Programming for Geo Informatics - Lab 5

Submitted By :

Ashwin E

SC24M136

Points

Program 1

1.1 - (Create Point geometric object(s) with coordinates)

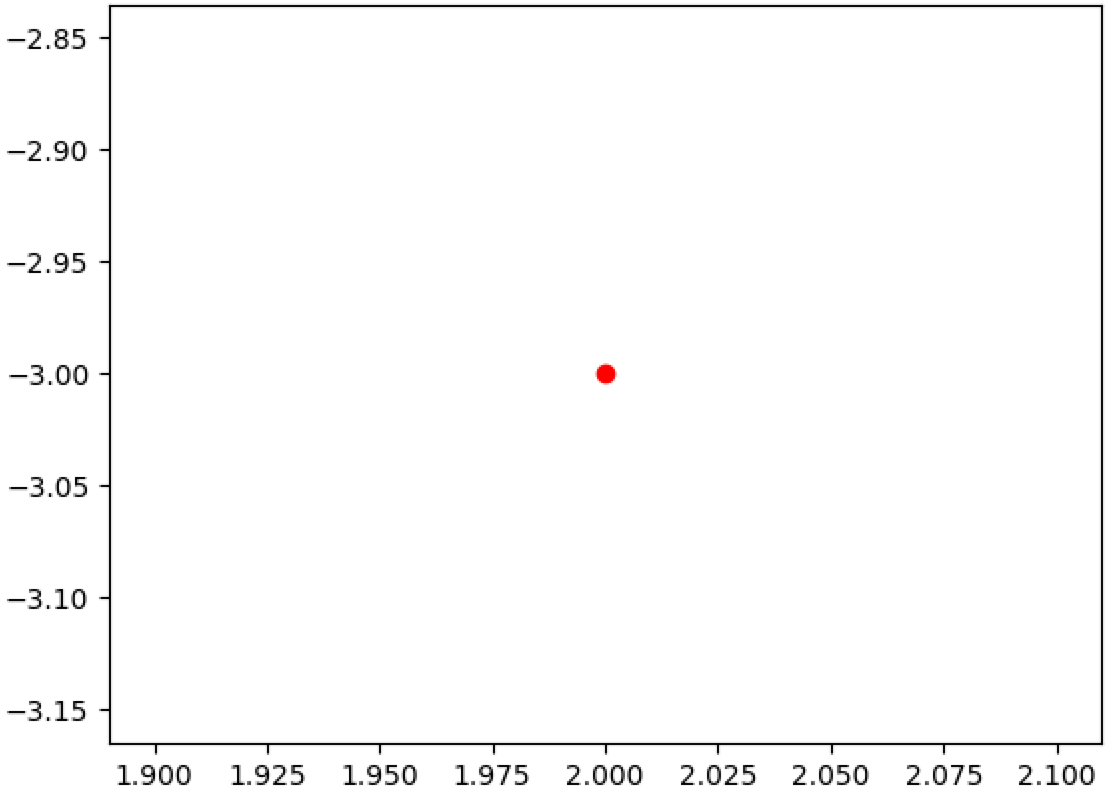
In [218…

```python
from shapely.geometry import Point
import matplotlib.pyplot as plot
point = Point(2.0,-3.0)
```

1.2 - (Display the point on screen)

In [219…

```python
figure, axes = plot.subplots()
x, y = point.xy
axes.plot(x, y, 'ro')
plot.show()
```



1.3 - (Print the Points)

In [220…

```
print(point)
```

```
POINT (2 -3)
```

1.4 - (Display the Type of the Point Data)

In [221…

```
print(f"Type of point1: {type(point)}")
```

```
Type of point1: <class 'shapely.geometry.point.Point'>
```

1.5 - (Getting the xy coordinate of points)

In [222…

```
print(point.xy)
```

```
(array('d', [2.0]), array('d', [-3.0]))
```

1.6 - (Read x and y coordinates separately and Display the coordinates)

