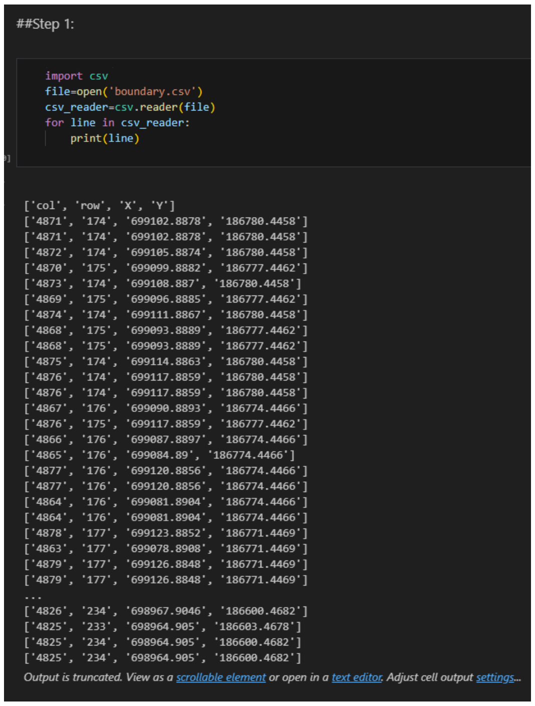
Name: Sudeep Kuikel

Guided Project2: GEOG 4057

This project includes a data as a csv file for location of water boundaries extracted from geotiff file flood_2class.tif. Our work is to use python to write script to obtain elevation values for the points from the csv file from Google Earth Engine and write the values to the shapefile. In the end, ArcGIS toolbox will be developed to provide a user interface:

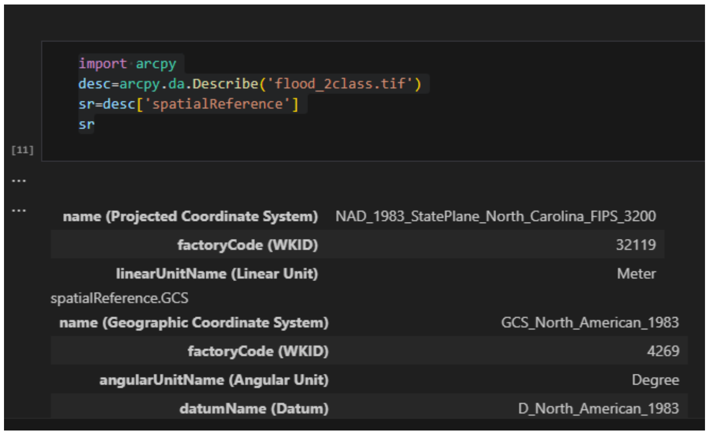
Explanations of each code parts are provided below wach snapshots of the coding that is done in notebook in Visual Studio Code:



```
##Step 1:

import csv
file=open('boundary.csv')
csv_reader=csv.reader(file)
for line in csv_reader:
    print(line)
```

```
['col', 'row', 'X', 'Y']
['4871', '174', '699102.8878', '186780.4458']
['4871', '174', '699102.8878', '186780.4458']
['4872', '174', '699105.8874', '186780.4458']
['4870', '175', '699099.8882', '186777.4462']
['4873', '174', '699108.887', '186780.4458']
['4869', '175', '699096.8885', '186777.4462']
['4874', '174', '699111.8867', '186780.4458']
['4868', '175', '699093.8889', '186777.4462']
['4868', '175', '699093.8889', '186777.4462']
['4875', '174', '699114.8863', '186780.4458']
['4876', '174', '699117.8859', '186780.4458']
['4876', '174', '699117.8859', '186780.4458']
['4867', '176', '699090.8893', '186774.4466']
['4876', '175', '699117.8859', '186777.4462']
['4866', '176', '699087.8897', '186774.4466']
['4865', '176', '699084.89', '186774.4466']
['4877', '176', '699120.8856', '186774.4466']
['4877', '176', '699120.8856', '186774.4466']
['4864', '176', '699081.8904', '186774.4466']
['4864', '176', '699081.8904', '186774.4466']
['4878', '177', '699123.8852', '186771.4469']
['4863', '177', '699078.8908', '186771.4469']
['4879', '177', '699126.8848', '186771.4469']
['4879', '177', '699126.8848', '186771.4469']
...
['4826', '234', '698967.9046', '186600.4682']
['4825', '233', '698964.905', '186603.4678']
['4825', '234', '698964.905', '186600.4682']
['4825', '234', '698964.905', '186600.4682']
Output is truncated. View as a scrollable element or open in a text editor. Adjust cell output settings...
```

