Visualizing Geospatial Data

Group 5 & 7 Presentation

Agenda

★ Introduction

★ Folium

★ Maps with Markers

★ Choropleth Map

★ Summary

Team Members

Aya Hellal

Fahad Bin Gias

Gift Jimoh

Jane Teergele

Manjula Kumari

Nazar Kazymov

Odichinma Ukah

Raghavi Ramasamy

Tamara Marie

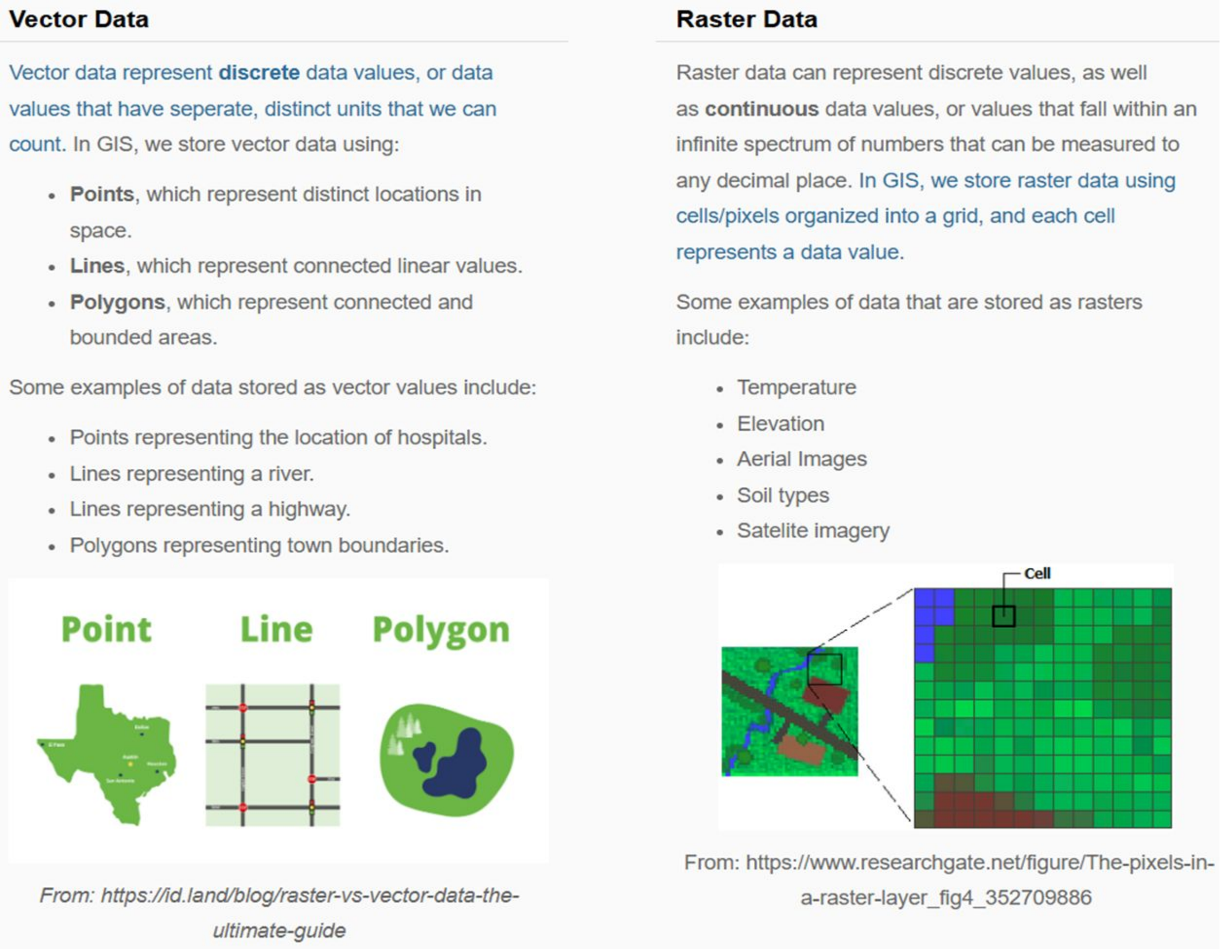
Yasmina Barkouch

What is Geospatial Data

Geospatial data refers to information tied to geographic locations, typically represented by coordinates (latitude/longitude), addresses, or boundaries.

Applications: Urban planning, disaster management (e.g., wildfire tracking), agriculture, and navigation (e.g., Google Maps).

Jane



Vector Data

Vector data represent **discrete** data values, or data values that have seperate, distinct units that we can count. In GIS, we store vector data using:

- **Points**, which represent distinct locations in space.
- **Lines**, which represent connected linear values.
- **Polygons**, which represent connected and bounded areas.

Some examples of data stored as vector values include:

- Points representing the location of hospitals.
- Lines representing a river.
- Lines representing a highway.
- Polygons representing town boundaries.

From: https://id.land/blog/raster-vs-vector-data-the-ultimate-guide

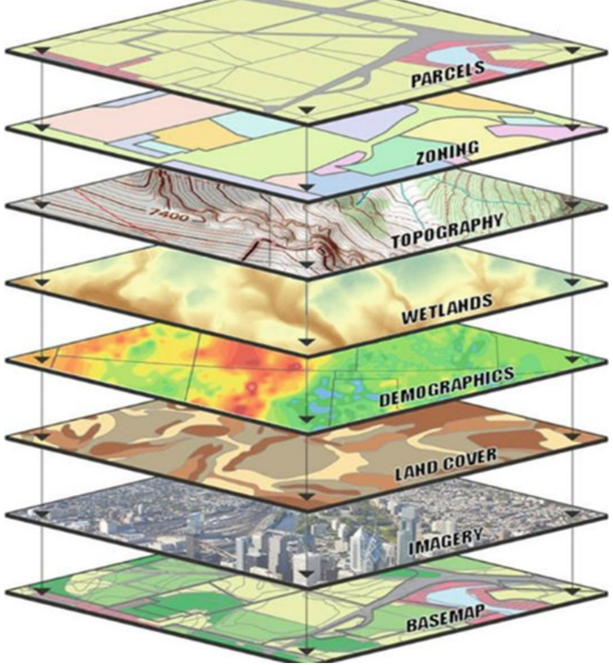
Raster Data

Raster data can represent discrete values, as well as **continuous** data values, or values that fall within an infinite spectrum of numbers that can be measured to any decimal place. In GIS, we store raster data using cells/pixels organized into a grid, and each cell represents a data value.

Some examples of data that are stored as rasters include:

- Temperature
- Elevation
- Aerial Images
- Soil types
- Satelite imagery

From: https://www.researchgate.net/figure/The-pixels-in-a-raster-layer_fig4_352709886

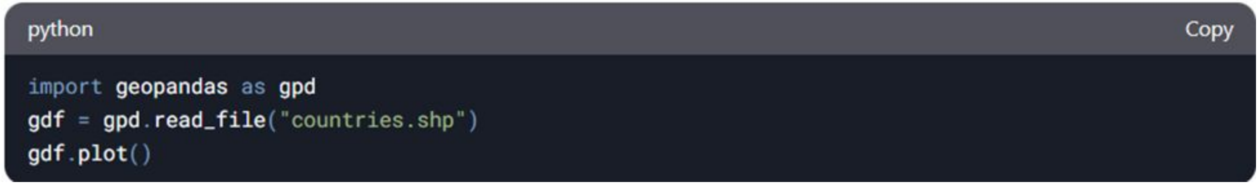


Jane

GeoPandas

Extends Pandas to support geospatial operations. Key features:

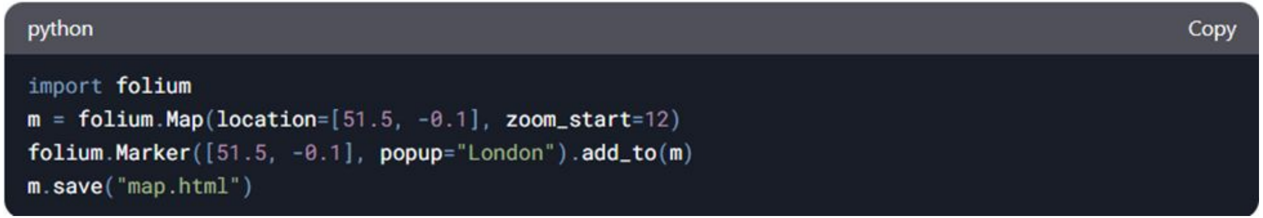
- Read/write shapefiles, GeoJSON.
- Spatial joins, buffering, and CRS transformations.



```python
import geopandas as gpd
gdf = gpd.read_file("countries.shp")
gdf.plot()
```

Folium

Folium allows creating Leaflet maps using Python. This makes it very easy to build interactive maps within Python environments like Jupyter Notebooks.



```python
import folium
m = folium.Map(location=[51.5, -0.1], zoom_start=12)
folium.Marker([51.5, -0.1], popup="London").add_to(m)
m.save("map.html")
```

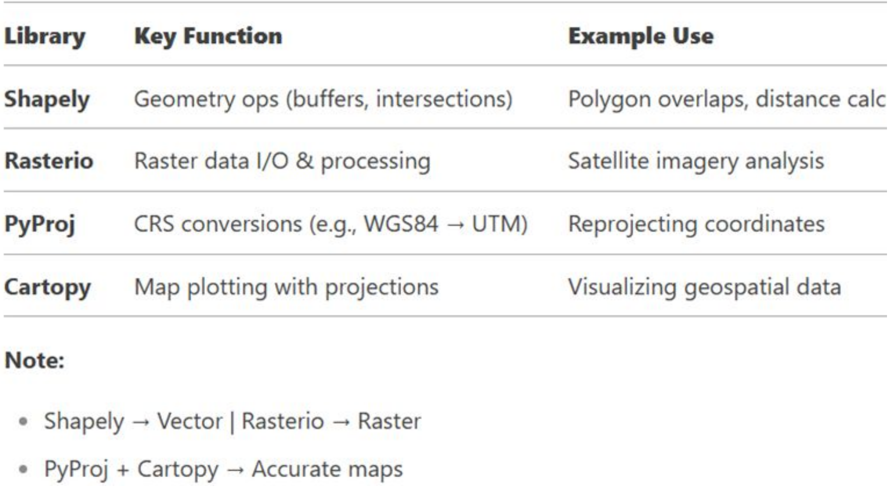
Jane

Choropleth Maps

A *thematic* map where regions are shaded based on a variable (e.g., population). Requires:

- ❏ GeoJSON/TopoJSON: Defines geographic boundaries.
- ❏ Data: Values to visualize (e.g., GDP by country).