In [223…

```
print(f"The x-coordinate is: {point.x}, and the y-coordinate is: {point.y}")
```

```
The x-coordinate is: 2.0, and the y-coordinate is: -3.0
```

1.7 - (Calculating the distance between two points)

In [224…

```
A = Point(5, 2)
B = Point(-3, 8)
AB = A.distance(B)
print(f"Distance between point_a and point_b: {AB:.3f} units")
```

```
Distance between point_a and point_b: 10.000 units
```

Program 2

2.1 - (Create a LineString from the Point objects)

In [225…

```
from shapely.geometry import Point, LineString
point1 = Point(7, 7)
point2 = Point(6, 6)
point3 = Point(5, 5)
line1 = LineString([point1, point2, point3])
print(line1)
```

```
LINESTRING (7 7, 6 6, 5 5)
```

2.2 - (Create a LineString using coordinate tuples)

In [226…

```
point1 = (7, 8)
point2 = (8, 6)
point3 = (5, 2)
line2 = LineString([point1, point2, point3])
print(line2)
```

```
LINESTRING (7 8, 8 6, 5 2)
```

2.3 - (Check if lines are identical)

In [227…

```
print(f"Whether the lines are identical? \n{line1.equals(line2)}")
print(f"Whether the lines are identical? \n{line1 == line2}")
```

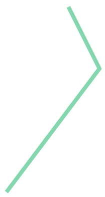
```
Whether the lines are identical?
False
Whether the lines are identical?
False
```

2.4 - (Display the linestring)

In [228…

```
line2
```

Out[228…



2.5 - (Print the Linestring)

In [229…

```python
print(line1)
```

```
LINESTRING (7 7, 6 6, 5 5)
```

2.6 - (Display the Type of the Line Object)

In [230…

```python
print(f"Type of line object : {type(line1)}")
```

```
Type of line object : <class 'shapely.geometry.linestring.LineString'>
```

2.7 - (Display the Geometry of the Line Object)

In [231…

```python
print ("Geometry of Line Object : ",line1.wkt)
```

```
Geometry of Line Object :  LINESTRING (7 7, 6 6, 5 5)
```

2.8 - (Get the xy coordinate tuples)

In [232…

```python
coords1 = list(line1.coords)
coords2 = list(line2.coords)
print("(X,Y) coordinates of line1 :", coords1)
print("(X,Y) coordinates of line2 :", coords2)
```

```
(X,Y) coordinates of line1 : [(7.0, 7.0), (6.0, 6.0), (5.0, 5.0)]
(X,Y) coordinates of line2 : [(7.0, 8.0), (8.0, 6.0), (5.0, 2.0)]
```

2.9 - (Read x and y coordinates separately and Display the coordinates)

In [233…

```python
x,y = line1.coords.xy
print(f"X Coordinate : {list(x)}")
print(f"Y Coordinate : {list(y)}")
```

```
X Coordinate : [7.0, 6.0, 5.0]
Y Coordinate : [7.0, 6.0, 5.0]
```

2.10 - (Calculate the length of the line)

In [234…

```python
len1 = line1.length
len2 = line2.length
```

```
print("Length of line 1 : ",len1)
print("Length of line 2 : ",len2)
```

```
Length of line 1 :  2.8284271247461903
Length of line 2 :  7.23606797749979
```

2.11 - (Calculate the centroid of the line)

In [235…

```
x,y = line1.centroid.coords.xy
print(line1)
print(f"The centroid of line1 is ({(x[0],y[0])})")
```

```
LINESTRING (7 7, 6 6, 5 5)
The centroid of line1 is ((6.0, 6.0))
```

POLYGON

Program 3

3.1 - (Create a Polygon from the coordinates)

In [236…

```
from shapely.geometry import Polygon
coordinates = [(2,0),(1,0),(4,3),(5,7),(8,1),(9,3)]
poly = Polygon(coordinates)
poly_test = Polygon(coordinates)
print(poly)
```

```
POLYGON ((2 0, 1 0, 4 3, 5 7, 8 1, 9 3, 2 0))
```