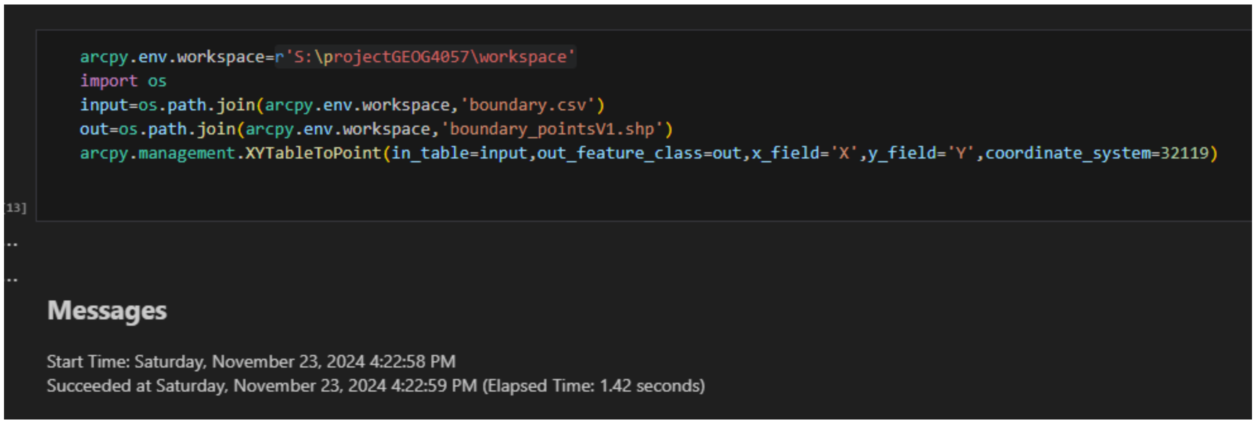
This code imports csv to work with CSV (Comma Separated Values). We opened a file named boundary.csv in read mode. The open function returns a file object, which is assigned to the variable file. The csv.reader() function took the opened file object as input and creates a CSV reader object (csv_reader). We used a for loop which iterates through the rows of the CSV file where each row is returned as a list of strings. Finally, we printed each row.



```
import arcpy
desc=arcpy.da.Describe('flood_2class.tif')
sr=desc['spatialReference']
sr
```

```
[11]
...
...
 name (Projected Coordinate System)   NAD_1983_StatePlane_North_Carolina_FIPS_3200
              factoryCode (WKID)                                            32119
        linearUnitName (Linear Unit)                                        Meter
spatialReference.GCS
 name (Geographic Coordinate System)                          GCS_North_American_1983
                  factoryCode (WKID)                                            4269
       angularUnitName (Angular Unit)                                         Degree
                   datumName (Datum)                               D_North_American_1983
```

