Geomatics Engineering Department

Second Year Geomatics

Computer Applications I

Lecture No: 2

Introduction to Geospatial Libraries

Dr. Eng. Reda FEKRY

Assistant Professor of Geomatics
reda.abdelkawy@feng.bu.edu.eg

Overview Of Today's Lecture

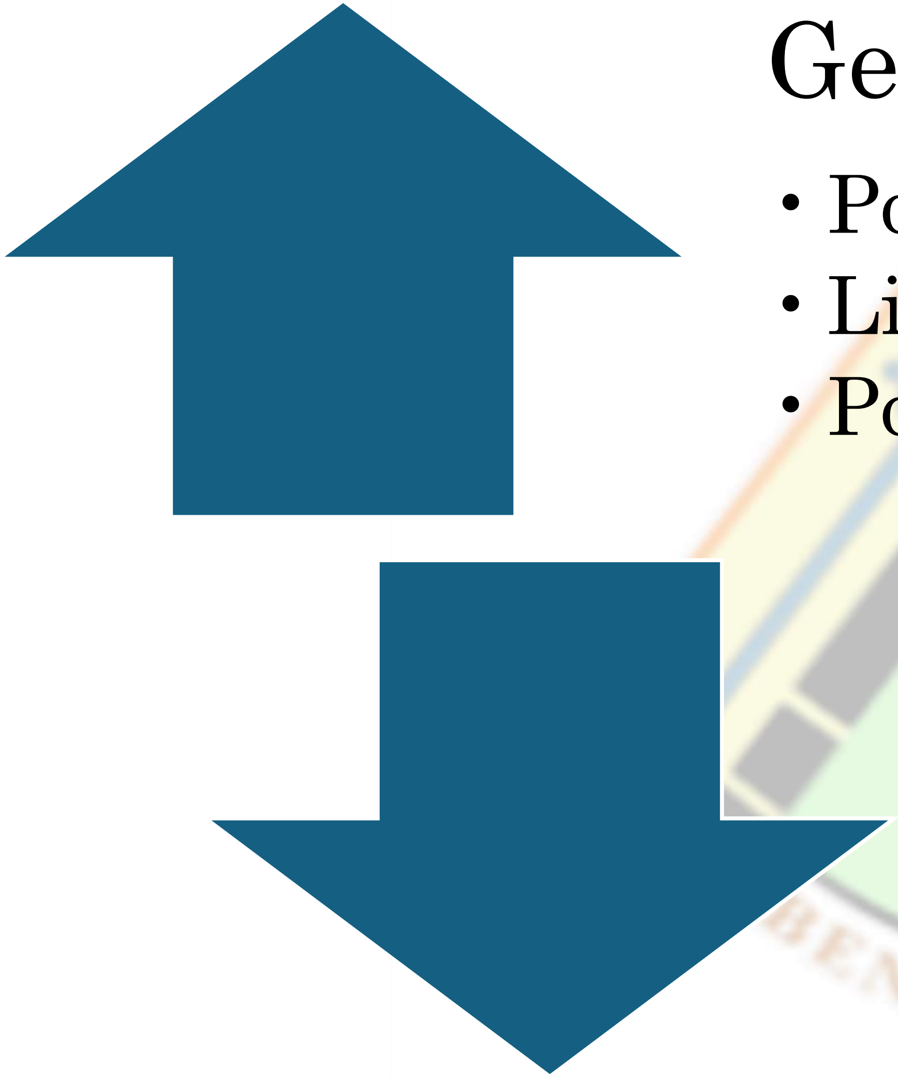
Overview of popular Python libraries for geospatial data analysis

