookbook/

Analizy rastrowe

- Tworzenie i zapis rastra
- Kadrowanie rastra na podstawie poligonu
- Algebra map
  - Obliczenia lokalne: NDVI

GDAL raster


Dataset
Band 1
Band 2
Band 3
GeoTransform
Data array
Overview 1
Overview 2
Data array

Funkcje GDAL raster

- GDAL
  - Open()
  - GetDriverByName()
  - RasterizeLayer()
- GDALDriver
  - Create()
- GDALDataset
  - GetRasterBand()
  - GetGeoTransform()
  - SetGeoTransform()
  - SetProjection() – ustawienie układu współrzędnych
  - RasterXSize – ilość kolumn
  - RasterYSize – ilość wierszy
- GDALBand
  - WriteArray()
  - ReadAsArray()
  - SetNoDataValue()

Tworzenie obrazu rastrowego

Funkcja Create(filename,nXSize, nYSize, nBands, GDALDataType)

```
gtiff_driver.Create('raster.tif', 3000, 3000, 1, gdal.GDT_Float32)
```

- GDT_Byte Unsigned 8-bit integer (byte)
- GDT_UInt16 Unsigned 16-bit integer
- GDT_Int16 Signed 16-bit integer
- GDT_UInt32 Unsigned 32-bit integer
- GDT_Int32 Signed 32-bit integer
- GDT_Float32 32-bit floating point
- GDT_Float64 64-bit floating point

Procedura tworzenia i zapisu do pliku rastrowego

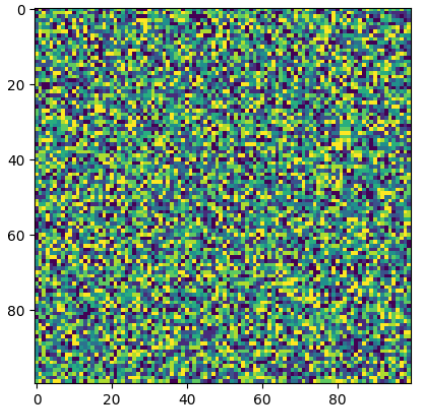
```
#utworzenie sterownika
gtiff_driver = gdal.GetDriverByName('GTiff')
#Klasa GDALDriver
#Funkcja Create(filename,nXSize, nYSize, nBands, GDALDataType)
out_ds = gtiff_driver.Create('raster.tif', 3000, 3000, 1, gdal.GDT_Float32)

#ustawienie projekcji na podstawie danych źródłowych
out_ds.SetProjection(in_ds.GetProjection())
out_ds.SetGeoTransform(in_ds.GetGeoTransform())
#wybór kanału do zapisu
out_band = out_ds.GetRasterBand(1)
#zapis tablicy data do pliku rastrowego
out_band.WriteArray(data)
```