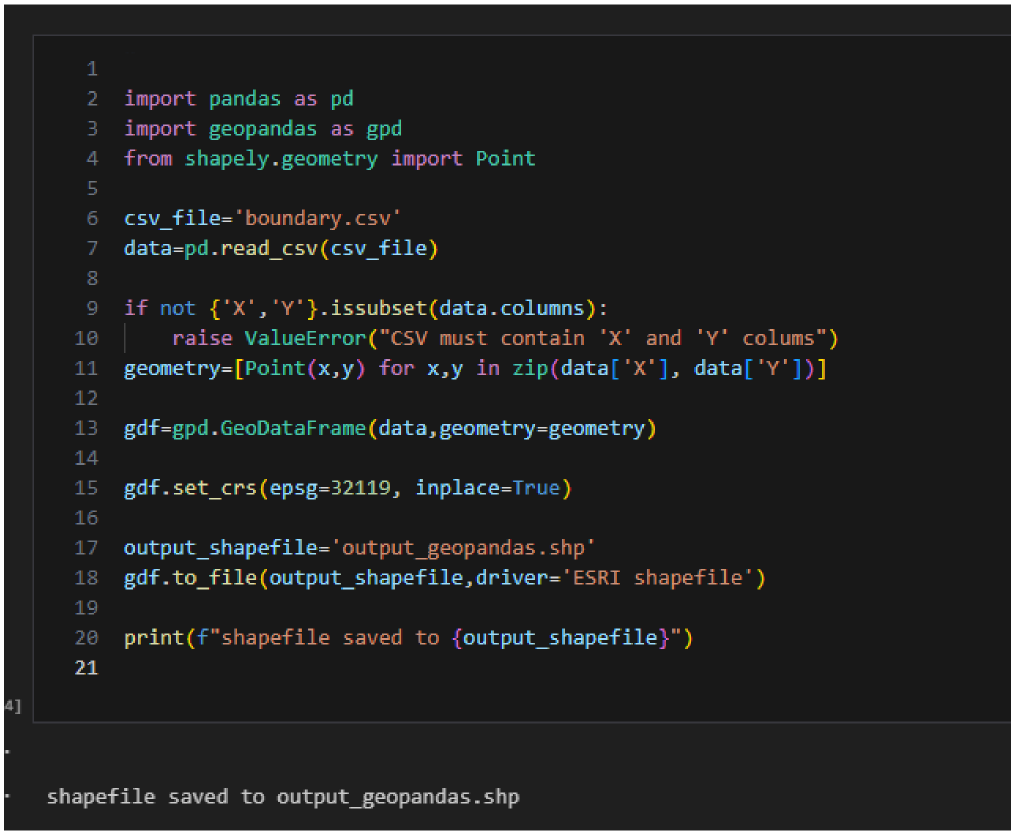
This part of the code imports arcpy with the aim to get the spatial reference of a raster file (flood_2class.tif). arcpy.da.Describe fetches metadata about the file and then desc['spatialReference'] retrieves the coordinate system deails. The last line sr gives the output with the information like the name of the spatial reference, its datum, and units, ensuring geographic alignment of data.



```
arcpy.env.workspace=r'S:\projectGEOG4057\workspace'
import os
input=os.path.join(arcpy.env.workspace,'boundary.csv')
out=os.path.join(arcpy.env.workspace,'boundary_pointsV1.shp')
arcpy.management.XYTableToPoint(in_table=input,out_feature_class=out,x_field='X',y_field='Y',coordinate_system=32119)
```

```
13]
..
..
Messages

Start Time: Saturday, November 23, 2024 4:22:58 PM
Succeeded at Saturday, November 23, 2024 4:22:59 PM (Elapsed Time: 1.42 seconds)
```

This part of the script converts a CSV file (boundary.csv) with XY coordinates into a shapefile (boundary_pointsV1.shp) using arcpy. We used arcpy.env.workspace to set the folder path for workspace of the project. We imported os to work with the file path then we used os.path.join to build input/output file paths. In the last line, arcpy.management.XYTableToPoint to convert the CSV into a point shapefile specifying X and Y fields. The code uses EPSG:32119 to define the spatial reference.

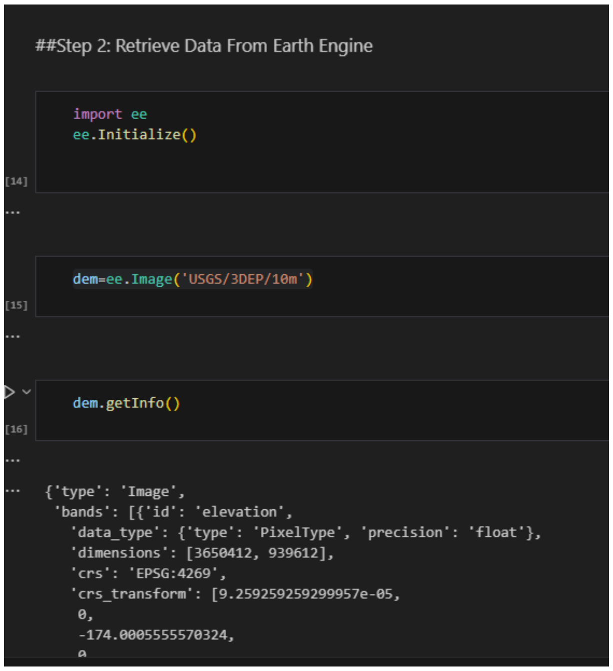


```

import pandas as pd
import geopandas as gpd
from shapely.geometry import Point

csv_file='boundary.csv'
data=pd.read_csv(csv_file)

if not {'X','Y'}.issubset(data.columns):
    raise ValueError("CSV must contain 'X' and 'Y' colums")
geometry=[Point(x,y) for x,y in zip(data['X'], data['Y'])]

gdf=gpd.GeoDataFrame(data,geometry=geometry)

gdf.set_crs(epsg=32119, inplace=True)

output_shapefile='output_geopandas.shp'
gdf.to_file(output_shapefile,driver='ESRI shapefile')

print(f"shapefile saved to {output_shapefile}")