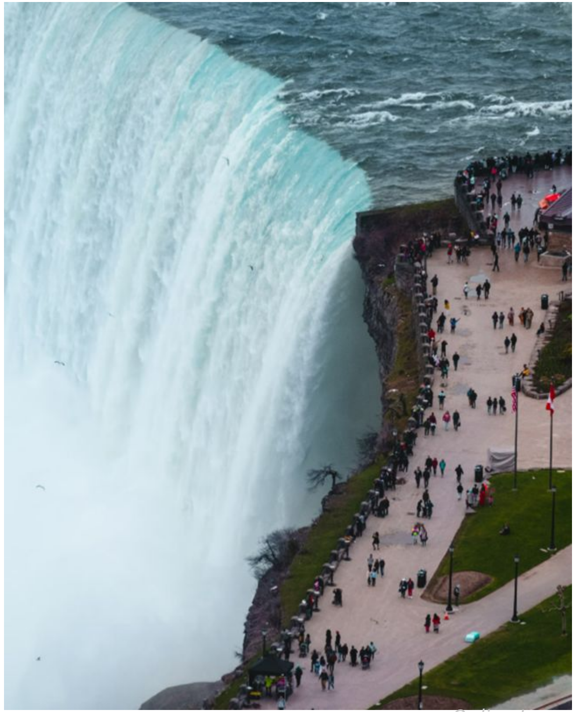
Additional Python libraries for geometry editing

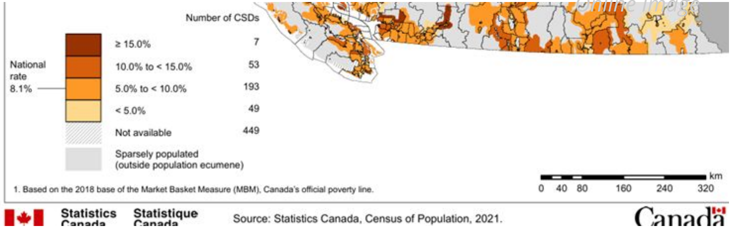


Library	Key Function	Example Use
Shapely	Geometry ops (buffers, intersections)	Polygon overlaps, distance calc
Rasterio	Raster data I/O & processing	Satellite imagery analysis
PyProj	CRS conversions (e.g., WGS84 → UTM)	Reprojecting coordinates
Cartopy	Map plotting with projections	Visualizing geospatial data

Note:

- Shapely → Vector | Rasterio → Raster
- PyProj + Cartopy → Accurate maps





1. Based on the 2018 base of the Market Basket Measure (MBM), Canada's official poverty line.

Source: Statistics Canada, Census of Population, 2021.

Jane

What is Folium

- A **Python** library for creating interactive, web-based maps with minimal code.
- Built on top of **Leaflet.js**, a popular JavaScript library for geospatial visualizations.
- Combines Python’s data manipulation (**Pandas**, **NumPy**) with **JavaScript**’s interactivity.
- Ideal for:
    - Visualizing location-based data.
    - Plotting routes and spatial trends.
    - Creating dynamic, user-friendly maps.
- Enables easy sharing via **HTML** exports.

Yasmina

Benefits and Disadvantages

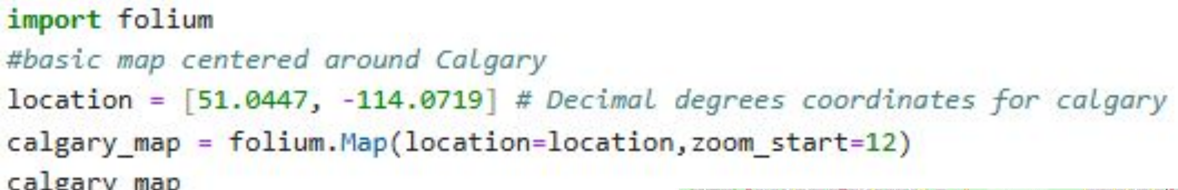
		Benefit	Limitation
1	Easy to use(simple syntaxe)	✓	
2	Integration with Python Data Ecosystem (like Pandas, GeoPandas, and NumPy)	✓	
3	Quick Export to HTML	✓	
4	Not Ideal for Heavy Geographic Information System Analysis		✗
5	Limited in customization compared to JavaScript-based tools.		✗
6	Browser Performance It may not handle very large datasets well		✗

Yasmina

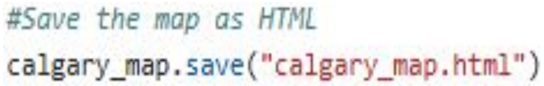
Get started with Folium

Installation : `!pip install folium`

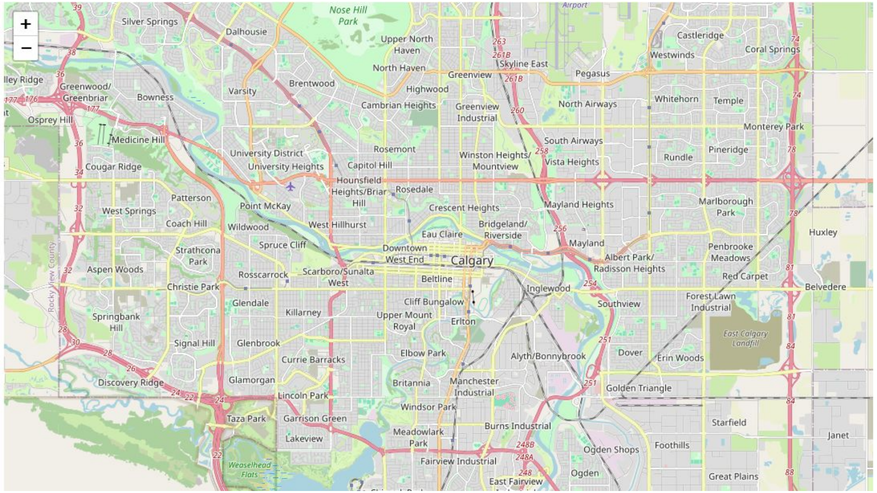
Create a basic map :



```
import folium
#basic map centered around Calgary
location = [51.0447, -114.0719] # Decimal degrees coordinates for calgary
calgary_map = folium.Map(location=location,zoom_start=12)
calgary_map
```



```
#Save the map as HTML
calgary_map.save("calgary_map.html")
```



Yasmina

Built-in tile options in Folium

OpenStreetMap (default) : Open-source and regularly updated street maps.

Stamen Terrain :Physical terrain features like mountains, rivers, etc.

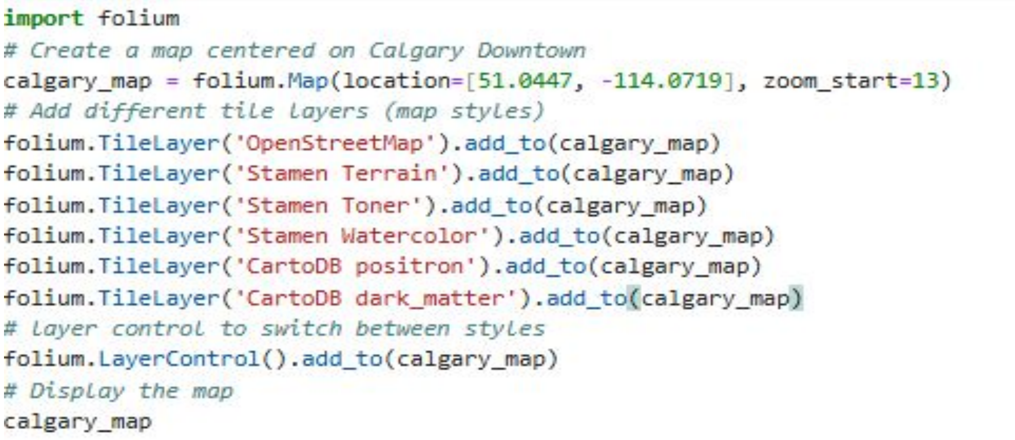
Stamen Toner:High-contrast black-and-white map, good for data overlays.

Stamen Watercolor: Artistic watercolor style.

CartoDB positron: Light-themed modern base map.

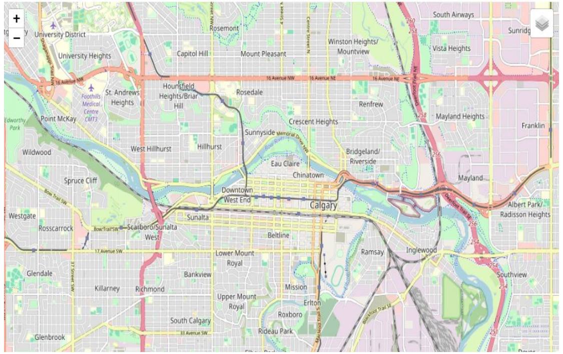
CartoDB dark_matter: Dark-themed version of CartoDB, good for night mode or highlighting overlays.