Przykład: zapis tablicy numpy do pliku rastrowego

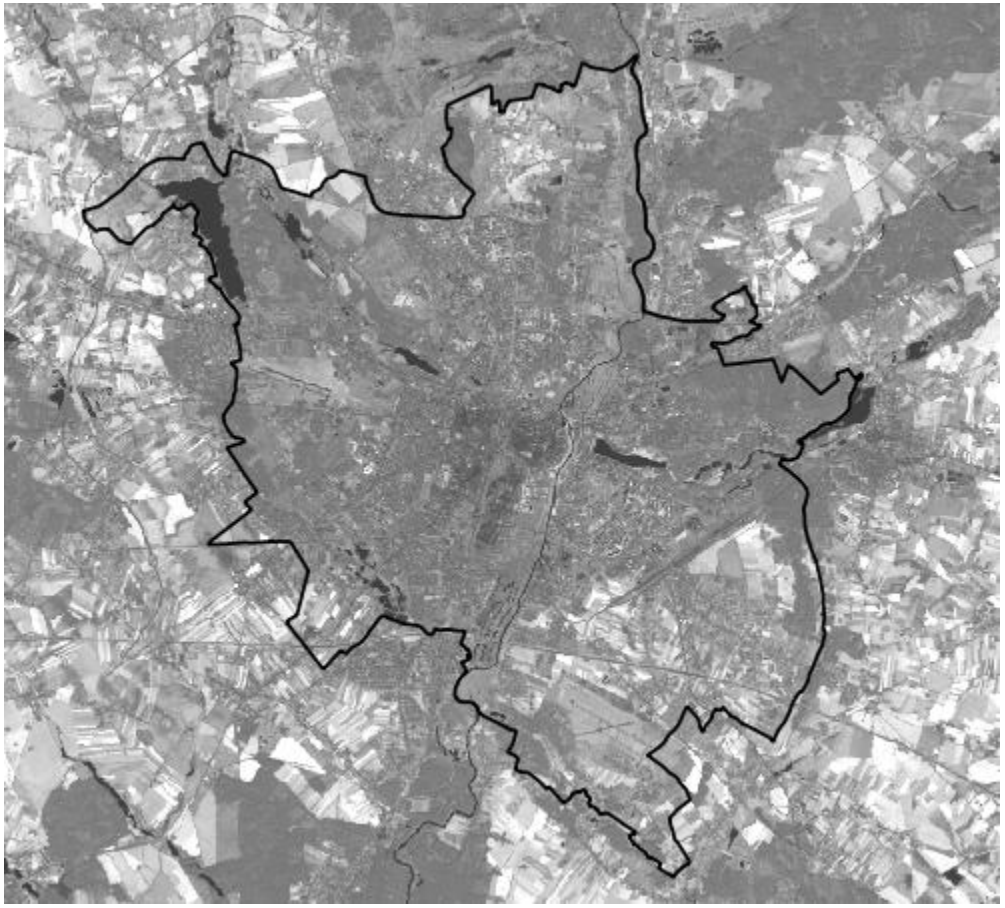
```
import gdal, osr
import numpy as np
import matplotlib.pyplot as plt

raster_fn = r'C:/GIS/test.tiff‘
#wybór sterownika
driver = gdal.GetDriverByName("GTiff")
#utworzenie nowego pliku
dstFile = driver.Create(raster_fn, 100, 100, 1,gdal.GDT_Int16)

#utworzenie zmiennej dla ukadu współrzędnych
spatialReference = osr.SpatialReference()
spatialReference. ImportFromEPSG(2180)

#nadanie układu współrzędnych
dstFile.SetProjection(spatialReference.ExportToWkt())
band = dstFile.GetRasterBand(1)
```

Zapis tablicy do pliku rastrowego

```
#definicja współrzędnych  początku obrazu i wielkości komórki rastra
originX = 334692
originY = 524966
cellWidth = 100
cellHeight = 100
geoTransform = [originX, cellWidth, 0, originY, 0, -cellHeight]
#nadanie parametrów geoTransform
dstFile.SetGeoTransform(geoTransform)

#utworzenie macierzy z losowymi danymi
data=np.random.randint(0, 9, 10000,dtype=np.int16).reshape(100,100)
plt.imshow(data)
plt.show()

#zapis macierzy do pliku rastrowego
band.WriteArray(data)
band.SetNoDataValue(-500)
del dstFile
```



