# Geodatenverarbeitung mit OpenSource Komponenten 2024

Kurs ETH Zürich - Planung 2025

Hans-Jörg Stark

Hilfreiche Referenz: <https://pcjericks.github.io/py-gdalogr-cookbook/layers.html>

import os  
from osgeo import ogr  
import sys

### Tag 1

# Initiieren des korrekten Treibers und Laden der Geodaten  
  
drv = ogr.GetDriverByName("Esri Shapefile")  
path2ds = os.path.join("Data","Gemeinden\_Solothurn.shp")  
print(path2ds)  
  
datasource = drv.Open(path2ds)  
print(datasource)

Data/Gemeinden\_Solothurn.shp  
<osgeo.ogr.DataSource; proxy of <Swig Object of type 'OGRDataSourceShadow \*' at 0x11ea82630> >

# Definition des Layers der geladenen Shapedatei  
  
layer = datasource.GetLayer(0)  
print(layer)

<osgeo.ogr.Layer; proxy of <Swig Object of type 'OGRLayerShadow \*' at 0x11ea7e720> >

# Ermittlung der Anzahl Datensätze im Layer  
  
ftrCnt = layer.GetFeatureCount()  
print(f"Die Anzahl der Gemeinden im Kanton Solothurn beträgt {ftrCnt}.")

Die Anzahl der Gemeinden im Kanton Solothurn beträgt 109.

# Shape in GeoJSON umwandeln  
gjsonFl = os.path.join("Data","gemSo.geojson")  
cmd = f'ogr2ogr -f GeoJSON {gjsonFl} {path2ds}'  
print(cmd)  
  
os.system(cmd)

ogr2ogr -f GeoJSON Data/gemSo.geojson Data/Gemeinden\_Solothurn.shp

0

# dasselbe mit GeoJSON Daten  
  
drv = ogr.GetDriverByName("GeoJSON")  
#path2ds = os.path.join("Data","gemSo.geojson")  
datasource = drv.Open(gjsonFl)  
layer = datasource.GetLayer(0)  
ftrCnt = layer.GetFeatureCount()  
print(f"Die Anzahl der Gemeinden im Kanton Solothurn beträgt {ftrCnt}.")

Die Anzahl der Gemeinden im Kanton Solothurn beträgt 109.

# Zugriff auf Attributinformationen  
  
lyrDef = layer.GetLayerDefn()  
fldCnt = lyrDef.GetFieldCount()  
print(f"Die Anzahl der Attribute im Geodatensatz beträgt {fldCnt}.")

Die Anzahl der Attribute im Geodatensatz beträgt 11.

for i in range(fldCnt):  
 attName = lyrDef.GetFieldDefn(i).GetName()  
 print(f"{i+1}. Attribut heisst {attName}")

1. Attribut heisst name  
2. Attribut heisst gem\_bfs  
3. Attribut heisst gmde\_name  
4. Attribut heisst gmde\_nr  
5. Attribut heisst bzrk\_nr  
6. Attribut heisst eg\_nr  
7. Attribut heisst plz  
8. Attribut heisst ktn\_nr  
9. Attribut heisst new\_date  
10. Attribut heisst archive\_da  
11. Attribut heisst archive

# SRS Extraktion  
  
mySrs = layer.GetSpatialRef()  
print(mySrs)

PROJCS["CH1903 / LV03",  
 GEOGCS["CH1903",  
 DATUM["CH1903",  
 SPHEROID["Bessel 1841",6377397.155,299.1528128,  
 AUTHORITY["EPSG","7004"]],  
 AUTHORITY["EPSG","6149"]],  
 PRIMEM["Greenwich",0,  
 AUTHORITY["EPSG","8901"]],  
 UNIT["degree",0.0174532925199433,  
 AUTHORITY["EPSG","9122"]],  
 AUTHORITY["EPSG","4149"]],  
 PROJECTION["Hotine\_Oblique\_Mercator\_Azimuth\_Center"],  
 PARAMETER["latitude\_of\_center",46.9524055555556],  
 PARAMETER["longitude\_of\_center",7.43958333333333],  
 PARAMETER["azimuth",90],  
 PARAMETER["rectified\_grid\_angle",90],  
 PARAMETER["scale\_factor",1],  
 PARAMETER["false\_easting",600000],  
 PARAMETER["false\_northing",200000],  
 UNIT["metre",1,  
 AUTHORITY["EPSG","9001"]],  
 AXIS["Easting",EAST],  
 AXIS["Northing",NORTH],  
 AUTHORITY["EPSG","21781"]]

# Ausdehnung der Geometien, dh MBR ermitteln  
  
myExtent = layer.GetExtent()

print(f"Dies sind die 4 Eckpunkte des MBR: \n 1. Punkt: {myExtent[0]} / {myExtent[2]} \n 2. Punkt: {myExtent[0]} / {myExtent[3]} \n 3. Punkt: {myExtent[1]} / {myExtent[3]} \n 4. Punkt: {myExtent[1]} / {myExtent[2]}")

Dies sind die 4 Eckpunkte des MBR:   
 1. Punkt: 592560.389 / 213702.99   
 2. Punkt: 592560.389 / 261329.631   
 3. Punkt: 644759.038 / 261329.631   
 4. Punkt: 644759.038 / 213702.99

#Geometrietyp der Ebene  
  
geomRefNm = ogr.GeometryTypeToName(lyrDef.GetGeomType())  
print(f"Der Geometrietyp der Ebene ist: {geomRefNm}")

Der Geometrietyp der Ebene ist: Unknown (any)

# Zugriff auf einzelne Features/Objekte/Datensätze  
  
myFtr = layer.GetFeature(0)  
print(myFtr)

<osgeo.ogr.Feature; proxy of <Swig Object of type 'OGRFeatureShadow \*' at 0x11dd62030> >

layer.SetAttributeFilter("")  
ftr2search = input("Gemeindename:")  
for feature in layer:  
 if feature.GetField("name") == ftr2search:  
 print(f"BFS Nummer für {ftr2search} lautet: {feature.GetField('gem\_bfs')}")

Gemeindename:Olten  
BFS Nummer für Olten lautet: 2581

layer.SetAttributeFilter(f"name = '{ftr2search}'")  
for feature in layer:  
 print(feature.GetField('gem\_bfs'))  
myftr = layer

2581

layer.SetAttributeFilter("name = 'Solothurn'")  
print(feature.GetField('gem\_bfs'))

2581

for feature in layer:  
 print(feature.GetField('gem\_bfs'))  
 #print(feature.GetGeometryRef())

2498  
2571  
2614  
2401  
2585  
2497  
2428  
2532  
2553  
2555  
2611  
2612  
2620  
2622  
2518  
2574  
2547  
2500  
2427  
2580  
2578  
2429  
2492  
2426  
2481  
2530  
2463  
2526  
2616  
2581  
2407  
2477  
2514  
2476  
2528  
2541  
2579  
2422  
2543  
2529  
2572  
2525  
2576  
2502  
2573  
2479  
2615  
2546  
2493  
2583  
2516  
2527  
2424  
2406  
2549  
2495  
2402  
2474  
2517  
2519  
2499  
2551  
2544  
2408  
2524  
2534  
2513  
2601  
2455  
2471  
2520  
2584  
2545  
2503  
2456  
2457  
2472  
2478  
2511  
2535  
2464  
2548  
2445  
2556  
2423  
2491  
2617  
2473  
2550  
2425  
2403  
2575  
2404  
2586  
2405  
2523  
2421  
2501  
2619  
2542  
2582  
2461  
2475  
2613  
2554  
2480  
2618  
2621  
2465

def analyzeGeometry(geometry, indent=0):  
 s = []  
 s.append(" " \* indent)  
 s.append(geometry.GetGeometryName())  
 if geometry.GetPointCount() > 0:  
 s.append(" mit %d Stuetzpunkten" % geometry.GetPointCount())  
 if geometry.GetGeometryCount() > 0:  
 s.append(" enthaelt:")  
  
 print ("".join(s))  
  
 for i in range(geometry.GetGeometryCount()):  
 print(i)  
 analyzeGeometry(geometry.GetGeometryRef(i), indent+1)

# Geometrie eines Features:  
curFtr = layer.GetFeature(16)  
print(curFtr.GetField('name'))  
myGeometry = curFtr.GetGeometryRef()  
#print(myGeometry)

Günsberg

analyzeGeometry(myGeometry)

POLYGON enthaelt:  
0  
 LINEARRING mit 377 Stuetzpunkten

### Tag 2

# Hausaufgaben:  
ftrDict = {}  
for feature in layer:  
 geometry = feature.GetGeometryRef()  
 geomCnt = geometry.GetGeometryCount()  
 for i in range(geomCnt):   
 if geometry.GetGeometryRef(i).GetPointCount() > 0:  
 pntCnt = geometry.GetGeometryRef(i).GetPointCount()  
 ftrDict[feature.GetField('gmde\_name')] = pntCnt  
sortedFtrDict = sorted(ftrDict.items(), key=lambda x:x[1])  
  
if len(ftrDict) >0:  
 avgPntCnt = int(sum(ftrDict.values()) / len(ftrDict))  
 print(f"Die Gemeinde in Solothurn mit:\n den wenigsten Stützpunkten:{sortedFtrDict[0]}\n den meisten Stützpunkten:{sortedFtrDict[-1]}\n(Durchschnittliche Anzahl Stützpunkte pro Gemeinde: {avgPntCnt})")

Die Gemeinde in Solothurn mit:  
 den wenigsten Stützpunkten:('Rodersdorf', 112)  
 den meisten Stützpunkten:('Erlinsbach SO', 1407)  
(Durchschnittliche Anzahl Stützpunkte pro Gemeinde: 440)

# Umprojektion von Geodaten  
  
from osgeo import osr  
source = osr.SpatialReference()  
source.ImportFromEPSG(2056)  
destination = osr.SpatialReference()  
destination.ImportFromEPSG(4326)  
  
transformationdef = osr.CoordinateTransformation(source, destination)  
retransformationdef = osr.CoordinateTransformation(destination,source)  
  
point = ogr.CreateGeometryFromWkt("POINT (2618579 1244235)")  
point2 = ogr.CreateGeometryFromWkt("POINT (2618579 1244235)")  
print(f"Ausgangspunkt LV95: {point.ExportToWkt()}")  
point.Transform(transformationdef)  
point2.Transform(transformationdef)  
point2.Transform(retransformationdef)  
  
pointWGS84 = ogr.CreateGeometryFromWkt(point.ExportToWkt())  
print(f"Transformierter Punkt in WGS84: {pointWGS84.ExportToWkt()}")  
print(f"Rücktransformierter Punkt in LV95: {point2.ExportToWkt()}")  
  
x=str(point.ExportToWkt()).split('(')[1].split(' ')[0]  
y=str(point.ExportToWkt()).split('(')[1].split(' ')[1].split(')')[0]  
print(x,y)

Ausgangspunkt LV95: POINT (2618579 1244235)  
Transformierter Punkt in WGS84: POINT (47.3487071641269 7.68450280319522)  
Rücktransformierter Punkt in LV95: POINT (2618579.00056614 1244235.00115685)  
47.3487071641269 7.68450280319522

# Extraktion aller Zentroide der Gemeinden  
centroidFl = os.path.join('Data','centroidsGemSo.csv')  
with open(centroidFl, 'w', encoding='utf-8') as centroidFl:  
 centroidFl.write("GemName,X,Y\n")  
 for feature in layer:  
 geometry = feature.GetGeometryRef()  
 minEasting,maxEasting,minNorthing,maxNorthing = geometry.GetEnvelope()  
 centerX = (minEasting + maxEasting)/2  
 centerY = (minNorthing + maxNorthing)/2  
 gemName = feature.GetField('gmde\_name')  
 ftrInfo = f"{gemName},{centerX},{centerY}\n"  
 centroidFl.write(ftrInfo)

# Lösung Lukas  
  
import csv  
dict = [['Gemeinde', 'Centroid']]  
for feature in layer:  
 name = feature.GetField("name")  
 polygon = feature.GetGeometryRef()  
 centroid = polygon.Centroid()  
 dict.append([name, centroid])  
   
with open('solothurn\_center.csv', 'w', newline='', encoding='utf-8') as csvfile:  
   
 csvwriter = csv.writer(csvfile)  
 csvwriter.writerows(dict)  
  
with open('solothurn\_center.csv', 'r', encoding='utf-8') as csvfile:  
 for row in csvfile:  
 print(row)

Gemeinde,Centroid  
  
Rohr,POINT (638811.067723848 251323.78765764)  
  
Boningen,POINT (631433.595773726 239262.035954597)  
  
Büsserach,POINT (608007.64649093 248773.663224415)  
  
Egerkingen,POINT (627043.811838396 241736.272661099)  
  
Walterswil,POINT (639960.470487599 242144.311875436)  
  
Obergösgen,POINT (638934.329313055 246052.650570903)  
  
Mümliswil-Ramiswil,POINT (618003.971930186 244294.456525332)  
  
Subingen,POINT (614034.28281 227700.799157357)  
  
Oberdorf,POINT (604131.310525347 232134.918698141)  
  
Rüttenen,POINT (607177.455185779 232189.058574687)  
  
Bärschwil,POINT (602041.400873615 248245.881417738)  
  
Beinwil,POINT (611593.992994595 245057.236566273)  
  
Meltingen,POINT (611348.131941383 247620.039666757)  
  
Zullwil,POINT (612451.775428482 248585.288451597)  
  
Etziken,POINT (615701.217512814 226350.139124971)  
  
Eppenberg-Wöschnau,POINT (644248.80371216 247849.348360601)  
  
Günsberg,POINT (610269.365628435 234682.372325948)  
  
Trimbach,POINT (634056.114021128 246641.457210408)  
  
Matzendorf,POINT (614247.566813471 239341.209706477)  
  
Kappel,POINT (631672.950151432 241224.200440726)  
  
Gunzgen,POINT (629993.758303797 240288.550390144)  
  
Welschenrohr,POINT (606199.538609237 236130.90332478)  
  
Kienberg,POINT (639872.020872591 254528.07676231)  
  
Laupersdorf,POINT (616394.591960493 240394.646971907)  
  
Witterswil,POINT (606586.07831469 259642.857231669)  
  
Recherswil,POINT (612354.101648409 223221.73253674)  
  
Unterramsern,POINT (603201.813690976 218662.88540585)  
  
Lohn-Ammannsegg,POINT (606762.955888702 224715.796553722)  
  
Fehren,POINT (610675.21750708 249597.739307418)  
  
Olten,POINT (634877.492818944 243881.966240918)  
  
Oensingen,POINT (621486.029485321 237489.177159903)  
  
Metzerlen,POINT (602445.634719349 257233.151319923)  
  
Bolken,POINT (616921.564768987 226926.043137156)  
  
Hofstetten-Flüh,POINT (605249.900002531 257934.124955763)  
  
Obergerlafingen,POINT (611026.368100862 223421.024034916)  
  
Balm b. Günsberg,POINT (608063.023490863 234479.756555913)  
  
Hägendorf,POINT (629163.103379836 243905.923706656)  
  
Balsthal,POINT (619787.96318737 240429.748079579)  
  
Bettlach,POINT (598505.652419201 228623.065182069)  
  
Oekingen,POINT (613134.768478408 225322.19657433)  
  
Däniken,POINT (640564.113254274 244402.643816253)  
  
Kriegstetten,POINT (611851.581266237 224888.832201904)  
  
Gretzenbach,POINT (642195.592640336 244775.13573035)  
  
Wisen,POINT (634071.878117282 249418.402121967)  
  
Dulliken,POINT (638357.840161193 243893.194725876)  
  
Rodersdorf,POINT (600580.452982357 259311.992729399)  
  
Erschwil,POINT (608028.53300881 246785.97907521)  
  
Grenchen,POINT (596133.454219402 227364.873690246)  
  
Lostorf,POINT (637460.49107056 248750.357862273)  
  
Schönenwerd,POINT (643116.536243601 247081.668017978)  
  
Deitingen,POINT (613792.043086808 229685.180237287)  
  
Luterbach,POINT (611220.540445675 229460.628641485)  
  
Herbetswil,POINT (609885.910508681 238069.784379882)  
  
Oberbuchsiten,POINT (624291.790640502 240195.073148854)  
  
Kammersrohr,POINT (611632.152716883 233983.233147486)  
  