```

```
4]
.
·  shapefile saved to output_geopandas.shp
```

Here we imported the pandas, geopandas and Point from shapely.geometery. This is the script which we prepared to convert a CSV file (boundary.csv) that contains X and Y coordinates into a GeoDataFrame and saves it as a shapefile (output_geopandas.shp).Within this code, pd.read_csv reads the CSV file. Shapely.geometry.Point converts X and Y values into geometruc Point objects. In the code GeoDataFrame creates a GeodataFrame with the point geometries and set_crs(epsg=32119) sets the coordinate system and saves the GeoDataFrame as an ESRI shapefile. The output of the code is the shapefile created.

The next part of our project is to retreive data from Earth Engine for this we went through following steps:



##Step 2: Retrieve Data From Earth Engine

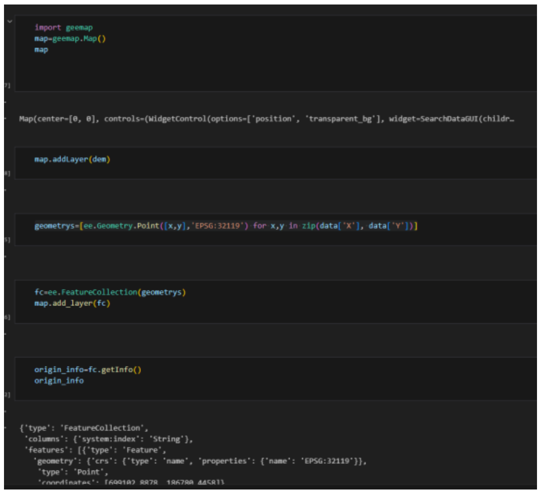
```
import ee
ee.Initialize()
```

```
dem=ee.Image('USGS/3DEP/10m')
```

```
dem.getInfo()
```

```
{'type': 'Image',
 'bands': [{'id': 'elevation',
   'data_type': {'type': 'PixelType', 'precision': 'float'},
   'dimensions': [3650412, 939612],
   'crs': 'EPSG:4269',
   'crs_transform': [9.259259259299957e-05,
    0,
    -174.0005555570324,
```

We initialized the Earth Engine Python API for use using ee.Initialize() and then loaded a DEM dataset of 10m resolution from the USGS 3DEP collection using ee.Image('USGS/3DEP/10m'). Then in the last part dem.getInfo() fetched the metadata of the image which could be helpful to understand the structure and properties of the DEM data.



```
import geemap
map=geemap.Map()
map
```

```
Map(center=[0, 0], controls=(WidgetControl(options=['position', 'transparent_bg'], widget=SearchDataGUI(childr…
```

```
map.addLayer(dem)
```

