basic_GIS_functionality_Python

June 17, 2024

1 Basic Geographic Information System (GIS) functionality in Python

1.1 Libraries and settings

[ ]:

```
# Libraries
import os
import folium
import openpyxl
import platform
import pandas as pd
import geopandas as gpd

# Ignore warnings
import warnings
warnings.filterwarnings('ignore')
```

1.2 Import a map of municipalities

[ ]:

```
# Polygonmap als .json-File
polys = gpd.read_file("GEN_A4_GEMEINDEN_2019_epsg4326.json")

# Structure and type
print("nrows, ncols", polys.shape)
print("-------------------------------------------------------")
print("Type:", type(polys))

# Object 'polys' is a GeoDataFrame
polys.head()
```

```
nrows, ncols (162, 6)
-------------------------------------------------------
Type: <class 'geopandas.geodataframe.GeoDataFrame'>
```

[ ]:

```
   BFS           NAME BEZIRKSNAM  ART_TEXT  ART_CODE  \
0  117         Hinwil     Hinwil  Gemeinde         1
1  131       Adliswil     Horgen  Gemeinde         1
2    3     Bonstetten  Affoltern  Gemeinde         1
```

```
3  154   Küsnacht (ZH)     Meilen  Gemeinde         1
4  135  Kilchberg (ZH)     Horgen  Gemeinde         1

                                            geometry
0  POLYGON ((8.84778 47.32410, 8.85861 47.32162, …
1  POLYGON ((8.53489 47.32502, 8.53662 47.32100, …
2  POLYGON ((8.46026 47.33326, 8.46753 47.33410, …
3  POLYGON ((8.60977 47.33352, 8.61127 47.32749, …
4  POLYGON ((8.54625 47.33441, 8.54875 47.33113, …
```

1.3 Plotting the map

[ ]:

```
"""
Parameters:
- location (list): The latitude and longitude coordinates of the map's center.
- zoom_start (int): The initial zoom level of the map.
- geo_data (str): The path to the geojson file containing the polygon data.
- fill_color (str): The color used to fill the polygons on the map.
Returns:
- folium.Map: The map object with the choropleth layer added.
Example usage:
m = create_choropleth_map([47.44, 8.65], 10, 'polys.geojson', 'greenyellow')
"""

# Initialisierung der Map
m = folium.Map(location=[47.44, 8.65], zoom_start=10)



# Map settings
folium.Choropleth( # Choropleth layer
    geo_data=polys, # GeoDataFrame with polygon data
    name='polys', # Name of the layer
    fill_color='greenyellow'
).add_to(m)

folium.LayerControl().add_to(m)

# Plot map
m
```

[ ]:

```
<folium.folium.Map at 0x20ad0bb7c50>
```

1.4 Creating a spatial subset

[ ]:

```
# Subset is formed by using indexing
#idx_winti = polys[polys['NAME'] == 'Winterthur'].index[0] # Index of Winterthur
#polys.iloc[[idx_winti]] # Subset of Winterthur

# Erstellen Sie ein Subset des Geodatensatzes welches die Gemeinden 'Zürich'␣
  ↪und 'Uster' enthält.
idx_zurich = polys[polys['NAME'] == 'Zürich'].index[0] # Index of Zürich
idx_uster = polys[polys['NAME'] == 'Uster'].index[0] # Index of Uster
polys.iloc[[idx_zurich, idx_uster]] # Subset of Zürich and Uster
```

[ ]:

```
    BFS    NAME BEZIRKSNAM  ART_TEXT  ART_CODE  \
69  261  Zürich     Zürich  Gemeinde         1
20  198   Uster      Uster  Gemeinde         1

                                             geometry
69  POLYGON ((8.52697 47.43175, 8.52950 47.43449, …
20  POLYGON ((8.74370 47.37630, 8.74284 47.37384, …
```

1.5 Plotting the spatial subset

[ ]:

```
m = folium.Map(location=[47.44, 8.65], zoom_start=11)

# Map settings
folium.Choropleth(
    #geo_data=polys.iloc[[idx_winti]],
    geo_data=polys.iloc[[idx_zurich, idx_uster]],
    name='polys',
    fill_color='greenyellow'
).add_to(m)

folium.LayerControl().add_to(m)

# Plot map
m
```

[ ]:

```
<folium.folium.Map at 0x20ad0d97320>
```

1.6 Importing municipality data

[ ]:

```
data = pd.read_excel('municipalities_kt_zh_data.xlsx', index_col=None)
print(type(data))

data.head(5)
```

```
<class 'pandas.core.frame.DataFrame'>
```

```
[ ]:   BFS   municipality_name  residents  percentage_foreigners  area_km2  \
     0   21             Adlikon        665                    9.2      6.58
     1  131            Adliswil      18803                   35.3      7.77
     2  241           Aesch (ZH)       1348                   15.7      5.24
     3    1      Aeugst am Albis       1941                   12.7      7.91
     4    2   Affoltern am Albis      12146                   27.6     10.59

        residents_per_km2
     0         101.063830
     1        2419.948520
     2         257.251908
     3         245.385588
     4        1146.931067
```

1.7 Creating a choropleth map

```
[ ]: """
     This code creates a choropleth map using the Folium library in Python. It␣
      ↪visualizes a variable called 'residents' on a map using polygon data stored␣
      ↪in the 'polys' variable. The map is centered at latitude 47.44 and longitude␣
      ↪8.65 with a zoom level of 10.

     The code defines a function 'folium_del_legend' to hide the default legend of␣
      ↪the choropleth map. It also calculates the bins for the color range based on␣
      ↪the quantiles of the 'residents' variable.

     The map is initialized using the Folium library and a choropleth layer is added␣
      ↪to it. The choropleth layer is configured with the polygon data, attribute␣
      ↪data, key to match the attribute data, fill color, opacity, legend name, and␣
      ↪bins. The layer control is also added to the map.

     Finally, the map is plotted and displayed.
     """
     # Variable to plot
     var = 'residents'

     # Function for hiding the default legend
     def folium_del_legend(choropleth: folium.Choropleth):
             del_list = []
             for child in choropleth._children:
                     if child.startswith('color_map'):
                             del_list.append(child)
                             for del_item in del_list:
                                     choropleth._children.pop(del_item)
                                     return choropleth

     # Bins for the color range
```

```python
bins = list(data[var].quantile([0.00, 0.25, 0.50, 0.75, 1.00]))

# Initialize map
m = folium.Map(location=[47.44, 8.65], zoom_start=10)

# Map-Settings (key_on contains the key to match the attribute data)
folium.Choropleth(
        geo_data=polys,
        name='choropleth',
        data=data,
        columns=['BFS', var],
        key_on='feature.properties.BFS',
        fill_color='RdGy',
        fill_opacity=0.7,
        line_opacity=0.5,
        legend_name=var,
        bins=bins,
        reset=True
).add_to(m)

# Layer controls
folium.LayerControl().add_to(m)

# Plotting the map
m
```

```
[ ]: <folium.folium.Map at 0x20ad0bb7e30>
```