Different map styles using tile parameter

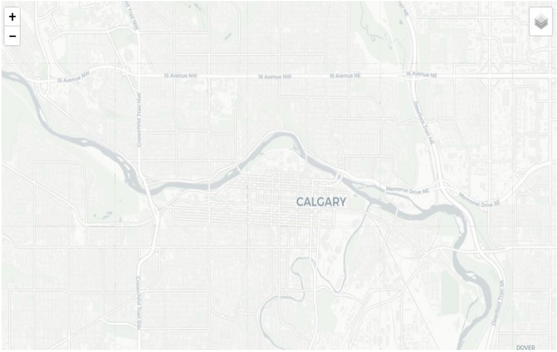


```
import folium
# Create a map centered on Calgary Downtown
calgary_map = folium.Map(location=[51.0447, -114.0719], zoom_start=13)
# Add different tile layers (map styles)
folium.TileLayer('OpenStreetMap').add_to(calgary_map)
folium.TileLayer('Stamen Terrain').add_to(calgary_map)
folium.TileLayer('Stamen Toner').add_to(calgary_map)
folium.TileLayer('Stamen Watercolor').add_to(calgary_map)
folium.TileLayer('CartoDB positron').add_to(calgary_map)
folium.TileLayer('CartoDB dark_matter').add_to(calgary_map)
# layer control to switch between styles
folium.LayerControl().add_to(calgary_map)
# Display the map
calgary_map
```

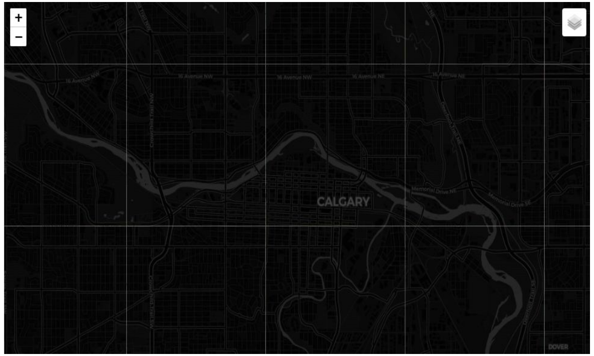
OpenStreetMap



CartoDB positron



CartoDB dark_matter



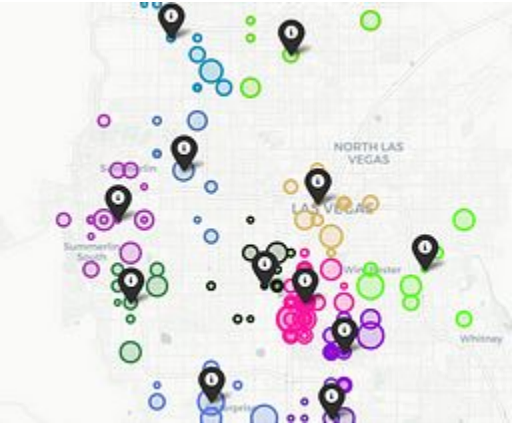
Yasmina

Folium Real world Uses cases

🏥**Public Health**: Disease spread maps

🚚 **Logistics**: Delivery point tracking

🧑‍🏫**Education**: Teaching geography and data visualization



📍 **Business**: Customer Location Analysis

Use Case:
A business wants to understand where their customers are located to improve marketing and delivery services.

Example:
A coffee shop chain collects customer addresses through loyalty programs. By converting addresses into coordinates (geocoding), they can:

- Map all customer locations on a Folium map
- Identify clusters of customers
- Decide where to open a new store

Yasmina

Maps with markers

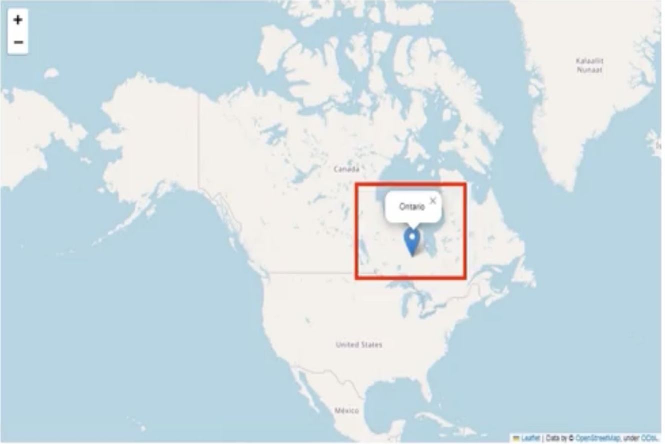
Markers are like **signposts** that guide us through the map, highlighting important elements

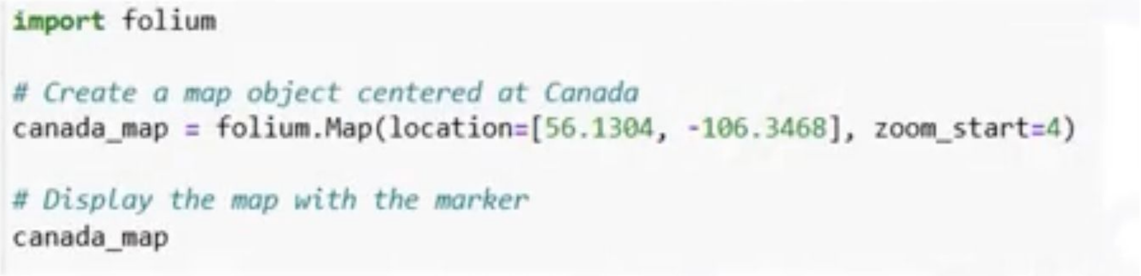
They represent **specific locations** or **points of interest**, providing **additional information** when clicked

Markers play a vital role in enhancing **interactivity** and adding **context** to maps

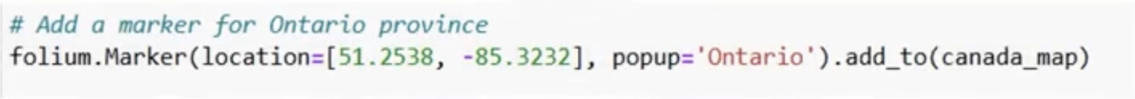
Aya

Add marker and label





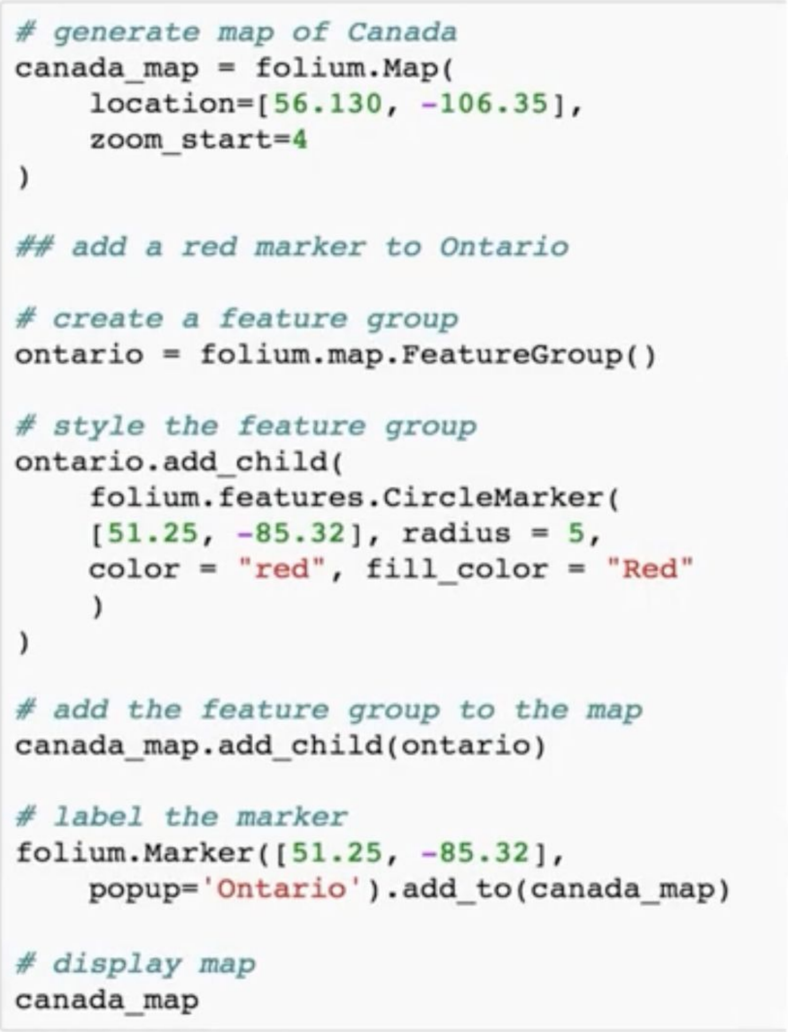
```
import folium

# Create a map object centered at Canada
canada_map = folium.Map(location=[56.1304, -106.3468], zoom_start=4)

# Display the map with the marker
canada_map
```



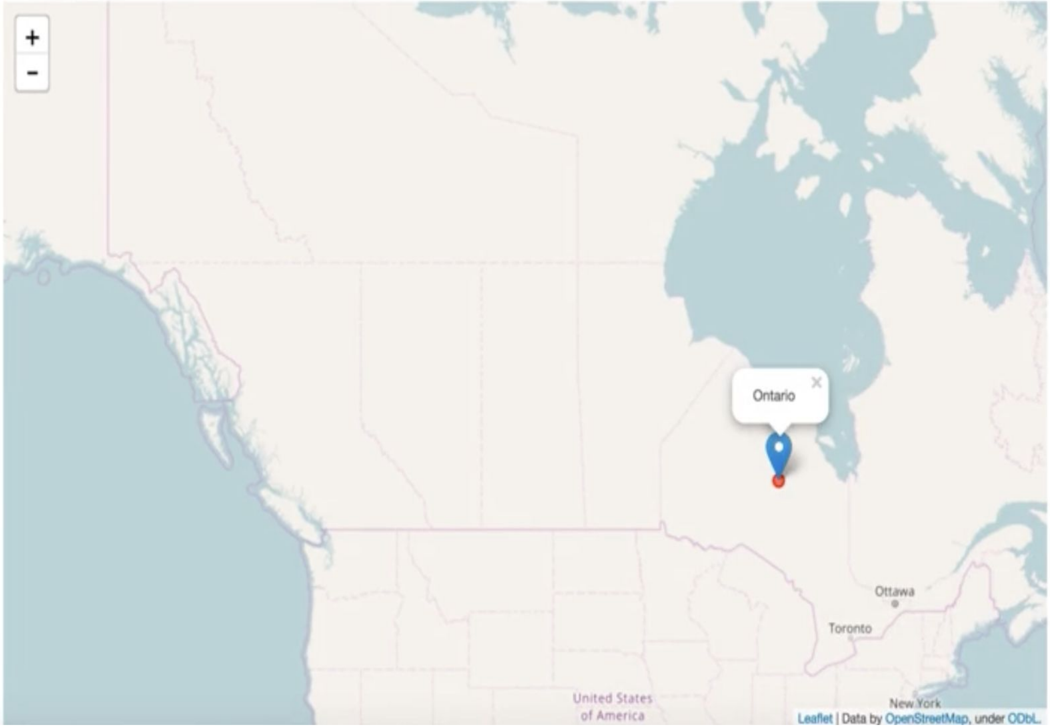
```
# Add a marker for Ontario province
folium.Marker(location=[51.2538, -85.3232], popup='Ontario').add_to(canada_map)
```