Niedergösgen,POINT (641186.08741839 247105.763832229)  
  
Härkingen,POINT (628950.061793274 239367.818717828)  
  
Gempen,POINT (616615.892781101 258648.18240322)  
  
Derendingen,POINT (611460.268868428 226775.059303974)  
  
Gerlafingen,POINT (610110.677071172 224407.196210611)  
  
Stüsslingen,POINT (640077.064675392 249484.958629292)  
  
Lommiswil,POINT (602220.177000283 230829.318903991)  
  
Feldbrunnen-St.Niklaus,POINT (608604.263177888 230453.699497546)  
  
Wolfwil,POINT (627211.604202042 235533.465816431)  
  
Hüniken,POINT (614939.095250359 225844.777970856)  
  
Zuchwil,POINT (609464.239807126 228320.100651157)  
  
Biberist,POINT (608279.036566801 226015.567869377)  
  
Solothurn,POINT (606926.054477269 228603.906663258)  
  
Lüterkofen-Ichertswil,POINT (604887.027619466 223541.123854605)  
  
Bättwil,POINT (605378.377859914 260100.371829598)  
  
Halten,POINT (613136.423952142 224148.987690678)  
  
Starrkirch-Wil,POINT (636913.504129006 243690.591432498)  
  
Flumenthal,POINT (612388.127660385 231489.650032801)  
  
Erlinsbach SO,POINT (641951.940536451 249977.64262756)  
  
Lüterswil-Gächliwil,POINT (600285.28033927 218909.013671193)  
  
Messen,POINT (600892.250414604 216125.925177282)  
  
Büren,POINT (617695.396048907 255250.069711762)  
  
Nuglar-St.Pantaleon,POINT (618851.166121071 257835.660106996)  
  
Aeschi,POINT (617482.90028883 224417.394659348)  
  
Drei Höfe,POINT (615036.692687786 223315.446043306)  
  
Lüsslingen-Nennigkofen,POINT (604210.946550592 226135.326879185)  
  
Hubersdorf,POINT (611361.397596019 232686.963454021)  
  
Biezwil,POINT (598520.291254263 217862.121365934)  
  
Selzach,POINT (600741.505787643 229591.325940843)  
  
Gänsbrunnen,POINT (602145.26871366 234564.051662884)  
  
Hauenstein-Ifenthal,POINT (631521.595320352 246777.623418662)  
  
Grindel,POINT (605186.547530429 247656.965809906)  
  
Dornach,POINT (613995.162103315 258440.134623004)  
  
Langendorf,POINT (605821.350413531 230084.457265437)  
  
Holderbank,POINT (623651.246339452 242509.701813483)  
  
Kestenholz,POINT (623809.786903147 236206.519754123)  
  
Fulenbach,POINT (629817.423431481 236727.140942019)  
  
Neuendorf,POINT (627398.330596242 238545.628228105)  
  
Wangen b. Olten,POINT (632720.848382521 243561.430302223)  
  
Niederbuchsiten,POINT (625686.390565297 237583.20643501)  
  
Horriwil,POINT (614172.416122923 225504.343008831)  
  
Aedermannsdorf,POINT (611655.235422699 240148.260651508)  
  
Winznau,POINT (636714.602175095 246670.143153908)  
  
Kleinlützel,POINT (598429.568882318 252863.227805084)  
  
Bellach,POINT (604326.008112389 228839.378214005)  
  
Rickenbach,POINT (631482.314402676 243608.550755119)  
  
Schnottwil,POINT (596402.935968414 217531.157780923)  
  
Hochwald,POINT (615156.000339653 255983.693089594)  
  
Breitenbach,POINT (608945.005979864 250636.501226574)  
  
Riedholz,POINT (609927.875140837 231762.426759498)  
  
Seewen,POINT (615592.760662654 252765.514148811)  
  
Himmelried,POINT (611466.340900365 252676.104842196)  
  
Nunningen,POINT (613418.874802064 249692.247651751)  
  
Buchegg,POINT (602975.823906844 220891.628040008)

### Rasterdaten

from osgeo import gdal  
rb = os.path.join('Data','ortho14\_5m\_rgb\_solothurn.tif')  
  
def getDifferentRasterInformatioN(rasFl):  
 ds = gdal.Open(rasFl)  
 cols = ds.RasterXSize  
 rows = ds.RasterYSize  
 bands = ds.RasterCount  
   
 print(f"Anzahl Spalten: {cols}, Anzahl Zeilen: {rows}, Anzahl Bänder: {bands}")  
   
 addInfo = ds.GetGeoTransform()  
 orig = f"Ursprung: {addInfo[0]},{addInfo[3]}"  
 pxsize = f"Pixelgrösse: x = {addInfo[1]}, y = {addInfo[5]}"  
 rot = f"Rotation: Achse 1 = {addInfo[2]}, y = {addInfo[4]}"  
 print(f"{orig}\n{pxsize}\n{rot}\n")  
  
 #Bandinformationen  
 for bandnr in range(bands):  
 band = ds.GetRasterBand(bandnr+1)  
 print ('Band-Typ: ',gdal.GetDataTypeName(band.DataType))  
   
 if not band is None:  
 min = band.GetMinimum()  
 max = band.GetMaximum()  
 ct = band.GetColorTable()  
 if not ct is None:  
 print ('Band hat ', ct, ' Farbpalette.')  
   
 if min is None or max is None:  
 (min,max) = band.ComputeRasterMinMax(1)  
   
 print ('Min=%.3f, Max=%.3f' % (min,max))  
   
 if band.GetOverviewCount() > 0:  
 print ('Band hat ', band.GetOverviewCount(), ' Übersichten.')  
   
 if not band.GetRasterColorTable() is None:  
 print ('Band hat eine Farbtabelle mit ', \  
 band.GetRasterColorTable().GetCount(), ' Einträgen.')   
  
getDifferentRasterInformatioN(rb)

Anzahl Spalten: 5800, Anzahl Zeilen: 4800, Anzahl Bänder: 3  
Ursprung: 592000.0,237000.0  
Pixelgrösse: x = 5.0, y = -5.0  
Rotation: Achse 1 = 0.0, y = 0.0  
  
Band-Typ: Byte  
Min=0.000, Max=189.000  
Band hat 7 Übersichten.  
Band-Typ: Byte  
Min=0.000, Max=182.000  
Band hat 7 Übersichten.  
Band-Typ: Byte  
Min=0.000, Max=172.000  
Band hat 7 Übersichten.

scndPic = os.path.join('Data','myimg.png')  
getDifferentRasterInformatioN(scndPic)

Anzahl Spalten: 1244, Anzahl Zeilen: 962, Anzahl Bänder: 4  
Ursprung: 0.0,0.0  
Pixelgrösse: x = 1.0, y = 1.0  
Rotation: Achse 1 = 0.0, y = 0.0  
  
Band-Typ: Byte  
Min=23.000, Max=255.000  
Band-Typ: Byte  
Min=20.000, Max=255.000  
Band-Typ: Byte  
Min=12.000, Max=255.000  
Band-Typ: Byte  
Min=255.000, Max=255.000

thrdPic = r"\\fs.bauwelt.ethz.ch\baug-bauwelt-home$\starkhan\Downloads\lubis-luftbilder\_farbe\_000-288-671\_op\_2056.tif"  
getDifferentRasterInformatioN(thrdPic)  
  
ds = r'Data/swissalti3d.tif'  
getDifferentRasterInformatioN(ds)

Anzahl Spalten: 15793, Anzahl Zeilen: 15701, Anzahl Bänder: 3  
Ursprung: 2598337.5,1200922.5  
Pixelgrösse: x = 0.5, y = -0.5  
Rotation: Achse 1 = 0.0, y = 0.0  
  
Band-Typ: Byte  
Min=0.000, Max=234.000  
Band hat 6 Übersichten.  
Band-Typ: Byte  
Min=0.000, Max=240.000  
Band hat 6 Übersichten.  
Band-Typ: Byte  
Min=0.000, Max=232.000  
Band hat 6 Übersichten.  
Anzahl Spalten: 2000, Anzahl Zeilen: 2000, Anzahl Bänder: 1  
Ursprung: 2601000.0,1202000.0  
Pixelgrösse: x = 0.5, y = -0.5  
Rotation: Achse 1 = 0.0, y = 0.0  
  
Band-Typ: Float32  
Min=515.212, Max=559.813  
Band hat 3 Übersichten.

# obtained from http://www.gis.usu.edu/~chrisg/python/2009/lectures/ospy\_slides4.pdf and adapted  
# script to get pixel values at a set of coordinate by reading in one pixel at a time  
  
import os, sys, numpy, time, csv  
from osgeo import gdal  
from osgeo.gdalconst import \*  
  
# start timing  
startTime = time.time()  
# coordinates to get pixel values for  
#xValues = [594000.0, 604000.0, 613500.0,599594.0]  
#yValues = [229500.0, 231000.0, 222800.0,226081.0]  
  
centrPnts = os.path.join("Data","centroidsGemSo.csv")  
with open(centrPnts, 'r', encoding='utf-8') as centroidFl:  
 cpwri = os.path.join('Data','centroidsGemSoWithRasterInformation.csv')  
 with open(cpwri, 'w', encoding='utf-8') as cpwriFl:  
   
 # register all of the drivers  
 gdal.AllRegister()  
 # open the image  
 ds = gdal.Open('Data/ortho14\_5m\_rgb\_solothurn.tif', GA\_ReadOnly)  
 if ds is None:  
 print ('Could not open image')  
 sys.exit()  
   
 # get image size  
 rows = ds.RasterYSize  
 cols = ds.RasterXSize  
 bands = ds.RasterCount  
 # get georeference info  
 transform = ds.GetGeoTransform()  
 xOrigin = transform[0]  
 yOrigin = transform[3]  
 pixelWidth = transform[1]  
 pixelHeight = transform[5]  
 outStr = 'X , Y , xOffset , yOffset , Wert Band 1 , Wert Band 2 , Wert Band 3 \n'  
 #print(outStr)  
 cpwriFl.write(outStr)  
   
 output=False  
 cnt = 0  
 lineCnt = 0  
   
 # loop through the coordinates  
 for lines in centroidFl:   
 # get x,y  
 if lineCnt > 0:  
 x = float(lines.split(",")[1])  
 y = float(lines.split(",")[2])  
 # compute pixel offset  
 xOffset = int((x - xOrigin) / pixelWidth)  
 yOffset = int((y - yOrigin) / pixelHeight)  
   
 # create a string to print out  
 s = str(x) + ', ' + str(y) + ', ' + str(xOffset) + ', ' + str(yOffset) + ', '  
 # loop through the bands  
 for j in range(bands):  
 band = ds.GetRasterBand(j+1) # 1-based index  
 #read data and add the value to the string  
 data = band.ReadAsArray(xOffset, yOffset, 1, 1)  
 try:  
 value = data[0,0]  
 #print(data)  
 #value2 = numpy.median(data)  
 s = s + str(value) + ', '  
 output=True  
 except:  
 output=False  
 pass  
   
 #print out the data string  
 if output:  
 print(s)  
 cpwriFl.write(f"{s[:-1]}\n")  
 cnt += 1  
 lineCnt += 1  
# figure out how long the script took to run  
endTime = time.time()  
print()  
print ('The script took %.3f seconds' %(endTime - startTime))  
print(f"{cnt} Punkte innerhalb des Rasterbildes von {lineCnt}")

614157.919, 227592.20500000002, 4431, 1881, 69, 92, 78,   
603837.217, 232008.2035, 2367, 998, 61, 63, 69,   
607279.375, 232042.40364837286, 3055, 991, 63, 88, 71,   
615644.5155, 226142.72999999998, 4728, 2171, 167, 162, 150,   
610056.551, 234662.75400000002, 3611, 467, 201, 163, 149,   
606360.3695, 235851.001, 2872, 229, 111, 122, 105,   
612544.7455, 223336.429, 4108, 2732, 72, 94, 85,   
603189.7409999999, 218655.7115, 2237, 3668, 137, 148, 163,   
606760.286, 224852.19199999998, 2952, 2429, 83, 79, 78,   
616792.931, 226934.4895, 4958, 2013, 110, 115, 101,   
610964.84, 223321.053, 3792, 2735, 178, 171, 164,   
607948.3489999999, 234330.3205, 3189, 533, 91, 104, 84,   
598558.324, 228315.2745, 1311, 1736, 57, 79, 67,   
613621.3470000001, 225518.5635, 4324, 2296, 98, 93, 92,   
611828.5774999999, 224931.9575, 3965, 2413, 66, 89, 78,   
596021.1945, 227677.90000000002, 804, 1864, 37, 49, 56,   
613586.88, 229625.0325, 4317, 1474, 115, 114, 104,   
611280.4985, 229238.60499999998, 3856, 1552, 76, 81, 77,   
611624.1315, 233993.3555, 3924, 601, 176, 161, 148,   
611492.8400000001, 226978.29, 3898, 2004, 40, 56, 61,   
610172.0575, 224359.0495, 3634, 2528, 75, 82, 88,   
602367.3605, 230895.532, 2073, 1220, 102, 108, 96,   
608734.5650053136, 230506.52850000001, 3346, 1298, 80, 93, 80,   
614869.5495, 225811.2, 4573, 2237, 66, 85, 74,   
609386.2864999999, 228368.36, 3477, 1726, 161, 156, 156,   
608013.115, 225600.97749999998, 3202, 2279, 136, 143, 138,   
606983.1625, 228575.47, 2996, 1684, 48, 60, 68,   
604749.475, 223627.49, 2549, 2674, 98, 107, 96,   
613170.3955, 224155.0275, 4234, 2568, 91, 91, 90,   
612424.591, 231557.4575, 4084, 1088, 78, 91, 79,   
600263.113, 218942.5685, 1652, 3611, 115, 120, 109,   
600340.899, 216274.81449999998, 1668, 4145, 82, 91, 81,   
617800.1, 224313.4217262693, 5160, 2537, 44, 59, 65,   
614851.5904999999, 223283.313, 4570, 2743, 71, 89, 79,   
604106.7885, 226323.2745, 2421, 2135, 93, 97, 95,   
611382.5935, 232672.09649999999, 3876, 865, 113, 106, 106,   
598673.729, 217723.1065, 1334, 3855, 7, 23, 33,   
600657.421, 229652.11, 1731, 1469, 71, 89, 76,   
602119.7760000001, 235192.42599999998, 2023, 361, 0, 0, 0,   
605776.6215, 230076.075, 2755, 1384, 143, 141, 141,   
614388.371, 225532.14812910568, 4477, 2293, 202, 190, 179,   
604358.158, 228806.52000000002, 2471, 1638, 109, 109, 111,   
596381.255, 217565.419, 876, 3886, 124, 109, 104,   
610119.3984999999, 231787.552, 3623, 1042, 120, 118, 100,   
602816.9375, 220644.685, 2163, 3271, 133, 125, 121,   
  