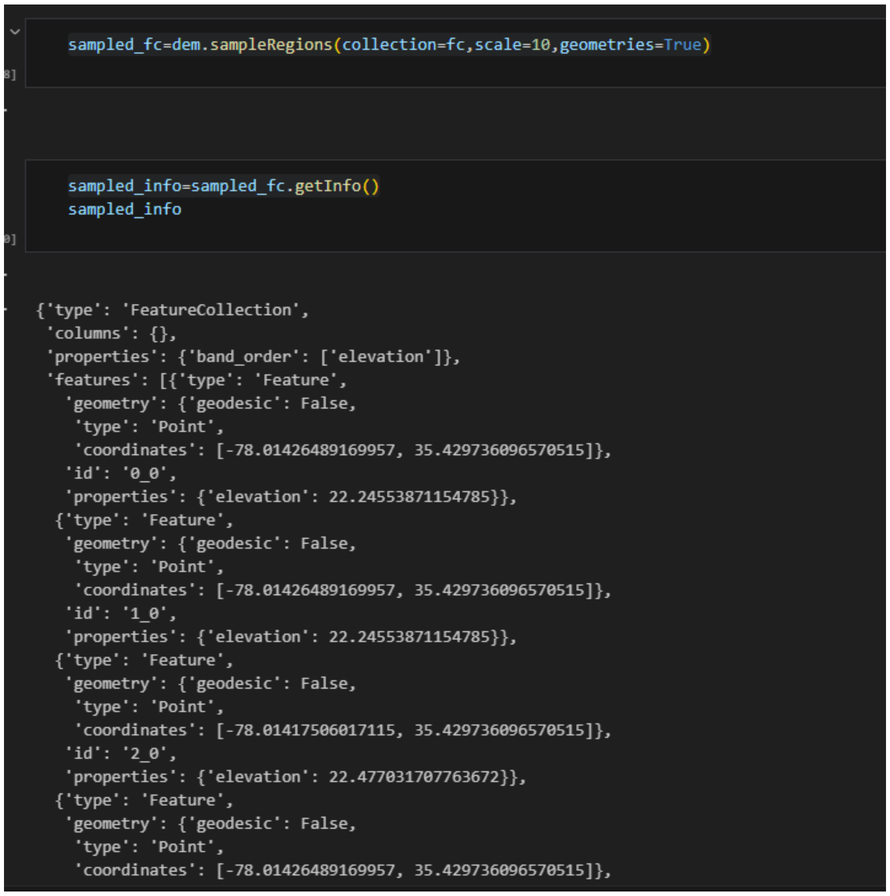
```
geometrys=[ee.Geometry.Point([x,y],'EPSG:32119') for x,y in zip(data['X'], data['Y'])]
```

```
fc=ee.FeatureCollection(geometrys)
map.add_layer(fc)
```

```
origin_info=fc.getInfo()
origin_info
```

```
{'type': 'FeatureCollection',
 'columns': {'system:index': 'String'},
 'features': [{'type': 'Feature',
   'geometry': {'crs': {'type': 'name', 'properties': {'name': 'EPSG:32119'}},
    'type': 'Point',
    'coordinates': [699102.8878, 186780.4458]},
```

These code snippet shows how to use the geemap and google earth engine API to visualize DEM data and create a feature collection from the point coordinates. Firstly, we imported geemap which initializes geemap to work with GEE data and then gee.Map() is used to create an interactive map. We used map.addLayer(dem) to overlay the DEM data on the map for visualization. For converting X and Y coordinates into geometry points using the specified reference system, we used ee.Geometry.Point([x,y], 'EPSG:32119'). Then we converted list of point geometries into a GEE feature collection using ee.FeatureCollection(geometrys). Map.add_layer(fc) added the feature collection layer to the map for visualization and finally fc.getInfo() retrieved metadata such as feature, geometry and CRS about the feature collection.

