Package Installation for Geodata Processing

In [60]:
```
import warnings
warnings.filterwarnings('ignore')
```

In [ ]:
```
import sys,os
!{sys.executable} -m pip install gdal
!{sys.executable} -m pip install psycopg2
!{sys.executable} -m pip install shapely
!{sys.executable} -m pip install fiona
!{sys.executable} -m pip install rasterio
!{sys.executable} -m pip install geopandas
!{sys.executable} -m pip install folium
!{sys.executable} -m pip install ipython-notebook
```

If the following commands may not be installed via Jupyter Notebook. Try it and if it fails, run the following commands in a command shell:

In [62]:
```
import os
os.system('conda install -c conda-forge ipyleaflet')
os.system('conda install -c conda-forge ipython-notebook')
```

```
Collecting package metadata (current_repodata.json): ...working... done
Solving environment: ...working... done

# All requested packages already installed.

Collecting package metadata (current_repodata.json): ...working... done
Solving environment: ...working... failed with initial frozen solve. Retrying
with flexible solve.
Collecting package metadata (repodata.json): ...working... done
Solving environment: ...working... failed with initial frozen solve. Retrying
with flexible solve.
PackagesNotFoundError: The following packages are not available from current c
hannels:

  - ipython-notebook

Current channels:

  - https://conda.anaconda.org/conda-forge/osx-64
  - https://conda.anaconda.org/conda-forge/noarch
  - https://repo.anaconda.com/pkgs/main/osx-64
  - https://repo.anaconda.com/pkgs/main/noarch
  - https://repo.anaconda.com/pkgs/r/osx-64
  - https://repo.anaconda.com/pkgs/r/noarch

To search for alternate channels that may provide the conda package you're
looking for, navigate to

    https://anaconda.org

and use the search bar at the top of the page.
```

Out[62]:
```
256
```

Test the Installation

In [63]:
```
import importlib
def importPackages(modules):
    for library in modules:
        try:
            exec("from {module} import *".format(module=library))
            print("Package {} successfully imported!".format(library))
        except Exception as e:
            print("\n\n E R R O R !!! With Package {}".format(library))
            print(e)
    #print(dir()) # Exactly working as thought
pckList = []
pckList.append('gdal')
pckList.append('psycopg2')
pckList.append('shapely')
pckList.append('fiona')
pckList.append('rasterio')
pckList.append('geopandas')
pckList.append('folium')

importPackages(pckList)
```

```
Package gdal successfully imported!
Package psycopg2 successfully imported!
Package shapely successfully imported!
Package fiona successfully imported!
Package rasterio successfully imported!
Package geopandas successfully imported!
Package folium successfully imported!
```

Test Database connection to PostgreSQL/PostGIS

In [64]:
```
import psycopg2
def postgres_test(dbn,h,u,pwd,p):
    try:
        conn = psycopg2.connect(dbname=dbn, host=h, user=u, password=pwd, por
        conn.close()
        return True
    except:
        return False

database = "postgis"
host = "ikgsql2.ethz.ch"
usr = "postgres"
pwd = "tur4finupum9"
port = "5432"
print(postgres_test(database,usr,host,pwd,port))
```

```
False
```

Test Reading Data

Copy from one format (Esri Shape) to another (MapInfo) using commandline tool `ogr2ogr`

In [65]:
```
import os
os.system('ogr2ogr -f "MapInfo File" ../Data/Gemeinden_Solothurn.tab ../Data/(
```

```
Warning 1: The output driver does not seem to natively support Integer64 type
```

```
for field gem_bfs. Converting it to Real instead. -mapFieldType can be used to
control field type conversion.
Warning 1: The output driver does not seem to natively support Integer64 type
for field gmde_nr. Converting it to Real instead. -mapFieldType can be used to
control field type conversion.
Warning 1: The output driver does not seem to natively support Integer64 type
for field bzrk_nr. Converting it to Real instead. -mapFieldType can be used to
control field type conversion.
Warning 1: The output driver does not seem to natively support Integer64 type
for field eg_nr. Converting it to Real instead. -mapFieldType can be used to c
ontrol field type conversion.
Warning 1: The output driver does not seem to natively support Integer64 type
for field plz. Converting it to Real instead. -mapFieldType can be used to con
trol field type conversion.
Warning 1: The output driver does not seem to natively support Integer64 type
for field archive. Converting it to Real instead. -mapFieldType can be used to
control field type conversion.
```

Out[65]:
```
0
```

Read some data..

You should see something like:

```
<osgeo.ogr.DataSource; proxy of <Swig Object of type
'OGRDataSourceShadow *' at 0x7fab289df960> >
Gemeinden_Solothurn
109
Der Datensatz 'Gemeinden_Solothurn' hat 109 Gemeinden
```

In [66]:
```
import ogr
drv = ogr.GetDriverByName("ESRI Shapefile")
path2ds = "../Data/Gemeinden_Solothurn.shp"
datasource = drv.Open(path2ds)
print(datasource)

mylayer = datasource.GetLayer(0)
print(mylayer.GetName())

ftrcnt = mylayer.GetFeatureCount()
print(ftrcnt)

print("Der Datensatz '%s' hat %i Gemeinden" %(mylayer.GetName(),int(ftrcnt)))
```

```
<osgeo.ogr.DataSource; proxy of <Swig Object of type 'OGRDataSourceShadow *' a
t 0x7fab1b152090> >
Gemeinden_Solothurn
109
Der Datensatz 'Gemeinden_Solothurn' hat 109 Gemeinden
```

Umprojektion

You should see something like:

```
Ausgangskoordinaten: POINT (1120351.57 741921.42)
Transformierte Koordinaten: POINT (47.3488013802885
-122.598135130878)
```