3.2 - (Create a Polygon based on information from the Shapely points)

In [237…

```
poly2 = Polygon([point1,point2,point3])
print(poly2)
```

```
POLYGON ((7 8, 8 6, 5 2, 7 8))
```

3.3 - (Check if Polygons are identical)

In [238…

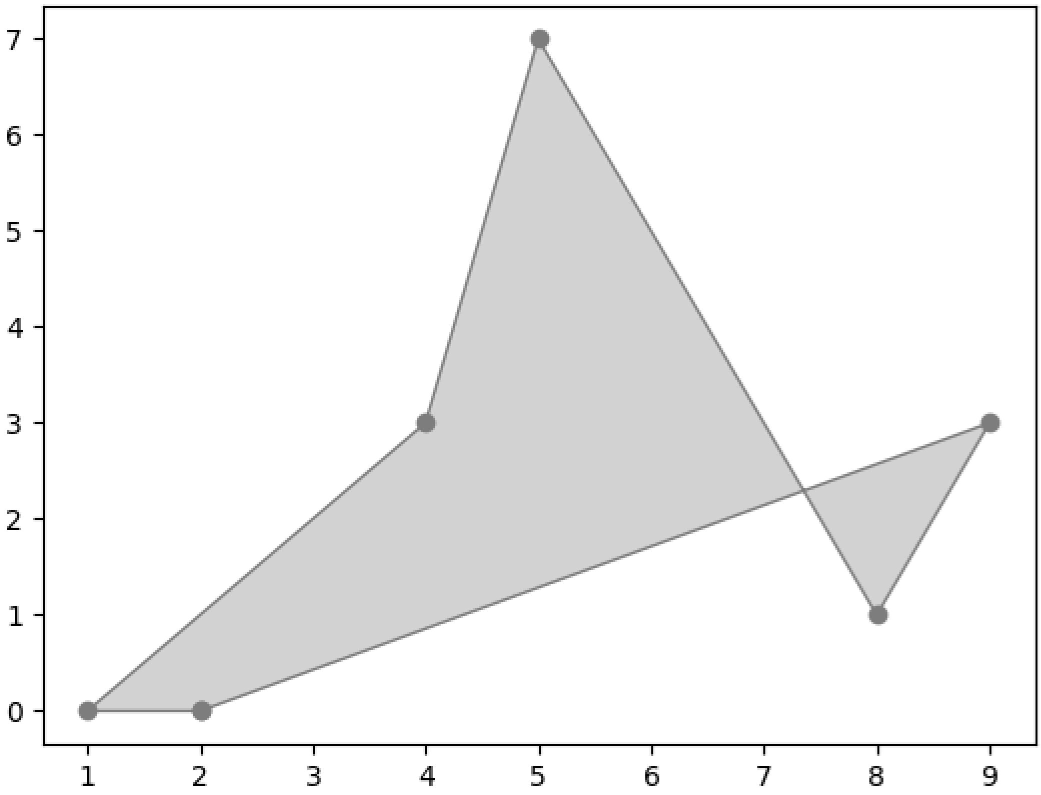
```
print(poly == poly2)
print(poly == poly_test)
```

```
False
True
```

3.4 - (Display the Polygon on screen)

In [239…

```
from shapely.plotting import plot_polygon as pp
figure, axes = plot.subplots()
pp(poly, ax=axes, color='grey',facecolor='lightgrey', edgecolor='grey')
plot.show()
```



3.5 - (Print the Polygon)

In [240…

```
print(poly)
```

```
POLYGON ((2 0, 1 0, 4 3, 5 7, 8 1, 9 3, 2 0))
```

3.6 - (Display the type of the polygon object)

