BAKHAT ALI

Institute of Geoinformatics and Earth Observation,

Pir Mehr Ali Shah *Arid* Agriculture *University* Rawalpindi , Punjab, Pakistan

bakhtali21uaar@gmail.com

Assignment – Python for GIS Applications Subject: GIS Topic: Python in GIS Total Marks: 20

Question Paper Q1. Python Environment Setup (2 Marks) a) List and explain any three Python libraries used in GIS. b) Write a command to install geopandas and explain its role in spatial analysis.

Q1 Why Python for GIS?

Python for Geographic Information Systems (GIS) is a powerful combination due to Python's versatility and the rich ecosystem of libraries designed for spatial data handling. Below is a guide that covers essential libraries, workflows, and examples for using Python in GIS.

- Easy to learn and use
  - Extensive GIS libraries (ArcPy, GeoPandas, GDAL)
  - Strong community support
  - Integration with GIS software
- Using Python for GIS
- Key Libraries for GIS in Python

Department Tools: Google Colab in GIS:

- Google Colab is a cloud-based platform that allows users to write, run, and share Python code in an interactive environment without the need for local setup. It is especially useful in GIS for collaborative analysis, data processing, and machine learning applications.
- Features of Google Colab for GIS:
- Cloud-Based Environment: No need for local installation; access powerful computing resources directly from a web browser.
- Pre-Installed Libraries: Comes with many popular Python libraries such as GeoPandas, Rasterio, Shapely, and Pyproj, facilitating geospatial data analysis.

➢ Google Colab in Google Earth Engine (GEE)

Google Colab can be integrated with Google Earth Engine (GEE) to facilitate advanced geospatial analysis using Python in a cloud environment. This combination leverages Colab's ease of use and collaboration features with GEE's extensive satellite imagery and geospatial data processing capabilities

Using Google Colab with Google Earth Engine (GEE) Python API

Google Colab provides an interactive environment to run Python scripts that interact with Google Earth Engine (GEE), allowing users to access, analyze, and visualize large geospatial datasets in the cloud.

Steps to Set Up GEE in Google Colab:

- **Install the GEE Python API**

```
!pip install earthengine-api
```

- **Authenticate and Initialize GEE**

Import and authenticate

```
import ee

import geemap

ee.Authenticate()

ee.Initialize(project='ee-bakhtali21uaar')  # ← your project ID
```

A)Google colab Jupiter notebook python use Api gee libraries

1. **B)Geopandas**
   - Builds on pandas to handle geospatial data.
   - Provides easy-to-use data manipulation with geographic data.

   Installation:

   !pip install Geopandas

   Explanation of geopandas in spatial analysis:

   geopandas is a Python library that extends the capabilities of the popular pandas library to work with geospatial data. It simplifies handling and analyzing spatial datasets by providing easy-to-use data structures and functions for reading, writing, and manipulating vector data such as shapefiles, GeoJSON, and other GIS formats.

   Role in spatial analysis:

- **Reading and writing spatial data:** Load shapefiles, GeoJSON, KML, etc.
- **Spatial operations:** Perform spatial joins, overlays, buffering, and intersection.
- **Coordinate reference systems (CRS):** Manage and transform coordinate systems.
- **Visualization:** Plot spatial data directly using matplotlib.
- **Analysis:** Conduct spatial queries, aggregations, and calculations.

2. **Shapely**
   - For manipulation and analysis of planar geometric objects.

   Installation:

   !pip install shapely
3. **Fiona**
   - For reading and writing vector data.

   Installation:

   !pip install fiona
4. **Rasterio**
   - For reading and writing raster data.

   Installation:

   pip install rasterio
5. **Folium**
   - For visualizing spatial data with interactive maps.

   Installation:

   !pip install folium
6. **Pyproj**
   - For coordinate transformation and cartographic projections.

