Application of GIS with Python

Submitted By: Prakash Gharti Magar
Class Roll No.: PAS075BGE029

Submitted To: Er. Saurav Gautam
Department of Geomatics Engineering

Tribhuvan University

Institute of Engineering

Pashchimanchal Campus

Department of Geomatics Engineering

Lamachaur, Pokhara

1. **Shapefile: what are the different files that constitute a shapefile? Discuss about their importance.**

⇨A shapefile is an ESRI vector data storage format used to store location, shape, spatial index, projection, and attributes of geographic features. It is developed and regulated by ESRI as a mostly open specification for data interoperability among ESRI and other GIS software products. It can describe all features in vector format using point, line, and polygon. The '.shp' file extension alone is incomplete for the distribution of spatial data so shapefile must have at least three file formats like '.shp' for the geometry of feature, '.shx' for an index of geometry, and '.dbf' for attribute data storage of features. It may contain other files with different extension like, '.prj' for projection, '.sbn' and '.sbx' for spatial index etc.

2. **Use overpass turbo.eu and download GeoJOSN, GPX and KML file.**

⇨Source Code,

- ⇨OSM(OpenStreetMap): It is possible to download map data from the OpenStreetMap dataset in a number of ways. The full dataset is available from the OpenStreetMap website download area. It is also possible to select smaller areas to download. Data normally comes in the form of XML formatted .osm files. But, if you want to download entire data of some extent then you need to download .osm file.
- ⇨GPX(GPS Exchange Format): GPX is an XML file format for storing coordinate data. It can store waypoints, tracks, and routes in a way that is easy to process and convert to other forms. GPX is simply a text file with geographic information such as waypoints, tracks, and routes saved in it.
- ⇨KML(Keyhole Markup Language): KML is a file format used to display geographic data in an Earth browser such as Google Earth. You can create KML files to pinpoint locations, add image overlays, and expose rich data in new ways. KML is an international standard maintained by the Open Geospatial Consortium, Inc. (OGC). The KML format was originally developed by

Keyhole, Inc. for Keyhole Earth Viewer, a mapping program that was acquired by Google in 2004. The format eventually became a worldwide standard for geographic annotation and visualization in 2D and 3D geographic mapping programs. KML files are primarily associated with the Google Earth web application and Google Earth Pro desktop program. They are also supported by other mapping applications, such as Blue Marble Geographics Global Mapper and ESRI ArcGIS Pro.

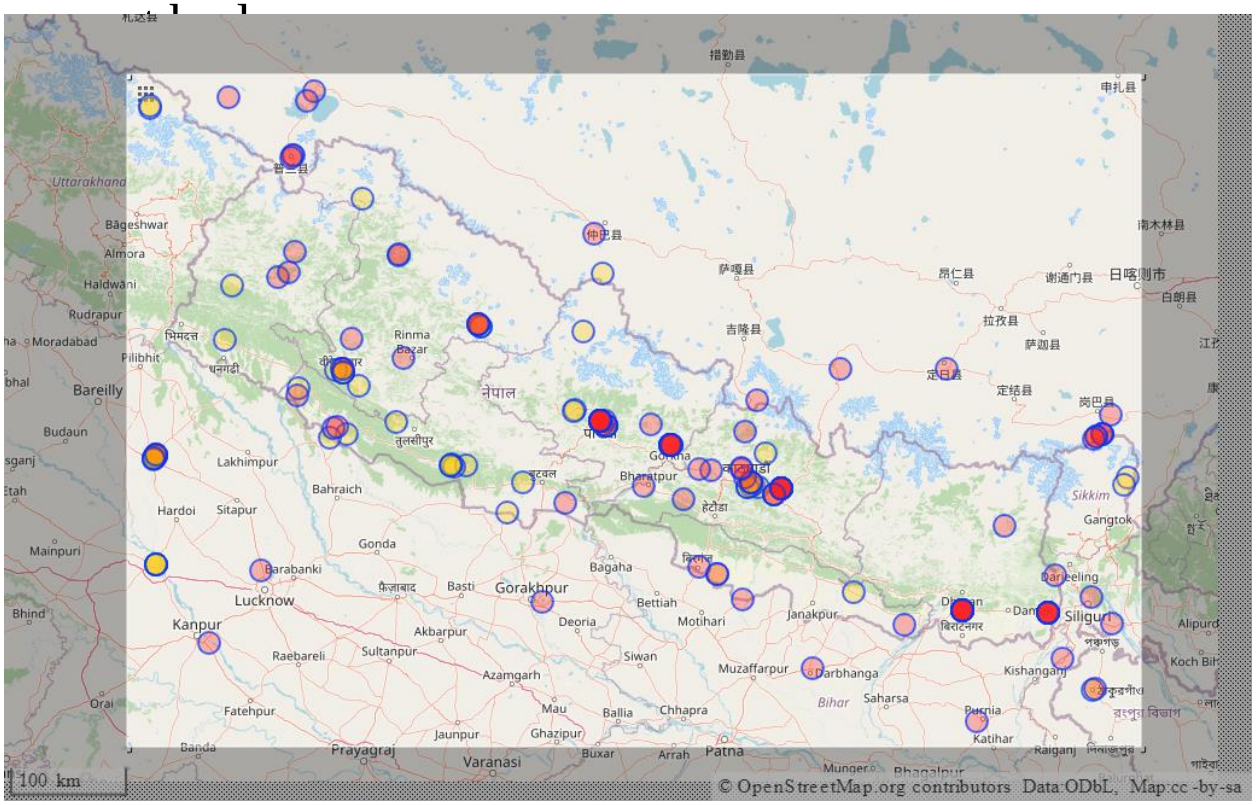
⇨ GeoJSON(Geographic JavaScript Object Notation): GeoJSON is an open standard geospatial data interchange format that represents simple geographic features and their nonspatial attributes. Based on JavaScript Object Notation (JSON), GeoJSON is a format for encoding a variety of geographic data structures. GeoJSON supports the following geometry types: Point, LineString, Polygon, MultiPoint, MultiLineString, and MultiPolygon. Geometric objects with additional properties are Feature objects. Sets of features are contained by FeatureCollection objects.

