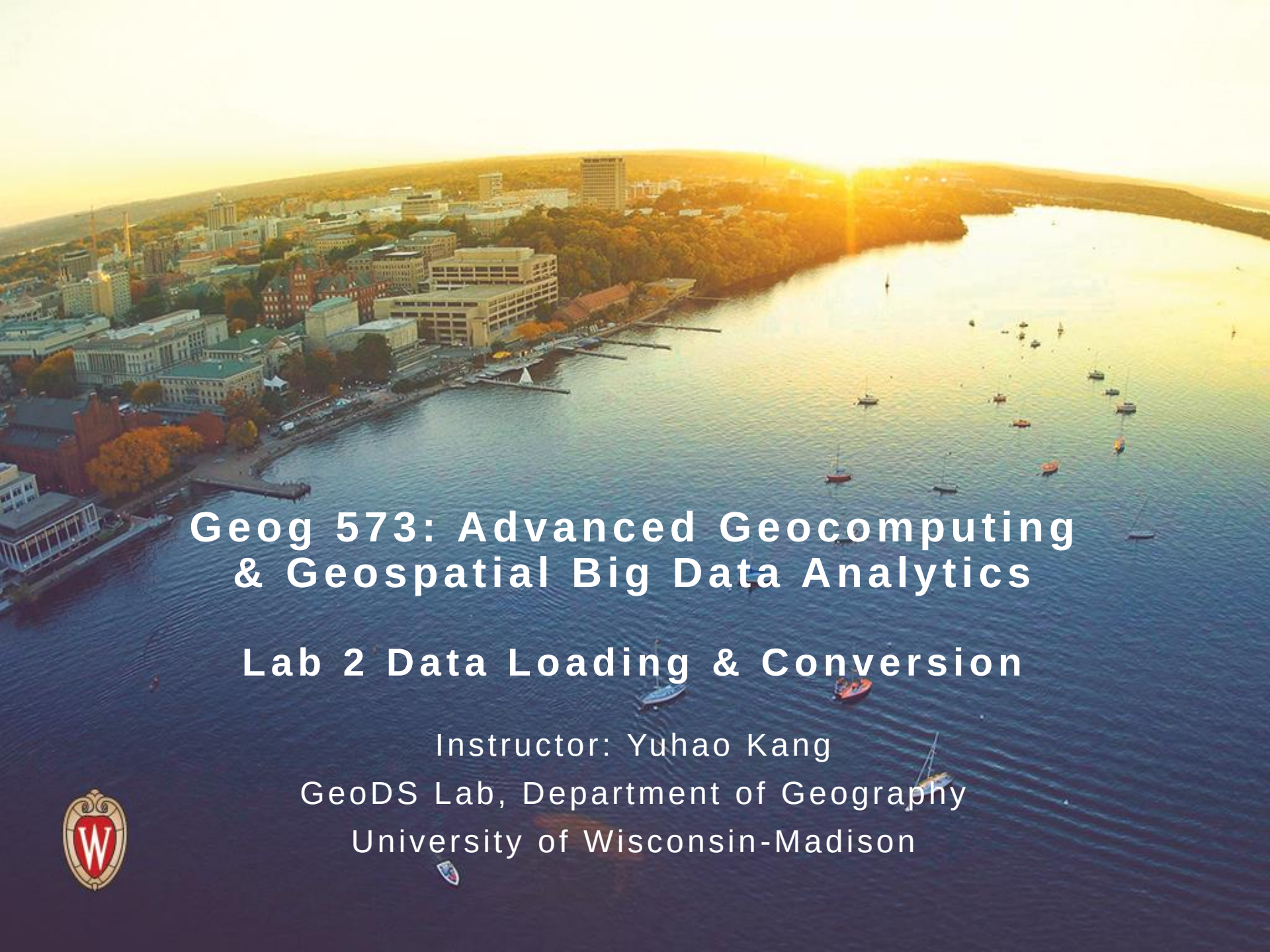


Geog 573: Advanced Geocomputing & Geospatial Big Data Analytics

Lab 2 Data Loading & Conversion

Instructor: Yuhao Kang

GeoDS Lab, Department of Geography

University of Wisconsin-Madison

Review and Notice

Scientific computing basic

- Anaconda
- Jupyter Lab/Notebook
- Google CoLab
- Install packages

Pandas DataFrame

- Array, Dictionary to DataFrame
- Basic functions of DataFrame

Objectives

- Popular data structures for geospatial analysis
  - CSV
  - JSON
  - GeoJSON
  - XML
  - Shapefile
- Conversions among these data structures.