Performing spatial operations

Geospatial data visualization and mapping

GEOSPATIAL DATA TYPES

Spatial Data Types



Geometric Data

- Points
- Lines
- Polygons

Non-Geometric Data

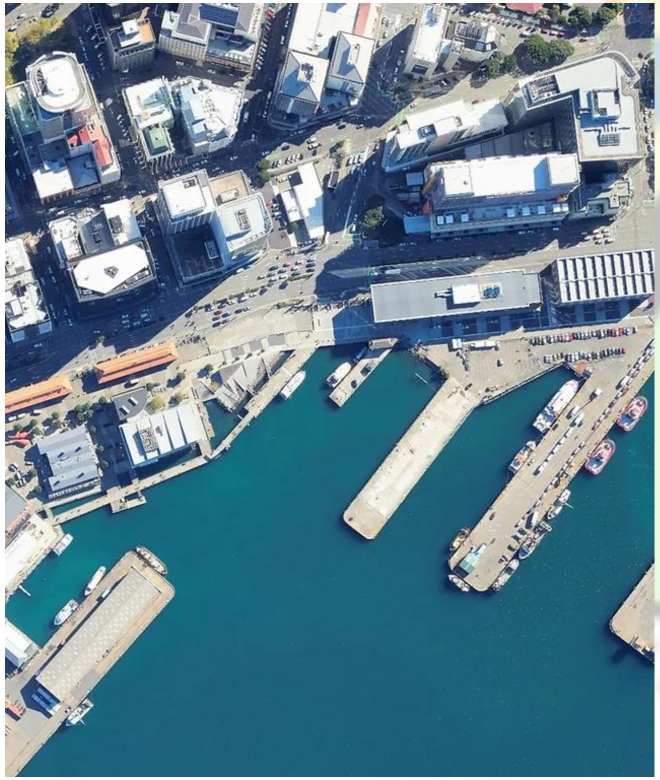
- Attribute Data
- Temporal Data

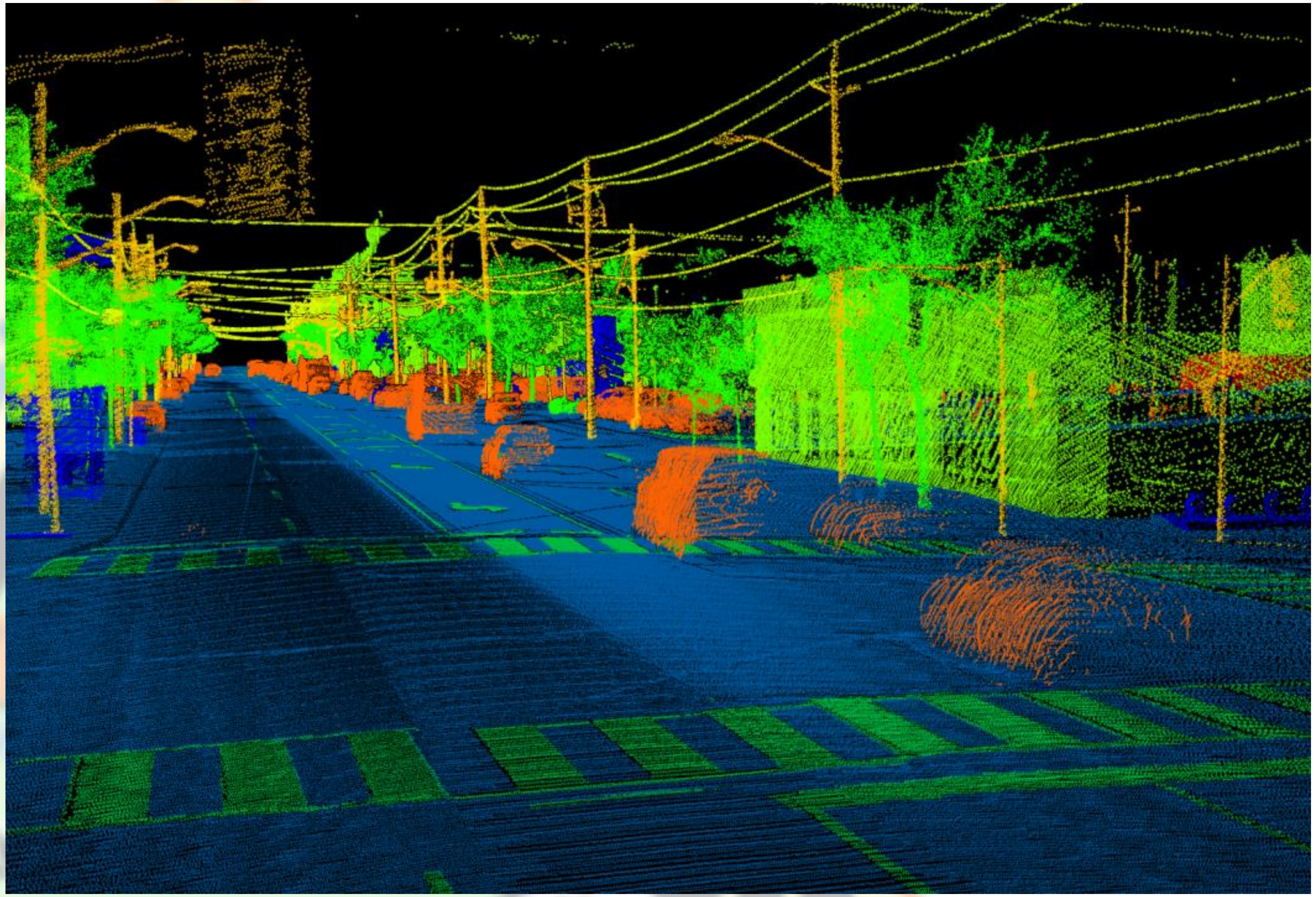
Computer Applications I - Dr. Eng. Reda Fekry