Aya

Add Marker with feature group

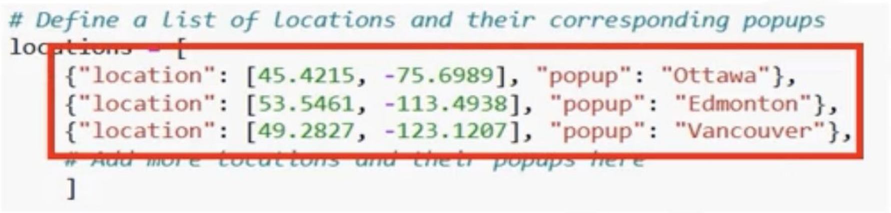


```
# generate map of Canada
canada_map = folium.Map(
    location=[56.130, -106.35],
    zoom_start=4
)

## add a red marker to Ontario

# create a feature group
ontario = folium.map.FeatureGroup()

# style the feature group
ontario.add_child(
    folium.features.CircleMarker(
    [51.25, -85.32], radius = 5,
    color = "red", fill_color = "Red"
    )
)

# add the feature group to the map
canada_map.add_child(ontario)

# label the marker
folium.Marker([51.25, -85.32],
    popup='Ontario').add_to(canada_map)

# display map
canada_map
```

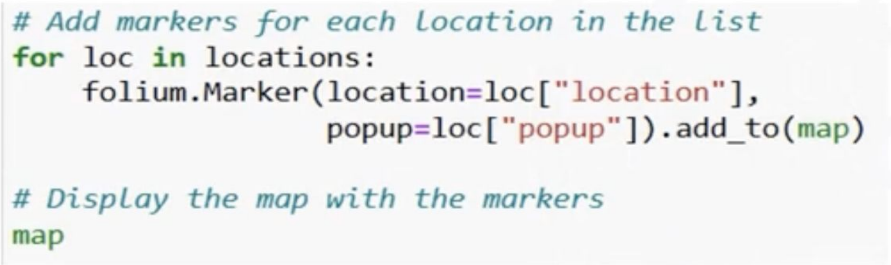


Aya

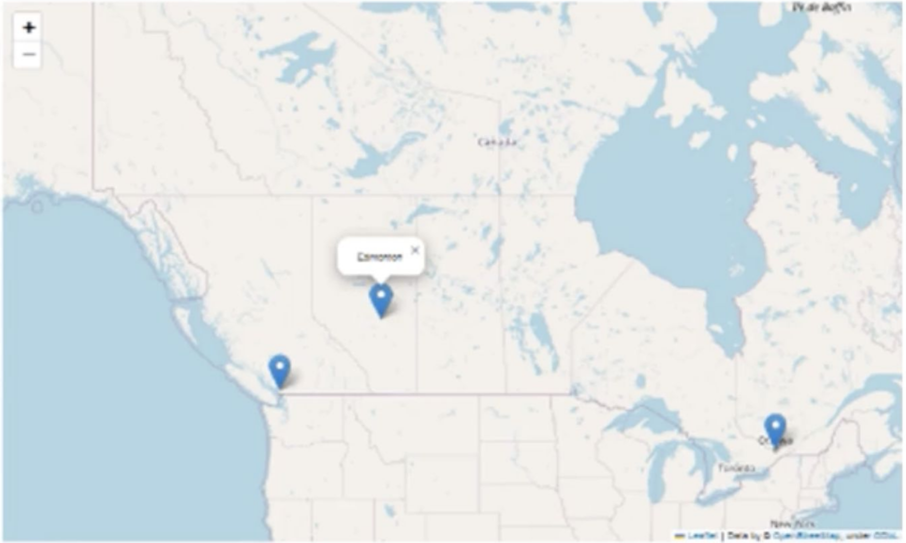
Multiple markers



```
# Define a list of locations and their corresponding popups
locations = [
    {"location": [45.4215, -75.6989], "popup": "Ottawa"},
    {"location": [53.5461, -113.4938], "popup": "Edmonton"},
    {"location": [49.2827, -123.1207], "popup": "Vancouver"},
    # Add more locations and their popups here
    ]
```



```
# Add markers for each location in the list
for loc in locations:
    folium.Marker(location=loc["location"],
                  popup=loc["popup"]).add_to(map)

# Display the map with the markers
map
```



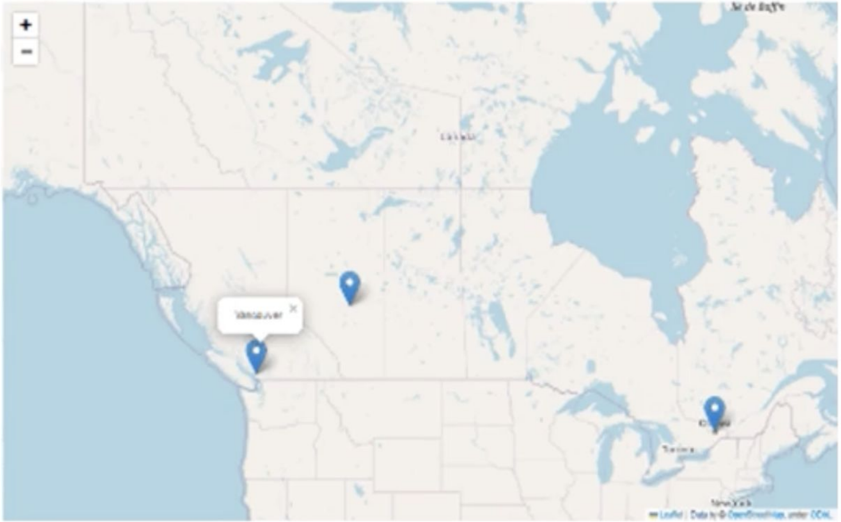
MarkerCluster feature

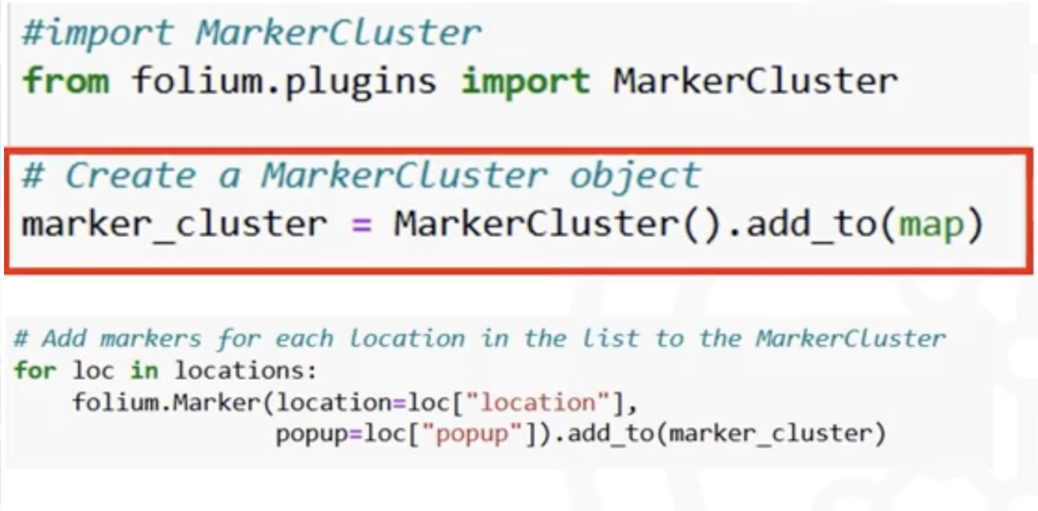
This clustering feature enhances the visual presentation by **preventing overcrowding** and ensuring a **clear representation** primarily when numerous markers are close.

The markers within the MarkerCluster will be **intelligently grouped** based on their **proximity** when the map is displayed.

Aya

Multiple markers using MarkerCluster feature

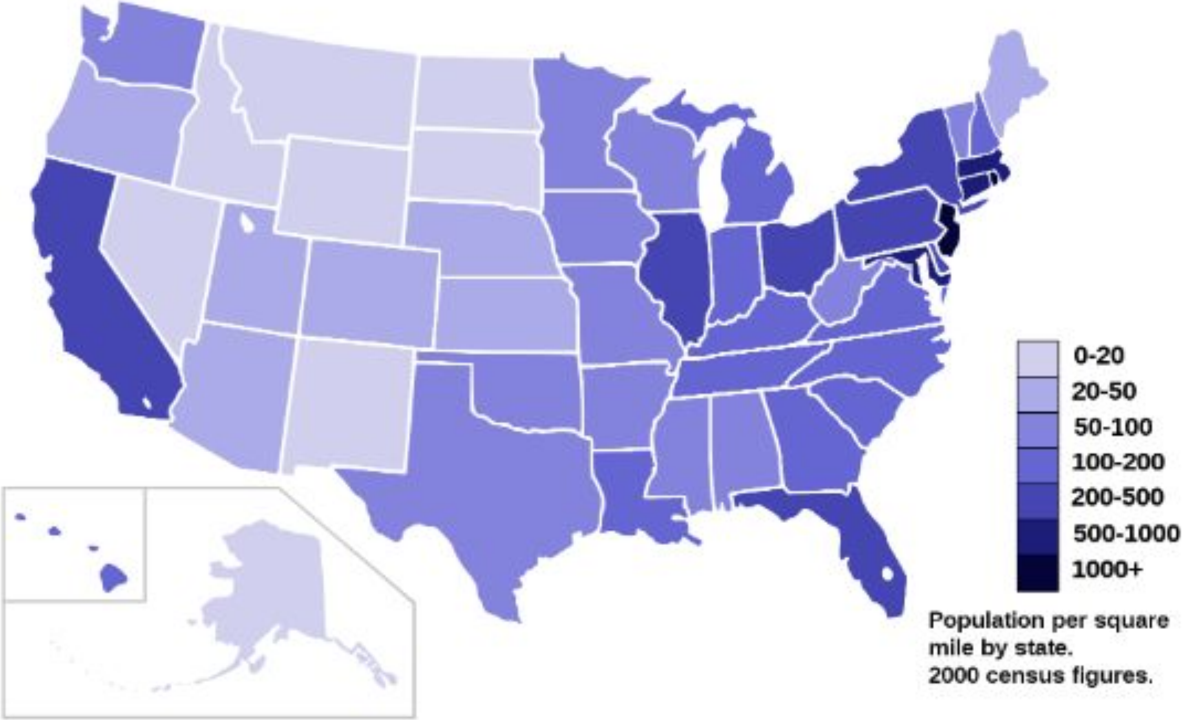




```
#import MarkerCluster
from folium.plugins import MarkerCluster

# Create a MarkerCluster object
marker_cluster = MarkerCluster().add_to(map)

# Add markers for each location in the list to the MarkerCluster
for loc in locations:
    folium.Marker(location=loc["location"],
                  popup=loc["popup"]).add_to(marker_cluster)
```