GeoJSON Format Data:


```
{
  "type": "Feature",
  "geometry": {
    "type": "Point",
    "coordinates": [125.6, 10.1]
  },
  "properties": {
    "name": "Dinagat Islands"
  }
}
```

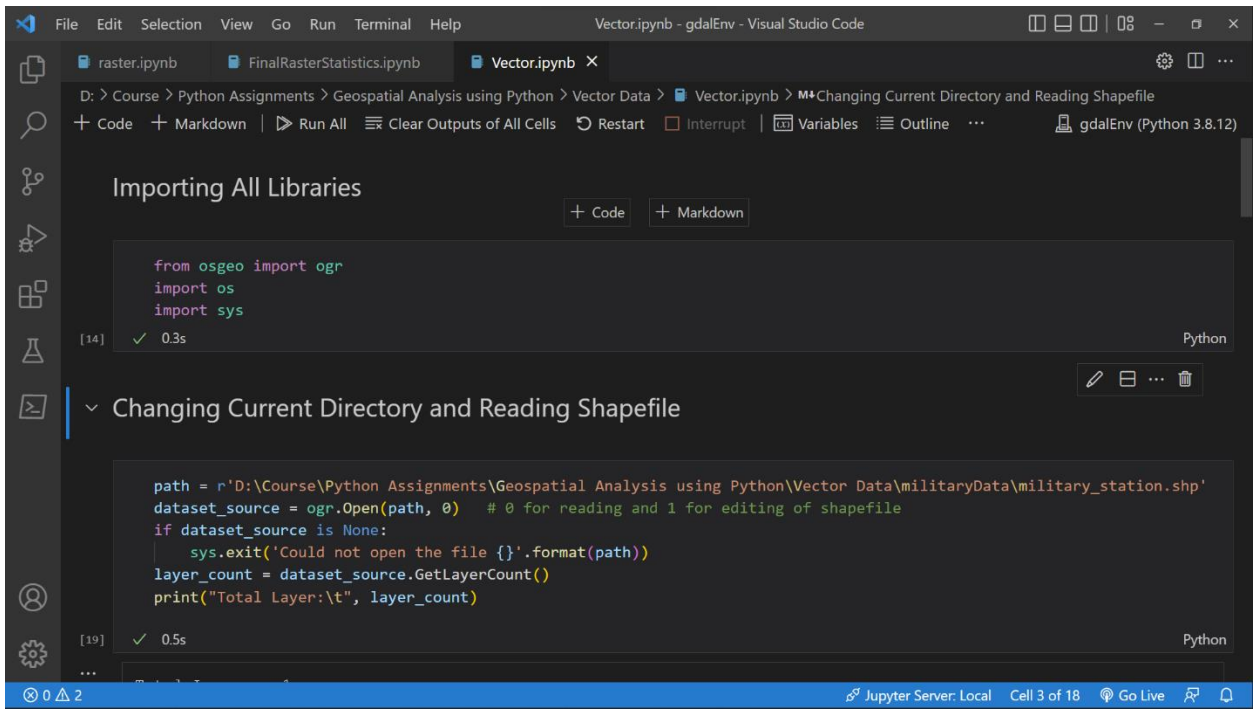

```
/*
This has been generated by the overpass-turbo wizard.
The original search was:
“military=*”
*/
[out:json][timeout:25];
// gather results
(
```