The script took 0.194 seconds  
45 Punkte innerhalb des Rasterbildes von 110

ERROR 5: Data/ortho14\_5m\_rgb\_solothurn.tif, band 1: Access window out of range in RasterIO(). Requested  
(9351,-2853) of size 1x1 on raster of 5800x4800.  
ERROR 5: Data/ortho14\_5m\_rgb\_solothurn.tif, band 2: Access window out of range in RasterIO(). Requested  
(9351,-2853) of size 1x1 on raster of 5800x4800.  
ERROR 5: Data/ortho14\_5m\_rgb\_solothurn.tif, band 3: Access window out of range in RasterIO(). Requested  
(9351,-2853) of size 1x1 on raster of 5800x4800.  
ERROR 5: Data/ortho14\_5m\_rgb\_solothurn.tif, band 1: Access window out of range in RasterIO(). Requested  
(7983,-398) of size 1x1 on raster of 5800x4800.  
ERROR 5: Data/ortho14\_5m\_rgb\_solothurn.tif, band 2: Access window out of range in RasterIO(). Requested  
(7983,-398) of size 1x1 on raster of 5800x4800.  
ERROR 5: Data/ortho14\_5m\_rgb\_solothurn.tif, band 3: Access window out of range in RasterIO(). Requested  
(7983,-398) of size 1x1 on raster of 5800x4800.  
ERROR 5: Data/ortho14\_5m\_rgb\_solothurn.tif, band 1: Access window out of range in RasterIO(). Requested  
(3245,-2357) of size 1x1 on raster of 5800x4800.  
ERROR 5: Data/ortho14\_5m\_rgb\_solothurn.tif, band 2: Access window out of range in RasterIO(). Requested  
(3245,-2357) of size 1x1 on raster of 5800x4800.  
ERROR 5: Data/ortho14\_5m\_rgb\_solothurn.tif, band 3: Access window out of range in RasterIO(). Requested  
(3245,-2357) of size 1x1 on raster of 5800x4800.  
ERROR 5: Data/ortho14\_5m\_rgb\_solothurn.tif, band 1: Access window out of range in RasterIO(). Requested  
(7020,-941) of size 1x1 on raster of 5800x4800.  
ERROR 5: Data/ortho14\_5m\_rgb\_solothurn.tif, band 2: Access window out of range in RasterIO(). Requested  
(7020,-941) of size 1x1 on raster of 5800x4800.  
ERROR 5: Data/ortho14\_5m\_rgb\_solothurn.tif, band 3: Access window out of range in RasterIO(). Requested  
(7020,-941) of size 1x1 on raster of 5800x4800.  
ERROR 5: Data/ortho14\_5m\_rgb\_solothurn.tif, band 1: Access window out of range in RasterIO(). Requested  
(9625,-1010) of size 1x1 on raster of 5800x4800.  
ERROR 5: Data/ortho14\_5m\_rgb\_solothurn.tif, band 2: Access window out of range in RasterIO(). Requested  
(9625,-1010) of size 1x1 on raster of 5800x4800.  
ERROR 5: Data/ortho14\_5m\_rgb\_solothurn.tif, band 3: Access window out of range in RasterIO(). Requested  
(9625,-1010) of size 1x1 on raster of 5800x4800.  
ERROR 5: Data/ortho14\_5m\_rgb\_solothurn.tif, band 1: Access window out of range in RasterIO(). Requested  
(9366,-1791) of size 1x1 on raster of 5800x4800.  
ERROR 5: Data/ortho14\_5m\_rgb\_solothurn.tif, band 2: Access window out of range in RasterIO(). Requested  
(9366,-1791) of size 1x1 on raster of 5800x4800.  
ERROR 5: Data/ortho14\_5m\_rgb\_solothurn.tif, band 3: Access window out of range in RasterIO(). Requested  
(9366,-1791) of size 1x1 on raster of 5800x4800.  
ERROR 5: Data/ortho14\_5m\_rgb\_solothurn.tif, band 1: Access window out of range in RasterIO(). Requested  
(5008,-1455) of size 1x1 on raster of 5800x4800.  
ERROR 5: Data/ortho14\_5m\_rgb\_solothurn.tif, band 2: Access window out of range in RasterIO(). Requested  
(5008,-1455) of size 1x1 on raster of 5800x4800.  
ERROR 5: Data/ortho14\_5m\_rgb\_solothurn.tif, band 3: Access window out of range in RasterIO(). Requested  
(5008,-1455) of size 1x1 on raster of 5800x4800.  
ERROR 5: Data/ortho14\_5m\_rgb\_solothurn.tif, band 1: Access window out of range in RasterIO(). Requested  
(2023,-2290) of size 1x1 on raster of 5800x4800.  
ERROR 5: Data/ortho14\_5m\_rgb\_solothurn.tif, band 2: Access window out of range in RasterIO(). Requested  
(2023,-2290) of size 1x1 on raster of 5800x4800.  
ERROR 5: Data/ortho14\_5m\_rgb\_solothurn.tif, band 3: Access window out of range in RasterIO(). Requested  
(2023,-2290) of size 1x1 on raster of 5800x4800.  
ERROR 5: Data/ortho14\_5m\_rgb\_solothurn.tif, band 1: Access window out of range in RasterIO(). Requested  
(3896,-1565) of size 1x1 on raster of 5800x4800.  
ERROR 5: Data/ortho14\_5m\_rgb\_solothurn.tif, band 2: Access window out of range in RasterIO(). Requested  
(3896,-1565) of size 1x1 on raster of 5800x4800.  
ERROR 5: Data/ortho14\_5m\_rgb\_solothurn.tif, band 3: Access window out of range in RasterIO(). Requested  
(3896,-1565) of size 1x1 on raster of 5800x4800.  
ERROR 5: Data/ortho14\_5m\_rgb\_solothurn.tif, band 1: Access window out of range in RasterIO(). Requested  
(3913,-2160) of size 1x1 on raster of 5800x4800.  
ERROR 5: Data/ortho14\_5m\_rgb\_solothurn.tif, band 2: Access window out of range in RasterIO(). Requested  
(3913,-2160) of size 1x1 on raster of 5800x4800.  
ERROR 5: Data/ortho14\_5m\_rgb\_solothurn.tif, band 3: Access window out of range in RasterIO(). Requested  
(3913,-2160) of size 1x1 on raster of 5800x4800.  
ERROR 5: Data/ortho14\_5m\_rgb\_solothurn.tif, band 1: Access window out of range in RasterIO(). Requested  
(4079,-2315) of size 1x1 on raster of 5800x4800.  
ERROR 5: Data/ortho14\_5m\_rgb\_solothurn.tif, band 2: Access window out of range in RasterIO(). Requested  
(4079,-2315) of size 1x1 on raster of 5800x4800.  
ERROR 5: Data/ortho14\_5m\_rgb\_solothurn.tif, band 3: Access window out of range in RasterIO(). Requested  
(4079,-2315) of size 1x1 on raster of 5800x4800.  
ERROR 5: Data/ortho14\_5m\_rgb\_solothurn.tif, band 1: Access window out of range in RasterIO(). Requested  
(10440,-2176) of size 1x1 on raster of 5800x4800.  
ERROR 5: Data/ortho14\_5m\_rgb\_solothurn.tif, band 2: Access window out of range in RasterIO(). Requested  
(10440,-2176) of size 1x1 on raster of 5800x4800.  
ERROR 5: Data/ortho14\_5m\_rgb\_solothurn.tif, band 3: Access window out of range in RasterIO(). Requested  
(10440,-2176) of size 1x1 on raster of 5800x4800.  
ERROR 5: Data/ortho14\_5m\_rgb\_solothurn.tif, band 1: Access window out of range in RasterIO(). Requested  
(8376,-1950) of size 1x1 on raster of 5800x4800.  
ERROR 5: Data/ortho14\_5m\_rgb\_solothurn.tif, band 2: Access window out of range in RasterIO(). Requested  
(8376,-1950) of size 1x1 on raster of 5800x4800.  
ERROR 5: Data/ortho14\_5m\_rgb\_solothurn.tif, band 3: Access window out of range in RasterIO(). Requested  
(8376,-1950) of size 1x1 on raster of 5800x4800.  
ERROR 5: Data/ortho14\_5m\_rgb\_solothurn.tif, band 1: Access window out of range in RasterIO(). Requested  
(4476,-535) of size 1x1 on raster of 5800x4800.  
ERROR 5: Data/ortho14\_5m\_rgb\_solothurn.tif, band 2: Access window out of range in RasterIO(). Requested  
(4476,-535) of size 1x1 on raster of 5800x4800.  
ERROR 5: Data/ortho14\_5m\_rgb\_solothurn.tif, band 3: Access window out of range in RasterIO(). Requested  
(4476,-535) of size 1x1 on raster of 5800x4800.  
ERROR 5: Data/ortho14\_5m\_rgb\_solothurn.tif, band 1: Access window out of range in RasterIO(). Requested  
(7917,-865) of size 1x1 on raster of 5800x4800.  
ERROR 5: Data/ortho14\_5m\_rgb\_solothurn.tif, band 2: Access window out of range in RasterIO(). Requested  
(7917,-865) of size 1x1 on raster of 5800x4800.  
ERROR 5: Data/ortho14\_5m\_rgb\_solothurn.tif, band 3: Access window out of range in RasterIO(). Requested  
(7917,-865) of size 1x1 on raster of 5800x4800.  
ERROR 5: Data/ortho14\_5m\_rgb\_solothurn.tif, band 1: Access window out of range in RasterIO(). Requested  
(7584,-611) of size 1x1 on raster of 5800x4800.  
ERROR 5: Data/ortho14\_5m\_rgb\_solothurn.tif, band 2: Access window out of range in RasterIO(). Requested  
(7584,-611) of size 1x1 on raster of 5800x4800.  
ERROR 5: Data/ortho14\_5m\_rgb\_solothurn.tif, band 3: Access window out of range in RasterIO(). Requested  
(7584,-611) of size 1x1 on raster of 5800x4800.  
ERROR 5: Data/ortho14\_5m\_rgb\_solothurn.tif, band 1: Access window out of range in RasterIO(). Requested  
(9572,-3548) of size 1x1 on raster of 5800x4800.  
ERROR 5: Data/ortho14\_5m\_rgb\_solothurn.tif, band 2: Access window out of range in RasterIO(). Requested  
(9572,-3548) of size 1x1 on raster of 5800x4800.  
ERROR 5: Data/ortho14\_5m\_rgb\_solothurn.tif, band 3: Access window out of range in RasterIO(). Requested  
(9572,-3548) of size 1x1 on raster of 5800x4800.  
ERROR 5: Data/ortho14\_5m\_rgb\_solothurn.tif, band 1: Access window out of range in RasterIO(). Requested  
(4820,-580) of size 1x1 on raster of 5800x4800.  
ERROR 5: Data/ortho14\_5m\_rgb\_solothurn.tif, band 2: Access window out of range in RasterIO(). Requested  
(4820,-580) of size 1x1 on raster of 5800x4800.  
ERROR 5: Data/ortho14\_5m\_rgb\_solothurn.tif, band 3: Access window out of range in RasterIO(). Requested  
(4820,-580) of size 1x1 on raster of 5800x4800.  
ERROR 5: Data/ortho14\_5m\_rgb\_solothurn.tif, band 1: Access window out of range in RasterIO(). Requested  
(2910,-4531) of size 1x1 on raster of 5800x4800.  
ERROR 5: Data/ortho14\_5m\_rgb\_solothurn.tif, band 2: Access window out of range in RasterIO(). Requested  
(2910,-4531) of size 1x1 on raster of 5800x4800.  
ERROR 5: Data/ortho14\_5m\_rgb\_solothurn.tif, band 3: Access window out of range in RasterIO(). Requested  
(2910,-4531) of size 1x1 on raster of 5800x4800.  
ERROR 5: Data/ortho14\_5m\_rgb\_solothurn.tif, band 1: Access window out of range in RasterIO(). Requested  
(3730,-2532) of size 1x1 on raster of 5800x4800.  
ERROR 5: Data/ortho14\_5m\_rgb\_solothurn.tif, band 2: Access window out of range in RasterIO(). Requested  
(3730,-2532) of size 1x1 on raster of 5800x4800.  
ERROR 5: Data/ortho14\_5m\_rgb\_solothurn.tif, band 3: Access window out of range in RasterIO(). Requested  
(3730,-2532) of size 1x1 on raster of 5800x4800.  
ERROR 5: Data/ortho14\_5m\_rgb\_solothurn.tif, band 1: Access window out of range in RasterIO(). Requested  
(8601,-1266) of size 1x1 on raster of 5800x4800.  
ERROR 5: Data/ortho14\_5m\_rgb\_solothurn.tif, band 2: Access window out of range in RasterIO(). Requested  
(8601,-1266) of size 1x1 on raster of 5800x4800.  
ERROR 5: Data/ortho14\_5m\_rgb\_solothurn.tif, band 3: Access window out of range in RasterIO(). Requested  
(8601,-1266) of size 1x1 on raster of 5800x4800.  
ERROR 5: Data/ortho14\_5m\_rgb\_solothurn.tif, band 1: Access window out of range in RasterIO(). Requested  
(5833,-10) of size 1x1 on raster of 5800x4800.  
ERROR 5: Data/ortho14\_5m\_rgb\_solothurn.tif, band 2: Access window out of range in RasterIO(). Requested  
(5833,-10) of size 1x1 on raster of 5800x4800.  
ERROR 5: Data/ortho14\_5m\_rgb\_solothurn.tif, band 3: Access window out of range in RasterIO(). Requested  
(5833,-10) of size 1x1 on raster of 5800x4800.  
ERROR 5: Data/ortho14\_5m\_rgb\_solothurn.tif, band 1: Access window out of range in RasterIO(). Requested  
(2075,-4041) of size 1x1 on raster of 5800x4800.  
ERROR 5: Data/ortho14\_5m\_rgb\_solothurn.tif, band 2: Access window out of range in RasterIO(). Requested  
(2075,-4041) of size 1x1 on raster of 5800x4800.  
ERROR 5: Data/ortho14\_5m\_rgb\_solothurn.tif, band 3: Access window out of range in RasterIO(). Requested  
(2075,-4041) of size 1x1 on raster of 5800x4800.  
ERROR 5: Data/ortho14\_5m\_rgb\_solothurn.tif, band 1: Access window out of range in RasterIO(). Requested  
(2658,-4222) of size 1x1 on raster of 5800x4800.  
ERROR 5: Data/ortho14\_5m\_rgb\_solothurn.tif, band 2: Access window out of range in RasterIO(). Requested  
(2658,-4222) of size 1x1 on raster of 5800x4800.  
ERROR 5: Data/ortho14\_5m\_rgb\_solothurn.tif, band 3: Access window out of range in RasterIO(). Requested  
(2658,-4222) of size 1x1 on raster of 5800x4800.  
ERROR 5: Data/ortho14\_5m\_rgb\_solothurn.tif, band 1: Access window out of range in RasterIO(). Requested  
(7446,-1389) of size 1x1 on raster of 5800x4800.  
ERROR 5: Data/ortho14\_5m\_rgb\_solothurn.tif, band 2: Access window out of range in RasterIO(). Requested  
(7446,-1389) of size 1x1 on raster of 5800x4800.  
ERROR 5: Data/ortho14\_5m\_rgb\_solothurn.tif, band 3: Access window out of range in RasterIO(). Requested  
(7446,-1389) of size 1x1 on raster of 5800x4800.  
ERROR 5: Data/ortho14\_5m\_rgb\_solothurn.tif, band 1: Access window out of range in RasterIO(). Requested  
(5672,-651) of size 1x1 on raster of 5800x4800.  
ERROR 5: Data/ortho14\_5m\_rgb\_solothurn.tif, band 2: Access window out of range in RasterIO(). Requested  
(5672,-651) of size 1x1 on raster of 5800x4800.  
ERROR 5: Data/ortho14\_5m\_rgb\_solothurn.tif, band 3: Access window out of range in RasterIO(). Requested  
(5672,-651) of size 1x1 on raster of 5800x4800.  
ERROR 5: Data/ortho14\_5m\_rgb\_solothurn.tif, band 1: Access window out of range in RasterIO(). Requested  
(9710,-1514) of size 1x1 on raster of 5800x4800.  
ERROR 5: Data/ortho14\_5m\_rgb\_solothurn.tif, band 2: Access window out of range in RasterIO(). Requested  
(9710,-1514) of size 1x1 on raster of 5800x4800.  
ERROR 5: Data/ortho14\_5m\_rgb\_solothurn.tif, band 3: Access window out of range in RasterIO(). Requested  
(9710,-1514) of size 1x1 on raster of 5800x4800.  
ERROR 5: Data/ortho14\_5m\_rgb\_solothurn.tif, band 1: Access window out of range in RasterIO(). Requested  
(9983,-1497) of size 1x1 on raster of 5800x4800.  
ERROR 5: Data/ortho14\_5m\_rgb\_solothurn.tif, band 2: Access window out of range in RasterIO(). Requested  
(9983,-1497) of size 1x1 on raster of 5800x4800.  
ERROR 5: Data/ortho14\_5m\_rgb\_solothurn.tif, band 3: Access window out of range in RasterIO(). Requested  
(9983,-1497) of size 1x1 on raster of 5800x4800.  
ERROR 5: Data/ortho14\_5m\_rgb\_solothurn.tif, band 1: Access window out of range in RasterIO(). Requested  
(8406,-2496) of size 1x1 on raster of 5800x4800.  
ERROR 5: Data/ortho14\_5m\_rgb\_solothurn.tif, band 2: Access window out of range in RasterIO(). Requested  
(8406,-2496) of size 1x1 on raster of 5800x4800.  
ERROR 5: Data/ortho14\_5m\_rgb\_solothurn.tif, band 3: Access window out of range in RasterIO(). Requested  
(8406,-2496) of size 1x1 on raster of 5800x4800.  
ERROR 5: Data/ortho14\_5m\_rgb\_solothurn.tif, band 1: Access window out of range in RasterIO(). Requested  
(9274,-1388) of size 1x1 on raster of 5800x4800.  
ERROR 5: Data/ortho14\_5m\_rgb\_solothurn.tif, band 2: Access window out of range in RasterIO(). Requested  
(9274,-1388) of size 1x1 on raster of 5800x4800.  
ERROR 5: Data/ortho14\_5m\_rgb\_solothurn.tif, band 3: Access window out of range in RasterIO(). Requested  
(9274,-1388) of size 1x1 on raster of 5800x4800.  
ERROR 5: Data/ortho14\_5m\_rgb\_solothurn.tif, band 1: Access window out of range in RasterIO(). Requested  
(1707,-4457) of size 1x1 on raster of 5800x4800.  
ERROR 5: Data/ortho14\_5m\_rgb\_solothurn.tif, band 2: Access window out of range in RasterIO(). Requested  
(1707,-4457) of size 1x1 on raster of 5800x4800.  
ERROR 5: Data/ortho14\_5m\_rgb\_solothurn.tif, band 3: Access window out of range in RasterIO(). Requested  
(1707,-4457) of size 1x1 on raster of 5800x4800.  
ERROR 5: Data/ortho14\_5m\_rgb\_solothurn.tif, band 1: Access window out of range in RasterIO(). Requested  
(3194,-1950) of size 1x1 on raster of 5800x4800.  
ERROR 5: Data/ortho14\_5m\_rgb\_solothurn.tif, band 2: Access window out of range in RasterIO(). Requested  
(3194,-1950) of size 1x1 on raster of 5800x4800.  
ERROR 5: Data/ortho14\_5m\_rgb\_solothurn.tif, band 3: Access window out of range in RasterIO(). Requested  