In [241…

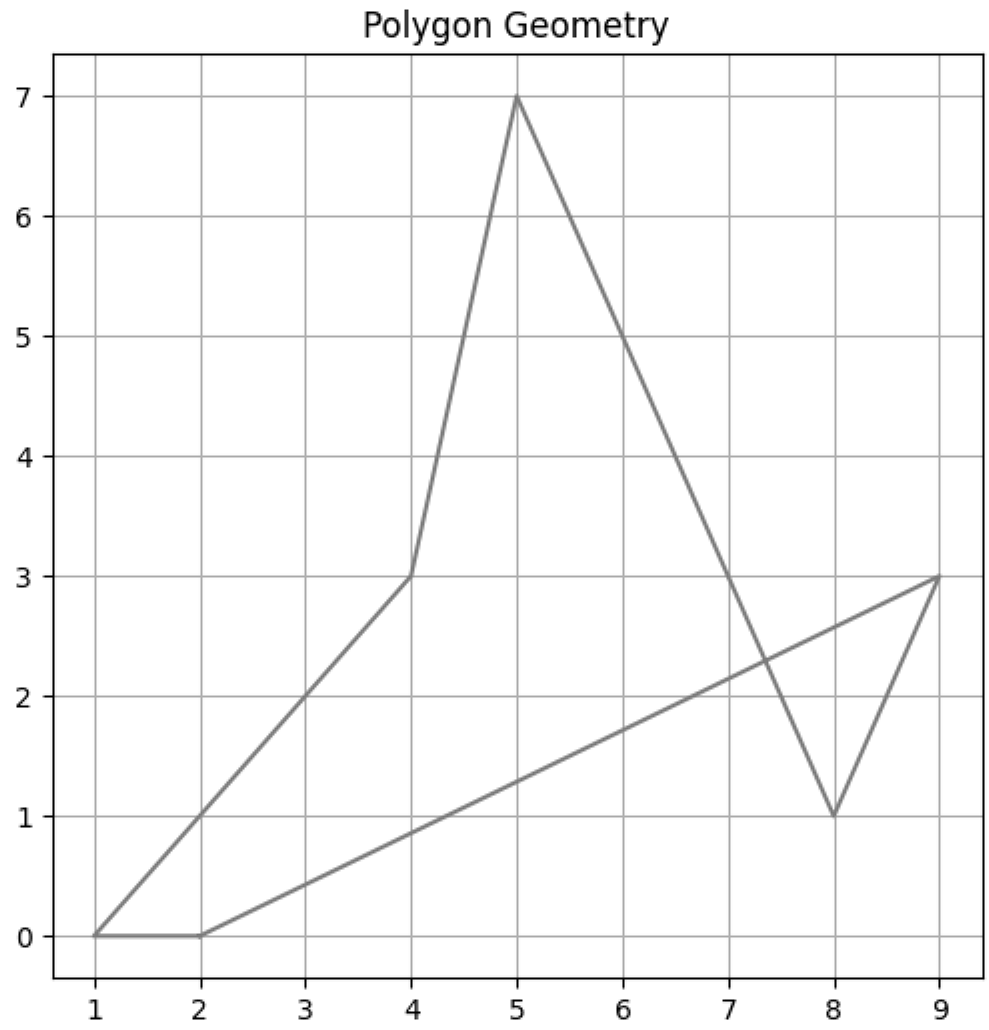
```
print(type(poly))
```

```
<class 'shapely.geometry.polygon.Polygon'>
```

3.7 - (Display the geometry of the polygon object)

In [242…

```
x, y = poly.exterior.xy
plot.figure(figsize=(6, 6))
plot.plot(x, y, color='grey', linewidth=1.5)
plot.title('Polygon Geometry')
plot.grid(True)
plot.show()
```



3.8 - (Create a hollow polygon)

In [243…

```
exterior_ring = Polygon([(2, 3), (3, 1), (4, 1), (1, 0), (0, 0)])
inner_ring = Polygon([(1, 1), (3, 3), (6, 6),(3,3)])
hollow_polygon = Polygon(shell=exterior_ring, holes=inner_ring)
hollow_polygon.area
```

