

Web Map Creation with Python(Folium)

Table of content

1. Brief Background
2. Webmap code breakdown
3. Result snippets

Brief Background

This code is designed to create an interactive web map using folium, a Python library for visualizing geospatial data. The script demonstrates a simple yet effective approach to retrieve data from a PostgreSQL database and visualize it using folium. The application runs on a local host and consists of two main components: the database and the visualization.

Database:

The script connects to a PostgreSQL database, which hosts two tables: "Health" and "Schools." Both tables contain records with names and geographical coordinates (x for longitude and y for latitude). This data serves as the basis for the visualization.

Visualization:

The visualization component is developed using folium, which offers a range of tools for creating interactive maps. The retrieved data is fed into the folium library

to generate a web map. Feature groups are created for health facilities and schools to differentiate between the two types of markers. The markers' coordinates are extracted from the database, and custom icons are assigned to each type.

Additionally, a popup feature displays the name of the facility or school when the user clicks on the respective marker.

Integration:

The script demonstrates how Python's data manipulation capabilities can be effectively combined with the folium library to create informative and interactive visualizations. The data is retrieved, processed, and visualized within the same script, simplifying the process of generating maps.

Local Host:

As the application runs on the local host, the resulting HTML map file can be opened and viewed in any web browser. This ensures easy access to the visualization and provides users with an intuitive interface to explore the geographical distribution of health facilities and schools.

Summary:

In conclusion, the code combines Python's capabilities for database interaction and data manipulation with the powerful folium library to generate an interactive web

map. The database provides geographical data, which is then processed and visualized on the map, enabling users to explore and analyze the locations of health facilities and schools. This approach offers a seamless solution to working with geospatial data and presenting it effectively.

Webmap Code breakdown

This code is written in Python and uses the psycopg2 library to connect to a PostgreSQL database. The primary purpose of the code is to retrieve data from two tables, "Health" and "Schools," to create a map visualization using the folium library. The code demonstrates a connection to a PostgreSQL database, extraction of data, and integration with the folium library to generate an interactive map with markers.

Here's a step-by-step breakdown of the code:

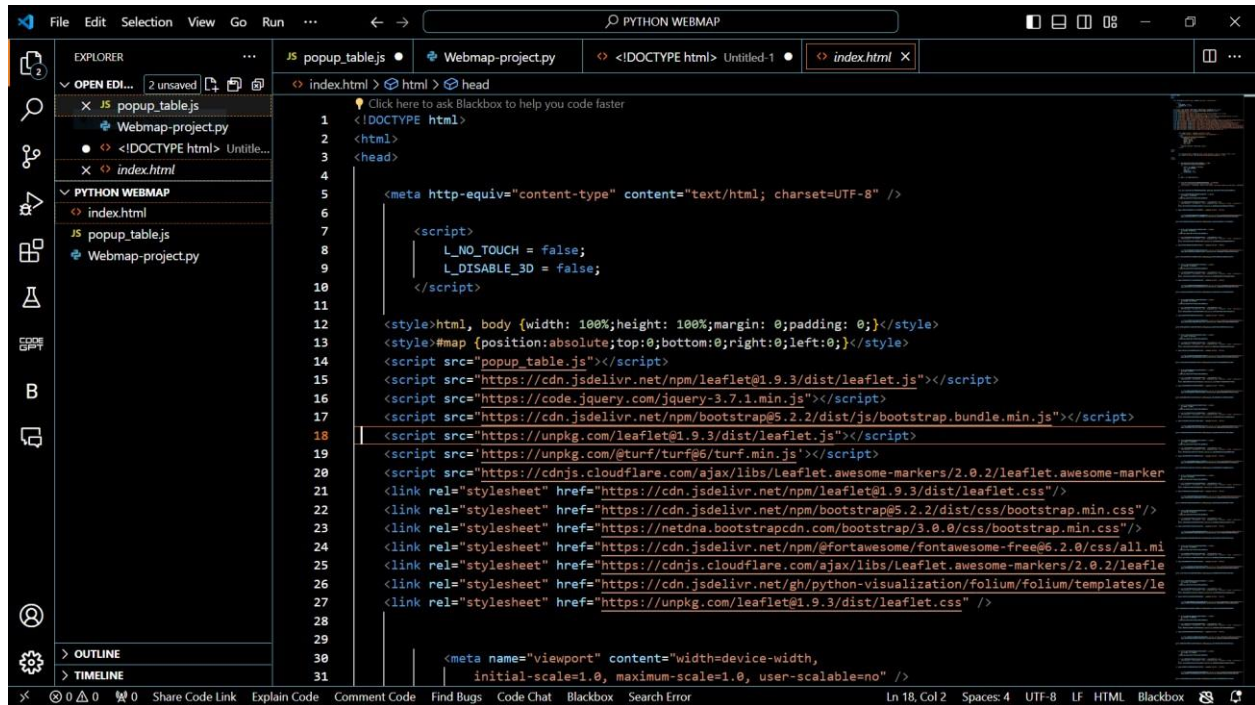
1. **Import Libraries:** The script begins by importing the necessary libraries: psycopg2 for PostgreSQL database interaction and folium for map visualization.
2. **Define PostgreSQL Connection Parameters:** The script defines the PostgreSQL connection parameters: user, password, database, host, and port.
3. **Connect to PostgreSQL Database:** The connect_to_postgres() function establishes a connection to the PostgreSQL database using the defined parameters. It returns the connection object if successful or raises an exception.