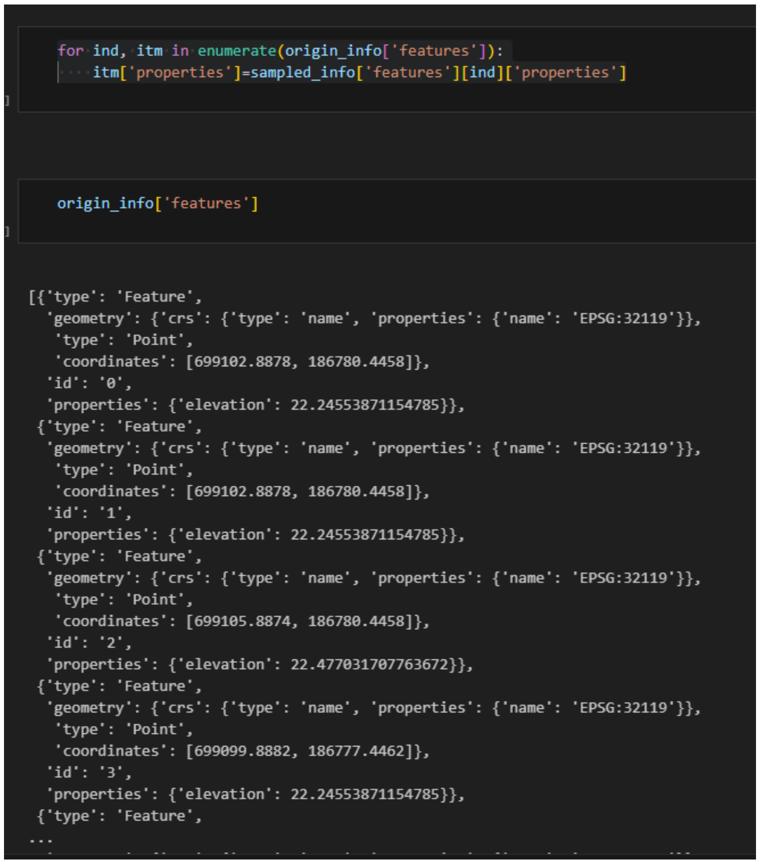


```
sampled_fc=dem.sampleRegions(collection=fc,scale=10,geometries=True)
```

```
sampled_info=sampled_fc.getInfo()
sampled_info
```

```
{'type': 'FeatureCollection',
 'columns': {},
 'properties': {'band_order': ['elevation']},
 'features': [{'type': 'Feature',
   'geometry': {'geodesic': False,
    'type': 'Point',
    'coordinates': [-78.01426489169957, 35.429736096570515]},
   'id': '0_0',
   'properties': {'elevation': 22.24553871154785}},
  {'type': 'Feature',
   'geometry': {'geodesic': False,
    'type': 'Point',
    'coordinates': [-78.01426489169957, 35.429736096570515]},
   'id': '1_0',
   'properties': {'elevation': 22.24553871154785}},
  {'type': 'Feature',
   'geometry': {'geodesic': False,
    'type': 'Point',
    'coordinates': [-78.01417506017115, 35.429736096570515]},
   'id': '2_0',
   'properties': {'elevation': 22.477031707763672}},
  {'type': 'Feature',
   'geometry': {'geodesic': False,
    'type': 'Point',
    'coordinates': [-78.01426489169957, 35.429736096570515]},
```