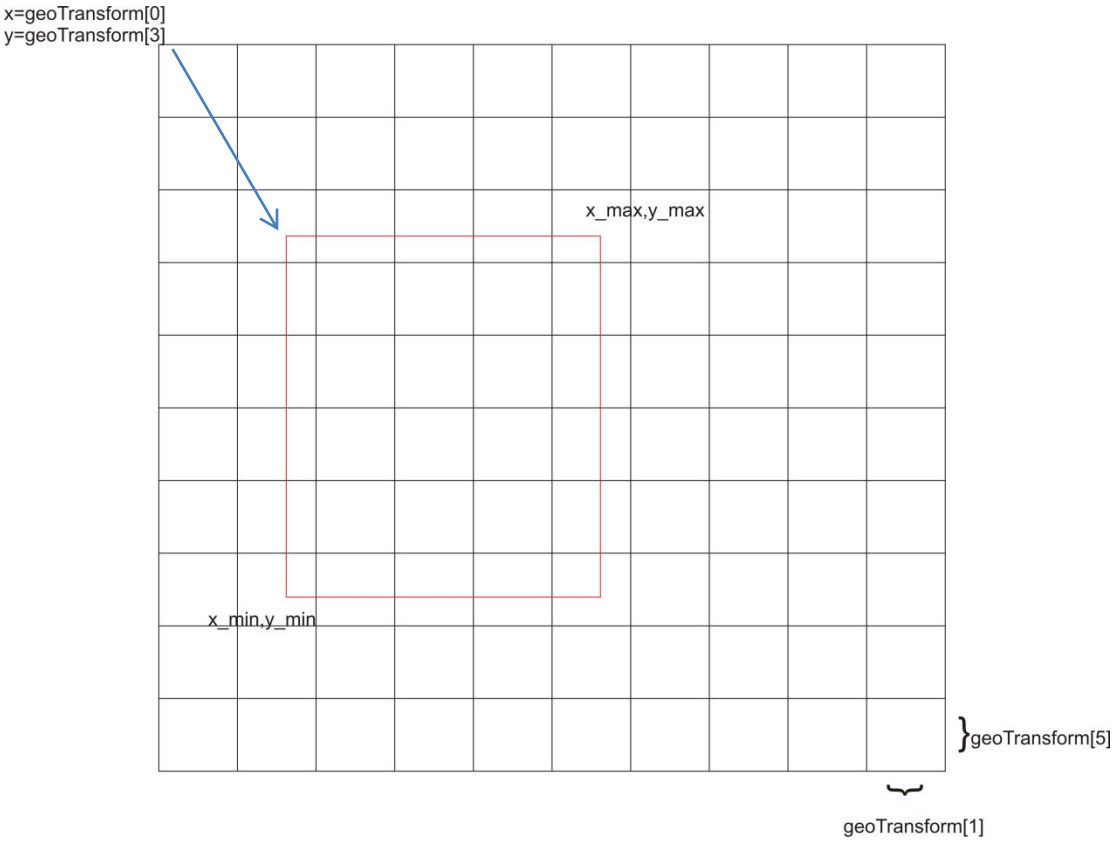
Kadrowanie rastra



Kadrowanie rastra

```
import gdal, ogr
import numpy as np

#funkcja przeliczająca przesunięcie w wierszach i kolumnach względem obrazu źródłowego
def world2Pixel(geoMatrix, x, y):
    ulX = geoMatrix[0]   #współrzędna x górnego narożnika
    ulY = geoMatrix[3]   #współrzędna y górnego narożnika
    xDist = geoMatrix[1]  #szerokość pixela
    yDist = geoMatrix[5]  #wysokość pixela
    pixel = int((x - ulX) / xDist)
    line = int((ulY - y) / abs(yDist))
    return (pixel, line)
```

Kadrowanie rastra



pixel=(x_min-geoTransform[0])/geoTransform[1]

line=(geoTransform[3]-y_max)/geoTransform[5]

Kadrowanie rastra

```
#otwarcie pliku  shapefile z poligonem do kadrowania
ds_clip= ogr.Open(r"C:/GIS")
clip_shp= ds_clip.GetLayer('poznan')

#otwarcie rastra źródłowego dem
raster_path='C:/GIS/Landsat/LC8190024201511 1LGN00.tif'
ds_raster = gdal.Open(raster_path)
geoTrans = ds_raster.GetGeoTransform() #pobranie parametrów geoTrans
```

Kadrowanie rastra

```
#pobranie zasięgu poligonu
x_min, x_max, y_min, y_max = clip_shp.GetExtent()

#obliczenia zasięgu pixeli dla wierszy i kolumn
ulX, ulY = world2Pixel(geoTrans, x_min, y_max)
lrX, lrY = world2Pixel(geoTrans, x_max, y_min)

# obliczenie rozdzielczości nowego obrazu
x_res = int(lrX - ulX)
y_re = int(lrY - ulY)

# obliczenie rozmiarów  pixela dla nowego obrazu
pixel_size = (x_max-x_min)/x_res
pixel_size2 = (y_max-y_min)/y_res

# Utworzenie docelowego pliku rastrowego
raster_fn = 'C:/GIS/poznan_5band.tiff‘
target_ds = gdal.GetDriverByName('GTiff').Create(raster_fn, x_res, y_res, 1, gdal.GDT_Int16)
target_ds.SetGeoTransform((x_min, pixel_size, 0, y_max, 0, -pixel_size2))
band = target_ds.GetRasterBand(1)
```

NumPy.choose (kadrowanie tablicy)

```
>>> mask = np.random.randint(0, 2, 16) .reshape(4,4)
>>> mask
array([[1, 1, 0, 1],
       [0, 1, 1, 1],
       [0, 1, 1, 1],
       [0, 0, 1, 0]])
>>> clip = np.arange(16).reshape(4,4)
>>> clip
array([[ 0,  1,  2,  3],
       [ 4,  5,  6,  7],
       [ 8,  9, 10, 11],
       [12, 13, 14, 15]])
>>> np.choose(mask, (clip,0)) #wstaw 0 gdzie jest 1
array([[ 0,  0,  2,  0],
       [ 4,  0,  0,  0],
       [ 8,  0,  0,  0],
       [12, 13,  0, 15]])
```