(3194,-1950) of size 1x1 on raster of 5800x4800.  
ERROR 5: Data/ortho14\_5m\_rgb\_solothurn.tif, band 1: Access window out of range in RasterIO(). Requested  
(9057,-2371) of size 1x1 on raster of 5800x4800.  
ERROR 5: Data/ortho14\_5m\_rgb\_solothurn.tif, band 2: Access window out of range in RasterIO(). Requested  
(9057,-2371) of size 1x1 on raster of 5800x4800.  
ERROR 5: Data/ortho14\_5m\_rgb\_solothurn.tif, band 3: Access window out of range in RasterIO(). Requested  
(9057,-2371) of size 1x1 on raster of 5800x4800.  
ERROR 5: Data/ortho14\_5m\_rgb\_solothurn.tif, band 1: Access window out of range in RasterIO(). Requested  
(10228,-2022) of size 1x1 on raster of 5800x4800.  
ERROR 5: Data/ortho14\_5m\_rgb\_solothurn.tif, band 2: Access window out of range in RasterIO(). Requested  
(10228,-2022) of size 1x1 on raster of 5800x4800.  
ERROR 5: Data/ortho14\_5m\_rgb\_solothurn.tif, band 3: Access window out of range in RasterIO(). Requested  
(10228,-2022) of size 1x1 on raster of 5800x4800.  
ERROR 5: Data/ortho14\_5m\_rgb\_solothurn.tif, band 1: Access window out of range in RasterIO(). Requested  
(3448,-267) of size 1x1 on raster of 5800x4800.  
ERROR 5: Data/ortho14\_5m\_rgb\_solothurn.tif, band 2: Access window out of range in RasterIO(). Requested  
(3448,-267) of size 1x1 on raster of 5800x4800.  
ERROR 5: Data/ortho14\_5m\_rgb\_solothurn.tif, band 3: Access window out of range in RasterIO(). Requested  
(3448,-267) of size 1x1 on raster of 5800x4800.  
ERROR 5: Data/ortho14\_5m\_rgb\_solothurn.tif, band 1: Access window out of range in RasterIO(). Requested  
(6424,-647) of size 1x1 on raster of 5800x4800.  
ERROR 5: Data/ortho14\_5m\_rgb\_solothurn.tif, band 2: Access window out of range in RasterIO(). Requested  
(6424,-647) of size 1x1 on raster of 5800x4800.  
ERROR 5: Data/ortho14\_5m\_rgb\_solothurn.tif, band 3: Access window out of range in RasterIO(). Requested  
(6424,-647) of size 1x1 on raster of 5800x4800.  
ERROR 5: Data/ortho14\_5m\_rgb\_solothurn.tif, band 1: Access window out of range in RasterIO(). Requested  
(9822,-2054) of size 1x1 on raster of 5800x4800.  
ERROR 5: Data/ortho14\_5m\_rgb\_solothurn.tif, band 2: Access window out of range in RasterIO(). Requested  
(9822,-2054) of size 1x1 on raster of 5800x4800.  
ERROR 5: Data/ortho14\_5m\_rgb\_solothurn.tif, band 3: Access window out of range in RasterIO(). Requested  
(9822,-2054) of size 1x1 on raster of 5800x4800.  
ERROR 5: Data/ortho14\_5m\_rgb\_solothurn.tif, band 1: Access window out of range in RasterIO(). Requested  
(7380,-462) of size 1x1 on raster of 5800x4800.  
ERROR 5: Data/ortho14\_5m\_rgb\_solothurn.tif, band 2: Access window out of range in RasterIO(). Requested  
(7380,-462) of size 1x1 on raster of 5800x4800.  
ERROR 5: Data/ortho14\_5m\_rgb\_solothurn.tif, band 3: Access window out of range in RasterIO(). Requested  
(7380,-462) of size 1x1 on raster of 5800x4800.  
ERROR 5: Data/ortho14\_5m\_rgb\_solothurn.tif, band 1: Access window out of range in RasterIO(). Requested  
(4960,-4380) of size 1x1 on raster of 5800x4800.  
ERROR 5: Data/ortho14\_5m\_rgb\_solothurn.tif, band 2: Access window out of range in RasterIO(). Requested  
(4960,-4380) of size 1x1 on raster of 5800x4800.  
ERROR 5: Data/ortho14\_5m\_rgb\_solothurn.tif, band 3: Access window out of range in RasterIO(). Requested  
(4960,-4380) of size 1x1 on raster of 5800x4800.  
ERROR 5: Data/ortho14\_5m\_rgb\_solothurn.tif, band 1: Access window out of range in RasterIO(). Requested  
(9601,-2486) of size 1x1 on raster of 5800x4800.  
ERROR 5: Data/ortho14\_5m\_rgb\_solothurn.tif, band 2: Access window out of range in RasterIO(). Requested  
(9601,-2486) of size 1x1 on raster of 5800x4800.  
ERROR 5: Data/ortho14\_5m\_rgb\_solothurn.tif, band 3: Access window out of range in RasterIO(). Requested  
(9601,-2486) of size 1x1 on raster of 5800x4800.  
ERROR 5: Data/ortho14\_5m\_rgb\_solothurn.tif, band 1: Access window out of range in RasterIO(). Requested  
(7016,312) of size 1x1 on raster of 5800x4800.  
ERROR 5: Data/ortho14\_5m\_rgb\_solothurn.tif, band 2: Access window out of range in RasterIO(). Requested  
(7016,312) of size 1x1 on raster of 5800x4800.  
ERROR 5: Data/ortho14\_5m\_rgb\_solothurn.tif, band 3: Access window out of range in RasterIO(). Requested  
(7016,312) of size 1x1 on raster of 5800x4800.  
ERROR 5: Data/ortho14\_5m\_rgb\_solothurn.tif, band 1: Access window out of range in RasterIO(). Requested  
(2654,-4649) of size 1x1 on raster of 5800x4800.  
ERROR 5: Data/ortho14\_5m\_rgb\_solothurn.tif, band 2: Access window out of range in RasterIO(). Requested  
(2654,-4649) of size 1x1 on raster of 5800x4800.  
ERROR 5: Data/ortho14\_5m\_rgb\_solothurn.tif, band 3: Access window out of range in RasterIO(). Requested  
(2654,-4649) of size 1x1 on raster of 5800x4800.  
ERROR 5: Data/ortho14\_5m\_rgb\_solothurn.tif, band 1: Access window out of range in RasterIO(). Requested  
(8954,-1379) of size 1x1 on raster of 5800x4800.  
ERROR 5: Data/ortho14\_5m\_rgb\_solothurn.tif, band 2: Access window out of range in RasterIO(). Requested  
(8954,-1379) of size 1x1 on raster of 5800x4800.  
ERROR 5: Data/ortho14\_5m\_rgb\_solothurn.tif, band 3: Access window out of range in RasterIO(). Requested  
(8954,-1379) of size 1x1 on raster of 5800x4800.  
ERROR 5: Data/ortho14\_5m\_rgb\_solothurn.tif, band 1: Access window out of range in RasterIO(). Requested  
(9951,-2629) of size 1x1 on raster of 5800x4800.  
ERROR 5: Data/ortho14\_5m\_rgb\_solothurn.tif, band 2: Access window out of range in RasterIO(). Requested  
(9951,-2629) of size 1x1 on raster of 5800x4800.  
ERROR 5: Data/ortho14\_5m\_rgb\_solothurn.tif, band 3: Access window out of range in RasterIO(). Requested  
(9951,-2629) of size 1x1 on raster of 5800x4800.  
ERROR 5: Data/ortho14\_5m\_rgb\_solothurn.tif, band 1: Access window out of range in RasterIO(). Requested  
(5110,-3647) of size 1x1 on raster of 5800x4800.  
ERROR 5: Data/ortho14\_5m\_rgb\_solothurn.tif, band 2: Access window out of range in RasterIO(). Requested  
(5110,-3647) of size 1x1 on raster of 5800x4800.  
ERROR 5: Data/ortho14\_5m\_rgb\_solothurn.tif, band 3: Access window out of range in RasterIO(). Requested  
(5110,-3647) of size 1x1 on raster of 5800x4800.  
ERROR 5: Data/ortho14\_5m\_rgb\_solothurn.tif, band 1: Access window out of range in RasterIO(). Requested  
(5362,-4131) of size 1x1 on raster of 5800x4800.  
ERROR 5: Data/ortho14\_5m\_rgb\_solothurn.tif, band 2: Access window out of range in RasterIO(). Requested  
(5362,-4131) of size 1x1 on raster of 5800x4800.  
ERROR 5: Data/ortho14\_5m\_rgb\_solothurn.tif, band 3: Access window out of range in RasterIO(). Requested  
(5362,-4131) of size 1x1 on raster of 5800x4800.  
ERROR 5: Data/ortho14\_5m\_rgb\_solothurn.tif, band 1: Access window out of range in RasterIO(). Requested  
(7919,-1970) of size 1x1 on raster of 5800x4800.  
ERROR 5: Data/ortho14\_5m\_rgb\_solothurn.tif, band 2: Access window out of range in RasterIO(). Requested  
(7919,-1970) of size 1x1 on raster of 5800x4800.  
ERROR 5: Data/ortho14\_5m\_rgb\_solothurn.tif, band 3: Access window out of range in RasterIO(). Requested  
(7919,-1970) of size 1x1 on raster of 5800x4800.  
ERROR 5: Data/ortho14\_5m\_rgb\_solothurn.tif, band 1: Access window out of range in RasterIO(). Requested  
(2602,-2140) of size 1x1 on raster of 5800x4800.  
ERROR 5: Data/ortho14\_5m\_rgb\_solothurn.tif, band 2: Access window out of range in RasterIO(). Requested  
(2602,-2140) of size 1x1 on raster of 5800x4800.  
ERROR 5: Data/ortho14\_5m\_rgb\_solothurn.tif, band 3: Access window out of range in RasterIO(). Requested  
(2602,-2140) of size 1x1 on raster of 5800x4800.  
ERROR 5: Data/ortho14\_5m\_rgb\_solothurn.tif, band 1: Access window out of range in RasterIO(). Requested  
(4421,-4277) of size 1x1 on raster of 5800x4800.  
ERROR 5: Data/ortho14\_5m\_rgb\_solothurn.tif, band 2: Access window out of range in RasterIO(). Requested  
(4421,-4277) of size 1x1 on raster of 5800x4800.  
ERROR 5: Data/ortho14\_5m\_rgb\_solothurn.tif, band 3: Access window out of range in RasterIO(). Requested  
(4421,-4277) of size 1x1 on raster of 5800x4800.  
ERROR 5: Data/ortho14\_5m\_rgb\_solothurn.tif, band 1: Access window out of range in RasterIO(). Requested  
(6396,-1061) of size 1x1 on raster of 5800x4800.  
ERROR 5: Data/ortho14\_5m\_rgb\_solothurn.tif, band 2: Access window out of range in RasterIO(). Requested  
(6396,-1061) of size 1x1 on raster of 5800x4800.  
ERROR 5: Data/ortho14\_5m\_rgb\_solothurn.tif, band 3: Access window out of range in RasterIO(). Requested  
(6396,-1061) of size 1x1 on raster of 5800x4800.  
ERROR 5: Data/ortho14\_5m\_rgb\_solothurn.tif, band 1: Access window out of range in RasterIO(). Requested  
(6391,140) of size 1x1 on raster of 5800x4800.  
ERROR 5: Data/ortho14\_5m\_rgb\_solothurn.tif, band 2: Access window out of range in RasterIO(). Requested  
(6391,140) of size 1x1 on raster of 5800x4800.  
ERROR 5: Data/ortho14\_5m\_rgb\_solothurn.tif, band 3: Access window out of range in RasterIO(). Requested  
(6391,140) of size 1x1 on raster of 5800x4800.  
ERROR 5: Data/ortho14\_5m\_rgb\_solothurn.tif, band 1: Access window out of range in RasterIO(). Requested  
(7547,61) of size 1x1 on raster of 5800x4800.  
ERROR 5: Data/ortho14\_5m\_rgb\_solothurn.tif, band 2: Access window out of range in RasterIO(). Requested  
(7547,61) of size 1x1 on raster of 5800x4800.  
ERROR 5: Data/ortho14\_5m\_rgb\_solothurn.tif, band 3: Access window out of range in RasterIO(). Requested  
(7547,61) of size 1x1 on raster of 5800x4800.  
ERROR 5: Data/ortho14\_5m\_rgb\_solothurn.tif, band 1: Access window out of range in RasterIO(). Requested  
(7077,-303) of size 1x1 on raster of 5800x4800.  
ERROR 5: Data/ortho14\_5m\_rgb\_solothurn.tif, band 2: Access window out of range in RasterIO(). Requested  
(7077,-303) of size 1x1 on raster of 5800x4800.  
ERROR 5: Data/ortho14\_5m\_rgb\_solothurn.tif, band 3: Access window out of range in RasterIO(). Requested  
(7077,-303) of size 1x1 on raster of 5800x4800.  
ERROR 5: Data/ortho14\_5m\_rgb\_solothurn.tif, band 1: Access window out of range in RasterIO(). Requested  
(8105,-1251) of size 1x1 on raster of 5800x4800.  
ERROR 5: Data/ortho14\_5m\_rgb\_solothurn.tif, band 2: Access window out of range in RasterIO(). Requested  
(8105,-1251) of size 1x1 on raster of 5800x4800.  
ERROR 5: Data/ortho14\_5m\_rgb\_solothurn.tif, band 3: Access window out of range in RasterIO(). Requested  
(8105,-1251) of size 1x1 on raster of 5800x4800.  
ERROR 5: Data/ortho14\_5m\_rgb\_solothurn.tif, band 1: Access window out of range in RasterIO(). Requested  
(6737,-116) of size 1x1 on raster of 5800x4800.  
ERROR 5: Data/ortho14\_5m\_rgb\_solothurn.tif, band 2: Access window out of range in RasterIO(). Requested  
(6737,-116) of size 1x1 on raster of 5800x4800.  
ERROR 5: Data/ortho14\_5m\_rgb\_solothurn.tif, band 3: Access window out of range in RasterIO(). Requested  
(6737,-116) of size 1x1 on raster of 5800x4800.  
ERROR 5: Data/ortho14\_5m\_rgb\_solothurn.tif, band 1: Access window out of range in RasterIO(). Requested  
(3794,-513) of size 1x1 on raster of 5800x4800.  
ERROR 5: Data/ortho14\_5m\_rgb\_solothurn.tif, band 2: Access window out of range in RasterIO(). Requested  
(3794,-513) of size 1x1 on raster of 5800x4800.  
ERROR 5: Data/ortho14\_5m\_rgb\_solothurn.tif, band 3: Access window out of range in RasterIO(). Requested  
(3794,-513) of size 1x1 on raster of 5800x4800.  
ERROR 5: Data/ortho14\_5m\_rgb\_solothurn.tif, band 1: Access window out of range in RasterIO(). Requested  
(8860,-1928) of size 1x1 on raster of 5800x4800.  
ERROR 5: Data/ortho14\_5m\_rgb\_solothurn.tif, band 2: Access window out of range in RasterIO(). Requested  
(8860,-1928) of size 1x1 on raster of 5800x4800.  
ERROR 5: Data/ortho14\_5m\_rgb\_solothurn.tif, band 3: Access window out of range in RasterIO(). Requested  
(8860,-1928) of size 1x1 on raster of 5800x4800.  
ERROR 5: Data/ortho14\_5m\_rgb\_solothurn.tif, band 1: Access window out of range in RasterIO(). Requested  
(1254,-3217) of size 1x1 on raster of 5800x4800.  
ERROR 5: Data/ortho14\_5m\_rgb\_solothurn.tif, band 2: Access window out of range in RasterIO(). Requested  
(1254,-3217) of size 1x1 on raster of 5800x4800.  
ERROR 5: Data/ortho14\_5m\_rgb\_solothurn.tif, band 3: Access window out of range in RasterIO(). Requested  
(1254,-3217) of size 1x1 on raster of 5800x4800.  
ERROR 5: Data/ortho14\_5m\_rgb\_solothurn.tif, band 1: Access window out of range in RasterIO(). Requested  
(7919,-1298) of size 1x1 on raster of 5800x4800.  
ERROR 5: Data/ortho14\_5m\_rgb\_solothurn.tif, band 2: Access window out of range in RasterIO(). Requested  
(7919,-1298) of size 1x1 on raster of 5800x4800.  
ERROR 5: Data/ortho14\_5m\_rgb\_solothurn.tif, band 3: Access window out of range in RasterIO(). Requested  
(7919,-1298) of size 1x1 on raster of 5800x4800.  
ERROR 5: Data/ortho14\_5m\_rgb\_solothurn.tif, band 1: Access window out of range in RasterIO(). Requested  
(4683,-3782) of size 1x1 on raster of 5800x4800.  
ERROR 5: Data/ortho14\_5m\_rgb\_solothurn.tif, band 2: Access window out of range in RasterIO(). Requested  
(4683,-3782) of size 1x1 on raster of 5800x4800.  
ERROR 5: Data/ortho14\_5m\_rgb\_solothurn.tif, band 3: Access window out of range in RasterIO(). Requested  
(4683,-3782) of size 1x1 on raster of 5800x4800.  
ERROR 5: Data/ortho14\_5m\_rgb\_solothurn.tif, band 1: Access window out of range in RasterIO(). Requested  
(3379,-2685) of size 1x1 on raster of 5800x4800.  
ERROR 5: Data/ortho14\_5m\_rgb\_solothurn.tif, band 2: Access window out of range in RasterIO(). Requested  
(3379,-2685) of size 1x1 on raster of 5800x4800.  
ERROR 5: Data/ortho14\_5m\_rgb\_solothurn.tif, band 3: Access window out of range in RasterIO(). Requested  
(3379,-2685) of size 1x1 on raster of 5800x4800.  
ERROR 5: Data/ortho14\_5m\_rgb\_solothurn.tif, band 1: Access window out of range in RasterIO(). Requested  
(4674,-3210) of size 1x1 on raster of 5800x4800.  
ERROR 5: Data/ortho14\_5m\_rgb\_solothurn.tif, band 2: Access window out of range in RasterIO(). Requested  
(4674,-3210) of size 1x1 on raster of 5800x4800.  
ERROR 5: Data/ortho14\_5m\_rgb\_solothurn.tif, band 3: Access window out of range in RasterIO(). Requested  
(4674,-3210) of size 1x1 on raster of 5800x4800.  
ERROR 5: Data/ortho14\_5m\_rgb\_solothurn.tif, band 1: Access window out of range in RasterIO(). Requested  
(3922,-3118) of size 1x1 on raster of 5800x4800.  
ERROR 5: Data/ortho14\_5m\_rgb\_solothurn.tif, band 2: Access window out of range in RasterIO(). Requested  
(3922,-3118) of size 1x1 on raster of 5800x4800.  
ERROR 5: Data/ortho14\_5m\_rgb\_solothurn.tif, band 3: Access window out of range in RasterIO(). Requested  
(3922,-3118) of size 1x1 on raster of 5800x4800.  
ERROR 5: Data/ortho14\_5m\_rgb\_solothurn.tif, band 1: Access window out of range in RasterIO(). Requested  
(4236,-2408) of size 1x1 on raster of 5800x4800.  
ERROR 5: Data/ortho14\_5m\_rgb\_solothurn.tif, band 2: Access window out of range in RasterIO(). Requested  
(4236,-2408) of size 1x1 on raster of 5800x4800.  
ERROR 5: Data/ortho14\_5m\_rgb\_solothurn.tif, band 3: Access window out of range in RasterIO(). Requested  
(4236,-2408) of size 1x1 on raster of 5800x4800.