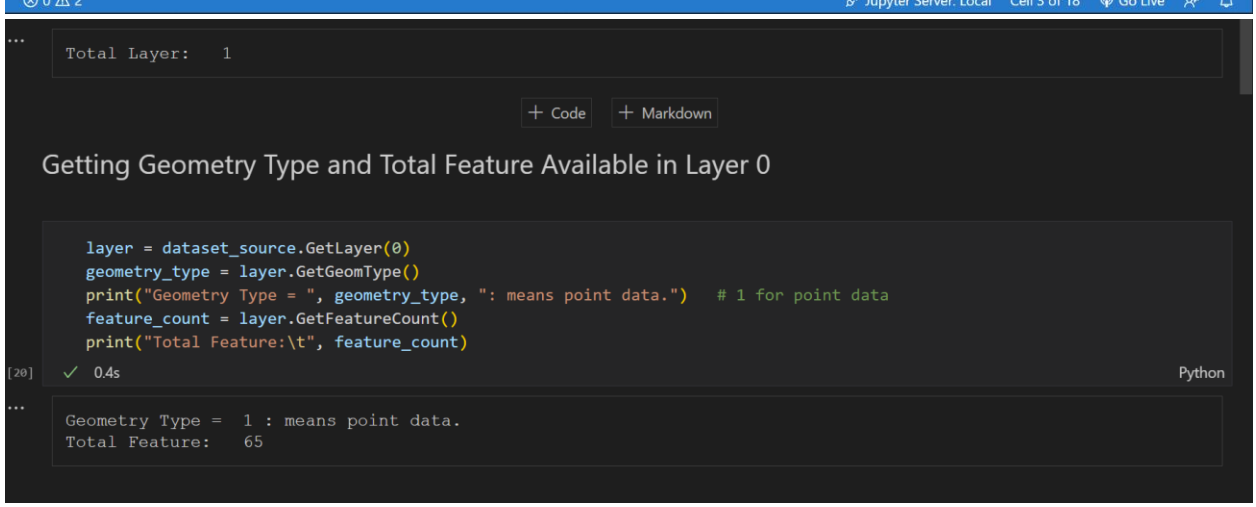
```
// query part for: “military=*”
node["military"](25.55235365216549,79.6728515625,30.939924331023445,88.9013671875);
way["military"](25.55235365216549,79.6728515625,30.939924331023445,88.9013671875);
relation["military"](25.55235365216549,79.6728515625,30.939924331023445,88.9013671875);
);
// print results
```



3. Run the code given in Geoprocessing with python for reading and writing shp. Read the various vector file formats using a text editor and compare between them. (Combined Q3, Q4 and Bonus Q5)

1. For Reading Vector Data (ESRI Shapefile)





Reading and Tabulating Some Records

```python
i=0
print("Military", "\t", "Name", "\t", 'X-Coordinate', "\t", 'Y-Coordinate\n')
for feature in layer:
    point = feature.geometry()
    name = feature.GetField('name')
    military = feature.GetField('military')
    x_coordinate = point.GetX()
    y_coordinate = point.GetY()
    print(military, "\t",  name, "\t", x_coordinate, "\t", y_coordinate)
    if i >= 10:
        break
    i = i + 1
```

[57] ✓ 0.3s Python Python

```
Military   Name    X-Coordinate    Y-Coordinate

checkpoint   None    85.4817685    27.9409208
checkpoint   None    79.9214553    27.8973949
barracks   None    80.6243525    29.2740792
```