4. **Retrieve Data:** Two SELECT queries retrieve data from the "Health" and "Schools" tables. The retrieved data contains names, and x (longitude) and y (latitude) coordinates.
5. **Initialize Map:** A folium map object is initialized with a specific location and zoom level.
6. **Create Feature Groups:** Two feature groups, health_group and school_group, are created to hold markers for health facilities and schools, respectively. Each group is assigned a name.
7. **Add Markers:** The script adds markers to each feature group by iterating over the retrieved data. Markers are positioned based on the latitude and longitude coordinates, and a popup is added to each marker with the facility or school's name. Custom icons are used for both types of markers, with different colors and icons.
8. **Add Feature Groups to Map:** Both feature groups are added to the map to visualize the health facilities and schools.
9. **Add Layer Control to Map:** LayerControl is added to the map to enable users to toggle the visibility of the health facilities and schools layers.
10. **Save Map to HTML File:** The script saves the generated map to an HTML file for easy viewing and sharing.

In summary, this Python script establishes a connection to a PostgreSQL database, retrieves data from two tables, and generates an interactive map using folium. The map shows markers representing health facilities and schools, each with a custom icon and a popup displaying the

name of the facility or school. The LayerControl feature allows users to toggle the visibility of the markers on the map.

Code Snippets:



```
1 <!DOCTYPE html>
2 <html>
3 <head>
4
5     <meta http-equiv="content-type" content="text/html; charset=UTF-8" />
6
7     <script>
8         L_NO_TOUCH = false;
9         L_DISABLE_3D = false;
10    </script>
11
12    <style>html, body {width: 100%;height: 100%;margin: 0;padding: 0;}</style>
13    <style>#map {position: absolute; top: 0; bottom: 0; right: 0; left: 0;}</style>
14    <script src="popup_table.js"></script>
15    <script src="https://cdn.jsdelivr.net/npm/leaflet@1.9.3/dist/leaflet.js"></script>
16    <script src="https://code.jquery.com/jquery-3.7.1.min.js"></script>
17    <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.2.2/dist/js/bootstrap.bundle.min.js"></script>
18    <script src="https://unpkg.com/leaflet@1.9.3/dist/leaflet.js"></script>
19    <script src="https://unpkg.com/@turf/turf@6/turf.min.js"></script>
20    <script src="https://cdnjs.cloudflare.com/ajax/libs/Leaflet.awesome-markers/2.0.2/leaflet.awesome-marker"
21    <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/leaflet@1.9.3/dist/leaflet.css">
22    <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap@5.2.2/dist/css/bootstrap.min.css">
23    <link rel="stylesheet" href="https://netdna.bootstrapcdn.com/bootstrap/3.0.0/css/bootstrap.min.css">
24    <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/@fortawesome/fontawesome-free@6.2.0/css/all.min"
25    <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/Leaflet.awesome-markers/2.0.2/leaflet"
26    <link rel="stylesheet" href="https://cdn.jsdelivr.net/gh/python-visualization/folium/folium/templates/le
27    <link rel="stylesheet" href="https://unpkg.com/leaflet@1.9.3/dist/leaflet.css" />
28
29
30    <meta name="viewport" content="width=device-width,
31          initial-scale=1.0, maximum-scale=1.0, user-scalable=no" />
```

The html code works with the python code that has folium plugins loaded in order to generate the web map.

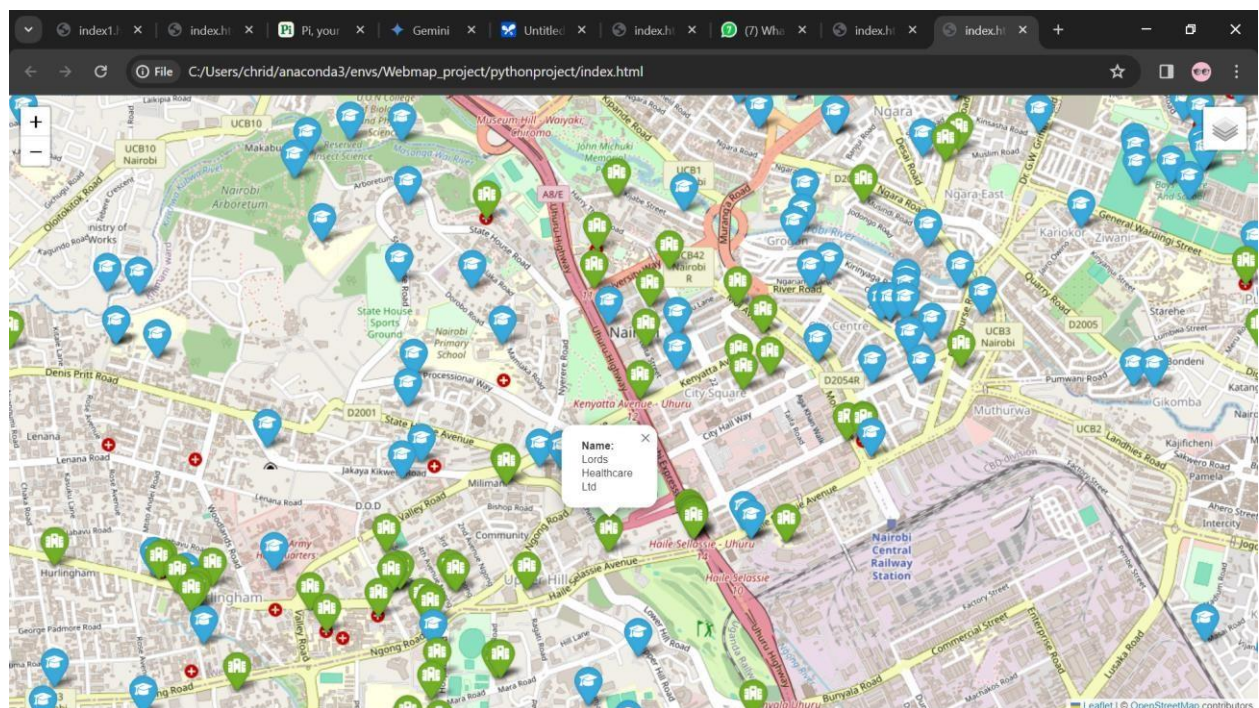
Python code snippet:

```
File Edit Selection View Go Run ... PYTHON WEBMAP
Webmap-project.py x <DOCTYPE html> Untitled-1 index.html
JS popup_table.js
Webmap-project.py
index.html
PYTHON WEBMAP
index.html
JS popup_table.js
Webmap-project.py
OUTLINE
TIMELINE
Share Code Link Explain Code Comment Code Find Bugs Code Chat Blackbox Search Error Spaces: 4 UTF-8 CRLF Python 3.11.5 (Webmap-project: conda) Blackbox

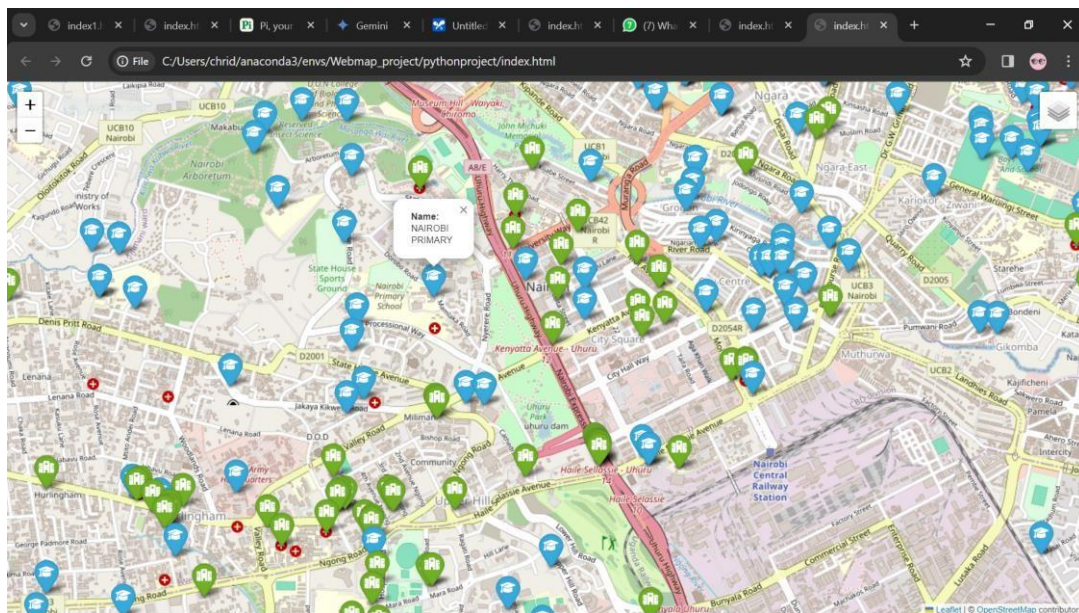
1 Click here to ask Blackbox to help you code faster
2 import psycopg2
3 import folium
4
5 # Connect to your PostgreSQL database
6 POSTGRES_USER = "postgres"
7 POSTGRES_PASSWORD = "Kinani"
8 POSTGRES_DB = "Morphius"
9 def connect_to_postgres():
10     try:
11         conn = psycopg2.connect(
12             user=POSTGRES_USER,
13             password=POSTGRES_PASSWORD,
14             database=POSTGRES_DB,
15             host="localhost",
16             port="5432",
17         )
18         print("Successfully connected to the PostgreSQL database")
19         return conn
20     except (Exception, psycopg2.DatabaseError) as error:
21         print("Error while connecting to PostgreSQL", error)
22
23 # Call the function to connect to the database
24 conn = connect_to_postgres()
25 conn.autocommit = False
26 if conn is not None:
27     cur = conn.cursor()
28     cur.execute("SELECT 'name', 'x', 'y' FROM 'Health'")
29     rows = cur.fetchall()
30
31     cur.execute("SELECT 'school_name', 'x_coord', 'y_coord' FROM 'Schools'")
32     school_rows = cur.fetchall()
```

Results Snippet:

Health facilities have Green Markers



Schools have blue markers



Implementing Layer control at the top right corner of the webpage where we deselect health facilities to visualize schools alone in order to get better clarity