#### Höhenmodellinformationen ableiten

cmd = 'gdal\_contour -a hoehe -i 150 Data/Elevation\_raster.tif Data/contour150.shp'  
os.system(cmd)

0...10...20...30...40...50...60...70...80...90...100 - done.

0

cmdList = []  
cmdList.append('gdaldem slope Data/Elevation\_raster.tif Data/Ele\_slope.tif -p')  
cmdList.append('gdaldem aspect Data/Elevation\_raster.tif Data/Ele\_aspect.tif')  
cmdList.append('gdaldem hillshade Data/Elevation\_raster.tif Data/Ele\_hillshade.tif')  
for cmd in cmdList:  
 os.system(cmd)

0...10...20...30...40...50...60...70...80...90...100 - done.  
0...10...20...30...40...50...60...70...80...90...100 - done.  
0...10...20...30...40...50...60...70...80...90...100 - done.

import os, sys  
from osgeo import gdal  
from osgeo.gdalconst import \*  
  
# register all of the drivers  
gdal.AllRegister()  
  
#Open Rasterfile  
fn = 'Data/worldmap.jpg'  
ds = gdal.Open(fn)  
if ds is None:  
 print ('Datensatz %s konnte nicht geöffnet werden' %fn)  
 sys.exit(1)  
  
os.system('gdalinfo %s' %fn)   
translatecommand = 'gdal\_translate -projwin 1680 170 2200 550 %s Data/europe.tif' %fn  
print ("command to run: %s" %translatecommand )  
os.system(translatecommand)  
  
#kleinere Kopie von Europa  
translatecommand = 'gdal\_translate -projwin 1680 170 2200 550 -outsize 50%% 50%% %s Data/europesmall.tif' %fn  
print ("command to run: %s" %translatecommand)   
os.system(translatecommand)

Driver: JPEG/JPEG JFIF  
Files: Data/worldmap.jpg  
Size is 3600, 1800  
Image Structure Metadata:  
 COMPRESSION=JPEG  
 INTERLEAVE=PIXEL  
 SOURCE\_COLOR\_SPACE=YCbCr  
Corner Coordinates:  
Upper Left ( 0.0, 0.0)  
Lower Left ( 0.0, 1800.0)  
Upper Right ( 3600.0, 0.0)  
Lower Right ( 3600.0, 1800.0)  
Center ( 1800.0, 900.0)  
Band 1 Block=3600x1 Type=Byte, ColorInterp=Red  
 Overviews: 1800x900, 900x450, 450x225  
 Image Structure Metadata:  
 COMPRESSION=JPEG  
Band 2 Block=3600x1 Type=Byte, ColorInterp=Green  
 Overviews: 1800x900, 900x450, 450x225  
 Image Structure Metadata:  
 COMPRESSION=JPEG  
Band 3 Block=3600x1 Type=Byte, ColorInterp=Blue  
 Overviews: 1800x900, 900x450, 450x225  
 Image Structure Metadata:  
 COMPRESSION=JPEG  
command to run: gdal\_translate -projwin 1680 170 2200 550 Data/worldmap.jpg Data/europe.tif  
Input file size is 3600, 1800  
0...10...20...30...40...50...60...70...80...90...100 - done.  
command to run: gdal\_translate -projwin 1680 170 2200 550 -outsize 50% 50% Data/worldmap.jpg Data/europesmall.tif  
Input file size is 3600, 1800  
0...10...20...30...40...50...60...70...80...90...100 - done.

0

### Geodaten schreiben

from osgeo import ogr  
from osgeo import osr  
  
# Räumliche Referenzsystem setzen  
srs = osr.SpatialReference()  
#srs.SetWellKnownGeogCS('WGS84')  
#srs.SetFromUserInput("EPGS:4326")  
srs.ImportFromEPSG(4326)  
  
# Alternative: srs aus existierendem Layer verwenden  
#srs.ImportFromProj4(layer.GetSpatialRef().ExportToProj4())  
  
# Datei erstellen vom Typ Esri Shapefile  
driver = ogr.GetDriverByName("ESRI Shapefile")  
destFlNm = os.path.join("Data","myFirstLyr.shp")  
  
#GeoJSON  
driver = ogr.GetDriverByName("GeoJSON")  
destFlNm = os.path.join("Data","myFirstLyr.geojson")  
  
#GeoPackage  
driver = ogr.GetDriverByName("GPKG")  
destFlNm = os.path.join("Data","myFirstLyr.gpkg")  
  
  
if os.path.exists(destFlNm):  
 driver.DeleteDataSource(destFlNm)  
destinationFile = driver.CreateDataSource(destFlNm)  
destinationLyr = destinationFile.CreateLayer("lyr",srs)  
  
# GeoPackage Test  
#destinationFile2 = driver.Open(destFlNm)  
destinationLyr2 = destinationFile.CreateLayer("lyr2",srs)  
  
# Attributdefinition: 3 Attribute: name (String), bemerkung (String), wert (Integer)  
fieldDef = ogr.FieldDefn('name',ogr.OFTString)  
fieldDef.SetWidth(50)  
destinationLyr.CreateField(fieldDef)  
fieldDef = ogr.FieldDefn('bemerkung',ogr.OFTString)  
fieldDef.SetWidth(150)  
destinationLyr.CreateField(fieldDef)  
fieldDef = ogr.FieldDefn('wert',ogr.OFTInteger)  
destinationLyr.CreateField(fieldDef)  
destinationLyr2.CreateField(fieldDef)  
  
fieldDef = ogr.FieldDefn('flaeche',ogr.OFTReal)  
destinationLyr.CreateField(fieldDef)  
  
  
# Featuredefinition - Erstellung eines Eintrags in die erstellte Layerstruktur  
ftrName = 'square'  
minEasting = 7.5  
maxEasting = 7.6  
minNorthing = 46.5  
maxNorthing = 46.6  
  
# Linearer Ring erstellen  
lR = ogr.Geometry(ogr.wkbLinearRing)  
  
# Stützpunkte definieren  
lR.AddPoint(minEasting, minNorthing)  
lR.AddPoint(maxEasting, minNorthing)  
lR.AddPoint(maxEasting, maxNorthing)  
lR.AddPoint(minEasting, maxNorthing)  
lR.AddPoint(minEasting, minNorthing)  
  
# Instanzierung der Geometrie als WkbPolygon  
sqr = ogr.Geometry(ogr.wkbPolygon)  
  
# Zuweisen der Geometrie  
sqr.AddGeometry(lR)  
ftrarea = abs(sqr.GetArea())  
  
  
# Feature mit Attribut- und Geometriedefinition  
sqrFtr = ogr.Feature(destinationLyr.GetLayerDefn())  
sqrFtr.SetGeometry(sqr)  
sqrFtr.SetField("name",ftrName)  
sqrFtr.SetField("bemerkung","Hurra, ich bin ein quadrat!")  
sqrFtr.SetField("wert",9)  
sqrFtr.SetField("flaeche",ftrarea)  
# Feature dem Layer hinzufügen  
destinationLyr.CreateFeature(sqrFtr)  
#destinationLyr2.CreateFeature(sqrFtr)  
  
# Freigabe des Featureobjekts und der Datei  
sqrFtr.Destroy()  
destinationFile.Destroy()

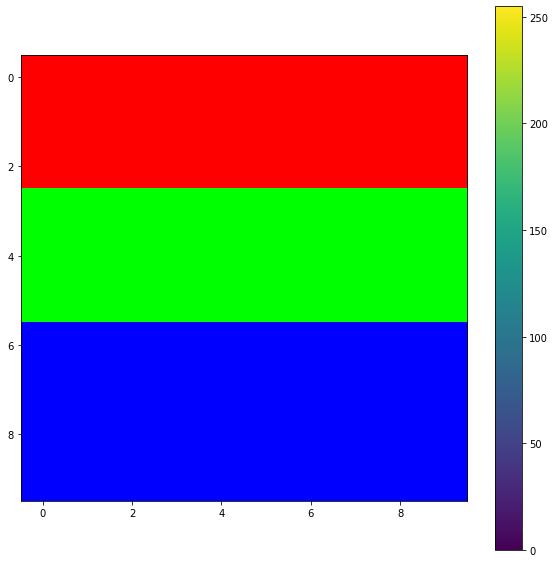
### Tag 3

#### Rasterdaten schreiben

from osgeo import gdal  
from osgeo import osr  
import numpy as np  
import matplotlib.pyplot as plt  
import matplotlib as mpl  
import matplotlib.image as mpimg  
cmap = mpl.colormaps['viridis']  
  
fn\_create = "Data/gdalCreateRaster3.tif" # filename for new raster  
driver\_gtiff = gdal.GetDriverByName('GTiff')  
ds\_create = driver\_gtiff.Create(fn\_create, xsize=10, ysize=10, bands=3, eType=gdal.GDT\_Byte)  
  
srs = osr.SpatialReference()  
srs.ImportFromEPSG(21781)  
ds\_create.SetProjection(srs.ExportToWkt())  
  
geot\_create = [600000, 10.0, 0.0, 200000, 0.0, -10.0]  
ds\_create.SetGeoTransform(geot\_create)  
print(ds\_create.GetGeoTransform())  
  
data\_createR = np.zeros((10, 10))  
data\_createR[:3, :10] = 255 # values to 1, leave outer as 0 (no data)  
data\_createG = np.zeros((10, 10))  
data\_createG[3:6, :10] = 255 # values to 1, leave outer as 0 (no data)  
data\_createB = np.zeros((10, 10))  
data\_createB[6:10, :10] = 255 # values to 1, leave outer as 0 (no data)  
  
ds\_create.GetRasterBand(1).WriteArray(data\_createR) # write the array to the raster  
ds\_create.GetRasterBand(1).SetNoDataValue(0) # set the no data value  
ds\_create.GetRasterBand(2).WriteArray(data\_createG) # write the array to the raster  
ds\_create.GetRasterBand(2).SetNoDataValue(0) # set the no data value  
ds\_create.GetRasterBand(3).WriteArray(data\_createB) # write the array to the raster  
ds\_create.GetRasterBand(3).SetNoDataValue(0) # set the no data value  
  
ds\_create = None # properly close the raster  
  
image = mpimg.imread(fn\_create)  
# plot the data values we created  
plt.figure(figsize=(10, 10))  
plt.imshow(image)  
plt.colorbar()

(600000.0, 10.0, 0.0, 200000.0, 0.0, -10.0)