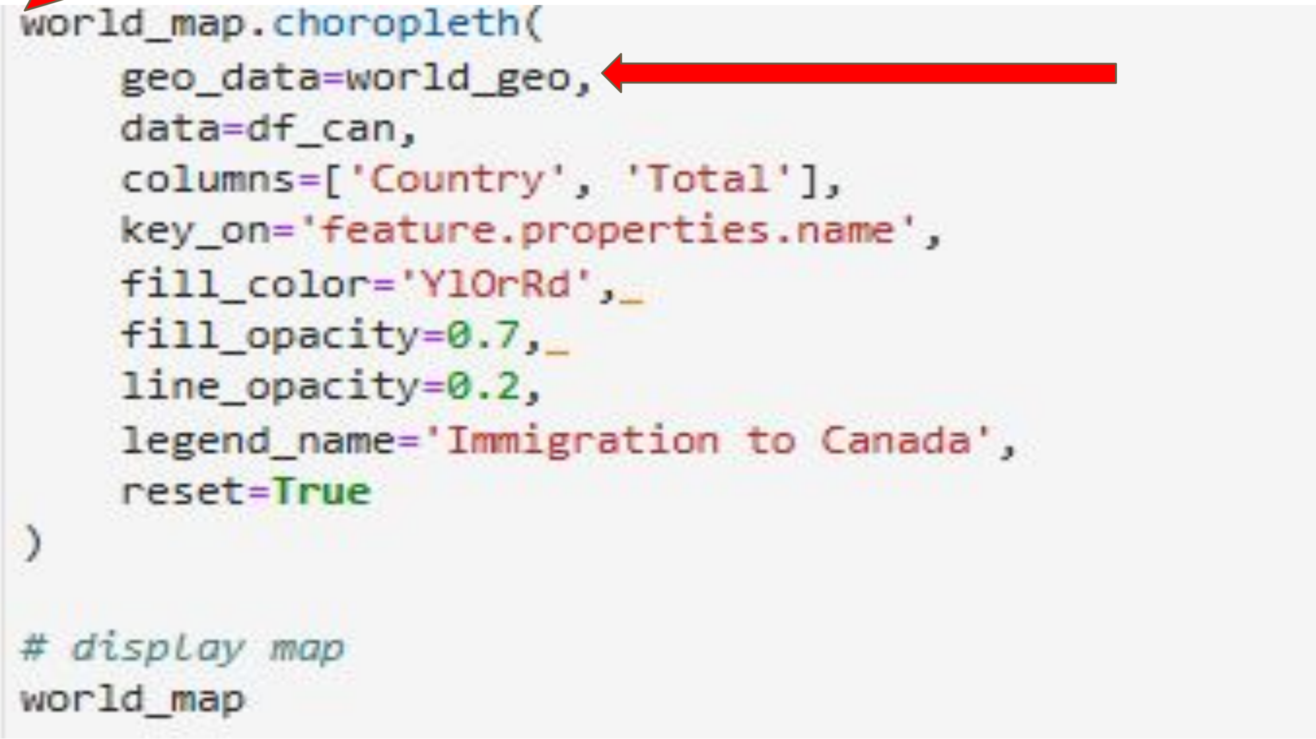
Aya

Choropleth Maps



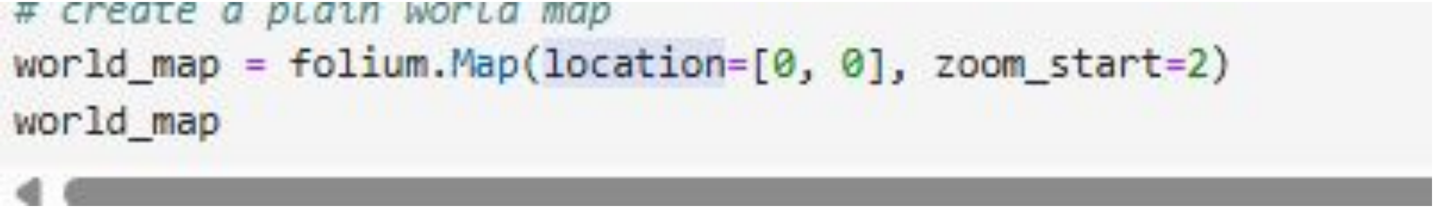
Tamara

Now, let's create our own `Choropleth` map of the world depicting immigration from various countries to Canada.



```
world_map.choropleth(
    geo_data=world_geo,
    data=df_can,
    columns=['Country', 'Total'],
    key_on='feature.properties.name',
    fill_color='YlOrRd',_
    fill_opacity=0.7,_
    line_opacity=0.2,
    legend_name='Immigration to Canada',
    reset=True
)

# display map
world_map
```

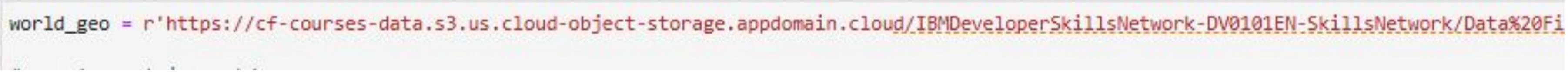
Tamara



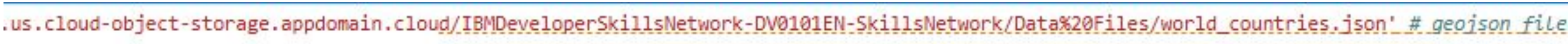
```
# create a plain world map
world_map = folium.Map(location=[0, 0], zoom_start=2)
world_map
```

+
−
Leaflet | Data by © OpenStreetMap, under ODbL.

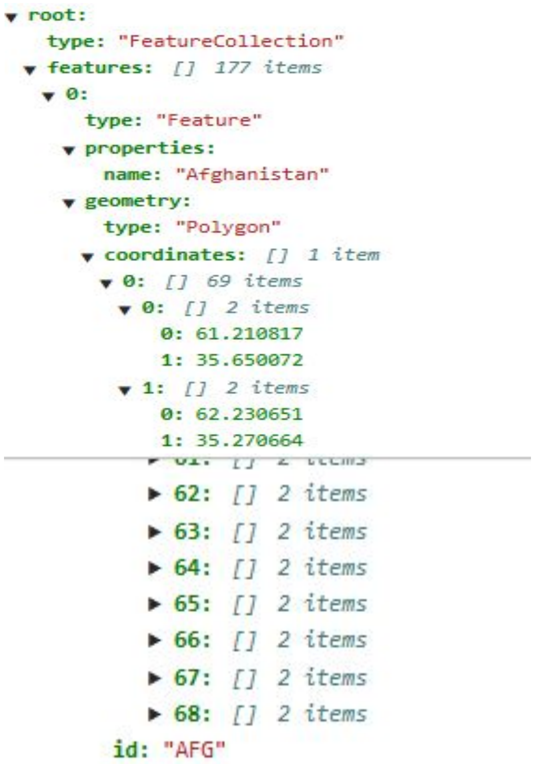
Tamara



```
world_geo = r'https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBMDeveloperSkillsNetwork-DV0101EN-SkillsNetwork/Data%20Fi
```



```
.us.cloud-object-storage.appdomain.cloud/IBMDeveloperSkillsNetwork-DV0101EN-SkillsNetwork/Data%20Files/world_countries.json' # geojson file
```

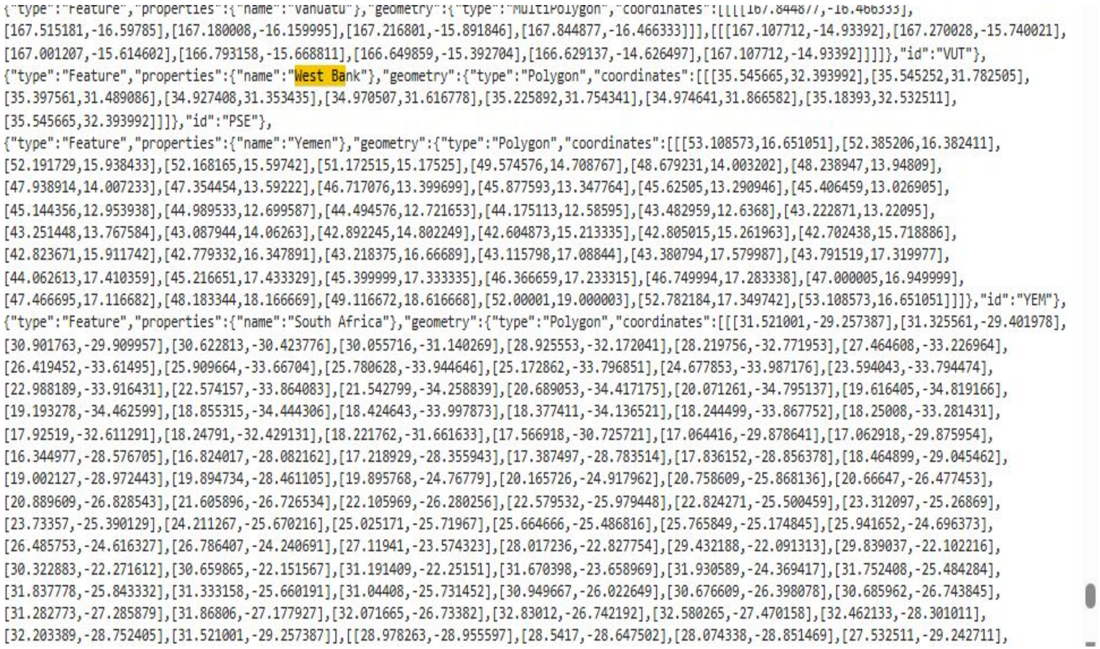


```
▼ root:
   type: "FeatureCollection"
 ▼ features: [] 177 items
   ▼ 0:
       type: "Feature"
     ▼ properties:
         name: "Afghanistan"
     ▼ geometry:
         type: "Polygon"
       ▼ coordinates: [] 1 item
         ▼ 0: [] 69 items
           ▼ 0: [] 2 items
               0: 61.210817
               1: 35.650072
           ▼ 1: [] 2 items
               0: 62.230651
               1: 35.270664
           ▶ 61: [] 2 items
           ▶ 62: [] 2 items
           ▶ 63: [] 2 items
           ▶ 64: [] 2 items
           ▶ 65: [] 2 items
           ▶ 66: [] 2 items
           ▶ 67: [] 2 items
           ▶ 68: [] 2 items
       id: "AFG"
```