   Installation:

   !pip install pyproj

Basic Workflow for GIS with Python

1. Import Libraries

```
import geopandas as gpd

import matplotlib.pyplot as plt
```

2. Reading Geospatial Data

Load a shapefile using GeoPandas.

```
    gdf = gpd.read_file('path/to/your/shapefile.shp')
```

3. Exploring Data

You can inspect the first few rows and get basic information.

```
print(gdf.head())

print(gdf.crs)  # Print the coordinate reference system
```

4. Data Manipulation

```
# Filtering data

urban_areas = gdf[gdf['type'] == 'Urban']

# Creating a new column

gdf['area'] = gdf.geometry.area
```

5. Plotting Data

Using Matplotlib to visualize the data.

```
gdf.plot()

plt.show()
```

6. Saving Data

After modifications, you might want to save your data back.

```
gdf.to_file('modified_shapefile.shp')
```

DATA visualization in Python in Gis

```
pip install matplotlib

!pip install seaborn

!pip install plotly

import matplotlib.pyplot as plt

import seaborn as sns

import plotly.express as px

import matplotlib.pyplot as plt

import seaborn as sns

import plotly.express as px

# Line Plot

plt.plot([1, 2, 3])

plt.show()

# Bar Chart

plt.bar([1, 2, 3], [4, 5, 6])

plt.show()

# Scatter Plot

plt.scatter([1, 2, 3], [4, 5, 6])

plt.show()

# Bar Chart

sns.barplot(x=[1, 2, 3], y=[4, 5, 6])

plt.show()

# Scatter Plot

sns.scatterplot(x=[1, 2, 3], y=[4, 5, 6])

plt.show()

# Heatmap

sns.heatmap([[1, 2], [3, 4]])
```

```
plt.show()
# Line Plot
fig = px.line(x=[1, 2, 3], y=[4, 5, 6])
fig.show()

# Bar Chart
fig = px.bar(x=[1, 2, 3], y=[4, 5, 6])
fig.show()

# Scatter Plot
fig = px.scatter(x=[1, 2, 3], y=[4, 5, 6])
fig.show()
```

7. Working with Raster Data

Using Rasterio to read a raster file.

```
import rasterio
with rasterio.open('path/to/raster.tif') as src:
  image = src.read()
print(src.profile)  # Print metadata of the raster file
```