Getting Spatial Reference of Layer

```python
spatial_reference = layer.GetSpatialRef()
print("Spatial Reference:\n", spatial_reference)
```

[43] ✓ 0.3s Python

```
Spatial Reference:
 GEOGCS["WGS 84",
    DATUM["WGS_1984",
        SPHEROID["WGS 84",6378137,298.257223563,
            AUTHORITY["EPSG","7030"]],
        AUTHORITY["EPSG","6326"]],
    PRIMEM["Greenwich",0,
        AUTHORITY["EPSG","8901"]],
    UNIT["degree",0.0174532925199433,
        AUTHORITY["EPSG","9122"]],
    AXIS["Latitude",NORTH],
    AXIS["Longitude",EAST],
    AUTHORITY["EPSG","4326"]]
```

2. For Writing Vector Data (ESRI Shapefile)

Importing ALl Required Libraries For Writing Vector Data

```python
from osgeo import ogr
import sys
```

[2] ✓ 0.2s Python

Opening Shapefile in Writing/Editing Mode

```python
file_path = r'D:\Course\Python Assignments\Geospatial Analysis using Python\Vector Data\militaryData\military_station.sh
dataset = ogr.Open(file_path, 1)
if dataset is None:
    sys.exit('Could not open the file {}'.format(file_path))
layer1 = dataset.GetLayer(0)
```

[46] ✓ 0.3s Python

Creating and Defining New Layer Named "prakash_new_layer"

```python
if dataset.GetLayer('prakash_new_layer'):
    dataset.DeleteLayer('prakash_new_layer')
output_layer = dataset.CreateLayer('prakash_new_layer', layer1.GetSpatialRef(), ogr.wkbPoint)
output_layer.CreateFields(layer1.schema)
define_layer = output_layer.GetLayerDefn()
blank_feature = ogr.Feature(define_layer)
```

✓ 0.1s Python

Writing to the layer

```python
for feature in layer1:
    if feature.GetField('name') == 'army camp':
        geometry = feature.geometry()
        blank_feature.SetGeometry(geometry)
        for i in range(feature.GetFieldCount()):
            value = feature.GetField(i)
            blank_feature.SetField(i, value)
        output_layer.CreateFeature(blank_feature)

del dataset
```

[48] ✓ 0.4s Python

3. For Reading Vector Data (GeoJSON)

Vector.ipynb - gdalEnv - Visual Studio Code

Raster (1).ipynb | FinalRasterStatistics.ipynb | Vector.ipynb

D: > Course > Python Assignments > Geospatial Analysis using Python > Vector Data > Vector.ipynb > Reading Other File Formats > For Reading .GeoJSON File

+ Code + Markdown | Run All | Clear Outputs of All Cells | Restart | Interrupt | Variables | Outline | gdalEnv (Python 3.8.12)

Reading Other File Formats

For Reading .GeoJSON File

+ Code + Markdown

```python
from osgeo import ogr
import os
import sys
```

[59] ✓ 0.3s Python

```python
path = r'D:\Course\Python Assignments\Geospatial Analysis using Python\Vector Data\Other Data Formats\military.geojson'
dataset_source = ogr.Open(path, 0)   # 0 for reading and 1 for editing of shapefile
if dataset_source is None:
    sys.exit('Could not open the file {}'.format(path))
layer_count = dataset_source.GetLayerCount()
print("Total Layer:\t", layer_count)
```

[60] ✓ 0.4s Python

```
Total Layer:    1
```

```python
layer = dataset_source.GetLayer(0)
geometry_type = layer.GetGeomType()
print("Geometry Type = ", geometry_type, ": means point data.")   # 1 for point data
feature_count = layer.GetFeatureCount()
print("Total Feature:\t", feature_count)
```

[61] ✓ 0.3s Python

```
Geometry Type =  0 : means point data.
Total Feature:   177
```

```python
spatial_reference = layer.GetSpatialRef()
print("Spatial Reference:\n", spatial_reference)
```