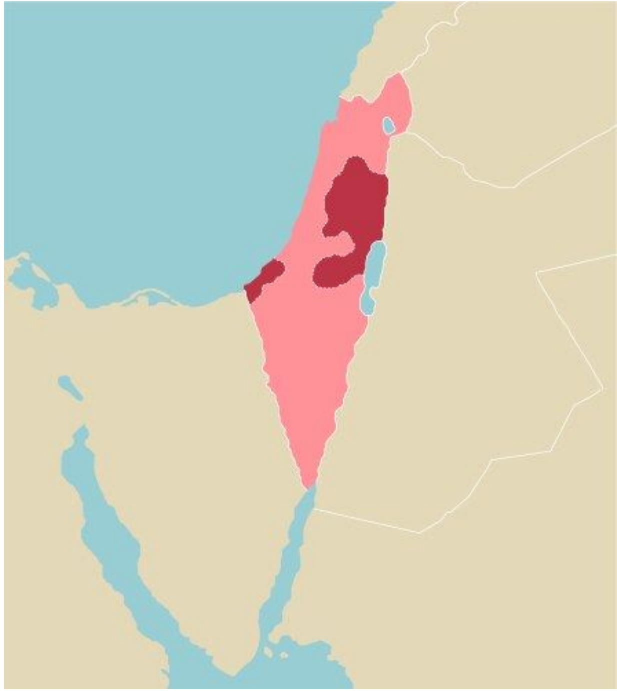
Read geojson data and rename it to world_geo

Tamara

Palestine geojson data: west bank and Gaza

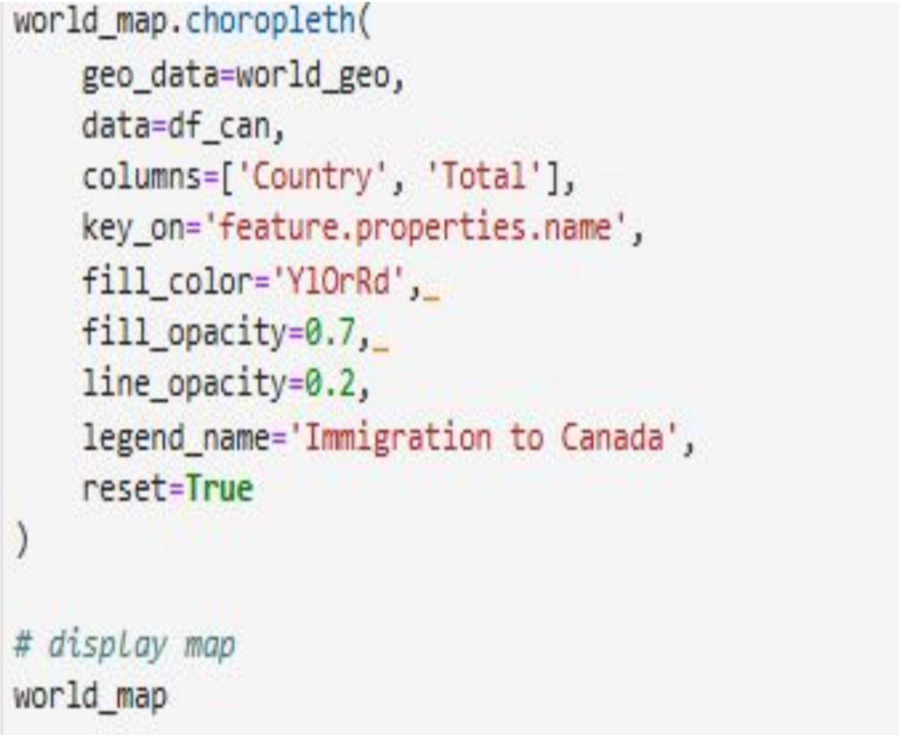


```
{"type":"Feature","properties":{"name":"Vanuatu"},"geometry":{"type":"MultiPolygon","coordinates":[[[[167.844877,-16.466333],
[167.515181,-16.59785],[167.180008,-16.159995],[167.216801,-15.891846],[167.844877,-16.466333]]],[[[167.107712,-14.93392],[167.270028,-15.740021],
[167.001207,-15.614602],[166.793158,-15.668811],[166.649859,-15.392704],[166.629137,-14.626497],[167.107712,-14.93392]]]]},"id":"VUT"},
{"type":"Feature","properties":{"name":"West Bank"},"geometry":{"type":"Polygon","coordinates":[[[35.545665,32.393992],[35.545252,31.782505],
[35.397561,31.489086],[34.927408,31.353435],[34.970507,31.616778],[35.225892,31.754341],[34.974641,31.866582],[35.18393,32.532511],
[35.545665,32.393992]]]},"id":"PSE"},
{"type":"Feature","properties":{"name":"Yemen"},"geometry":{"type":"Polygon","coordinates":[[[53.108573,16.651051],[52.385206,16.382411],
[52.191729,15.938433],[52.168165,15.59742],[51.172515,15.17525],[49.574576,14.708767],[48.679231,14.003202],[48.238947,13.94809],
[47.938914,14.007233],[47.354454,13.59222],[46.717076,13.399699],[45.877593,13.347764],[45.62505,13.290946],[45.406459,13.026905],
[45.144356,12.953938],[44.989533,12.699587],[44.494576,12.721653],[44.175113,12.58595],[43.482959,12.6368],[43.222871,13.22095],
[43.251448,13.767584],[43.087944,14.06263],[42.892245,14.802249],[42.604873,15.213335],[42.805015,15.261963],[42.702438,15.718886],
[42.823671,15.911742],[42.779332,16.347891],[43.218375,16.66689],[43.115798,17.08844],[43.380794,17.579987],[43.791519,17.319977],
[44.062613,17.410359],[45.216651,17.433329],[45.399999,17.333335],[46.366659,17.233315],[46.749994,17.283338],[47.000005,16.949999],
[47.466695,17.116682],[48.183344,18.166669],[49.116672,18.616668],[52.00001,19.000003],[52.782184,17.349742],[53.108573,16.651051]]]},"id":"YEM"},
{"type":"Feature","properties":{"name":"South Africa"},"geometry":{"type":"Polygon","coordinates":[[[31.521001,-29.257387],[31.325561,-29.401978],
[30.901763,-29.909957],[30.622813,-30.423776],[30.055716,-31.140269],[28.925553,-32.172041],[28.219756,-32.771953],[27.464608,-33.226964],
[26.419452,-33.61495],[25.909664,-33.66704],[25.780628,-33.944646],[25.172862,-33.796851],[24.677853,-33.987176],[23.594043,-33.794474],
[22.988189,-33.916431],[22.574157,-33.864083],[21.542799,-34.258839],[20.689053,-34.417175],[20.071261,-34.795137],[19.616405,-34.819166],
[19.193278,-34.462599],[18.855315,-34.444306],[18.424643,-33.997873],[18.377411,-34.136521],[18.244499,-33.867752],[18.25008,-33.281431],
[17.92519,-32.611291],[18.24791,-32.429131],[18.221762,-31.661633],[17.566918,-30.725721],[17.064416,-29.878641],[17.062918,-29.875954],
[16.344977,-28.576705],[16.824017,-28.082162],[17.218929,-28.355943],[17.387497,-28.783514],[17.836152,-28.856378],[18.464899,-29.045462],
[19.002127,-28.972443],[19.894734,-28.461105],[19.895768,-24.76779],[20.165726,-24.917962],[20.758609,-25.868136],[20.66647,-26.477453],
[20.889609,-26.828543],[21.605896,-26.726534],[22.105969,-26.280256],[22.579532,-25.979448],[22.824271,-25.500459],[23.312097,-25.26869],
[23.73357,-25.390129],[24.211267,-25.670216],[25.025171,-25.71967],[25.664666,-25.486816],[25.765849,-25.174845],[25.941652,-24.696373],
[26.485753,-24.616327],[26.786407,-24.240691],[27.11941,-23.574323],[28.017236,-22.827754],[29.432188,-22.091313],[29.839037,-22.102216],
[30.322883,-22.271612],[30.659865,-22.151567],[31.191409,-22.25151],[31.670398,-23.658969],[31.930589,-24.369417],[31.752408,-25.484284],
[31.837778,-25.843332],[31.333158,-25.660191],[31.04408,-25.731452],[30.949667,-26.022649],[30.676609,-26.398078],[30.685962,-26.743845],
[31.282773,-27.285879],[31.86806,-27.177927],[32.071665,-26.73382],[32.83012,-26.742192],[32.580265,-27.470158],[32.462133,-28.301011],
[32.203389,-28.752405],[31.521001,-29.257387]],[[28.978263,-28.955597],[28.5417,-28.647502],[28.074338,-28.851469],[27.532511,-29.242711],
```