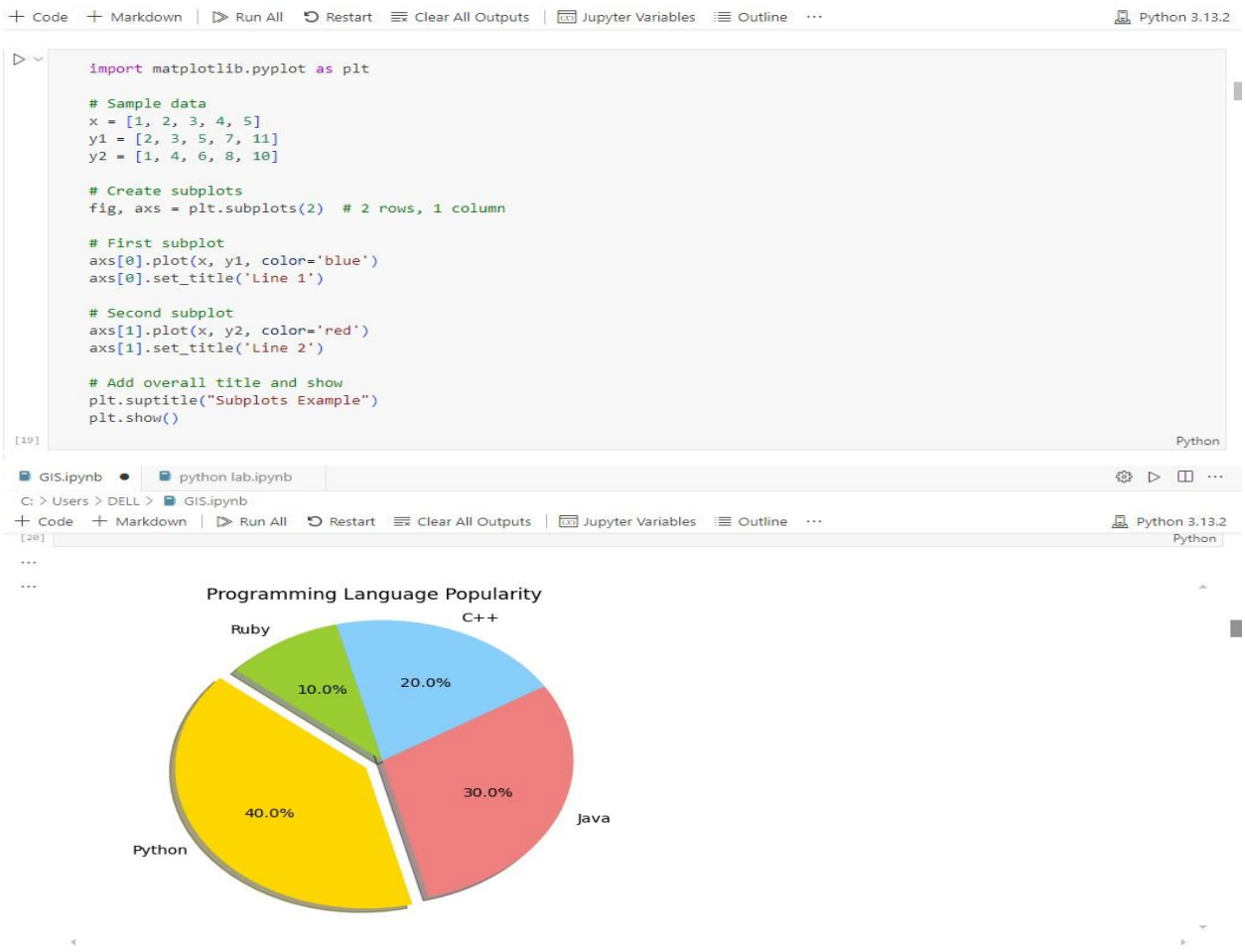
8. Visualization with Folium

Creating an interactive map.

```
import folium
# Create a map centered at a specific location
m = folium.Map(location=[latitude, longitude], zoom_start=10)
# Add GeoJSON data to the map
folium.GeoJson(gdf).add_to(m)
# Save the map to an HTML file
m.save('map.html')
```

Example: Plotting Points of Interest

```
import geopandas as gpd
import folium
# Load a GeoDataFrame with points of interest
gdf = gpd.read_file('points_of_interest.shp')
# Create a base map
m = folium.Map(location=[latitude, longitude], zoom_start=12)
# Add points to the map
for idx, row in gdf.iterrows():
 folium.Marker(location=[row.geometry.y, row.geometry.x],
 popup=row['name']).add_to(m)
                # Save the interactive map
                m.save('points_of_interest_map.html')
```

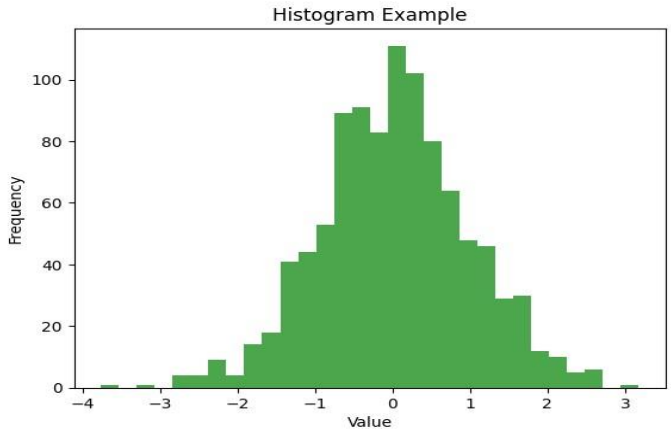


C: > Users > DELL > GIS.ipynb

+ Code + Markdown | Run All Restart Clear All Outputs | Jupyter Variables Outline ... Python 3.13.2

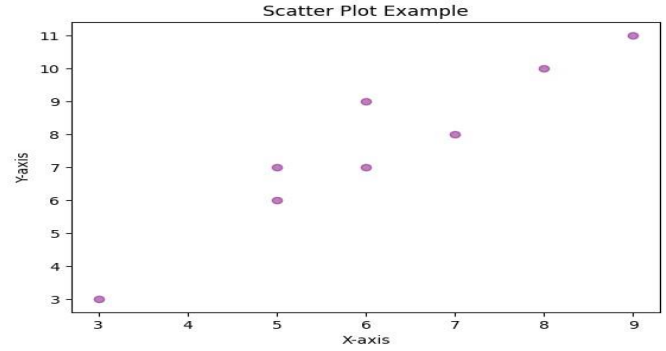
```python
import matplotlib.pyplot as plt
import numpy as np

# Generate random data
data = np.random.randn(1000)

# Create a histogram
plt.hist(data, bins=30, color='green', alpha=0.7)

# Add title and labels
plt.title("Histogram Example")
plt.xlabel("Value")
plt.ylabel("Frequency")

# Show the plot
plt.show()
```