In [67]:
```
import ogr
import osr
```

```
source = osr.SpatialReference()
source.ImportFromEPSG(2927)

target = osr.SpatialReference()
target.ImportFromEPSG(4326)

transform = osr.CoordinateTransformation(source, target)

origpnt = "POINT (1120351.57 741921.42)"
transfpoint = ogr.CreateGeometryFromWkt(origpnt)
transfpoint.Transform(transform)
print("Ausgangskoordinaten: %s" %origpnt)
print("Transformierte Koordinaten: %s" %transfpoint.ExportToWkt())
```

```
Ausgangskoordinaten: POINT (1120351.57 741921.42)
Transformierte Koordinaten: POINT (47.3488013802885 -122.598135130878)
```

Rasterdaten

You should see something like:

```
Anzahl Spalten: 5800
Anzahl Zeilen: 4800
Anzahl Baender: 3
```

In [68]:

```
import gdal

fn = '../Data/ortho14_5m_rgb_solothurn.tif'
ds = gdal.Open(fn)
if ds is None:
    print ('Datensatz %s konnte nicht geöffnet werden!' %fn)
    sys.exit()

#Dimension des Rasterbildes
cols = ds.RasterXSize
rows = ds.RasterYSize
bands = ds.RasterCount

print ("Anzahl Spalten: %d" %cols)
print ("Anzahl Zeilen: %d" %rows)
print ("Anzahl Baender: %d" %bands)
```

```
Anzahl Spalten: 5800
Anzahl Zeilen: 4800
Anzahl Baender: 3
```

Commandline commands

You should see something like:

```
command to run: gdaldem slope ../Data/Elevation_raster.tif
../Data/Ele_slope.tif -s 10000
0...10...20...30...40...50...60...70...80...90...100 - done.
```

In [69]:

```
imagefilename = "../Data/Elevation_raster.tif"
path = "../Data/"

sloapcommand = 'gdaldem slope %s %sEle_slope.tif -s 10000' %(imagefilename,pa
print ("command to run: %s" %sloapcommand)   #gdaldem slope input_dem output_
```

```
os.system(sloapcommand)
```

```
command to run: gdaldem slope ../Data/Elevation_raster.tif ../Data/Ele_slope.t
if -s 10000
0...10...20...30...40...50...60...70...80...90...100 - done.
```

Out[69]:
```
0
```

Geowebservices

You should see something like:

```
Successfully downloaded resource http://wms.geo.admin.ch/?
SERVICE=WMS&REQUEST=GetMap&VERSION=1.3.0&LAYERS=ch.bfs.arealstatistik-
1985-
04&STYLES=default&CRS=EPSG:21781&BBOX=550000,60000,660000,140000&WIDTH
```

In [70]:
```
# -*- coding: utf-8 -*-
import os, shutil, sys
import urllib.request
import gdal
from gdalconst import *

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.gif"
wmslink = "http://wms.geo.admin.ch/?SERVICE=WMS&REQUEST=GetMap&VERSION=1.3.0&
download(wmslink,path2save2,wmsfile)
```

```
Successfully downloaded resource http://wms.geo.admin.ch/?SERVICE=WMS&REQUEST=
GetMap&VERSION=1.3.0&LAYERS=ch.bfs.arealstatistik-1985-04&STYLES=default&CRS=E
PSG:21781&BBOX=550000,60000,660000,140000&WIDTH=800&HEIGHT=582&FORMAT=image/pn
g
```

Shapely

You should see something like:

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

In [71]:
```
import ogr
import shapely.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
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 = shapely.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 Fla
        cnt += 1
```

```
Gemeinde Rohr hat folgenden Zentroid: (638811.067724, 251323.787658) und folge
nde Flaeche 2229578.988870m2
```

Fiona

You should see something like:

```
Anzahl Datensätze: 109
Format: ESRI Shapefile
Geo-Referenzsystem: {'proj': 'somerc', 'lat_0':
46.9524055555556, 'lon_0': 7.43958333333333, 'k_0': 1, 'x_0':
600000, 'y_0': 200000, 'ellps': 'bessel', 'units': 'm',
'no_defs': True}
Ausdehnung: (592560.389, 213702.99, 644759.038, 261329.631)
```

In [72]:

```
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: {'proj': 'somerc', 'lat_0': 46.9524055555556, 'lon_0': 7.4
3958333333333, 'k_0': 1, 'x_0': 600000, 'y_0': 200000, 'ellps': 'bessel', 'uni
ts': 'm', 'no_defs': True}
Ausdehnung: (592560.389, 213702.99, 644759.038, 261329.631)
ERROR 1: PROJ: proj_identify: /Users/hansjoerg.stark/opt/anaconda3/share/proj/
proj.db lacks DATABASE.LAYOUT.VERSION.MAJOR / DATABASE.LAYOUT.VERSION.MINOR me
tadata. It comes from another PROJ installation.
ERROR 1: PROJ: proj_create_from_name: /Users/hansjoerg.stark/opt/anaconda3/sha
re/proj/proj.db lacks DATABASE.LAYOUT.VERSION.MAJOR / DATABASE.LAYOUT.VERSION.
MINOR metadata. It comes from another PROJ installation.
```

Folium

An interactive map located at EHT ZH Hönggerberg with a pin should appear.

In [73]:
```
import folium
m = folium.Map(location=[47.40875, 8.50778], zoom_start=17)

folium.Marker(
    location=[47.40875, 8.50778],
    popup="ETH",
    icon=folium.Icon(color="red", icon="info-sign"),
).add_to(m)

m
```

Out[73]: Make this Notebook Trusted to load map: File -> Trust Notebook

In [ ]:
```

```