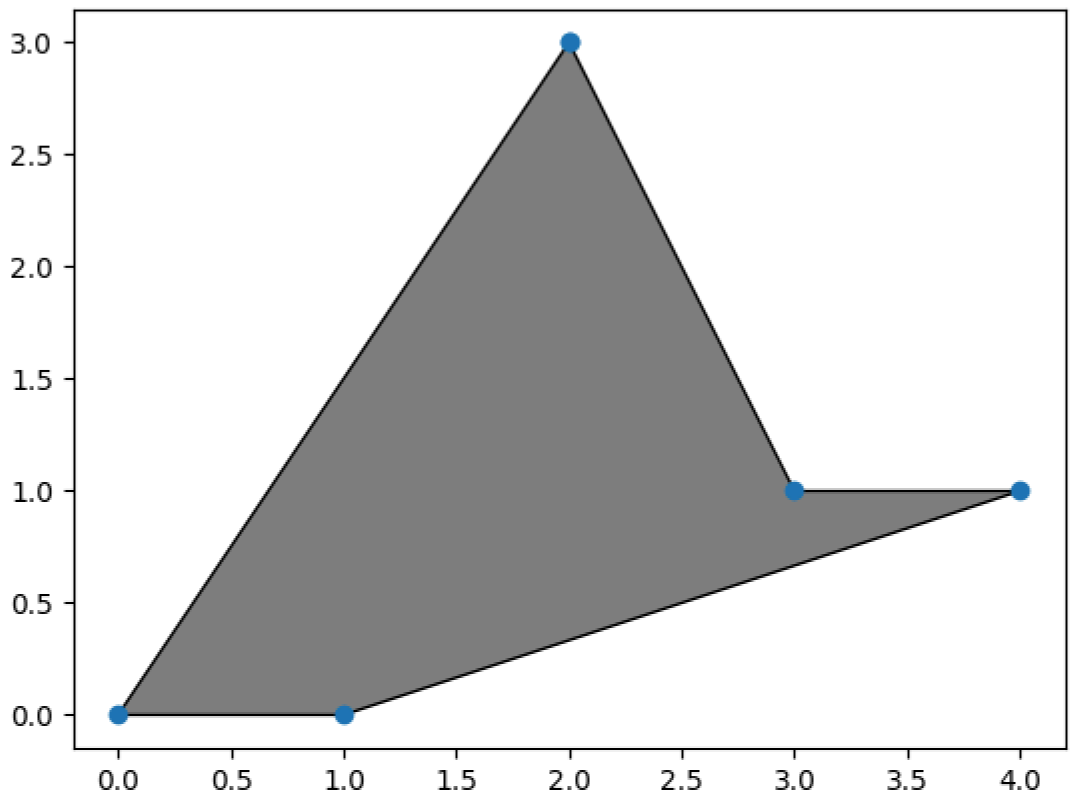
Out[243…

```
4.5
```

3.9 - (Display the Hollow Polygon)

In [244…

```
figure, axes = plot.subplots()
plot_polygon(hollow_polygon, ax=axes, facecolor='grey', edgecolor='black')
plot.show()
```



3.10 - (Display the parameters of the Polygon such as area, centroid, bounding box, exterior length)

