**Module 10: Developing Web Maps and Representing Information using Plots**

* **Use of Folium library**

1. Installing Folium in VS Code

* Go to Terminal
* Than go to new Terminal
* Write “pip install folium”
* Enter (it will automatically install folium)

Here’s a minimal example of creating an interactive map in VS Code:

import folium

# Create a map centered at a specific location (latitude, longitude)

m = folium.Map(location=[28.6139, 77.2090], zoom\_start=10) # Delhi

# Add a marker

folium.Marker(

location=[28.6139, 77.2090],

popup="New Delhi - Capital of India",

tooltip="Click here"

).add\_to(m)

# Save to HTML

m.save("map.html")

print("Map created! Open 'map.html' in your browser.")

**Explanation:**

* folium.Map() → creates a map object.
* location → sets the starting center of the map.
* zoom\_start → controls zoom level (lower = zoomed out).
* folium.Marker() → places a clickable pin.
* save() → saves the interactive map as an HTML file.

**Common Uses of Folium**

* Visualizing **sales or store locations**.
* Displaying **Earthquake or weather data**.
* Plotting **GPS tracking routes**.
* Creating **choropleth maps** (color-coded regions).
* Mapping **public transportation networks**.
* **Use of Pandas library**

**Pandas** library is like the “data backbone” when you’re building **Web Maps** and **Plots** in Python — it helps you handle and prepare data so that mapping libraries (like **Folium**, **Plotly**, or **Matplotlib**) can easily visualize it.

Let me break this into **two parts**:

1. Role of Pandas in Developing Web Maps

When building **interactive maps** (e.g., with Folium, Plotly, or Kepler.gl), Pandas is used to:

* **Load data** from files like CSV, Excel, or databases (e.g., locations, coordinates, population).
* **Clean and preprocess** data (remove missing values, fix inconsistent formats).
* **Filter data** for specific areas or conditions (e.g., only show cities with population > 1 million).
* **Merge datasets** (e.g., join location coordinates with demographic data).
* **Pass location data** (latitude & longitude) directly to map plotting functions.

**Example: Pandas + Folium for a Web Map**

import pandas as pd

import folium

# Load dataset with city name, latitude, longitude, population

df = pd.read\_csv("cities.csv")

# Create map

m = folium.Map(location=[20.5937, 78.9629], zoom\_start=5)

# Add markers for each city

for i, row in df.iterrows():

folium.Marker(

location=[row['Latitude'], row['Longitude']],

popup=f"{row['City']} - Pop: {row['Population']}"

).add\_to(m)

m.save("map.html")

**2. Role of Pandas in Representing Information using Plots**

When making plots (e.g., bar charts, line graphs, scatter plots), Pandas is used to:

* **Read and prepare** the dataset.
* **Group and summarize** data (groupby() for totals, averages, etc.).
* **Transform** data for different kinds of plots (pivot tables, aggregation).
* **Feed data directly** into plotting libraries like Matplotlib, Seaborn, or Plotly.

**Example: Pandas + Matplotlib for Visualization**

import pandas as pd

import matplotlib.pyplot as plt

# Read sales dataset

df = pd.read\_csv("sales.csv")

# Group sales by product

sales\_summary = df.groupby('Product')['Sales'].sum()

# Plot bar chart

sales\_summary.plot(kind='bar', title="Product Sales")

plt.ylabel("Total Sales")

plt.show()

* **Flow Chart of web map application**

This shows the logical steps when you build a small project with **Pandas + Folium/Plotly** to create maps and visualize data.

1.Start Application

2. Import Libraries │

(pandas, folium, etc.)

3. Load Dataset (CSV/Excel)│ → e.g., cities.csv

4. Clean & Process Data │ (lat, lon, values etc.)

5. Create Base Map (Folium/Leaflet) │ → folium.Map(location, zoom)

6. │ Add Markers / Layers │ → CircleMarker, Popup

7. Save Map as HTML │ → map.save("map.html")

8. │ Open in Browser / VS Code │ (View Interactive Map)

9. End Application

* **Developing web map using Folium and Pandas**

🌍 Example 2: Volcano Map with Folium & Pandas

import pandas as pd

import folium

# Load dataset

df = pd.read\_csv("volcanoes.csv")

# Function to color markers based on elevation

def color\_producer(elevation):

if elevation < 2000:

return "green"

elif 2000 <= elevation < 4000:

return "orange"

else:

return "red"

# Create base map

m = folium.Map(location=[20, 0], zoom\_start=2)

# Add markers

for i, row in df.iterrows():

folium.Marker(

location=[row["Latitude"], row["Longitude"]],

popup=f"{row['Name']} ({row['Elevation']} m)",

icon=folium.Icon(color=color\_producer(row["Elevation"]))

).add\_to(m)

# Save map

m.save("volcano\_map.html")

print("Volcano map created! Open volcano\_map.html in browser.")

* Volcanoes **below 2000m → Green markers**
* Volcanoes **2000–4000m → Orange markers**
* Volcanoes **above 4000m → Red markers**
* Click markers → popup shows **volcano name & elevation**.
* **Reading Information from titanic dataset and represent It using plots**

import seaborn as sns

import pandas as pd

import matplotlib.pyplot as plt

# Load Titanic dataset from seaborn

titanic = sns.load\_dataset("titanic")

print(titanic)

# (a) Countplot – Survival by Gender

sns.countplot(data=titanic, x="sex", hue="survived")

plt.title("Survival Count by Gender")

plt.show()

# (b) Bar Plot – Survival by Class

sns.countplot(data=titanic, x="class", hue="survived")

plt.title("Survival by Passenger Class")

plt.show()

# (c) Age Distribution – Histogram

sns.histplot(data=titanic, x="age", bins=30, kde=True)

plt.title("Age Distribution of Passengers")

plt.show()

# (d) Boxplot – Age vs Class

sns.boxplot(data=titanic, x="class", y="age")

plt.title("Passenger Age Distribution by Class")

plt.show()

# (e) Heatmap – Correlation Matrix

sns.heatmap(titanic.corr(numeric\_only=True), annot=True, cmap="coolwarm")

plt.title("Correlation Heatmap (Numeric Features)")

plt.show()

**Summary**

With these plots, you can quickly analyze:

* Which gender/class had higher survival rates
* How age was distributed
* Relationships between numerical variables