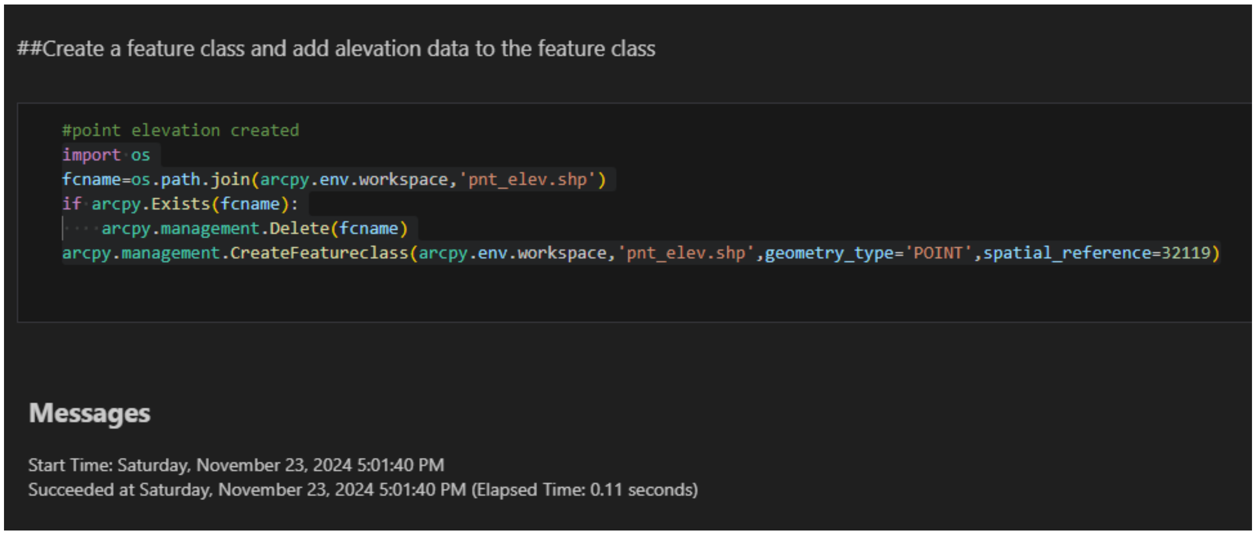
Dem.sampleRegions(collection=fc, scale=10, geometries=True) helps us to sample the DEM's elevation values at the location which is specified by the feature collection (fc). For the resolution, scale=10 specified 10m resolution and geometries=True ensures that the sample data retains the point geometries. After it, sampled_fc.getInfo() is used to retrieve the sampled data. The output here is the FeatureCollection which contains features for each point in fc with geometry and properties.


```
for ind, itm in enumerate(origin_info['features']):
    itm['properties']=sampled_info['features'][ind]['properties']
```

```
origin_info['features']
```

```
[{'type': 'Feature',
  'geometry': {'crs': {'type': 'name', 'properties': {'name': 'EPSG:32119'}},
   'type': 'Point',
   'coordinates': [699102.8878, 186780.4458]},
  'id': '0',
  'properties': {'elevation': 22.24553871154785}},
 {'type': 'Feature',
  'geometry': {'crs': {'type': 'name', 'properties': {'name': 'EPSG:32119'}},
   'type': 'Point',
   'coordinates': [699102.8878, 186780.4458]},
  'id': '1',
  'properties': {'elevation': 22.24553871154785}},
 {'type': 'Feature',
  'geometry': {'crs': {'type': 'name', 'properties': {'name': 'EPSG:32119'}},
   'type': 'Point',
   'coordinates': [699105.8874, 186780.4458]},
  'id': '2',
  'properties': {'elevation': 22.477031707763672}},
 {'type': 'Feature',
  'geometry': {'crs': {'type': 'name', 'properties': {'name': 'EPSG:32119'}},
   'type': 'Point',
   'coordinates': [699099.8882, 186777.4462]},
  'id': '3',
  'properties': {'elevation': 22.24553871154785}},
 {'type': 'Feature',
...
```

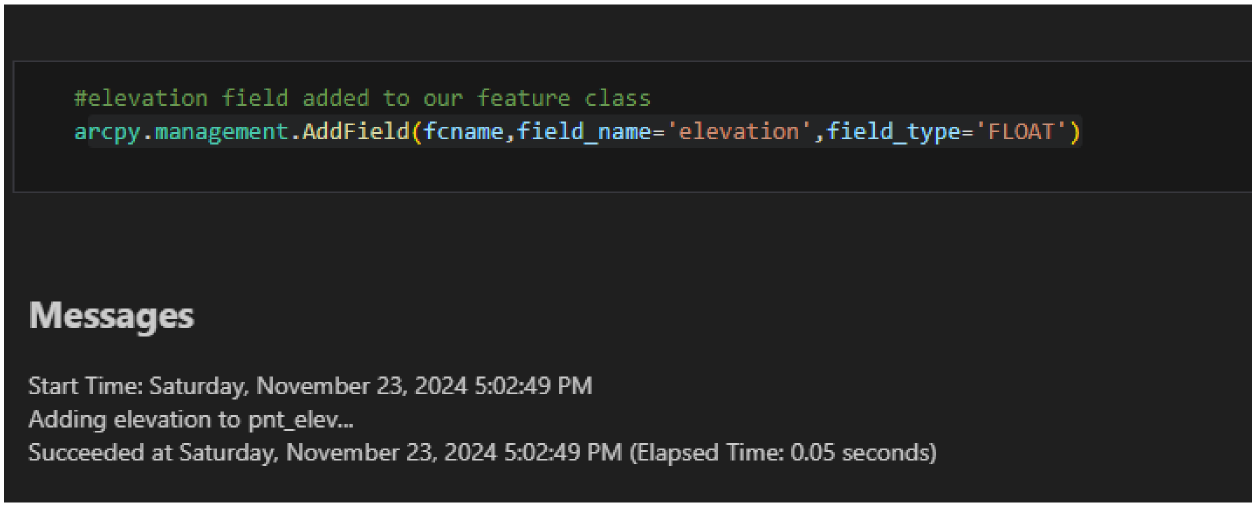
The for loop in the first cell iterates over each feature in origin_info['features'] using enumerate to access index(ind) and feature(itm). From the corresponding feature in sampled_info['features'], it updates the properties field field for each features. The other cell origin_info['features'] contains both the original geometry and the sampled elevation values under the properties field.