Kadrowanie rastra - rasteryzacja

```
#nodata ustwić na 1
band.SetNoDataValue(1)
target_ds.SetProjection(ds_raster.GetProjection())

# Rasteryzacja poligonu - wypalić 0 w rastrze jako obiekt
gdal.RasterizeLayer(target_ds, [1], clip_shp, burn_values=[0])

#wybranie kanału  rastra źródłowego do tablicy
srcArray=ds_raster.GetRasterBand(1).ReadAsArray()
#kadrowanie tablicy  źródłowej do rozmiaru maski
clip = srcArray[ulY:lrY, ulX:lrX]

#wybranie kanału maski do tablicy
mask = target_ds.GetRasterBand(1).ReadAsArray()

#maska w miejsce 1 - wstawia 0, w miejsce 0 wstawia liczbę z warstwy clip
clip = np.choose(mask,(clip, 0))
band.SetNoDataValue(0)
band.WriteArray(clip)
```

Rasteryzacja danych wektorowych

gdal.RasterizeLayer(dataset, bands, layer, [transformer], [transformArg], [burn_values], [options], [callback], [callback_data])

dataset – warstwa rastrowa przejmująca wyniki

bands – lista kanałów otrzymujących wyniki

layer – warstwa OGR, której obiekty zostaną wypalone w rastrze

burn_values – lista wartości wypalonych w rastrze

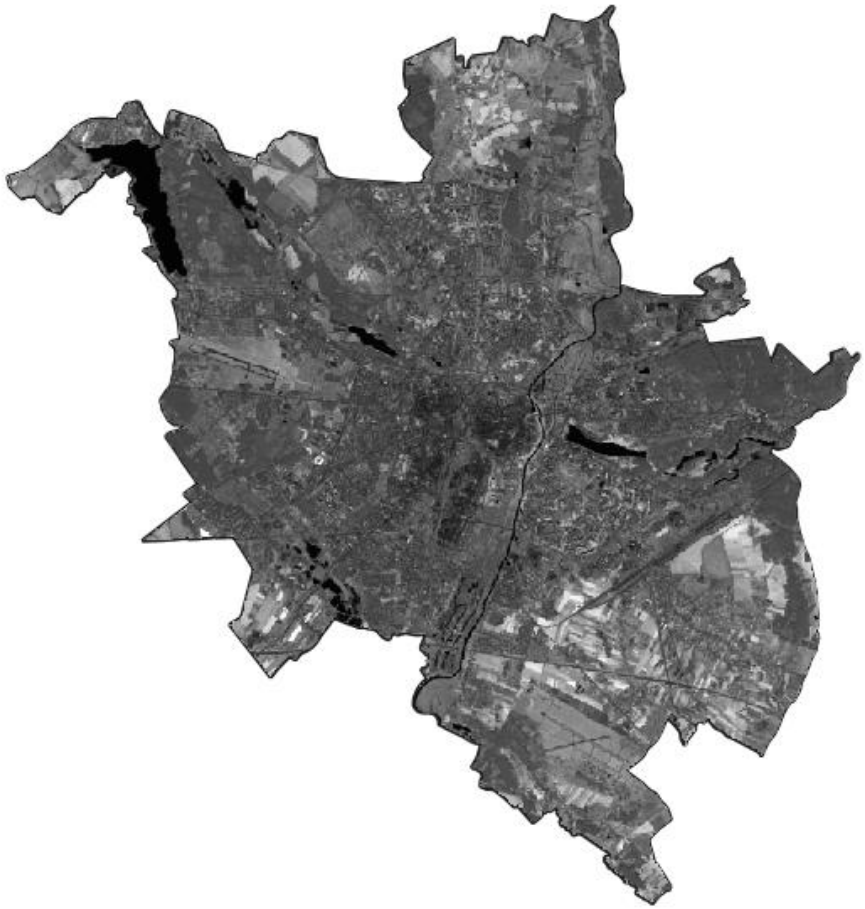
options – opcje zapisane w postaci łańcucha klucz-wartość

callback – funkcja zwrotna prezentująca postęp

callback_data – dane przekazane do funkcji zwrotnej

gdal.RasterizeLayer(target_ds, [1], clip_shp, burn_values=[0])

Efekt kadrowania



Algebra map

- Analiza lokalna
- Analiza ogniskowa/ruchomego okna
- Analiza strefowa
- Analiza globalna

Analiza lokalna

- Metoda polegająca na operacjach na pojedynczych komórkach
- Np. obliczenia wskaźnika NDVI
- ndvi = (nir - red) / (nir + red)