In [245…

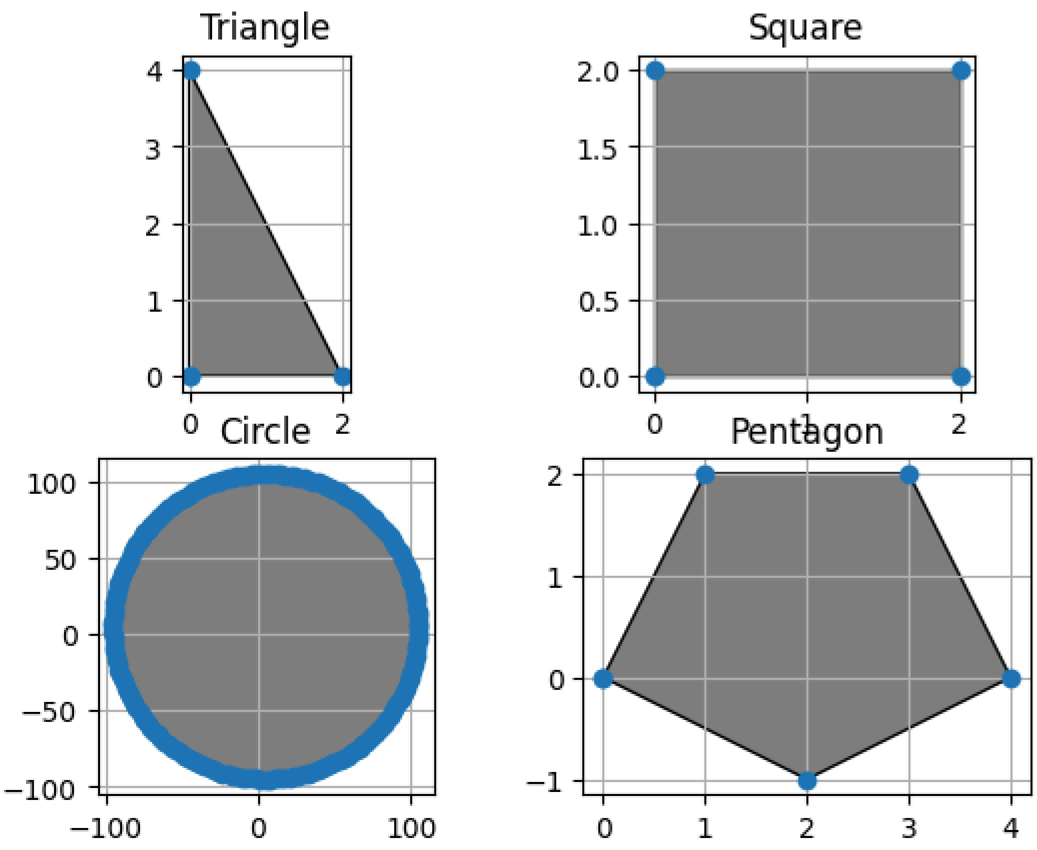
```
print(f"Area of the Polygon : {poly.area} ")
print(f"Centroid of the Polygon : {poly.centroid} ")
print(f"Bounding box of the Polygon : {poly.bounds} ")
print(f"Exterior length of the Polygon : {poly.exterior.length} ")
```

```
Area of the Polygon : 13.0
Centroid of the Polygon : POINT (4.384615384615385 2.9102564102564106)
Bounding box of the Polygon : (1.0, 0.0, 9.0, 7.0)
Exterior length of the Polygon : 25.925791328600013
```

3.11 - (Display Geometric Shapes Triangle, Square, Circle, Pentagon)

In [246…

```
square = Polygon([(0, 0), (2, 0), (2,2),(0,2)])
circle = Point(5,5).buffer(100,50)
triangle = Polygon([(0, 0), (2, 0), (0, 4),(0,0)])
pentagon = Polygon([(0, 0), (1, 2), (3, 2), (4, 0), (2, -1)])
plot.subplot(2,2,1)
plot.title("Triangle")
pp(triangle, facecolor='grey', edgecolor='black')
plot.subplot(2,2,2)
plot.title("Square")
pp(square, facecolor='grey', edgecolor='black')
plot.subplot(2,2,3)
plot.title("Circle")
pp(circle, facecolor='grey', edgecolor='black')
plot.subplot(2,2,4)
plot.title("Pentagon")
pp(pentagon, facecolor='grey', edgecolor='black')
plot.show()
```



3.12 - (Export any shape into shapefile.)