[62] ✓ 0.4s Python

```
Spatial Reference:
 GEOGCS["WGS 84",
    DATUM["WGS_1984",
        SPHEROID["WGS 84",6378137,298.257223563,
            AUTHORITY["EPSG","7030"]],
        AUTHORITY["EPSG","6326"]],
    PRIMEM["Greenwich",0,
        AUTHORITY["EPSG","8901"]],
    UNIT["degree",0.0174532925199433,
        AUTHORITY["EPSG","9122"]],
    AXIS["Latitude",NORTH],
    AXIS["Longitude",EAST],
    AUTHORITY["EPSG","4326"]]
```

```python
i=0
print("Military", "\t", "Name", "\t", 'X-Coordinate', "\t", 'Y-Coordinate\n')
for feature in layer:
    point = feature.geometry()
    name = feature.GetField('name')
    military = feature.GetField('military')
    x_coordinate = point.GetX()
    y_coordinate = point.GetY()
    print(military, "\t", name, "\t", x_coordinate, "\t", y_coordinate)
    if i >= 5:
        break
    i = i + 1
```

[64] ✓ 0.3s Python

```
Military    Name    X-Coordinate    Y-Coordinate

airfield    Kanpur Chakeri Airport    0.0    0.0
airfield    None    0.0    0.0
airfield    Gorakhpur Air Force Station    0.0    0.0
airfield    Purnea Airport    0.0    0.0
barracks    चाराली आर्मी ब्यारांक    0.0    0.0
airfield    Lucknow Air Force Station    0.0    0.0
```

4. For Reading Vector Data (KML)

File Edit Selection View Go Run Terminal Help — Vector.ipynb - gdalEnv - Visual Studio Code

Raster (1).ipynb | FinalRasterStatistics.ipynb | Vector.ipynb

D: > Course > Python Assignments > Geospatial Analysis using Python > Vector Data > Vector.ipynb > Reading Other File Formats > For Reading .GeoJSON File

+ Code + Markdown | Run All | Clear Outputs of All Cells | Restart | Interrupt | Variables | Outline ··· gdalEnv (Python 3.8.12)

For Reading .kml File

```python
from osgeo import ogr
import os
import sys
```

[65] ✓ 0.3s Python

```python
path = r'D:\Course\Python Assignments\Geospatial Analysis using Python\Vector Data\Other Data Formats\military.kml'
dataset_source = ogr.Open(path, 0)   # 0 for reading and 1 for editing of shapefile
if dataset_source is None:
    sys.exit('Could not open the file {}'.format(path))
layer_count = dataset_source.GetLayerCount()
print("Total Layer:\t", layer_count)
```

[66] ✓ 0.6s Python

```
Total Layer:    1
```

```python
layer = dataset_source.GetLayer(0)
geometry_type = layer.GetGeomType()
print("Geometry Type = ", geometry_type, ": means point data.")   # 1 for point data
feature_count = layer.GetFeatureCount()
print("Total Feature:\t", feature_count)
```

[67] ✓ 0.6s Python

```
Geometry Type =  0 : means point data.
Total Feature:   177
```

```python
spatial_reference = layer.GetSpatialRef()
print("Spatial Reference:\n", spatial_reference)
```

[68] ✓ 0.4s Python

```
Spatial Reference:
 GEOGCS["WGS 84",
    DATUM["WGS_1984",
        SPHEROID["WGS 84",6378137,298.257223563,
            AUTHORITY["EPSG","7030"]],
        AUTHORITY["EPSG","6326"]],
    PRIMEM["Greenwich",0,
        AUTHORITY["EPSG","8901"]],
    UNIT["degree",0.0174532925199433,
        AUTHORITY["EPSG","9122"]],
    AXIS["Latitude",NORTH],
    AXIS["Longitude",EAST],
    AUTHORITY["EPSG","4326"]]
```

```python
i=0
print("Military", "\t", "Name", "\t", 'X-Coordinate', "\t", 'Y-Coordinate\n')
for feature in layer:
    point = feature.geometry()
    name = feature.GetField('name')
    military = feature.GetField('military')
    x_coordinate = point.GetX()
    y_coordinate = point.GetY()
    print(military, "\t",  name, "\t", x_coordinate, "\t", y_coordinate)
    if i >= 5:
        break
    i = i + 1
```

[69] ✓ 0.5s Python

```
Military   Name    X-Coordinate    Y-Coordinate

airfield   Kanpur Chakeri Airport    0.0   0.0
airfield   None    0.0   0.0
airfield   Gorakhpur Air Force Station   0.0   0.0
airfield   Purnea Airport    0.0   0.0
barracks   चाराली आर्मी ब्याराक    0.0   0.0
airfield   Lucknow Air Force Station   0.0   0.0
```