[21] Python



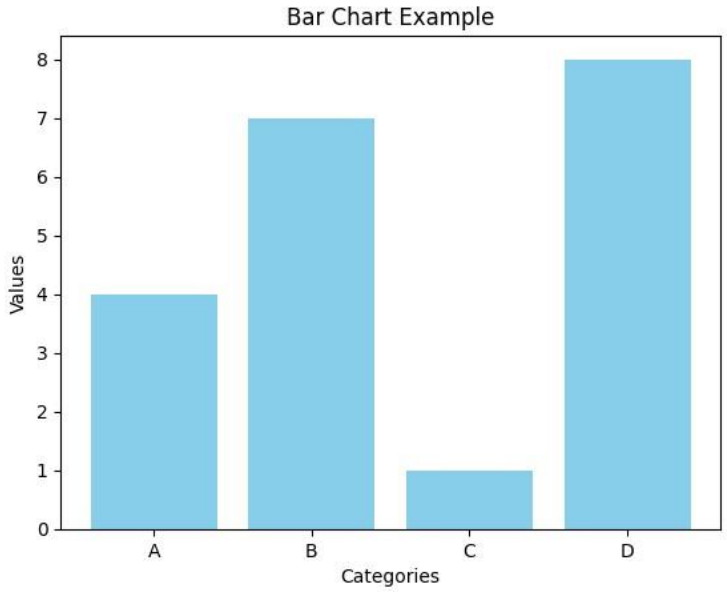

```python
import matplotlib.pyplot as plt

# Sample data
x = [5, 7, 8, 5, 6, 9, 3, 6]
y = [7, 8, 10, 6, 7, 11, 3, 9]

# Create a scatter plot
plt.scatter(x, y, color='purple', alpha=0.5)

# Add title and labels
plt.title("Scatter Plot Example")
plt.xlabel("X-axis")
plt.ylabel("Y-axis")

# Show the plot
plt.show()
```

Python



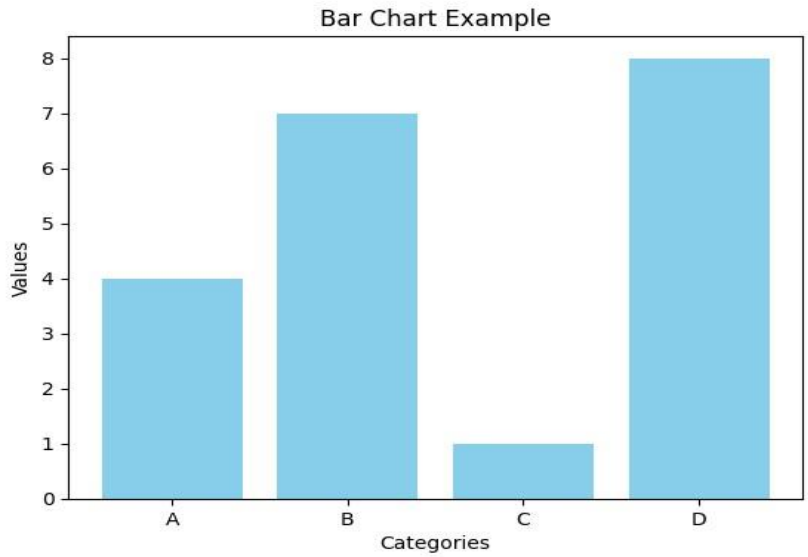
```python
import matplotlib.pyplot as plt

# Sample data
categories = ['A', 'B', 'C', 'D']
values = [4, 7, 1, 8]

# Create a bar chart
plt.bar(categories, values, color='skyblue')

# Add title and labels
plt.title("Bar Chart Example")
plt.xlabel("Categories")
plt.ylabel("Values")

# Show the plot
plt.show()
```

23] Python




```python
import matplotlib.pyplot as plt

# Sample data
categories = ['A', 'B', 'C', 'D']
values = [4, 7, 1, 8]

# Create a bar chart
plt.bar(categories, values, color='skyblue')

# Add title and labels
plt.title("Bar Chart Example")
plt.xlabel("Categories")
plt.ylabel("Values")

# Show the plot
plt.show()
```