In [247…

```
import geopandas as gpd
gdf = gpd.GeoDataFrame(geometry=[square],crs="EPSG:4326")
output_shapefile = 'square.shp'
gdf.to_file(output_shapefile)
```

Handling Shapefile

Program 4

4.1 - (From the given shapefile, display the number of records)

In [248…

```
import geopandas as gpd
shape_file = gpd.read_file(r"A:\IIST GEO INFORMATICS\Programming for geoinformatics
records = len(shape_file)
print(f"Number of records: {records}")
```

```
Number of records: 36
```

4.2 - (Display the projection system)

In [249…

```
print("Projection System => ", shape_file.crs)
```

```
Projection System =>  EPSG:3857
```

4.3 - (Make a copy of the file in the working directory)

In [250…

```
copy = shape_file.copy()
copy.to_file('duplicate.shp')
```

4.4 - (Compute the area of the Polygons)

In [251…

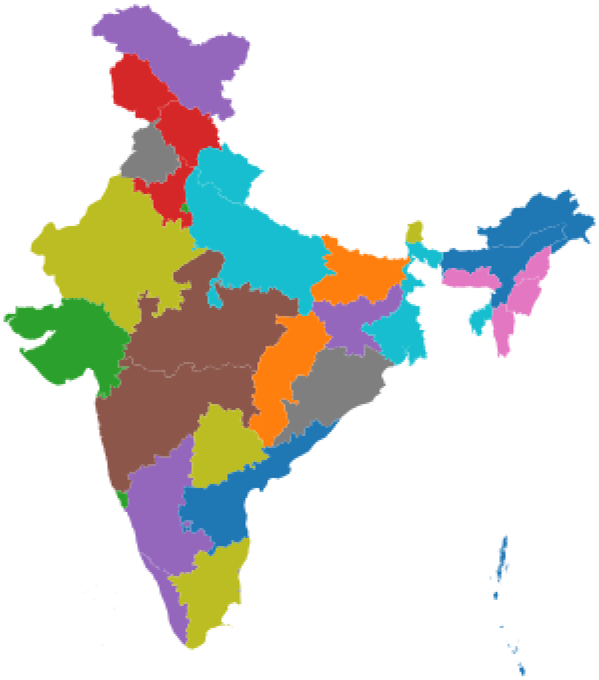
```
shape_file["area_km2"] = shape_file.area / 1000000
print(shape_file[["geometry", "area_km2"]])
```

```
                                             geometry      area_km2
0   MULTIPOLYGON (((10341718.474 1449533.161, 1034...    7658.811873
1   POLYGON ((8546255.616 3606050.813, 8546315.4 3...     155.608608
2   MULTIPOLYGON (((8122247.822 2312434.407, 81223...     663.666159
3   POLYGON ((8583390.57 3359116.19, 8583476.212 3...    1934.890744
4   POLYGON ((8524318.539 3516490.865, 8524451.392...   58208.866179
5   POLYGON ((9762288.285 2772949.712, 9762301.816...   95639.038325
6   MULTIPOLYGON (((8608594.474 2090389.205, 86086...  206403.564265
7   POLYGON ((8347733.191 1436381.747, 8347795.744...   40400.382330
8   MULTIPOLYGON (((8135256.29 930182.487, 8135260...      34.567532
9   POLYGON ((8724343.278 3106498.184, 8724579.382...  368604.132462
10  MULTIPOLYGON (((8280974.863 2515416.345, 82809...  347897.266280
11  MULTIPOLYGON (((9578537.936 2579790.782, 95786...  178402.811263
12  MULTIPOLYGON (((8939353.702 1513831.235, 89395...  135822.723752
13  POLYGON ((9275926.808 2765881.317, 9276185.437...  156543.125519
14  POLYGON ((8720284.876 2259244.214, 8720421.528...  124408.075203
15  POLYGON ((9426056.496 2174632.352, 9426228.484...  177174.558285
16  POLYGON ((8223217.424 1779394.764, 8223279.301...    4002.178917
17  POLYGON ((8548682.698 3929291.879, 8548760.706...   77509.763452
18  POLYGON ((8442331.679 3830799.529, 8442574.742...   68540.702257
19  POLYGON ((8234599.326 3529026.887, 8234599.327...  429932.423386
20  POLYGON ((7914780.837 2837315.493, 7915101.603...  221213.472052
21  POLYGON ((8801802.136 3692833.282, 8802083.049...   71697.034785
22  POLYGON ((8637489.997 3555885.598, 8637654.287...  304578.206289
23  POLYGON ((9864726.992 3265074.341, 9865469.61 ...    9087.828171
24  POLYGON ((10380499.251 2872443.723, 10380499.2...   98134.590488
25  POLYGON ((10696175.277 3434232.65, 10696981.87...  105686.083492
26  POLYGON ((10596805.532 3126858.281, 10597031.2...   20644.435936
27  POLYGON ((10527945.945 2960789.34, 10528432.78...   27179.632682
28  POLYGON ((10326423.582 2817021.246, 10326465.4...   25113.952104
29  POLYGON ((10260260.337 2818339.599, 10260273.8...   12527.445162
30  POLYGON ((10222042.434 3013858.327, 10222165.9...   27649.319403
31  POLYGON ((9800305.279 3151090.311, 9800377.779...  100467.932429
32  POLYGON ((9362949.333 3188807.607, 9362966.106...  116468.787731
33  POLYGON ((8550375.654 3927668.327, 8548619.625...  249153.605449
34  POLYGON ((8550375.654 3927668.327, 8550332.102...   83497.800923
35  MULTIPOLYGON (((8878474.16 1232399.36, 8878488...     417.722535
```