Spatial Data Types

- Additional Data Types

1. Satellite imagery
2. LiDAR point cloud

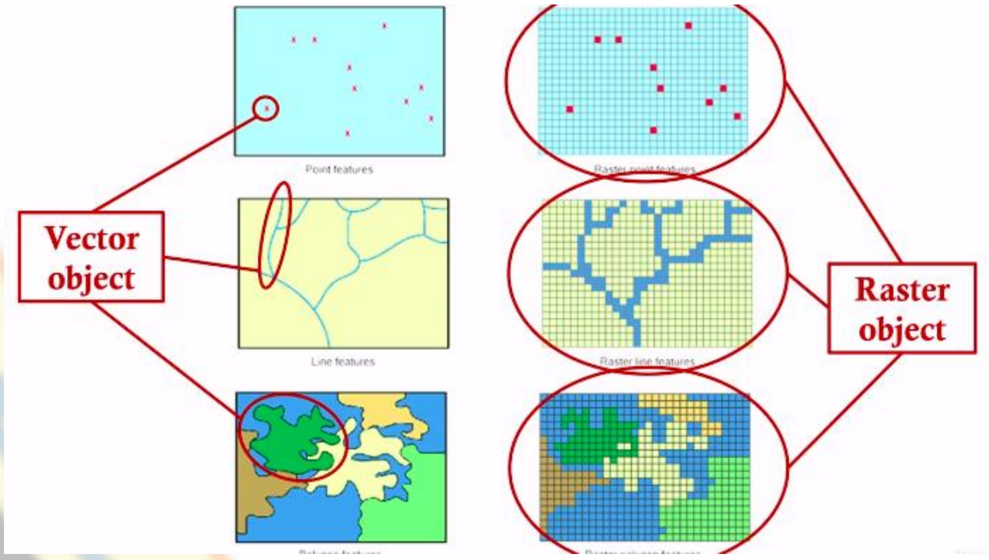


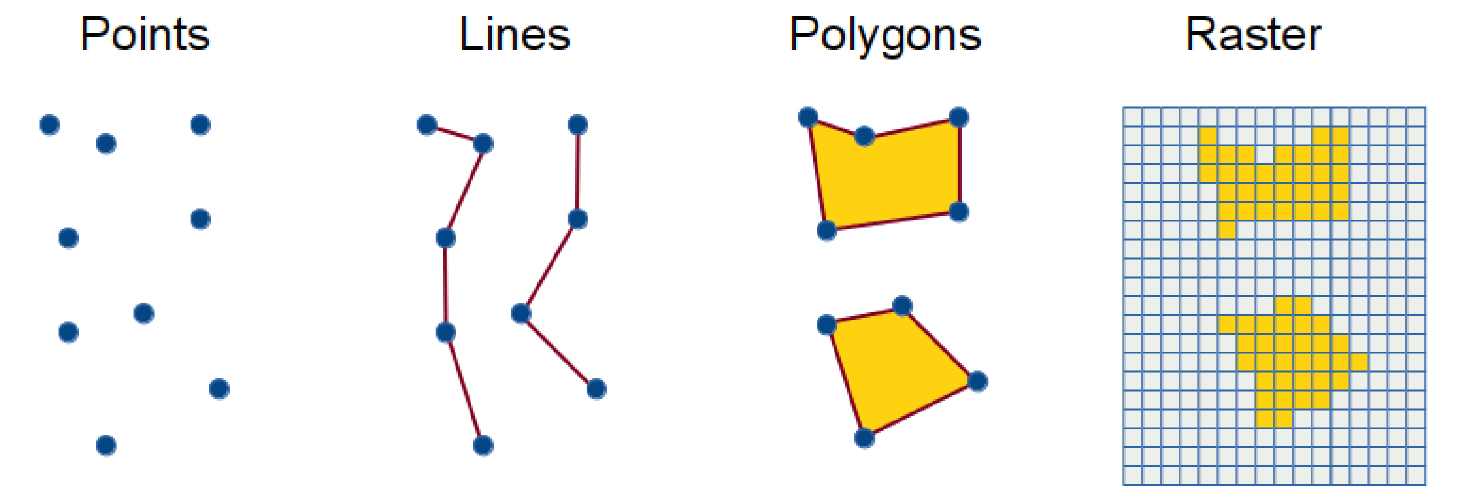