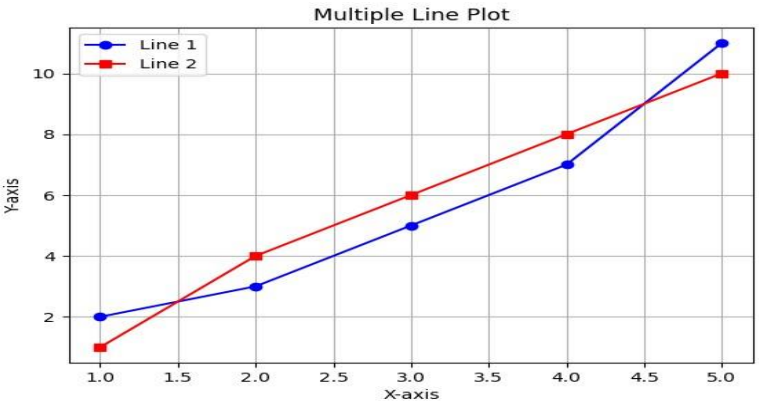
[24] Python




```python
import matplotlib.pyplot as plt

# Sample data for two lines
x = [1, 2, 3, 4, 5]
y1 = [2, 3, 5, 7, 11]
y2 = [1, 4, 6, 8, 10]

# Create a line plot
plt.plot(x, y1, label='Line 1', marker='o', color='blue')
plt.plot(x, y2, label='Line 2', marker='s', color='red')

# Add title and labels
plt.title("Multiple Line Plot")
plt.xlabel("X-axis")
plt.ylabel("Y-axis")
plt.legend()

# Show the plot
plt.grid()
plt.show()
```

5]

Python



```python
import folium
import geemap

# Define Rawalpindi center coordinates
rawalpindi_center = [33.6844, 73.0479]

# Create a Folium Map
m = folium.Map(location=rawalpindi_center, zoom_start=11)

# Add LULC Tile Layer from ESA WorldCover 2021 (Public Dataset)
esa_lulc_url = "https://services.terrascope.be/wmts/v2?request=GetCapabilities"

# Add LULC layer to the map
folium.raster_layers.WmsTileLayer(
    url=esa_lulc_url,
    layers="WORLDCOVER_2021_MAP",
    name="ESA WorldCover 2021",
    format="image/png",
    transparent=True,
).add_to(m)

# Add Layer Control
folium.LayerControl().add_to(m)

# Show the Map
m
```