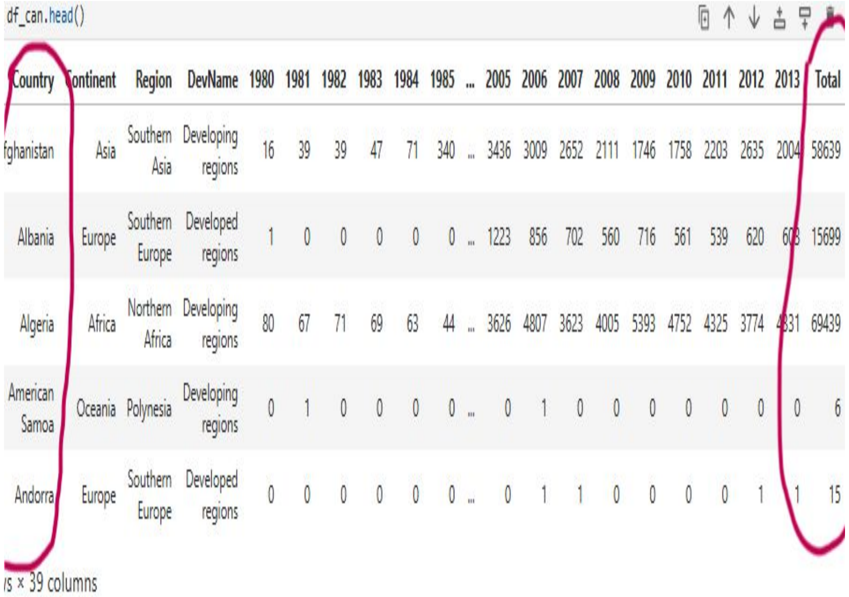


Tamara

Now, let's create our own `Choropleth` map of the world depicting immigration from various countries to Canada.



```
world_map.choropleth(
    geo_data=world_geo,
    data=df_can,
    columns=['Country', 'Total'],
    key_on='feature.properties.name',
    fill_color='YlOrRd',
    fill_opacity=0.7,
    line_opacity=0.2,
    legend_name='Immigration to Canada',
    reset=True
)

# display map
world_map
```

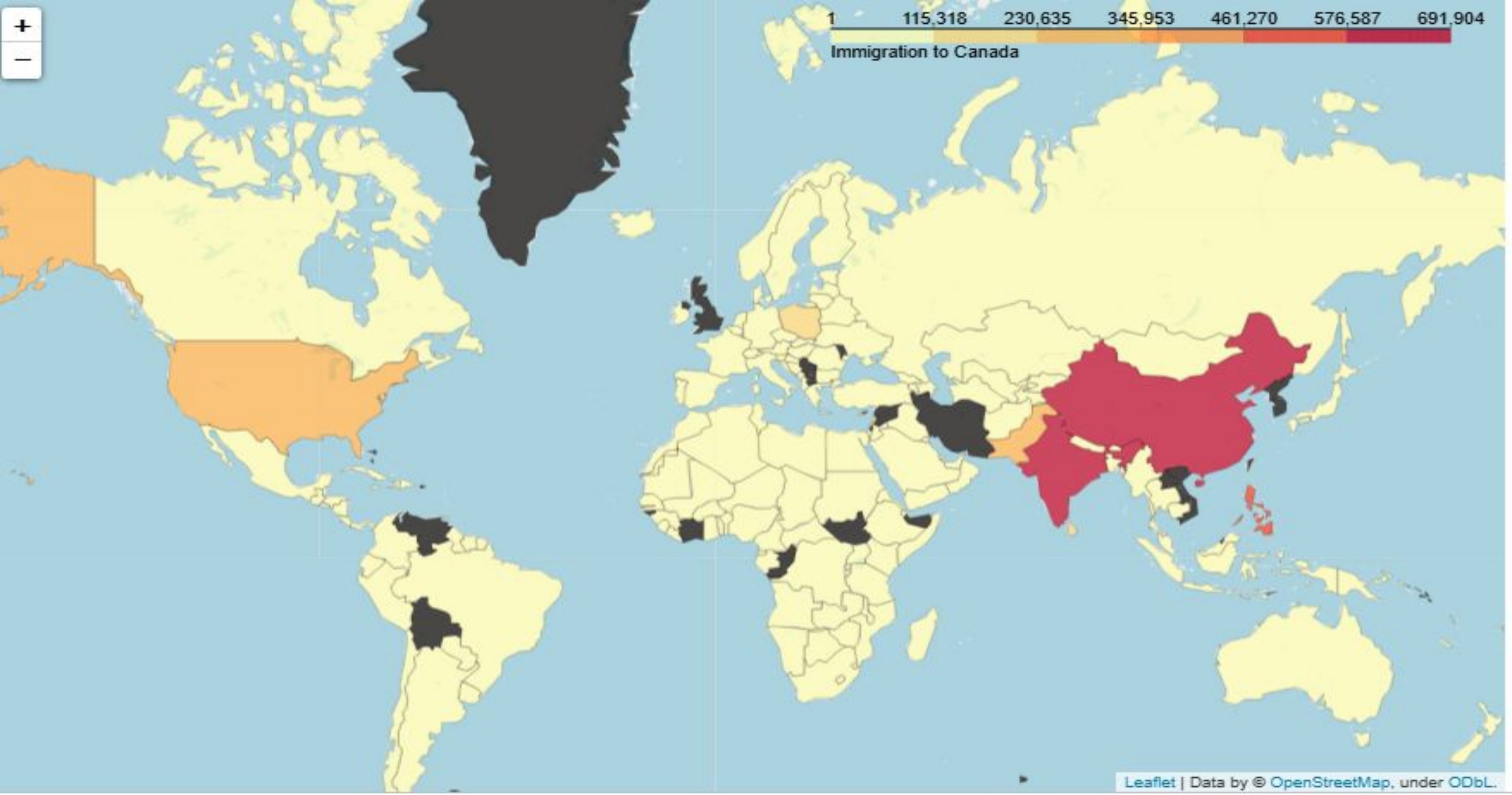


```
df_can.head()
```

Country	ontinent	Region	DevName	1980	1981	1982	1983	1984	1985	...	2005	2006	2007	2008	2009	2010	2011	2012	2013	Total
fghanistan	Asia	Southern Asia	Developing regions	16	39	39	47	71	340	...	3436	3009	2652	2111	1746	1758	2203	2635	2004	58639
Albania	Europe	Southern Europe	Developed regions	1	0	0	0	0	0	...	1223	856	702	560	716	561	539	620	603	15699
Algeria	Africa	Northern Africa	Developing regions	80	67	71	69	63	44	...	3626	4807	3623	4005	5393	4752	4325	3774	4331	69439
American Samoa	Oceania	Polynesia	Developing regions	0	1	0	0	0	0	...	0	1	0	0	0	0	0	0	0	6
Andorra	Europe	Southern Europe	Developed regions	0	0	0	0	0	0	...	0	1	1	0	0	0	0	1	1	15