<matplotlib.colorbar.Colorbar at 0x12539b310>



# Geometrie zu existierendem Datensatz hinzufügen  
  
from osgeo import ogr  
from osgeo import osr  
import random  
  
path = "Data/TM\_WORLD\_BORDERS-0.3/TM\_WORLD\_BORDERS-0.3.shp"  
driver = ogr.GetDriverByName("ESRI Shapefile")  
ds = driver.Open(path, 1)  
  
layer = ds.GetLayer(0)  
  
definition = layer.GetLayerDefn()  
  
print("Name | Type")  
for i in range(definition.GetFieldCount()):  
 name = definition.GetFieldDefn(i).GetName()  
 type = definition.GetFieldDefn(i).GetTypeName()  
 print(name, ' | ', type)  
  
feature = ogr.Feature(definition)  
  
feature.SetField("FIPS", "BLAB")  
feature.SetField("ISO2", "BLAB")  
feature.SetField("ISO3", "BLAB")  
feature.SetField("NAME", "BLABLA")  
  
start\_lat = round(random.uniform(30, 10), 2)  
start\_lon = round(random.uniform(30, 10), 2)  
  
s = str(start\_lat) + " " + str(start\_lon) + ", "  
  
for i in range(30 , 36, 2):  
 new\_lat = round(random.uniform(i, i + 1), 2)  
 new\_lon = round(random.uniform(i, i + 1) , 2)  
 s += str(new\_lat) + " " + str(new\_lon) + ", "  
  
for i in range(5, 1):  
 #new\_lat = round(random.uniform(i - 1, i), 2)  
 new\_lon = round(random.uniform(i - 1, i) , 2)  
 s += str(new\_lat) + " " + str(new\_lon) + ", "  
   
s += str(start\_lat) + " " + str(start\_lon)  
wkt = f"POLYGON(({s}))"  
print(wkt)  
polygon = ogr.CreateGeometryFromWkt(wkt)  
feature.SetGeometry(polygon)  
layer.CreateFeature(feature)  
feature.Destroy()  
ds.Destroy()

Name | Type  
FIPS | String  
ISO2 | String  
ISO3 | String  
UN | Integer  
NAME | String  
AREA | Integer  
POP2005 | Integer64  
REGION | Integer  
SUBREGION | Integer  
LON | Real  
LAT | Real  
POLYGON((14.1 14.94, 30.22 30.99, 32.32 32.2, 34.07 34.41, 14.1 14.94))

#### Stützpunkte einer Gemeinde

import osgeo.ogr as ogr  
import sys  
  
  
def extractPoints(geometry,expFl):  
 for (i) in range(geometry.GetPointCount()):  
 x,y,z = geometry.GetPoint(i)  
 expFl.write( f"{i+1},{x},{y}\n")  
   
 for i in range(geometry.GetGeometryCount()):  
 extractPoints(geometry.GetGeometryRef(i),expFl)  
  
gemname = input("Gemeindename:")  
  
logFl = f"Data/\_{gemname}.csv"  
exportfile = open(logFl, "w")  
  
  
shapefile = ogr.Open("Data/Gemeinden\_Solothurn.shp")  
if shapefile is None:  
 exportfile.write( "Datensatz konnte nicht geoeffnet werden.\n" + "\n")  
 sys.exit( 1 )  
  
layer = shapefile.GetLayer(0)  
#geometry = feature.GetGeometryRef()  
  
#Gemeindegeometry extrahieren:  
geometry = None  
for feature in layer:  
 if feature.GetField("NAME") == gemname:  
 geometry = feature.GetGeometryRef()  
 break  
  
if geometry is None:  
 exportfile.write( "\*" \* 20 + "\n")  
 exportfile.write( "Fuer %s konnte keine Geometrie ermittelt werden." %gemname + "\n")  
 exportfile.write( "\*" \* 20 + "\n")  
 sys.exit( 1 )  
   
exportfile.write( "pid,x,y\n")  
extractPoints(geometry,exportfile)  
exportfile.write( "-" \* 50 + "\n")  
print(f"Ausgabe siehe {logFl}")  
exportfile.close()

Gemeindename:Solothurn  
Ausgabe siehe Data/\_Solothurn.csv

#### Erstellung eines Datensatzes mit den MBRs aller Gemeinden

import osgeo.ogr as ogr  
import osgeo.osr as osr  
import osgeo.gdal  
import osgeo.gdalconst  
import sys  
  
sourcelayer = "Data/Gemeinden\_Solothurn.shp"  
destinationlayername = "Data/gemSoMBR.shp"  
destinationformat = "ESRI Shapefile"  
sourcefieldname = "name"  
  
shapefile = osgeo.ogr.Open(sourcelayer)  
sourcelayer = shapefile.GetLayer(0)  
srs = osr.SpatialReference()  
srs.ImportFromProj4(sourcelayer.GetSpatialRef().ExportToProj4())  
  
driver = osgeo.ogr.GetDriverByName(destinationformat)  
destinationFile = driver.CreateDataSource(destinationlayername)  
destinationLayer = destinationFile.CreateLayer(destinationlayername[0:len(destinationlayername)-4], srs)  
  
#Create Field to store the name  
fieldDef = osgeo.ogr.FieldDefn(sourcefieldname, osgeo.ogr.OFTString)  
fieldDef.SetWidth(100)  
destinationLayer.CreateField(fieldDef)  
  
feature = sourcelayer.GetNextFeature()  
while feature:  
 #Get value of Feature-Name  
 ftrName = feature.GetField(sourcefieldname)  
 #Get MBR  
 geometry = feature.GetGeometryRef()  
 minEasting,maxEasting,minNorthing,maxNorthing = geometry.GetEnvelope()  
 #print("\*"\*20)  
 #print(geometry.GetEnvelope())  
  
 linearRing = osgeo.ogr.Geometry(osgeo.ogr.wkbLinearRing)  
 linearRing.AddPoint(minEasting, minNorthing)  
 linearRing.AddPoint(maxEasting, minNorthing)  
 linearRing.AddPoint(maxEasting, maxNorthing)  
 linearRing.AddPoint(minEasting, maxNorthing)  
 linearRing.AddPoint(minEasting, minNorthing)  
 mbr = osgeo.ogr.Geometry(osgeo.ogr.wkbPolygon)  
 mbr.AddGeometry(linearRing)  
 mbrfeature = osgeo.ogr.Feature(destinationLayer.GetLayerDefn())  
 mbrfeature.SetGeometry(mbr)  
 mbrfeature.SetField(sourcefieldname, ftrName)  
 destinationLayer.CreateFeature(mbrfeature)  
 mbrfeature.Destroy()   
   
 feature = sourcelayer.GetNextFeature()  
  
shapefile.Destroy()  
destinationFile.Destroy()  
print ("Datei wurde erstellt: %s" %(destinationlayername))

Datei wurde erstellt: Data/gemSoMBR.shp

#### Datenbankverbindung

import os  
import psycopg2  
import sys  
import time  
from osgeo import ogr  
startTime = time.time()  
  
database = "postgis\_db"  
host = "ikgpgis.ethz.ch"  
usr = "casuser"  
pwd = "mrPope-2323-IKG"  
  
  
#Connect to PostgreSQL  
connection = psycopg2.connect(dbname=database, host=host, user=usr, password=pwd, port="5432")  
cursor = connection.cursor()  
  
print("connection worked")

connection worked

sqlstring = "select name, gmde\_nr, ST\_Area(geom),ST\_X(ST\_Centroid(geom)),ST\_Y(ST\_Centroid(geom)) from gemeinden\_solothurn where name like 'A%' order by name;"  
cursor.execute(sqlstring)  
  
for name, gemnr,flaeche,x,y in cursor:  
 print ("Gem: %s, Nr: %s, Fläche: %s, Zentroid X: %s / Y: %s" %(name, gemnr,flaeche,x,y))

Gem: Aedermannsdorf, Nr: 2421, Fläche: 12916288.030417854, Zentroid X: 611655.235422699 / Y: 240148.2606515081  
Gem: Aeschi, Nr: 2511, Fläche: 5476579.944437439, Zentroid X: 617482.9002888297 / Y: 224417.39465934777

sqlstring = "select ST\_X(ST\_Centroid(geom)),ST\_Y(ST\_Centroid(geom)) from gemeinden\_solothurn order by name;"  
cursor.execute(sqlstring)  
  
xList = []  
yList = []  
for x,y in cursor:  
 xList.append(x)  
 yList.append(y)

Import von CSV Daten in PostgreSQL/PostGIS

# Befehl für den Import einer ShapeDatei in eine PostgreSQL/PostGIS Datenbank  
cmd = 'ogr2ogr -f "PostgreSQL" -a\_srs "EPSG:21781" PG:"host=ikgpgis.ethz.ch port=5432 dbname=casdb2023 user=casuser password=mrPope-2323-IKG" "Data/Gemeinden\_Solothurn.shp"'  
os.system(cmd)  
  
# Befehl für den Import einer CSV Datei  
cmd = 'ogr2ogr -f "PostgreSQL" -a\_srs "EPSG:21781" PG:"host=ikgpgis.ethz.ch port=5432 dbname=casdb2023 user=casuser password=mrPope-2323-IKG" -oo X\_POSSIBLE\_NAMES=lng -oo Y\_POSSIBLE\_NAMES=lat -nlt POINT -nln "fromcsv" myfile.csv'

#### WMS Aufruf

# -\*- coding: utf-8 -\*-  
import os, shutil, sys  
import urllib.request  
from osgeo import gdal  
from osgeo.gdalconst import \*  
  
def createWorldFile(geotransform,fileName):  
 # create the 3-band raster file  
 dst\_ds = gdal.Open(fileName)  
 dst\_ds.SetGeoTransform(geotransform) # specify coords  
 srs = osr.SpatialReference() # establish encoding  
 srs.ImportFromEPSG(2056) # WGS84 lat/long  
 dst\_ds.SetProjection(srs.ExportToWkt()) # export coords to file  
 dst\_ds.FlushCache() # write to disk  
 dst\_ds = None  
  
def download(url, dest, fileName=None):  
 try:  
 r= urllib.request.urlopen(url)  
 fileName = os.path.join(dest, fileName)  
 with open(fileName, 'wb') as f:  
 shutil.copyfileobj(r,f)  
 r.close()  
 print("Successfully downloaded resource {}".format(url))  
 except:  
 print("ERROR Downloading resource {}".format(url))  
  
path2save2 = "Data/" #Zielpfad  
wmsfile = "wms.tif"  
#wmslink = "https://wms.geo.admin.ch/?SERVICE=WMS&REQUEST=GetMap&VERSION=1.3.0&LAYERS=ch.bafu.bundesinventare-bln&STYLES=default&CRS=EPSG:2056&BBOX=2550000,1060000,2660000,1140000&WIDTH=800&HEIGHT=582&FORMAT=image/png"  
minX = 2500000  
maxX = 2600000  
minY = 1060000  
maxY = 1140000  
width = 800  
height = 582  
pixSize = (maxX-minX)/width  
myBB = f"{minX},{minY},{maxX},{maxY}"  
geotransform = []  
geotransform.append(minX)  
geotransform.append(pixSize)  
geotransform.append(0.0)  
geotransform.append(minY)  
geotransform.append(0.0)  
geotransform.append(pixSize\*-1)  
  
wmslink = f"https://wms.geo.admin.ch/?SERVICE=WMS&REQUEST=GetMap&VERSION=1.3.0&LAYERS=ch.bafu.bundesinventare-bln&STYLES=default&CRS=EPSG:2056&BBOX={myBB}&WIDTH={width}&HEIGHT={height}&FORMAT=image/tiff"  
download(wmslink,path2save2,wmsfile)  
createWorldFile(geotransform,os.path.join(path2save2,wmsfile))

Successfully downloaded resource https://wms.geo.admin.ch/?SERVICE=WMS&REQUEST=GetMap&VERSION=1.3.0&LAYERS=ch.bafu.bundesinventare-bln&STYLES=default&CRS=EPSG:2056&BBOX=2500000,1060000,2600000,1140000&WIDTH=800&HEIGHT=582&FORMAT=image/tiff

Iteration über Punkte und Download der abgeleiteten Kacheln des WMS Dienstes

# -\*- coding: utf-8 -\*-  
import os, shutil, sys  
import urllib.request  
from osgeo import gdal  
from osgeo.gdalconst import \*  
  
printFlag = False  
  
def download(url, dest, fileName=None,printFlag=True):  
 try:  
 r= urllib.request.urlopen(url)  
 fileName = os.path.join(dest, fileName)  
 with open(fileName, 'wb') as f:  
 shutil.copyfileobj(r,f)  
 r.close()  
  
 if printFlag:  
 print("Successfully downloaded resource {}".format(url))  
 except Exception as e:   
 if printFlag:  
 print(e)  
 print("ERROR Downloading resource {}".format(url))  
  
path2save2 = "Data/" #Zielpfad  
for i in range(len(xList)):  
 x = xList[i]  
 y = yList[i]  
 bb = f"{x},{y},{x+5000},{y+5000}"  
  
 wmsfile = f"wms{bb}.gif"  
 wmslink = f"https://wms.geo.admin.ch/?SERVICE=WMS&REQUEST=GetMap&VERSION=1.3.0&LAYERS=ch.bafu.bundesinventare-bln&STYLES=default&CRS=EPSG:21781&BBOX={bb}&WIDTH=800&HEIGHT=582&FORMAT=image/png"  
 download(wmslink,path2save2,wmsfile,printFlag=False)

Fehlerhandling:

try:  
 a = 1/0  
except Exception as e:   
 print(e)

division by zero

### Shapely

'''  
!python -m pip uninstall shapely --yes  
  
  
!python -m pip install shapely  
'''

WARNING: Skipping shapely as it is not installed.

'\n!python -m pip install shapely\n'

!pip install shapely --no-binary shapely

DEPRECATION: --no-binary currently disables reading from the cache of locally built wheels. In the future --no-binary will not influence the wheel cache. pip 23.1 will enforce this behaviour change. A possible replacement is to use the --no-cache-dir option. You can use the flag --use-feature=no-binary-enable-wheel-cache to test the upcoming behaviour. Discussion can be found at https://github.com/pypa/pip/issues/11453  
Collecting shapely  
 Downloading shapely-2.0.3.tar.gz (280 kB)  
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 280.5/280.5 kB 617.3 kB/s eta 0:00:00a 0:00:01  
ents to build wheel ... etadata (pyproject.toml) ... ent already satisfied: numpy<2,>=1.14 in /opt/anaconda3/envs/GP4ETH/lib/python3.9/site-packages (from shapely) (1.24.1)  
Building wheels for collected packages: shapely  
 Building wheel for shapely (pyproject.toml) ... error: subprocess-exited-with-error  
   
 × Building wheel for shapely (pyproject.toml) did not run successfully.  
 │ exit code: 1  
 ╰─> [147 lines of output]  
 <string>:8: DeprecationWarning: pkg\_resources is deprecated as an API. See https://setuptools.pypa.io/en/latest/pkg\_resources.html  
 running bdist\_wheel  
 running build  
 running build\_py  
 creating build  
 creating build/lib.macosx-10.9-x86\_64-cpython-39  
 creating build/lib.macosx-10.9-x86\_64-cpython-39/shapely  
 copying shapely/predicates.py -> build/lib.macosx-10.9-x86\_64-cpython-39/shapely  
 copying shapely/\_enum.py -> build/lib.macosx-10.9-x86\_64-cpython-39/shapely  
 copying shapely/creation.py -> build/lib.macosx-10.9-x86\_64-cpython-39/shapely  
 copying shapely/strtree.py -> build/lib.macosx-10.9-x86\_64-cpython-39/shapely  
 copying shapely/coords.py -> build/lib.macosx-10.9-x86\_64-cpython-39/shapely  
 copying shapely/\_version.py -> build/lib.macosx-10.9-x86\_64-cpython-39/shapely  
 copying shapely/prepared.py -> build/lib.macosx-10.9-x86\_64-cpython-39/shapely  
 copying shapely/linear.py -> build/lib.macosx-10.9-x86\_64-cpython-39/shapely  
 copying shapely/plotting.py -> build/lib.macosx-10.9-x86\_64-cpython-39/shapely  
 copying shapely/io.py -> build/lib.macosx-10.9-x86\_64-cpython-39/shapely  
 copying shapely/measurement.py -> build/lib.macosx-10.9-x86\_64-cpython-39/shapely  
 copying shapely/affinity.py -> build/lib.macosx-10.9-x86\_64-cpython-39/shapely  
 copying shapely/\_\_init\_\_.py -> build/lib.macosx-10.9-x86\_64-cpython-39/shapely  
 copying shapely/ops.py -> build/lib.macosx-10.9-x86\_64-cpython-39/shapely  
 copying shapely/\_geometry.py -> build/lib.macosx-10.9-x86\_64-cpython-39/shapely  
 copying shapely/set\_operations.py -> build/lib.macosx-10.9-x86\_64-cpython-39/shapely  
 copying shapely/geos.py -> build/lib.macosx-10.9-x86\_64-cpython-39/shapely  
 copying shapely/wkt.py -> build/lib.macosx-10.9-x86\_64-cpython-39/shapely  
 copying shapely/errors.py -> build/lib.macosx-10.9-x86\_64-cpython-39/shapely  
 copying shapely/speedups.py -> build/lib.macosx-10.9-x86\_64-cpython-39/shapely  
 copying shapely/testing.py -> build/lib.macosx-10.9-x86\_64-cpython-39/shapely  
 copying shapely/constructive.py -> build/lib.macosx-10.9-x86\_64-cpython-39/shapely  
 copying shapely/\_ragged\_array.py -> build/lib.macosx-10.9-x86\_64-cpython-39/shapely  
 copying shapely/wkb.py -> build/lib.macosx-10.9-x86\_64-cpython-39/shapely  
 copying shapely/coordinates.py -> build/lib.macosx-10.9-x86\_64-cpython-39/shapely  
 copying shapely/validation.py -> build/lib.macosx-10.9-x86\_64-cpython-39/shapely  
 copying shapely/decorators.py -> build/lib.macosx-10.9-x86\_64-cpython-39/shapely  
 creating build/lib.macosx-10.9-x86\_64-cpython-39/shapely/algorithms  
 copying shapely/algorithms/\_oriented\_envelope.py -> build/lib.macosx-10.9-x86\_64-cpython-39/shapely/algorithms  
 copying shapely/algorithms/\_\_init\_\_.py -> build/lib.macosx-10.9-x86\_64-cpython-39/shapely/algorithms  
 copying shapely/algorithms/polylabel.py -> build/lib.macosx-10.9-x86\_64-cpython-39/shapely/algorithms  
 copying shapely/algorithms/cga.py -> build/lib.macosx-10.9-x86\_64-cpython-39/shapely/algorithms  
 creating build/lib.macosx-10.9-x86\_64-cpython-39/shapely/tests  
 copying shapely/tests/test\_constructive.py -> build/lib.macosx-10.9-x86\_64-cpython-39/shapely/tests  
 copying shapely/tests/test\_strtree.py -> build/lib.macosx-10.9-x86\_64-cpython-39/shapely/tests  
 copying shapely/tests/test\_plotting.py -> build/lib.macosx-10.9-x86\_64-cpython-39/shapely/tests  
 copying shapely/tests/test\_misc.py -> build/lib.macosx-10.9-x86\_64-cpython-39/shapely/tests  
 copying shapely/tests/test\_measurement.py -> build/lib.macosx-10.9-x86\_64-cpython-39/shapely/tests  
 copying shapely/tests/test\_set\_operations.py -> build/lib.macosx-10.9-x86\_64-cpython-39/shapely/tests  
 copying shapely/tests/\_\_init\_\_.py -> build/lib.macosx-10.9-x86\_64-cpython-39/shapely/tests  
 copying shapely/tests/test\_io.py -> build/lib.macosx-10.9-x86\_64-cpython-39/shapely/tests  
 copying shapely/tests/test\_creation.py -> build/lib.macosx-10.9-x86\_64-cpython-39/shapely/tests  
 copying shapely/tests/test\_ragged\_array.py -> build/lib.macosx-10.9-x86\_64-cpython-39/shapely/tests  
 copying shapely/tests/common.py -> build/lib.macosx-10.9-x86\_64-cpython-39/shapely/tests  
 copying shapely/tests/test\_coordinates.py -> build/lib.macosx-10.9-x86\_64-cpython-39/shapely/tests  
 copying shapely/tests/test\_predicates.py -> build/lib.macosx-10.9-x86\_64-cpython-39/shapely/tests  
 copying shapely/tests/test\_linear.py -> build/lib.macosx-10.9-x86\_64-cpython-39/shapely/tests  
 copying shapely/tests/test\_geometry.py -> build/lib.macosx-10.9-x86\_64-cpython-39/shapely/tests  
 copying shapely/tests/test\_creation\_indices.py -> build/lib.macosx-10.9-x86\_64-cpython-39/shapely/tests  
 copying shapely/tests/test\_testing.py -> build/lib.macosx-10.9-x86\_64-cpython-39/shapely/tests  
 creating build/lib.macosx-10.9-x86\_64-cpython-39/shapely/vectorized  
 copying shapely/vectorized/\_\_init\_\_.py -> build/lib.macosx-10.9-x86\_64-cpython-39/shapely/vectorized  
 creating build/lib.macosx-10.9-x86\_64-cpython-39/shapely/geometry  
 copying shapely/geometry/geo.py -> build/lib.macosx-10.9-x86\_64-cpython-39/shapely/geometry  
 copying shapely/geometry/conftest.py -> build/lib.macosx-10.9-x86\_64-cpython-39/shapely/geometry  
 copying shapely/geometry/multilinestring.py -> build/lib.macosx-10.9-x86\_64-cpython-39/shapely/geometry  
 copying shapely/geometry/multipolygon.py -> build/lib.macosx-10.9-x86\_64-cpython-39/shapely/geometry  
 copying shapely/geometry/polygon.py -> build/lib.macosx-10.9-x86\_64-cpython-39/shapely/geometry  
 copying shapely/geometry/linestring.py -> build/lib.macosx-10.9-x86\_64-cpython-39/shapely/geometry  
 copying shapely/geometry/multipoint.py -> build/lib.macosx-10.9-x86\_64-cpython-39/shapely/geometry  
 copying shapely/geometry/\_\_init\_\_.py -> build/lib.macosx-10.9-x86\_64-cpython-39/shapely/geometry  
 copying shapely/geometry/point.py -> build/lib.macosx-10.9-x86\_64-cpython-39/shapely/geometry  
 copying shapely/geometry/collection.py -> build/lib.macosx-10.9-x86\_64-cpython-39/shapely/geometry  
 copying shapely/geometry/base.py -> build/lib.macosx-10.9-x86\_64-cpython-39/shapely/geometry  
 creating build/lib.macosx-10.9-x86\_64-cpython-39/shapely/tests/legacy  
 copying shapely/tests/legacy/test\_prepared.py -> build/lib.macosx-10.9-x86\_64-cpython-39/shapely/tests/legacy  
 copying shapely/tests/legacy/test\_split.py -> build/lib.macosx-10.9-x86\_64-cpython-39/shapely/tests/legacy  
 copying shapely/tests/legacy/conftest.py -> build/lib.macosx-10.9-x86\_64-cpython-39/shapely/tests/legacy  
 copying shapely/tests/legacy/test\_transform.py -> build/lib.macosx-10.9-x86\_64-cpython-39/shapely/tests/legacy  
 copying shapely/tests/legacy/test\_products\_z.py -> build/lib.macosx-10.9-x86\_64-cpython-39/shapely/tests/legacy  
 copying shapely/tests/legacy/test\_substring.py -> build/lib.macosx-10.9-x86\_64-cpython-39/shapely/tests/legacy  
 copying shapely/tests/legacy/test\_empty\_polygons.py -> build/lib.macosx-10.9-x86\_64-cpython-39/shapely/tests/legacy  
 copying shapely/tests/legacy/test\_validation.py -> build/lib.macosx-10.9-x86\_64-cpython-39/shapely/tests/legacy  
 copying shapely/tests/legacy/test\_delaunay.py -> build/lib.macosx-10.9-x86\_64-cpython-39/shapely/tests/legacy  
 copying shapely/tests/legacy/test\_affinity.py -> build/lib.macosx-10.9-x86\_64-cpython-39/shapely/tests/legacy  
 copying shapely/tests/legacy/test\_buffer.py -> build/lib.macosx-10.9-x86\_64-cpython-39/shapely/tests/legacy  
 copying shapely/tests/legacy/test\_polygonize.py -> build/lib.macosx-10.9-x86\_64-cpython-39/shapely/tests/legacy  
 copying shapely/tests/legacy/test\_locale.py -> build/lib.macosx-10.9-x86\_64-cpython-39/shapely/tests/legacy  
 copying shapely/tests/legacy/test\_shape.py -> build/lib.macosx-10.9-x86\_64-cpython-39/shapely/tests/legacy  
 copying shapely/tests/legacy/test\_pickle.py -> build/lib.macosx-10.9-x86\_64-cpython-39/shapely/tests/legacy  
 copying shapely/tests/legacy/threading\_test.py -> build/lib.macosx-10.9-x86\_64-cpython-39/shapely/tests/legacy  
 copying shapely/tests/legacy/test\_make\_valid.py -> build/lib.macosx-10.9-x86\_64-cpython-39/shapely/tests/legacy  
 copying shapely/tests/legacy/test\_operations.py -> build/lib.macosx-10.9-x86\_64-cpython-39/shapely/tests/legacy  
 copying shapely/tests/legacy/test\_singularity.py -> build/lib.macosx-10.9-x86\_64-cpython-39/shapely/tests/legacy  
 copying shapely/tests/legacy/test\_parallel\_offset.py -> build/lib.macosx-10.9-x86\_64-cpython-39/shapely/tests/legacy  
 copying shapely/tests/legacy/test\_box.py -> build/lib.macosx-10.9-x86\_64-cpython-39/shapely/tests/legacy  
 copying shapely/tests/legacy/test\_operators.py -> build/lib.macosx-10.9-x86\_64-cpython-39/shapely/tests/legacy  
 copying shapely/tests/legacy/\_\_init\_\_.py -> build/lib.macosx-10.9-x86\_64-cpython-39/shapely/tests/legacy  
 copying shapely/tests/legacy/test\_snap.py -> build/lib.macosx-10.9-x86\_64-cpython-39/shapely/tests/legacy  
 copying shapely/tests/legacy/test\_persist.py -> build/lib.macosx-10.9-x86\_64-cpython-39/shapely/tests/legacy  
 copying shapely/tests/legacy/test\_orient.py -> build/lib.macosx-10.9-x86\_64-cpython-39/shapely/tests/legacy  
 copying shapely/tests/legacy/test\_ndarrays.py -> build/lib.macosx-10.9-x86\_64-cpython-39/shapely/tests/legacy  
 copying shapely/tests/legacy/test\_union.py -> build/lib.macosx-10.9-x86\_64-cpython-39/shapely/tests/legacy  
 copying shapely/tests/legacy/test\_voronoi\_diagram.py -> build/lib.macosx-10.9-x86\_64-cpython-39/shapely/tests/legacy  
 copying shapely/tests/legacy/test\_linear\_referencing.py -> build/lib.macosx-10.9-x86\_64-cpython-39/shapely/tests/legacy  
 copying shapely/tests/legacy/test\_cga.py -> build/lib.macosx-10.9-x86\_64-cpython-39/shapely/tests/legacy  
 copying shapely/tests/legacy/test\_geointerface.py -> build/lib.macosx-10.9-x86\_64-cpython-39/shapely/tests/legacy  
 copying shapely/tests/legacy/test\_create\_inconsistent\_dimensionality.py -> build/lib.macosx-10.9-x86\_64-cpython-39/shapely/tests/legacy  
 copying shapely/tests/legacy/test\_wkb.py -> build/lib.macosx-10.9-x86\_64-cpython-39/shapely/tests/legacy  
 copying shapely/tests/legacy/test\_shared\_paths.py -> build/lib.macosx-10.9-x86\_64-cpython-39/shapely/tests/legacy  
 copying shapely/tests/legacy/test\_clip\_by\_rect.py -> build/lib.macosx-10.9-x86\_64-cpython-39/shapely/tests/legacy  
 copying shapely/tests/legacy/test\_linemerge.py -> build/lib.macosx-10.9-x86\_64-cpython-39/shapely/tests/legacy  
 copying shapely/tests/legacy/test\_predicates.py -> build/lib.macosx-10.9-x86\_64-cpython-39/shapely/tests/legacy  
 copying shapely/tests/legacy/test\_invalid\_geometries.py -> build/lib.macosx-10.9-x86\_64-cpython-39/shapely/tests/legacy  
 copying shapely/tests/legacy/test\_polylabel.py -> build/lib.macosx-10.9-x86\_64-cpython-39/shapely/tests/legacy  
 copying shapely/tests/legacy/test\_equality.py -> build/lib.macosx-10.9-x86\_64-cpython-39/shapely/tests/legacy  
 copying shapely/tests/legacy/test\_svg.py -> build/lib.macosx-10.9-x86\_64-cpython-39/shapely/tests/legacy  
 copying shapely/tests/legacy/test\_nearest.py -> build/lib.macosx-10.9-x86\_64-cpython-39/shapely/tests/legacy  
 copying shapely/tests/legacy/test\_minimum\_clearance.py -> build/lib.macosx-10.9-x86\_64-cpython-39/shapely/tests/legacy  
 copying shapely/tests/legacy/test\_wkt.py -> build/lib.macosx-10.9-x86\_64-cpython-39/shapely/tests/legacy  
 copying shapely/tests/legacy/test\_vectorized.py -> build/lib.macosx-10.9-x86\_64-cpython-39/shapely/tests/legacy  
 copying shapely/tests/legacy/test\_mapping.py -> build/lib.macosx-10.9-x86\_64-cpython-39/shapely/tests/legacy  
 creating build/lib.macosx-10.9-x86\_64-cpython-39/shapely/tests/geometry  
 copying shapely/tests/geometry/test\_emptiness.py -> build/lib.macosx-10.9-x86\_64-cpython-39/shapely/tests/geometry  
 copying shapely/tests/geometry/test\_point.py -> build/lib.macosx-10.9-x86\_64-cpython-39/shapely/tests/geometry  
 copying shapely/tests/geometry/test\_decimal.py -> build/lib.macosx-10.9-x86\_64-cpython-39/shapely/tests/geometry  
 copying shapely/tests/geometry/\_\_init\_\_.py -> build/lib.macosx-10.9-x86\_64-cpython-39/shapely/tests/geometry  
 copying shapely/tests/geometry/test\_collection.py -> build/lib.macosx-10.9-x86\_64-cpython-39/shapely/tests/geometry  
 copying shapely/tests/geometry/test\_polygon.py -> build/lib.macosx-10.9-x86\_64-cpython-39/shapely/tests/geometry  
 copying shapely/tests/geometry/test\_multipolygon.py -> build/lib.macosx-10.9-x86\_64-cpython-39/shapely/tests/geometry  
 copying shapely/tests/geometry/test\_hash.py -> build/lib.macosx-10.9-x86\_64-cpython-39/shapely/tests/geometry  
 copying shapely/tests/geometry/test\_linestring.py -> build/lib.macosx-10.9-x86\_64-cpython-39/shapely/tests/geometry  
 copying shapely/tests/geometry/test\_multipoint.py -> build/lib.macosx-10.9-x86\_64-cpython-39/shapely/tests/geometry  
 copying shapely/tests/geometry/test\_coords.py -> build/lib.macosx-10.9-x86\_64-cpython-39/shapely/tests/geometry  
 copying shapely/tests/geometry/test\_geometry\_base.py -> build/lib.macosx-10.9-x86\_64-cpython-39/shapely/tests/geometry  
 copying shapely/tests/geometry/test\_equality.py -> build/lib.macosx-10.9-x86\_64-cpython-39/shapely/tests/geometry  
 copying shapely/tests/geometry/test\_multi.py -> build/lib.macosx-10.9-x86\_64-cpython-39/shapely/tests/geometry  
 copying shapely/tests/geometry/test\_multilinestring.py -> build/lib.macosx-10.9-x86\_64-cpython-39/shapely/tests/geometry  
 copying shapely/tests/geometry/test\_format.py -> build/lib.macosx-10.9-x86\_64-cpython-39/shapely/tests/geometry  
 copying shapely/\_geos.pxd -> build/lib.macosx-10.9-x86\_64-cpython-39/shapely  
 copying shapely/\_pygeos\_api.pxd -> build/lib.macosx-10.9-x86\_64-cpython-39/shapely  
 UPDATING build/lib.macosx-10.9-x86\_64-cpython-39/shapely/\_version.py  
 set build/lib.macosx-10.9-x86\_64-cpython-39/shapely/\_version.py to '2.0.3'  
 running build\_ext  
 building 'shapely.lib' extension  
 creating build/temp.macosx-10.9-x86\_64-cpython-39  
 creating build/temp.macosx-10.9-x86\_64-cpython-39/src  
 clang -Wno-unused-result -Wsign-compare -Wunreachable-code -DNDEBUG -fwrapv -O2 -Wall -fPIC -O2 -isystem /opt/anaconda3/envs/GP4ETH/include -fPIC -O2 -isystem /opt/anaconda3/envs/GP4ETH/include -I./src -I/opt/anaconda3/envs/GP4ETH/include -I/opt/anaconda3/envs/GP4ETH/include/python3.9 -I/private/var/folders/60/hv67rq\_53831s0j6stgblk580000gn/T/pip-build-env-l7owfl09/overlay/lib/python3.9/site-packages/numpy/core/include -c src/c\_api.c -o build/temp.macosx-10.9-x86\_64-cpython-39/src/c\_api.o  
 You have not agreed to the Xcode license agreements. Please run 'sudo xcodebuild -license' from within a Terminal window to review and agree to the Xcode and Apple SDKs license.  
 error: command '/usr/bin/clang' failed with exit code 69  
 [end of output]  
   
 note: This error originates from a subprocess, and is likely not a problem with pip.  
 ERROR: Failed building wheel for shapely  
ERROR: Could not build wheels for shapely, which is required to install pyproject.toml-based projects

from osgeo import ogr  
from shapely import wkt

shapefile = ogr.Open("Data/Gemeinden\_Solothurn.shp")  
if shapefile is None:  
 print ("Datensatz konnte nicht geoeffnet werden.\n")  
 sys.exit()  
  
layer = shapefile.GetLayer(0)  
  