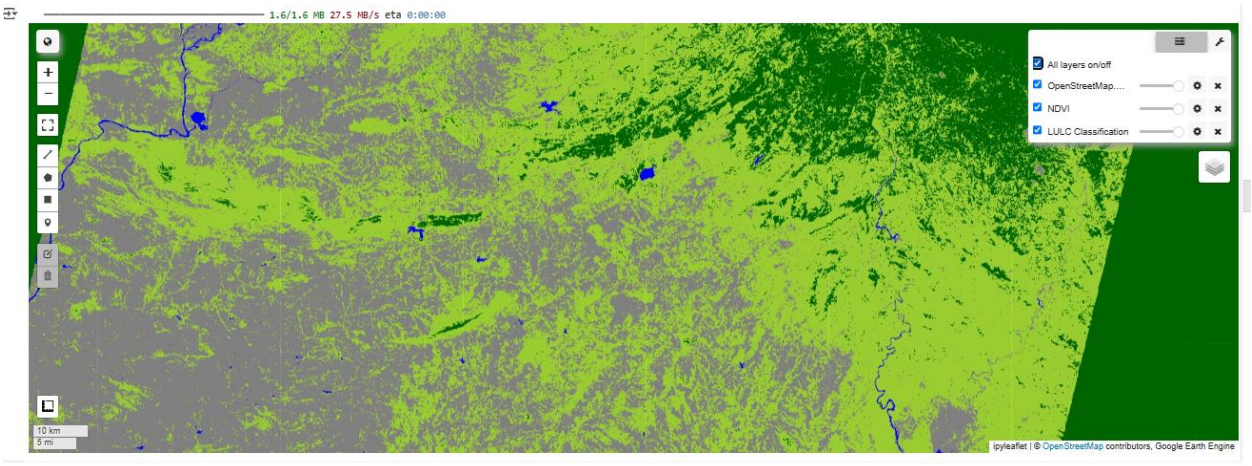
Lab 2 NDVI

Code :

```
import ee

import geemap

ee.Authenticate()

ee.Initialize(project='ee-bakhtali21uaar')  # ← your project ID

# STEP 3: Define Area of Interest (AOI)

aoi = ee.Geometry.BBox(73.00, 33.50, 73.20, 33.70)  # Adjust coordinates as needed

# STEP 4: Load and process Landsat 8 Level-2 imagery

image = (ee.ImageCollection("LANDSAT/LC08/C02/T1_L2")

        .filterBounds(aoi)

        .filterDate('2022-06-01', '2022-06-30')

        .filter(ee.Filter.lt('CLOUD_COVER', 10))

        .sort('CLOUD_COVER')

        .first())

# Apply scaling factors

def apply_scale_factors(img):

    optical = img.select(['SR_B.*']).multiply(0.0000275).add(-0.2)

    return img.addBands(optical, overwrite=True)

image = apply_scale_factors(image)

# STEP 5: Calculate NDVI

ndvi = image.normalizedDifference(['SR_B5', 'SR_B4']).rename('NDVI')
```

```
# STEP 6: NDVI-based LULC classification

lulc = ndvi.expression(

    "(b('NDVI') < 0) ? 1"          # Water

    ": (b('NDVI') < 0.2) ? 2"      # Urban/Barren

    ": (b('NDVI') < 0.5) ? 3"      # Vegetation

    ": 4"                          # Forest

).rename('LULC')

# STEP 7: Display results using geemap

Map = geemap.Map(center=[33.60, 73.10], zoom=10)

Map.addLayer(ndvi, {'min': -1, 'max': 1, 'palette': ['blue', 'white', 'green']}, 'NDVI')

Map.addLayer(lulc, {'min': 1, 'max': 4, 'palette': ['blue', 'gray', 'yellowgreen', 'darkgreen']},
'LULC Classification')

Map.addLayerControl()

Map
```

Output:



Q2. Working with Shapefiles (4 Marks) Using the geopandas library: a) Load any shapefile of your choice (e.g., districts of Punjab or world countries). b) Display the first five rows of the dataset. c) Plot the shapefile with appropriate title and color customization. Q3. Area Calculation (4 Marks) a) Calculate the area of each feature in the shapefile in square kilometers. b) Add a new column named 'Area_km2' to store this information. c) Filter and display only those features where area is greater than 10,000 km². Q4. Buffer and Spatial Join (6 Marks) a) Create a 10 km buffer around each polygon feature. b) Load a point shapefile (e.g., cities, schools, etc.) and perform a spatial join to count how many points fall within each polygon. c) Save the final output to a new shapefile named analysis_result.shp.

✅ **Setup in Google Colab**

python

CopyEdit

```
# Install required packages (if needed in Colab)
!pip install geopandas geemap shapely


# Import libraries
import geopandas as gpd
import matplotlib.pyplot as plt
import ee
```

```
import geemap

# Authenticate and initialize GEE
ee.Authenticate()
ee.Initialize(project='ee-bakhtali21uaar')
```