Spatial Data Types

- Data Formats

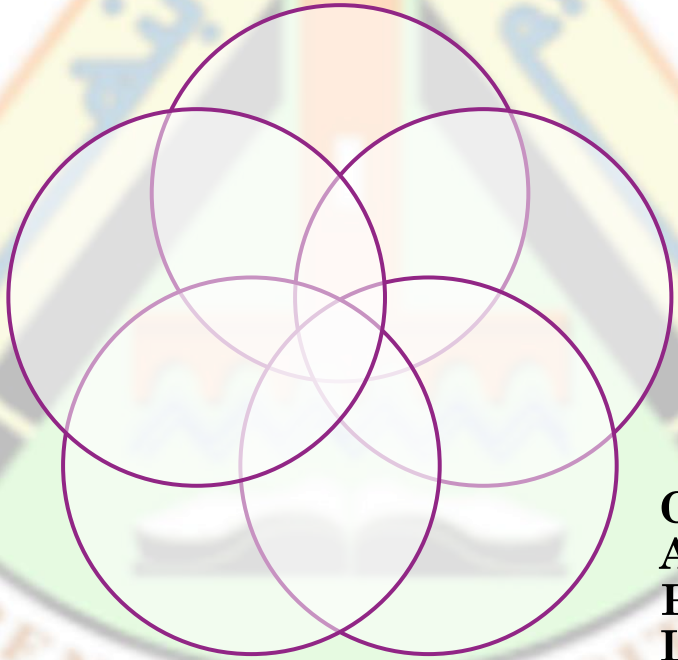
1. Vector data (points, lines, etc.,)
2. Raster data (a grid of cells/pixels)