Common Geospatial Data File Types

Structured Data

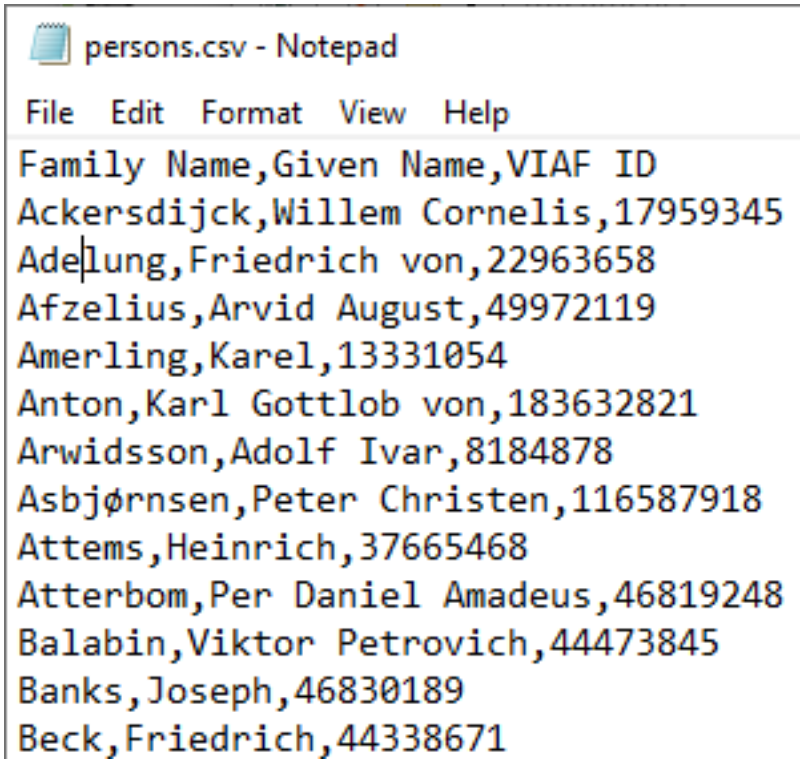
- CSV
- TSV
- …

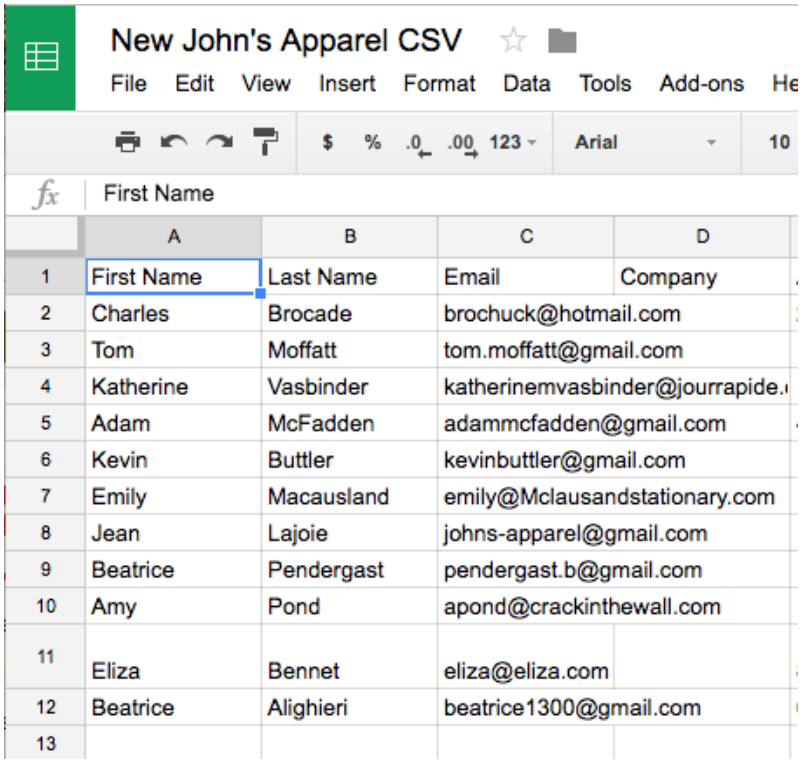
Unstructured Data

- JSON
- GeoJSON
- XML
- Shapefile
- …

CSV (Comma-Separated Values)

- CSV is a delimited text file that uses a comma to separate values. Each line of the file is a data record. Each record consists of one or more fields, separated by commas.