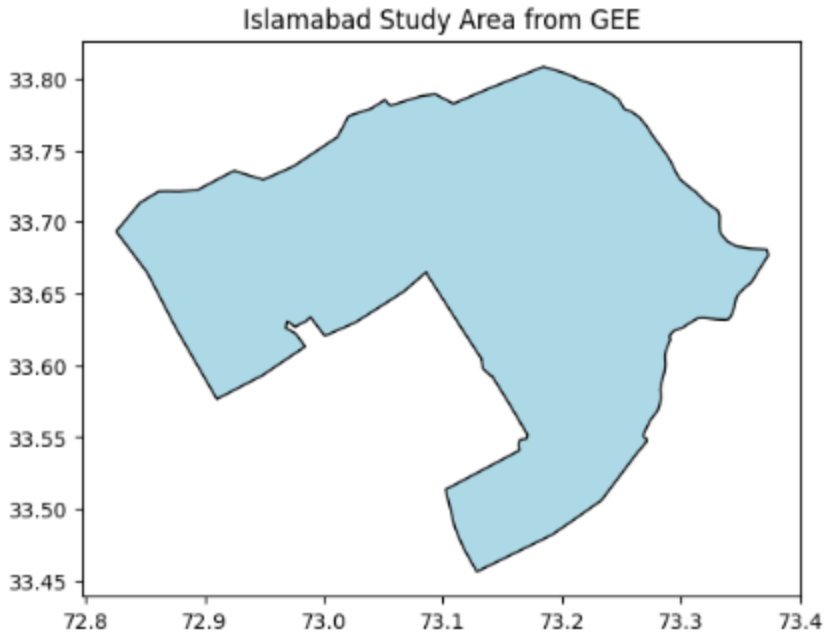
All-in-One Python + GEE Code for Q2, Q3, and Q4

```
# ✅ Step 1: Install required packages
!pip install -U geemap geopandas

# ✅ Step 2: Import libraries
import ee
import geemap
import geopandas as gpd
import matplotlib.pyplot as plt

# ✅ Step 3: Authenticate and Initialize GEE
ee.Authenticate()
ee.Initialize(project='ee-bakhtali21uaar')

# ✅ Step 4: Load Islamabad shapefile (study area) from GEE assets
study_area = ee.FeatureCollection('projects/ee-bakhtali21uaar/assets/isb')

# ✅ Step 5: Export GEE shapefile to GeoJSON for local use
geemap.ee_export_vector(study_area, filename='isb_study_area.geojson')
```

```
# ✅ Step 6: Load GeoJSON into GeoPandas
gdf = gpd.read_file('isb_study_area.geojson')

# ✅ Q2: Display and plot the shapefile
print("Q2: First 5 rows of shapefile:")
print(gdf.head())

gdf.plot(color='lightblue', edgecolor='black')
plt.title("Islamabad Study Area from GEE")
plt.show()

# ✅ Q3: Area Calculation
gdf_proj = gdf.to_crs(epsg=6933)  # EPSG:6933 = Cylindrical Equal Area (for accurate area)
gdf_proj['Area_km2'] = gdf_proj.geometry.area / 1e6  # Area in square kilometers

# b) Display updated dataframe
print("\nQ3: Area in km² added:")
print(gdf_proj[['Area_km2']])

# c) Filter features > 100 km² (if multiple polygons)
large_areas = gdf_proj[gdf_proj['Area_km2'] > 100]
print("\nFeatures with area > 100 km²:")
print(large_areas)

# ✅ Q4: Buffer + Spatial Join
```

```
# a) Create 10 km buffer around polygons
gdf_buffered = gdf_proj.copy()
gdf_buffered['geometry'] = gdf_buffered.geometry.buffer(10000)  # 10,000 meters = 10 km

# ✅ ✅ REPLACE THIS with your Islamabad points shapefile
points = gpd.read_file("isb_points.shp")  # Your local Islamabad points file
points_proj = points.to_crs(epsg=6933)

# b) Spatial join: count how many points fall within each buffer
joined = gpd.sjoin(gdf_buffered, points_proj, predicate='contains')

# Count points per polygon
point_counts = joined.groupby('index_left').size()
gdf_buffered['Point_Count'] = gdf_buffered.index.map(point_counts).fillna(0)

# ✅ Final Result
print("\nFinal Result with Area and Point Count:")
print(gdf_buffered[['Area_km2', 'Point_Count']])

# ✅ Export final shapefile for use in GIS software
gdf_buffered.to_file('isb_analysis_result.shp')
```

```
Q2: First 5 rows of shapefile:
                    id  OBJECTID  Shape_Leng   DISTRICT  \
0  0000000000000000000        55    1.691534  ISLAMABAD

                 ADMIN_UNIT  Shape_Area  Shape_Le_1  \
0  FEDERAL CAPITAL TERRITORY     0.08805    1.691534

                   PROVINCE  Shape_Le_2  \
0  FEDERAL CAPITAL TERRITORY    1.691534

                                            geometry
0  POLYGON ((72.82486 33.69389, 72.83472 33.68304...
```




```
Q3: Area in km² added:
     Area_km2
0  905.775433

Features with area > 100 km²:
                    id  OBJECTID  Shape_Leng   DISTRICT  \
0  0000000000000000000        55    1.691534  ISLAMABAD

                 ADMIN_UNIT  Shape_Area  Shape_Le_1  \
0  FEDERAL CAPITAL TERRITORY     0.08805    1.691534

                   PROVINCE  Shape_Le_2  \
0  FEDERAL CAPITAL TERRITORY    1.691534

                                            geometry    Area_km2
0  POLYGON ((7026599.966 4060516.772, 7027551.236...  905.775433
```