GEOSPATIAL LIBRARIES IN PYTHON

Geospatial Libraries in Python



Vector Data Libraries

- GeoPandas
- Shapely

Visualization Libraries

- Matplotlib
- Seaborn
- Cartopy
- plotly

Raster Data Libraries

- Rasterio
- RasterStats

Coordinate Reference Systems Libraries

- PyProj

Geospatial Data Access and Processing Libraries

- GDAL (Geospatial Data Abstraction Library)
- Fiona

Computer Applications I - Dr. Eng. Reda Fekry

Choosing Proper Library

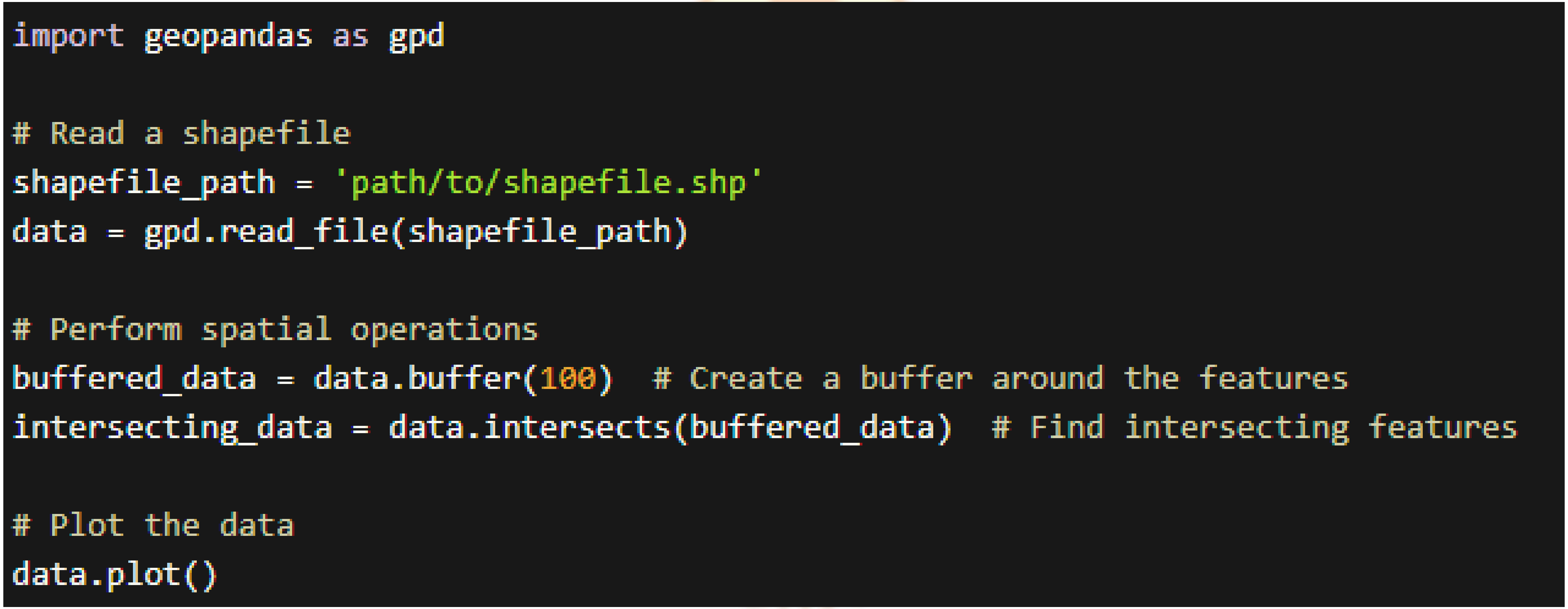
- **The selection of the most suitable library depends on application and data types:**

- **For basic geospatial data manipulation:** Use GeoPandas and Shapely for vector data and Rasterio for raster data.
- **For data access and processing:** Utilize GDAL and Fiona.
- **For coordinate transformations:** Employ PyProj.
- **For data visualization:** Consider Matplotlib, Seaborn, or Cartopy depending on the desired level of complexity and customization.