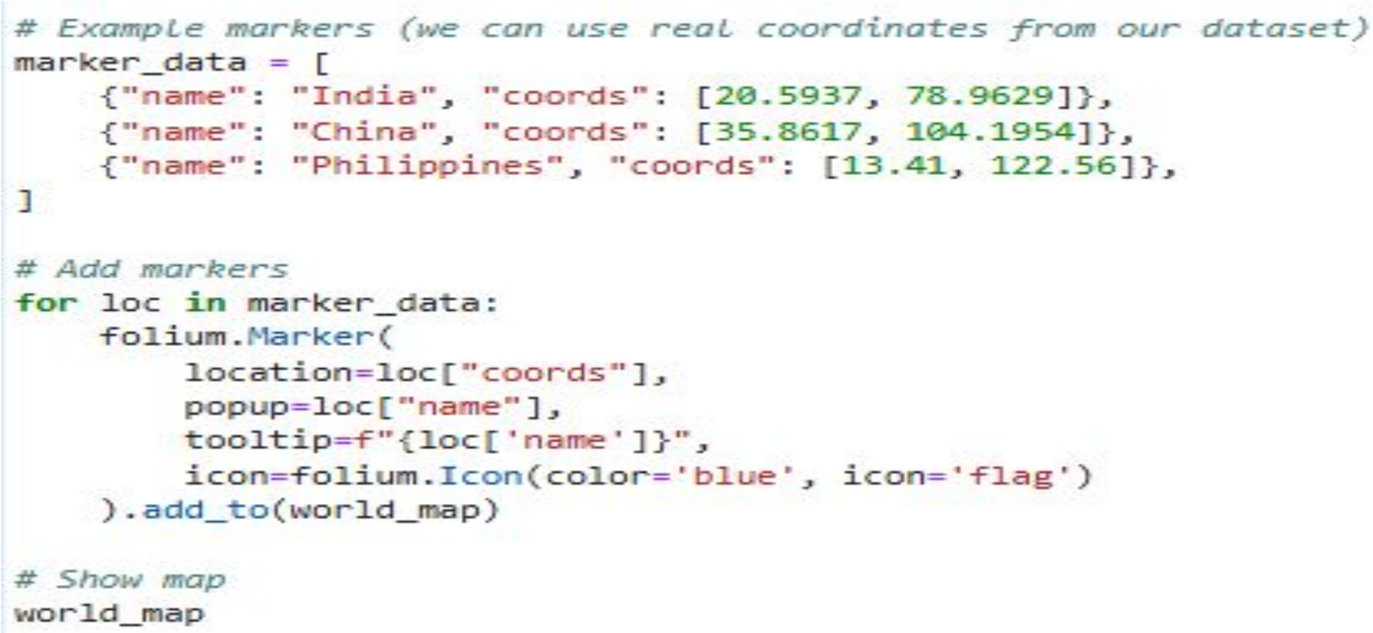
s × 39 columns

Tamara

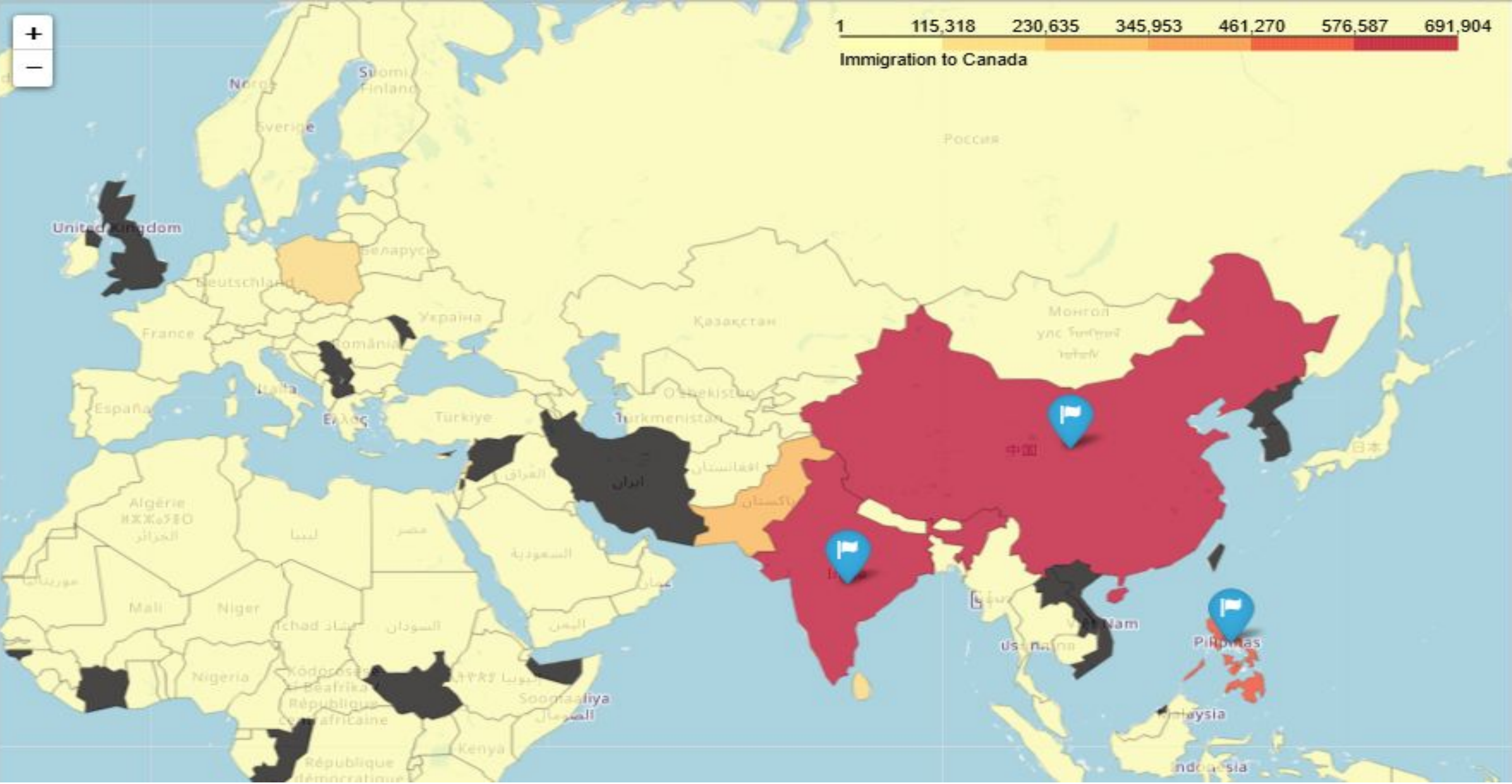

+
−
1
115,318
230,635
345,953
461,270
576,587
691,904
Immigration to Canada
Leaflet | Data by © OpenStreetMap, under ODbL.

Tamara

`Choropleth` map with some markers

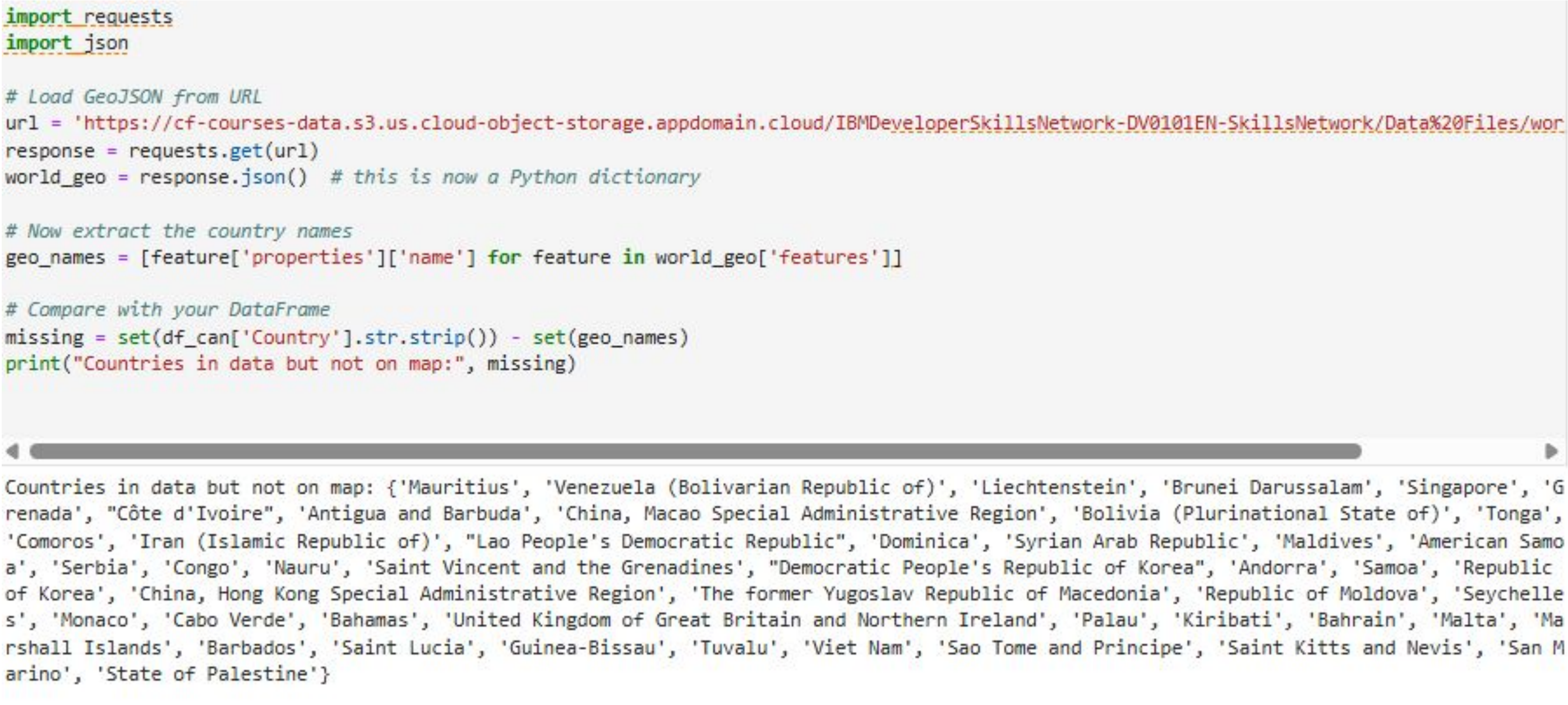


```
# Example markers (we can use real coordinates from our dataset)
marker_data = [
    {"name": "India", "coords": [20.5937, 78.9629]},
    {"name": "China", "coords": [35.8617, 104.1954]},
    {"name": "Philippines", "coords": [13.41, 122.56]},
]

# Add markers
for loc in marker_data:
    folium.Marker(
        location=loc["coords"],
        popup=loc["name"],
        tooltip=f"{loc['name']}",
        icon=folium.Icon(color='blue', icon='flag')
    ).add_to(world_map)

# Show map
world_map
```


+
−
1
115,318
230,635
345,953
461,270
576,587
691,904
Immigration to Canada
Suomi
Finland
Norge
Sverige
United Kingdom
Россия
Беларусь
Deutschland
Україна
France
România
Қазақстан
Монгол
Italia
España
Ελλάς
Türkiye
Oʻzbekiston
Turkmenistan
ایران
العراق
افغانستان
پاکستان
中国
日本
Algérie
الجزائر
ليبيا
مصر
السعودية
عمان
Mali
Niger
Tchad تشاد
السودان
اليمن
Việt Nam
Pilipinas
Nigeria
Ködörösêse tî Bêafrîka
République Centrafricaine
Soomaaliya
الصومال
Malaysia
Kenya
République démocratique
Indonesia

Tamara

Black means no data match



```
import requests
import json

# Load GeoJSON from URL
url = 'https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBMDeveloperSkillsNetwork-DV0101EN-SkillsNetwork/Data%20Files/wor
response = requests.get(url)
world_geo = response.json()  # this is now a Python dictionary

# Now extract the country names
geo_names = [feature['properties']['name'] for feature in world_geo['features']]

# Compare with your DataFrame
missing = set(df_can['Country'].str.strip()) - set(geo_names)
print("Countries in data but not on map:", missing)
```

```
Countries in data but not on map: {'Mauritius', 'Venezuela (Bolivarian Republic of)', 'Liechtenstein', 'Brunei Darussalam', 'Singapore', 'G
renada', "Côte d'Ivoire", 'Antigua and Barbuda', 'China, Macao Special Administrative Region', 'Bolivia (Plurinational State of)', 'Tonga',
'Comoros', 'Iran (Islamic Republic of)', "Lao People's Democratic Republic", 'Dominica', 'Syrian Arab Republic', 'Maldives', 'American Samo
a', 'Serbia', 'Congo', 'Nauru', 'Saint Vincent and the Grenadines', "Democratic People's Republic of Korea", 'Andorra', 'Samoa', 'Republic
of Korea', 'China, Hong Kong Special Administrative Region', 'The former Yugoslav Republic of Macedonia', 'Republic of Moldova', 'Seychelle
s', 'Monaco', 'Cabo Verde', 'Bahamas', 'United Kingdom of Great Britain and Northern Ireland', 'Palau', 'Kiribati', 'Bahrain', 'Malta', 'Ma
rshall Islands', 'Barbados', 'Saint Lucia', 'Guinea-Bissau', 'Tuvalu', 'Viet Nam', 'Sao Tome and Principe', 'Saint Kitts and Nevis', 'San M
arino', 'State of Palestine'}
```

Tamara

Conclusion: Visualizing Geospatial Data

- Folium is a powerful Python library for creating interactive maps, enabling easy visualization of geospatial data.

- With multiple map styles (e.g., street-level, Stamen, Mapbox), Folium allows users to customize their visualizations using the tiles parameter.

- Markers enhance map interactivity, providing context and labels through the popup parameter.

- Choropleth maps offer valuable insights by visualizing statistical data across regions, requiring GeoJSON files for precise geospatial representation.

Overall, Folium simplifies geospatial data visualization, making it accessible and effective for diverse applications.

References

IBM. (n.d.). *Data Visualization with Python* [Online course]. Coursera. Retrieved from https://www.coursera.org/learn/python-for-data-visualization

Data School. (2021, August 5). *Building Choropleth Maps with Folium and Pandas (Python)* [Video]. YouTube. https://www.youtube.com/watch?v=TDIo7s4SZA8