3	5	6	4
4	5	8	9
2	2	5	7
5	7	9	8

+

6	9	8	6
4	5	5	7
3	6	2	4
9	7	8	6

=

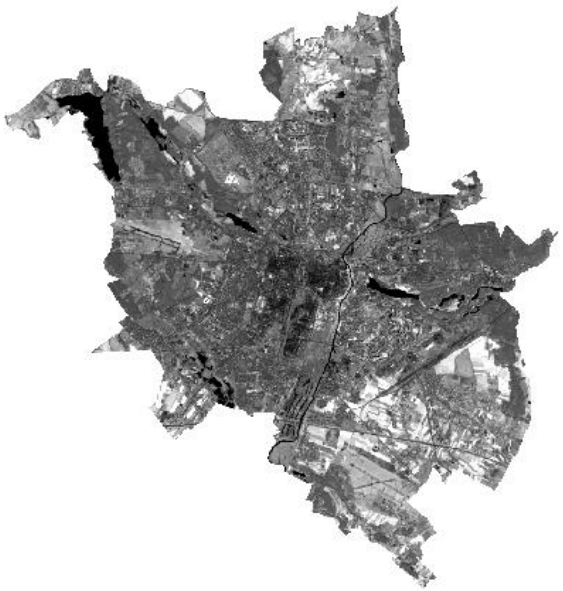
9	14	14	10
8	10	13	16
5	8	7	11
14	14	17	14

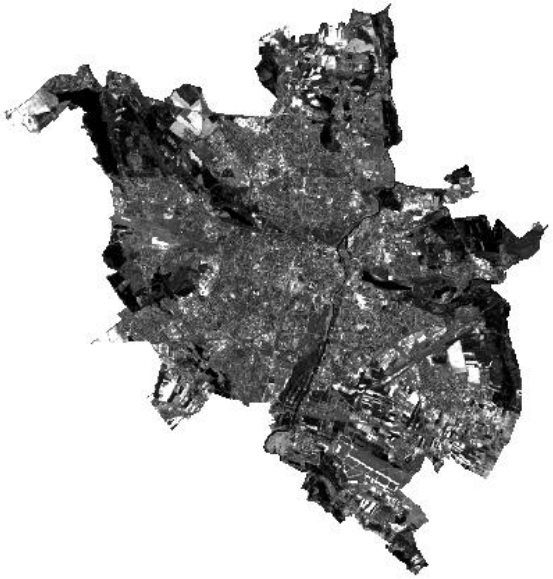
Analiza lokalna-działania na tablicach NumPy

```
>>> a=np.random.randint(0, 9, 16).reshape(4,4)
>>> a
array([[2, 6, 8, 3],
       [3, 0, 6, 3],
       [8, 0, 5, 0],
       [7, 2, 2, 6]])
>>> b = np.arange(16).reshape(4,4)
>>> b
array([[ 0,  1,  2,  3],
       [ 4,  5,  6,  7],
       [ 8,  9, 10, 11],
       [12, 13, 14, 15]])
>>> a+b
array([[ 2,  7, 10,  6],
       [ 7,  5, 12, 10],
       [16,  9, 15, 11],
       [19, 15, 16, 21]])
```

Obliczenia wskaźnika NDVI

```
# -*- coding: utf-8 -*-
import os, gdal
import numpy as np.

os.chdir(r'C:\Landsat\LC81900232015111LGN00')
band4_fn = 'LC81900232015111LGN00_B4.tif'#red
band5_fn = 'LC81900232015111LGN00_B5.tif'#ired

#otwarcie kanału 4
in_ds = gdal.Open(band4_fn)
in_band = in_ds.GetRasterBand(1)
red = in_band.ReadAsArray().astype(np.float)

#otwarcie kanału 5
in_ds = gdal.Open(band5_fn)
nir = in_ds.GetRasterBand(1).ReadAsArray()

#wykonanie obliczeń
#zabezpieczenie przed dzieleniem przez 0
red = np.ma.masked_where(nir + red == 0, red)
ndvi = (nir - red) / (nir + red)
ndvi = ndvi.filled(-9999)
```





Obliczenia NDVI – zapis wyniku

```
gtiff_driver = gdal.GetDriverByName('GTiff')
out_ds = gtiff_driver.Create('ndvi.tif', in_ds.RasterXSize, in_ds.RasterYSize, 1, gdal.GDT_Float32)

#ustawienie układu współrzędnych
out_ds.SetProjection(in_ds.GetProjection())
out_ds.SetGeoTransform(in_ds.GetGeoTransform())

#zapis tablicy do kanału 1
out_band = out_ds.GetRasterBand(1)
out_band.SetNoDataValue(-9999)
out_band.WriteArray(ndvi)
```