Geospatial Libraries – Code Examples

Geospatial Libraries – Code Examples

- Using GeoPandas to read a shapefile, perform a buffer operation, find intersecting features, and plot the data.

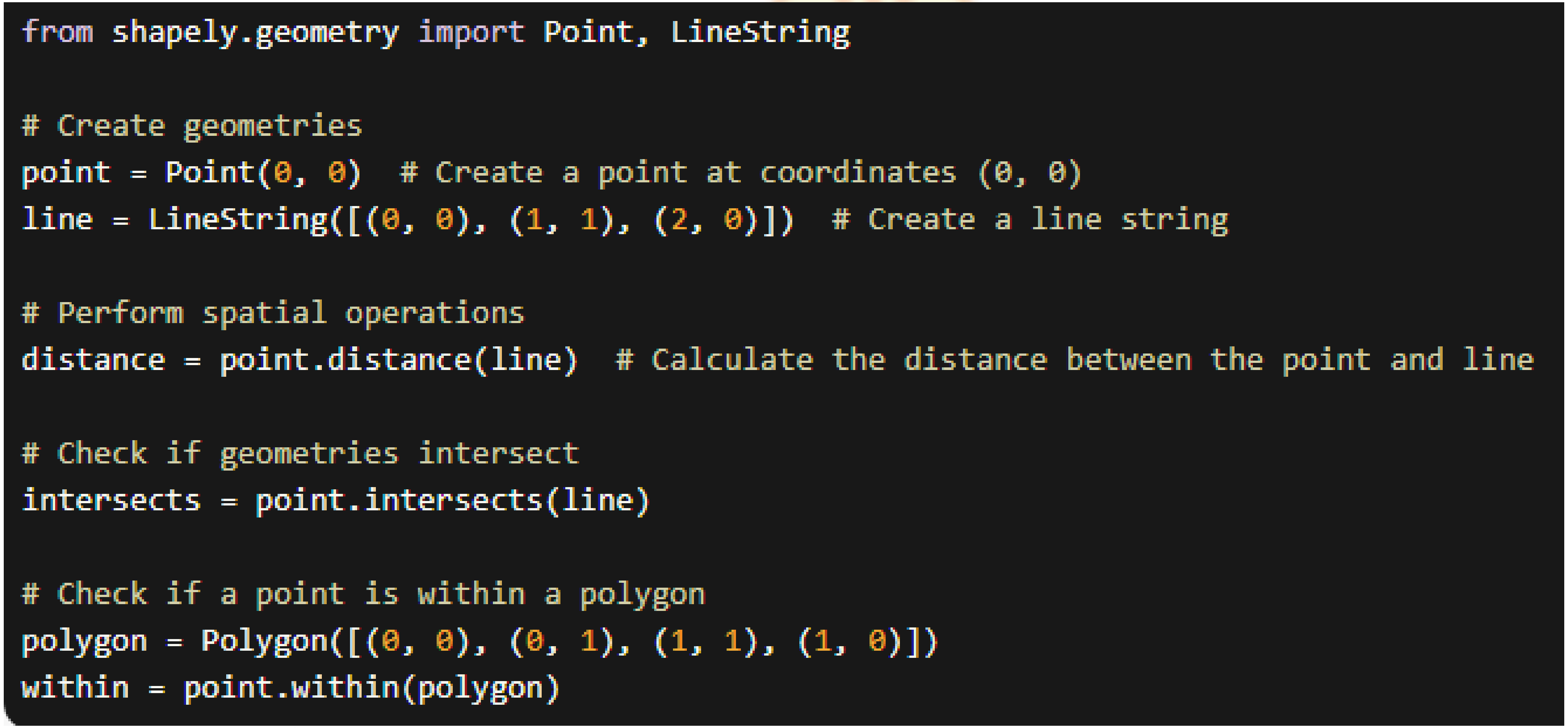


```
import geopandas as gpd

# Read a shapefile
shapefile_path = 'path/to/shapefile.shp'
data = gpd.read_file(shapefile_path)

# Perform spatial operations
buffered_data = data.buffer(100)  # Create a buffer around the features
intersecting_data = data.intersects(buffered_data)  # Find intersecting features

# Plot the data
data.plot()
```