CSV Processing

- import pandas as pd
- dataframe = pd.read_csv()
- dataframe.to_csv()

Advantages

1. Easier to enter.
2. Export quickly.
3. Stored as array.

Disadvantages

1. Large file handling (Excel).
2. Data missing.
3. Special characters.

JSON (JavaScript Object Notation)

- JSON is an open-standard file format that uses human-readable text to transmit data objects consisting of attribute/key–value pairs and array/list data types (or any other serializable types).
- Example


```
{
  "total": 1,
  "records": [
    {
      "key": "us/mt/dillon/33ebannackst",
      "long": "-112.63681",
      "lat": "45.21712",
      "city": "Dillon",
      "province": "MT",
      "country": "US",
      "address": "33 E Bannack St",
      "name": "Klondike Inn",
      "encoding": "UTF-8",
      "type": "location",
      "phone": "4066832141",
      "postalcode": "59725",
      "dateAdded": "2012-04-09 12:27:33",
      "dateUpdated": "2013-09-18T05:32:31Z",
```

JSON Processing

- import json
- r = {'placetype': 'coffee shop', 'name': 'starbucks', 'rating': 4.5}
- r = json.dumps(r)
- loaded_r = json.loads(r)

- Most modern Web APIs natively support JSON input and output.
- Several database technologies (including most NoSQL variations) support it.
- It’s significantly easier to work with most programming languages as well.

GeoJSON

- GeoJSON is an open standard geospatial data interchange format that represents simple geographic features and their nonspatial attributes based JSON.
- GeoJSON is a format for encoding a variety of geographic data structures.
- GeoJSON uses a geographic coordinate reference system, World Geodetic System 1984, and units of decimal degrees.
- Example: https://tools.ietf.org/html/rfc7946#section-1.5
- Documentation: https://geojson.org/

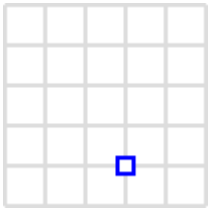
```
{
    "type": "FeatureCollection",
    "features": [{
        "type": "Feature",
        "geometry": {
            "type": "Point",
            "coordinates": [102.0, 0.5]
        },
        "properties": {
            "prop0": "value0"
        }
    }, {
        "type": "Feature",
        "geometry": {
            "type": "LineString",
            "coordinates": [
                [102.0, 0.0],
                [103.0, 1.0],
                [104.0, 0.0],
                [105.0, 1.0]
            ]
        },
        "properties": {
```

```
{
  "type": "Feature",
  "geometry": {
    "type": "Point",
    "coordinates": [125.6, 10.1]
  },
  "properties": {
    "name": "Dinagat Islands"
  }
}
```

GeoJSON Feature Types

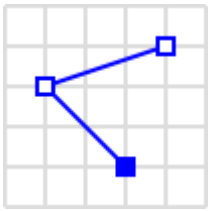
Point

```
{
  "type": "Point",
  "coordinates": [30, 10]
}
```



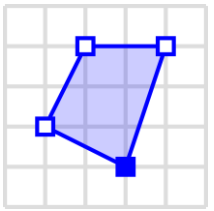
LineString

```
{
  "type": "LineString",
  "coordinates": [
    [30, 10], [10, 30], [40, 40]
  ]
}
```



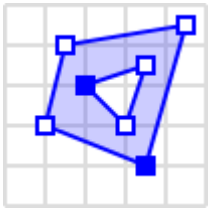
Polygon

```
{
  "type": "Polygon",
  "coordinates": [
    [[30, 10], [40, 40], [20, 40], [10, 20], [30, 10]]
  ]
}
```



Polygon with Hole

```
{
  "type": "Polygon",
  "coordinates": [
    [[35, 10], [45, 45], [15, 40], [10, 20], [35, 10]],
    [[20, 30], [35, 35], [30, 20], [20, 30]]
  ]
}
```



XML (Extensible Markup Language)

- XML is a markup language that defines a set of rules for encoding documents in a format that is both human-readable and machine-readable.
- KML:https://developers.google.com/kml (an XML language focused on geographic visualization).
- To modify and parse XML using python: https://docs.python.org/3.7/library/xml.etree.elementtree.html

```
<?xml version="1.0"?>
<data>
    <country name="Liechtenstein">
        <rank>1</rank>
        <year>2008</year>
        <gdppc>141100</gdppc>
        <neighbor name="Austria" direction="E"/>
        <neighbor name="Switzerland" direction="W"/>
    </country>
    <country name="Singapore">
        <rank>4</rank>
        <year>2011</year>
        <gdppc>59900</gdppc>
        <neighbor name="Malaysia" direction="N"/>
    </country>
    <country name="Panama">
        <rank>68</rank>
        <year>2011</year>
        <gdppc>13600</gdppc>
        <neighbor name="Costa Rica" direction="W"/>
        <neighbor name="Colombia" direction="E"/>
    </country>
</data>
```

Shp (Shapefile)

The shp format is a geospatial vector data format for geographic information system (GIS) software.

It is developed and regulated by Esri as a mostly open specification for data interoperability among Esri and other GIS software products.

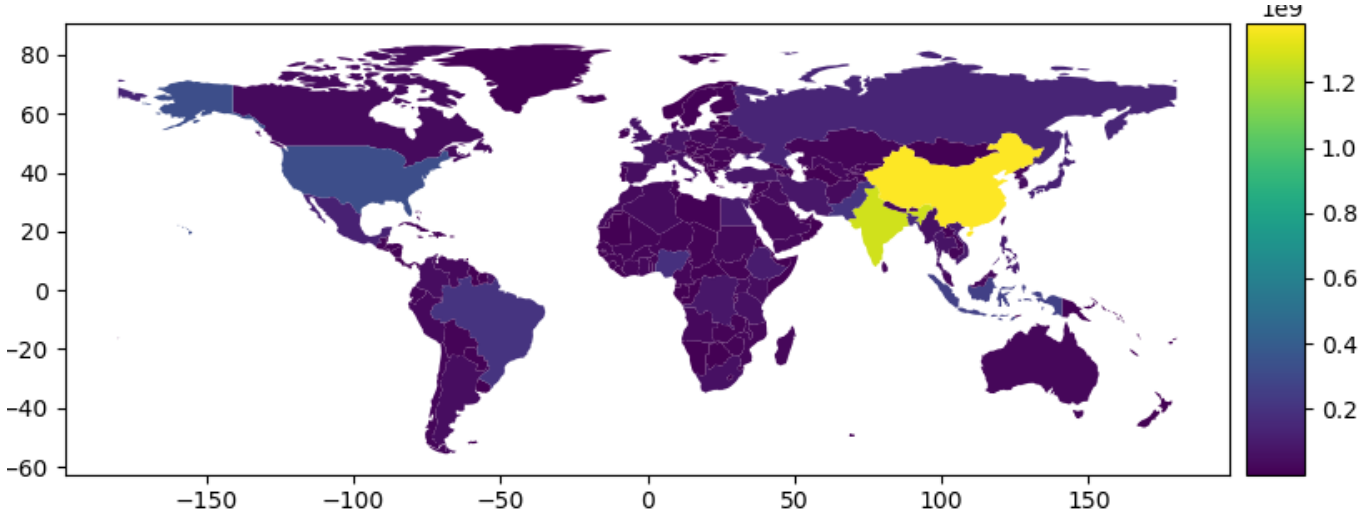
Shapefile shape format (.shp): the main file (.shp) contains the geometry data.

Shapefile attribute format (.dbf): this file stores the attributes for each shape.

Shapefile shape index format (.shx): the index file.

Shp (Shapefile) Processing

- import geopandas as gpd
- gdf = gpd.read_file('file.shp')
- gdf.to_file('file.shp')
- Plot geographic file
- gdf.plot()



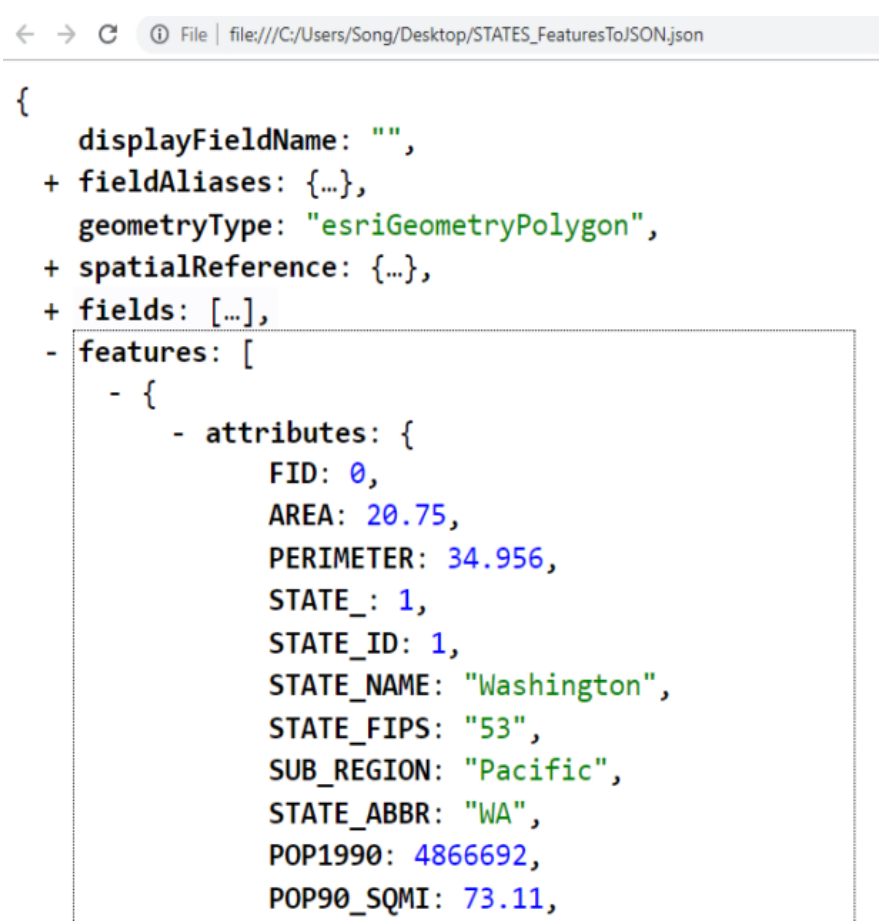
Discussion question: When it is appropriate to use CSV, json, geojson, and shp? Please provide some examples.

Additional Resources

- **MyGeodata Converter**
- Convert GeoJSON to Shp
- https://mygeodata.cloud/converter/geojson-to-shp

- **geojson.xyz**
- A simple, open source website, for the fast access of GeoJSON data for web mapping examples and experiments.
- http://geojson.xyz/

Esri vs. Open Standard GeoJSON



https://geography.wisc.edu/geods/archives/843

Lab Assignment (Due Feb. 8th)

Complete the "Discussion" Question for Lab 2 on Canvas first.

Compress all data files with the code file Yourname_Lab2.ipynb for submission.

Task 1:

- Convert the CityPop.csv file to a JSON file, save as CityPop.json.

Task 2:

- Convert the CityPop.json file to a GeoJSON file, save as CityPop_geojson.json. (hint: type: FeatureCollection)

Task 3:

- Convert the CityPop_geojson.json file to a Shp file, save as CityPop.shp.

Lab Assignment (Due Feb. 8th)

Compress all data files with the code file Yourname_Lab2.ipynb for submission.

Task 4:

- Load the CityPop.shp file and create a simple map with world map as basemap (hint: use gdf.plot()). Please try different style settings.

Task 5:

- Register the Flickr, Google geocoding developer account before the Lab 4, and Twitter developer account.

Check list:

- CityPop.csv
- CityPop.json
- CityPop_geojson.json
- CityPop.* (all shp files)
- Yourname_Lab2.ipynb


THANK YOU
QUESTIONS?