4.5 - (Plot the data)

In [252…

```
ax=shape_file.plot(column="State_Name")
ax.set_axis_off()
plot.show()
```



4.6 - (From the given shapefile, find out the entry with largest and smallest area)

In [253…

```
largest = shape_file.loc[shape_file["area_km2"].idxmax()]
smallest = shape_file.loc[shape_file["area_km2"].idxmin()]
print(f"Largest area :{largest}")
print(f"Smallest area : {smallest}")
```

```
Largest area :State_Name                                            Rajasthan
geometry      POLYGON ((8234599.326299999 3529026.8869000003...
area_km2                                          429932.423386
Name: 19, dtype: object
Smallest area : State_Name                                          Lakshadweep
geometry      MULTIPOLYGON (((8135256.290100001 930182.48690...
area_km2                                              34.567532
Name: 8, dtype: object
```

4.7 - (Extract the boundary of your homestate and project it into the appropriate coordinate system)

In [254…

```
kerala=shape_file.loc[shape_file["State_Name"]=="Kerala"]
kerala_boundary=kerala.boundary
kerala_boundary=kerala_boundary.to_crs("EPSG:32643")
ax=kerala_boundary.plot()
ax.set_axis_off()
plot.show()
```

4.8 - (Attempt to change the projection and save it as new shapefile)

In [255…

```
projection_change=kerala_boundary.set_crs("EPSG:7781",allow_override=True)
projection_change.to_file("kerala_projection_change.shp")
projection_change.crs
```

Out[255…

```
<Projected CRS: EPSG:7781>
Name: WGS 84 / Kerala
Axis Info [cartesian]:
- X[east]: Easting (metre)
- Y[north]: Northing (metre)
Area of Use:
- name: India - Kerala; Mayyazhi (Mahe) area of Pudacherry territory.
- bounds: (74.81, 8.25, 77.4, 12.8)
Coordinate Operation:
- name: Kerala NSF TM
- method: Transverse Mercator
Datum: World Geodetic System 1984 ensemble
- Ellipsoid: WGS 84
- Prime Meridian: Greenwich
```

In [ ]: