Creating a HTML App

13 December 2024

12:45

We will use Folium - first install Folium using:

pip install folium

Then we import using:

Import folium

**Step 1:**

To start by creating a map, we store this in a variable, map:

map = folium.Map(location=[Long Coord, Lang Coord], zoom\_start=12)

\*\*Note that zoom can be adjusted and coordinates any valid coordinate

**Step 2:**

Adding a marker to the map. Firstly we need to note that adding anything to the map will require:

map.add\_child()

And the script will have to end with:

map.save()

* We can use folium.FeatureGroup() to organize code by storing the add\_child in here

For the Location of the Markers:

folium.Marker(Coord, Coord)

For the Location Name:

popup="Name"

**Step 3:**

Getting Volcano markers - First we need to read the CSV we are extracting from:

Import pandas

df = pandas.read\_csv("CSV.txt")

lat = list(df["LAT"])

lon = list(df["LAT"])

* This stores the data for coordinates in a variable by reading from the headers in the CSV

Then we create multiple markers for the coordinates:

* We will use a For Loop

for lt, ln in zip(lat, lon):

    fg.add\_child(folium.Marker(location=[lt,ln]))

* We need to use zip() here because we have 2 variables in a For Loop

**Step 4:**

Adding elevation values from the CSV using dynamic markers:

* We declare it as a variable

lon = list(df["LAT"])

* Then include it in the For Loop

for lt, ln, el in zip(lat, lon, elev):

    fg.add\_child(folium.Marker(location=[lt,ln], popup=el))

**Step 5:**

Adding marker colors for elevations

* We define a function for this:

def colour\_producer(elevation):

    if elevation < 1000:

        return "green"

    elif 1000 <= elevation < 2000:

        return "orange"

    elif 2000 <= elevation < 3000:

        return "red"

    else:

        return "purple"

* We Then call this in the For Loop

for lt, ln, el in zip(lat, lon, elev):

    fg.add\_child(folium.Marker(location=[lt, ln], popup=el, icon=folium.Icon(color=colour\_producer(el))))

**Step 6:**

Adding Populations in Countries using a JSON file:

* First we open the json file:

fg.add\_child(folium.GeoJson(data=(open("world.json", 'r', encoding='utf-8-sig').read())))

* Next we need to use a lambda function (as it wont get called again) to search for an attribute in the JSON

fg.add\_child(folium.GeoJson(data=open("world.json", encoding='utf-8-sig').read(),

             style\_function=lambda x: {'fillColor':'green' if x['properties']['POP2005'] < 10000000

                                       else 'orange' if 10000000 <= x['properties']['POP2005'] < 20000000 else 'red'}))

\*\*This searches x as "POP2025" **in** "properties" **in** "x" in the JSON file and assigns colors based on the population Data

**Step 7:**

Adding layer toggles - We need to separate Volcanoes and Population into different Feature Groups.

We then add the below \*has to be after we add the feature groups to the map:

map.add\_child(folium.LayerControl())