Computer Applications I - Dr. Eng. Reda Fekry

Geospatial Libraries in Python

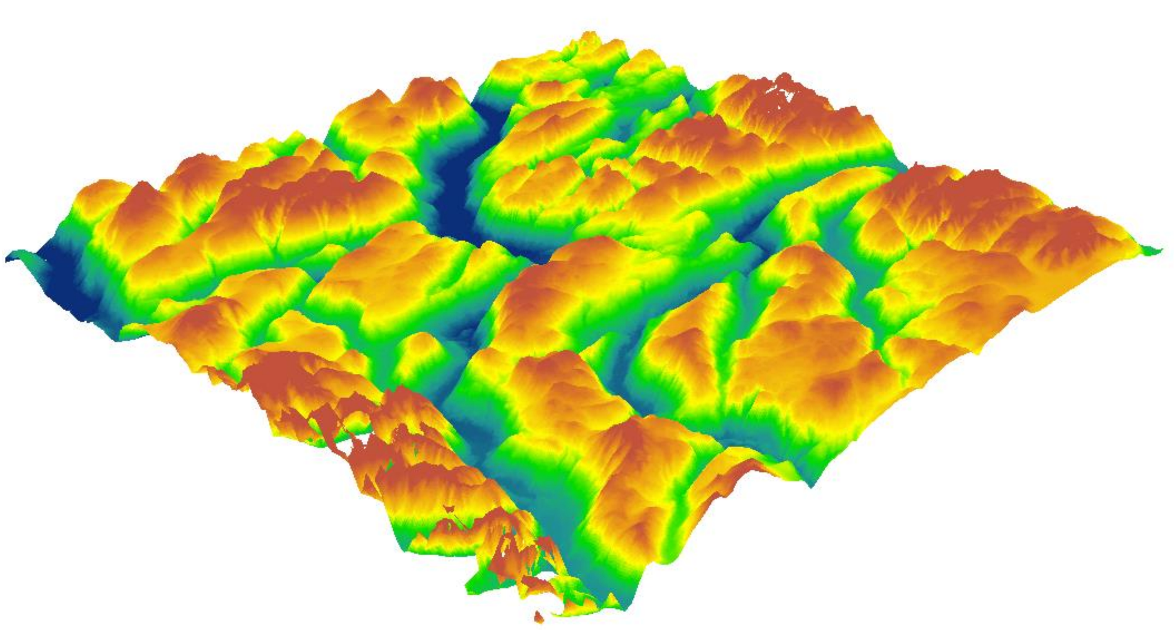
- Using Shapely to create and manipulate geometries.

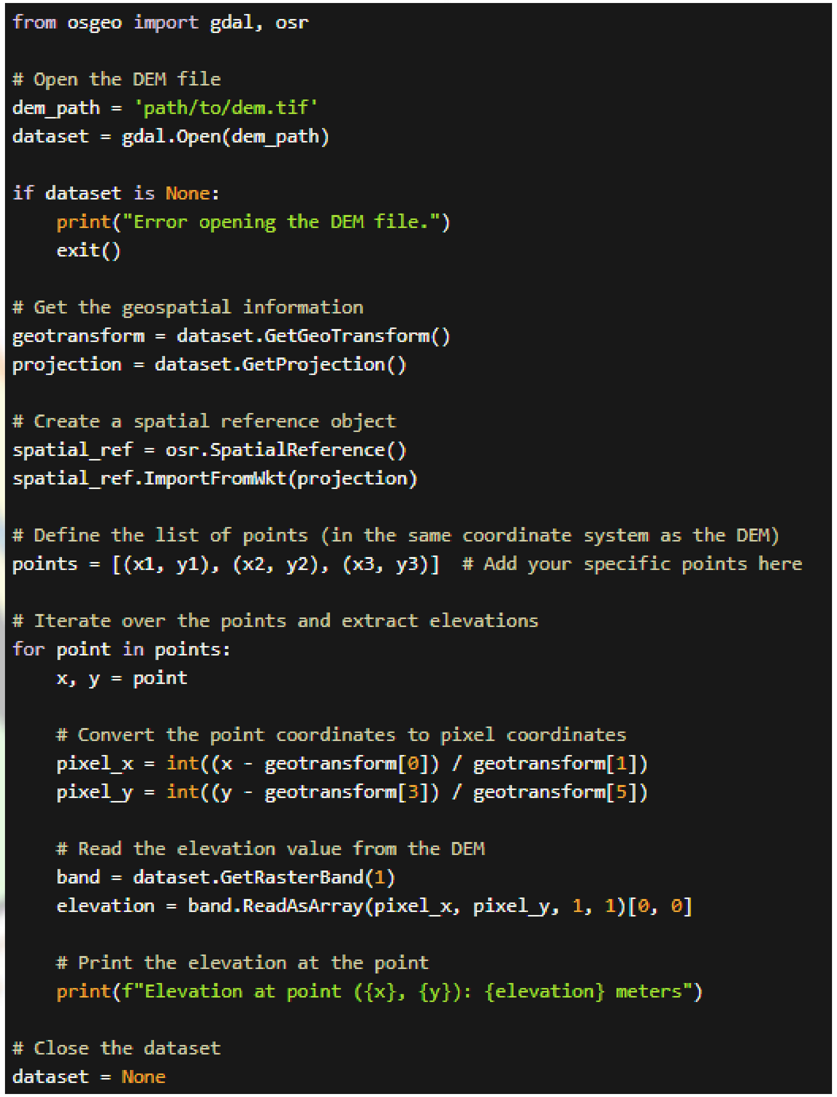


```
from shapely.geometry import Point, LineString

# Create geometries
point = Point(0, 0)  # Create a point at coordinates (0, 0)
line = LineString([(0, 0), (1, 1), (2, 0)])  # Create a line string

# Perform spatial operations
distance = point.distance(line)  # Calculate the distance between the point and line

# Check if geometries intersect
intersects = point.intersects(line)

# Check if a point is within a polygon
polygon = Polygon([(0, 0), (0, 1), (1, 1), (1, 0)])
within = point.within(polygon)
```

Geospatial Libraries in Python

- Using GDAL to read a Digital Elevation Model (DEM) file and extract elevations at specific points





```python
from osgeo import gdal, osr

# Open the DEM file
dem_path = 'path/to/dem.tif'
dataset = gdal.Open(dem_path)

if dataset is None:
    print("Error opening the DEM file.")
    exit()

# Get the geospatial information
geotransform = dataset.GetGeoTransform()
projection = dataset.GetProjection()

# Create a spatial reference object
spatial_ref = osr.SpatialReference()
spatial_ref.ImportFromWkt(projection)

# Define the list of points (in the same coordinate system as the DEM)
points = [(x1, y1), (x2, y2), (x3, y3)]  # Add your specific points here

# Iterate over the points and extract elevations
for point in points:
    x, y = point

    # Convert the point coordinates to pixel coordinates
    pixel_x = int((x - geotransform[0]) / geotransform[1])
    pixel_y = int((y - geotransform[3]) / geotransform[5])

    # Read the elevation value from the DEM
    band = dataset.GetRasterBand(1)
    elevation = band.ReadAsArray(pixel_x, pixel_y, 1, 1)[0, 0]

    # Print the elevation at the point
    print(f"Elevation at point ({x}, {y}): {elevation} meters")

# Close the dataset
dataset = None
```

END OF PRESENTATION

THANK YOU FOR ATTENTION!