#Gemeindegeometry extrahieren:  
geometry = None  
cnt = 0  
feature = layer.GetNextFeature()  
for feature in layer:  
 while cnt < 1:  
 #Extract Gemeinde-Name  
 gemname = feature.GetField("gmde\_name")  
 #Get Geometry (Polygon)  
 gemgeometry = feature.GetGeometryRef()  
 #"Convert" Geometry to shapely-geometry  
 gemgeomaswkt = gemgeometry.ExportToWkt()  
 shapelypolygon = wkt.loads(gemgeomaswkt)  
 #Extract Centroid  
 centroid\_point = shapelypolygon.centroid  
 x=centroid\_point.x  
 y=centroid\_point.y  
 area = shapelypolygon.area  
 #Printout Information  
 print ("Gemeinde %s hat folgenden Zentroid: (%f, %f) und folgende Flaeche %fm2" %(gemname, x, y, area))  
 cnt += 1  
 feature = layer.GetNextFeature()

Gemeinde Rohr hat folgenden Zentroid: (638811.067724, 251323.787658) und folgende Flaeche 2229578.988870m2

# Beispiel für die Ermittlung eines gemeinsamen Grenzverlaufs  
  
import osgeo.ogr  
import shapely.wkt  
import sys  
#from numpy import array  
#from pprint import pprint  
from shapely.geometry import LineString  
  
shapefile = osgeo.ogr.Open("Data/Gemeinden\_Solothurn.shp")  
if shapefile is None:  
 print ("Datensatz konnte nicht geoeffnet werden.\n")  
 sys.exit( 1 )  
  
layer = shapefile.GetLayer(0)  
#feature = layer.GetFeature(1)  
  
#Gemeindegeometry extrahieren:  
geometry = None  
for feature in layer:  
 #Extract Geometries for Seewen and Nunningen  
 if feature.GetField("NAME") == 'Seewen':  
 geomSeewen = feature.GetGeometryRef()  
 Seewengeomaswkt = geomSeewen.ExportToWkt()  
 shapelypolygonSeewen = shapely.wkt.loads(Seewengeomaswkt)  
 elif feature.GetField("NAME") == 'Nunningen':  
 geomNunningen = feature.GetGeometryRef()  
 Nunningengeomaswkt = geomNunningen.ExportToWkt()  
 shapelypolygonNunningen = shapely.wkt.loads(Nunningengeomaswkt)  
  
  
#compute intersection  
intersectionline = shapelypolygonSeewen.intersection(shapelypolygonNunningen)  
  
type = intersectionline.geom\_type  
vertices = len(intersectionline.geoms)  
print ("")  
print ("Laenge des gemeinsamen Grenzverlaufs (vom Typ %s): %fm mit %i Liniensegmenten" %(type, intersectionline.length, vertices))  
print ("")  
  
#Extraktion of Vertices of intersectionline  
i=0  
for vertex in intersectionline.geoms:  
 i=i+1  
 x1 = vertex.coords[0][0] # 1. Punkt der Linie, X-Koordinate  
 y1 = vertex.coords[0][1] # 1. Punkt der Linie, Y-Koordinate  
   
 print ("Stuetzpunkt[%i]: (x=%f, y=%f)" %(i,x1,y1))  
 if i==len(intersectionline.geoms):  
 x1=vertex.coords[1][0] # 2. Punkt der Linie, X-Koordinate  
 y1=vertex.coords[1][1] # 2. Punkt der Linie, Y-Koordinate  
 print ("Stuetzpunkt[%i]: (x=%f, y=%f)" %(i+1,x1,y1))

Laenge des gemeinsamen Grenzverlaufs (vom Typ MultiLineString): 2767.687874m mit 13 Liniensegmenten  
  
Stuetzpunkt[1]: (x=614649.601000, y=251042.156000)  
Stuetzpunkt[2]: (x=614278.328000, y=251220.798000)  
Stuetzpunkt[3]: (x=614213.930000, y=251268.670000)  
Stuetzpunkt[4]: (x=614072.524000, y=251380.196000)  
Stuetzpunkt[5]: (x=614038.401000, y=251406.899000)  
Stuetzpunkt[6]: (x=613985.761000, y=251448.735000)  
Stuetzpunkt[7]: (x=613299.161000, y=251527.460000)  
Stuetzpunkt[8]: (x=612600.435000, y=251326.330000)  
Stuetzpunkt[9]: (x=612400.456000, y=251417.959000)  
Stuetzpunkt[10]: (x=612310.557000, y=251563.695000)  
Stuetzpunkt[11]: (x=612279.430000, y=251595.266000)  
Stuetzpunkt[12]: (x=612242.692000, y=251633.715000)  
Stuetzpunkt[13]: (x=612208.757000, y=251649.142000)  
Stuetzpunkt[14]: (x=612177.337000, y=251674.815000)

### Fiona

import fiona  
  
c = fiona.open('Data/Gemeinden\_Solothurn.shp', 'r')  
print("Anzahl Datensätze: %i " %len(list(c)))  
print("Format: %s" %c.driver)  
print("Geo-Referenzsystem: %s" %c.crs)  
print("Ausdehnung: %s" %str(c.bounds))

Anzahl Datensätze: 109   
Format: ESRI Shapefile  
Geo-Referenzsystem: {'init': 'epsg:21781'}  
Ausdehnung: (592560.389, 213702.99, 644759.038, 261329.631)

import fiona  
# Shapely wird für die Definition der Geometrie benötigt  
from shapely.geometry import Point,LineString,Polygon, mapping  
import random  
#from pathvariable import datapath  
  
# Schema: einfaches Dictionary mit Geometrie und Properties als keys  
schema\_pnt = {'geometry': 'Point','properties': {'Id': 'int', 'Name': 'str'}}  
  
# Ein paar Punktgeometrien  
points = [Point(272830.63, 155125.73),Point(273770.32,155467.75),Point(273536.47,155914.07),Point(272033.12,152265.71)]  
expFl = os.path.join('Data', 'myrandompointshapes2.shp')  
with fiona.open(expFl, 'w', 'ESRI Shapefile', schema\_pnt) as pntlayer:  
 for cnt in range(1,400):  
 # Schema befüllen  
 elem = {}  
 # Geometrie wird mit der mapping function von shapely erstellt  
 y=random.randrange(1000000,2000000)  
 x=random.randrange(2000000,3000000)  
 pnt = Point(x,y)  
 elem['geometry'] = mapping(pnt)  
 # Attributwerte  
 elem['properties'] = {'Name': 'Punkt ' + str(cnt), 'Id' : cnt}  
 # Erstellen des neuen Datensatzes / Records  
 pntlayer.write(elem)

### Folium

import folium, json  
m = folium.Map(location=[47.3, 7.61], zoom\_start=10)  
  
infoHtmlText = """<table style="width:100%">  
 <tr>  
 <th>Company</th>  
 <th>Contact</th>  
 <th>Country</th>  
 </tr>  
 <tr>  
 <td>Alfreds Futterkiste</td>  
 <td>Maria Anders</td>  
 <td>Germany</td>  
 </tr>  
 <tr>  
 <td>Centro comercial Moctezuma</td>  
 <td>Francisco Chang</td>  
 <td>Mexico</td>  
 </tr>  
</table>"""  
  
folium.Marker(  
 location=[47.40875, 8.50778],  
 popup=infoHtmlText,  
 icon=folium.Icon(color="red", icon="info-sign"),  
).add\_to(m)  
  
  
rfile = open('Data/Gemeinden\_SolothurnWGS84.json', 'r', encoding='utf-8').read()   
jsonData = json.loads(rfile)   
style\_function = {  
 'fillColor': 'white',  
 }  
folium.GeoJson(jsonData, name='json\_data',#,  
 #style\_function=lambda x: style\_function  
  
 ).add\_to(m)  
  
m

<folium.folium.Map at 0x18147d400>

#### Leafmap

<https://leafmap.org/notebooks>

#!pip install leafmap

import leafmap.foliumap as leafmap  
m = leafmap.Map(center=(47.5, 7.65), zoom=12)  
m.add\_basemap("OpenTopoMap")  
  
in\_geojson = "Data/Gemeinden\_SolothurnWGS84.json"  
m.add\_geojson(in\_geojson, layer\_name="Gemeinden Solothurn")  
  
pk25 = "https://wms.geo.admin.ch/?"  
m.add\_wms\_layer(  
 url=pk25,  
 layers="ch.swisstopo.pixelkarte-farbe-pk25.noscale",  
 name="PK25",  
 #attribution="MRLC",  
 format="image/png",  
 shown=True,  
)  
  
#Wetterstationen Zürich  
in\_geojson = "https://data.stadt-zuerich.ch/dataset/ugz\_meteodaten\_tagesmittelwerte/download/uzg\_ogd\_metadaten.json"  
m.add\_geojson(in\_geojson, layer\_name="Wetterstationen Stadt Zürich")  
  
#m.add\_legend(title="Pixelkarte 1:25'000")  
m

---------------------------------------------------------------------------  
KeyError Traceback (most recent call last)  
Input In [46], in <cell line: 20>()  
 18 #Wetterstationen Zürich  
 19 in\_geojson = "https://data.stadt-zuerich.ch/dataset/ugz\_meteodaten\_tagesmittelwerte/download/uzg\_ogd\_metadaten.json"  
---> 20 m.add\_geojson(in\_geojson, layer\_name="Wetterstationen Stadt Zürich")  
 22 #m.add\_legend(title="Pixelkarte 1:25'000")  
 23 m  
  
File /opt/anaconda3/envs/GP4ETH/lib/python3.9/site-packages/leafmap/foliumap.py:1504, in Map.add\_geojson(self, in\_geojson, layer\_name, encoding, info\_mode, zoom\_to\_layer, \*\*kwargs)  
 1502 kwargs.pop("fields")  
 1503 else:  
-> 1504 props = list(data["features"][0]["properties"].keys())  
 1505 if info\_mode == "on\_hover":  
 1506 tooltip = folium.GeoJsonTooltip(fields=props)  
  
KeyError: 'features'

# add a heatmap  
m2 = leafmap.Map()  
  
in\_csv = "https://raw.githubusercontent.com/opengeos/leafmap/master/examples/data/world\_cities.csv"  
m2.add\_heatmap(  
 in\_csv,  
 latitude="latitude",  
 longitude="longitude",  
 value="pop\_max",  
 name="Heat map",  
 radius=20,  
)  
colors = ["blue", "lime", "red"]  
vmin = 0  
vmax = 10000  
m2.add\_colorbar(colors=colors, vmin=vmin, vmax=vmax)  
m2.add\_title("World Population Heat Map", font\_size="20px", align="center")  
m2

<leafmap.foliumap.Map at 0x16559c0d0>

# DuckDB

import duckdb  
  
# Erweiterung installieren  
duckdb.install\_extension('spatial')  
  
# Erweiterung laden  
duckdb.load\_extension('spatial')

cursor = duckdb.connect()  
print(cursor.execute('SELECT 42').fetchall())  
  
#FROM duckdb\_extensions();

[(42,)]

def runSql(sqlStmt):  
 print(sqlStmt)  
 result = duckdb.sql(sqlStmt)  
 print(result)  
 return result

testFl = '/Users/hansjoerg.stark/development/apachesuperset/Daten/fakeBIM.xlsx'  
# Lese Daten aus einer Excel-Datei  
sqlStmt = f"CREATE TABLE fakebim AS SELECT \* FROM st\_read('{testFl}');"  
  
#sqlStmt = f"SELECT \* FROM st\_read('{testFl}')"  
runSql(sqlStmt)

CREATE TABLE fakebim AS SELECT \* FROM st\_read('/Users/hansjoerg.stark/development/apachesuperset/Daten/fakeBIM.xlsx');

---------------------------------------------------------------------------  
CatalogException Traceback (most recent call last)  
Input In [36], in <cell line: 6>()  
 3 sqlStmt = f"CREATE TABLE fakebim AS SELECT \* FROM st\_read('{testFl}');"  
 5 #sqlStmt = f"SELECT \* FROM st\_read('{testFl}')"  
----> 6 runSql(sqlStmt)  
  
Input In [35], in runSql(sqlStmt)  
 1 def runSql(sqlStmt):  
 2 print(sqlStmt)  
----> 3 result = duckdb.sql(sqlStmt)  
 4 print(result)  
 5 return result  
  
CatalogException: Catalog Error: Table with name "fakebim" already exists!

sqlStmt = f"SELECT COLUMN\_NAME FROM INFORMATION\_SCHEMA.COLUMNS WHERE TABLE\_NAME = 'fakebim';"  
runSql(sqlStmt)

SELECT COLUMN\_NAME FROM INFORMATION\_SCHEMA.COLUMNS WHERE TABLE\_NAME = 'fakebim';  
┌───────────────────────┐  
│ column\_name │  
│ varchar │  
├───────────────────────┤  
│ id │  
│ projid │  
│ Bedarf │  
│ nutzung │  
│ ort │  
│ vorhaben │  
│ jahr │  
│ kOttoenprognose │  
│ Klasse │  
│ DatumEnde │  
│ Region │  
│ datumAbnahme │  
│ lon │  
│ lat │  
│ Kritisch │  
│ projektgestartet │  
│ tageseitprojektstart │  
│ projektabgeschlossen │  
│ anztagebisprojektende │  
├───────────────────────┤  
│ 19 rows │  
└───────────────────────┘

┌───────────────────────┐  
│ column\_name │  
│ varchar │  
├───────────────────────┤  
│ id │  
│ projid │  
│ Bedarf │  
│ nutzung │  
│ ort │  
│ vorhaben │  
│ jahr │  
│ kOttoenprognose │  
│ Klasse │  
│ DatumEnde │  
│ Region │  
│ datumAbnahme │  
│ lon │  
│ lat │  
│ Kritisch │  
│ projektgestartet │  
│ tageseitprojektstart │  
│ projektabgeschlossen │  
│ anztagebisprojektende │  
├───────────────────────┤  
│ 19 rows │  
└───────────────────────┘

sqlStmt = f"SELECT Region,count(\*) from fakebim group by Region;"  
result = runSql(sqlStmt)  
  
# Alle Zeilen abrufen  
rows = result.fetchall()  
  
# Iteration über die Zeilen  
for key,value in rows:  
 print(key,value)

SELECT Region,count(\*) from fakebim group by Region;  
┌──────────┬──────────────┐  
│ Region │ count\_star() │  
│ varchar │ int64 │  
├──────────┼──────────────┤  
│ Samuel │ 8 │  
│ NULL │ 6 │  
│ Monika │ 178 │  
│ Stefanie │ 51 │  
│ Zita │ 164 │  
│ Wolfgang │ 140 │  
│ Otto │ 94 │  
└──────────┴──────────────┘  
  
Zita 164  
Monika 178  
Stefanie 51  
Wolfgang 140  
Otto 94  
Samuel 8  
None 6