##Create a feature class and add alevation data to the feature class

```
#point elevation created
import os
fcname=os.path.join(arcpy.env.workspace,'pnt_elev.shp')
if arcpy.Exists(fcname):
    arcpy.management.Delete(fcname)
arcpy.management.CreateFeatureclass(arcpy.env.workspace,'pnt_elev.shp',geometry_type='POINT',spatial_reference=32119)
```

```
Messages

Start Time: Saturday, November 23, 2024 5:01:40 PM
Succeeded at Saturday, November 23, 2024 5:01:40 PM (Elapsed Time: 0.11 seconds)
```

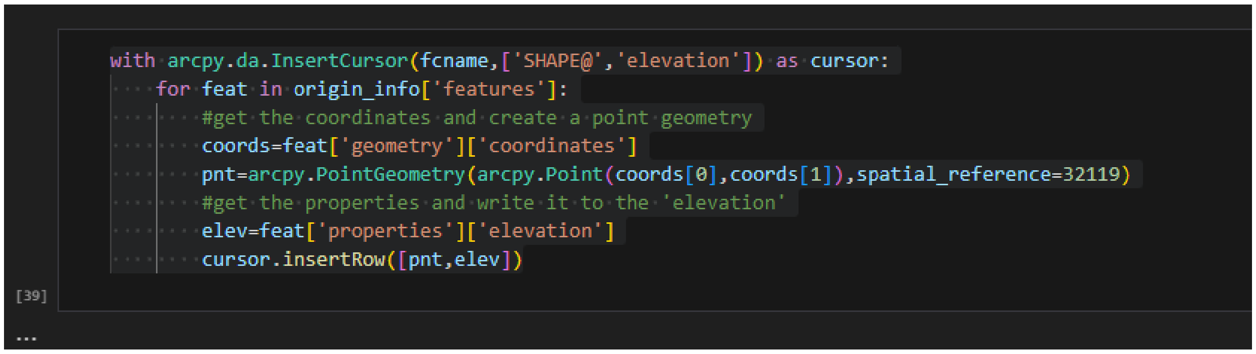
The last part of our coding is to create a feature class and add elevation data to the feature class. For this, we imported os to work with file path management. We constructed the file path for the shapefile pnt_elev.shp within the current ArcPy workspace. Then the code checks whether the name of shapefile already exists and delete the existing name using arcpy.management.CreateFeatureClass(). Then the arcpy.management.CretaeFeatureClass() creates a new feature class with ‘POINT’ geometry type with the spatial reference as 32119.



```
#elevation field added to our feature class
arcpy.management.AddField(fcname,field_name='elevation',field_type='FLOAT')
```

```
Messages

Start Time: Saturday, November 23, 2024 5:02:49 PM
Adding elevation to pnt_elev...
Succeeded at Saturday, November 23, 2024 5:02:49 PM (Elapsed Time: 0.05 seconds)
```

This code adds the elevation field to our feature class using arcpy.management.AddField(). The new field name is ‘elevation;’ within the feature class ‘pnt_elev.shp’ with field_type as ‘FLOAT’.



```
with arcpy.da.InsertCursor(fcname,['SHAPE@','elevation']) as cursor:
    for feat in origin_info['features']:
        #get the coordinates and create a point geometry
        coords=feat['geometry']['coordinates']
        pnt=arcpy.PointGeometry(arcpy.Point(coords[0],coords[1]),spatial_reference=32119)
        #get the properties and write it to the 'elevation'
        elev=feat['properties']['elevation']
        cursor.insertRow([pnt,elev])
[39]
...
```

Lastly, we populated the feature class with the point geometries and elevation values. We used InsertCursor to insert geometry(SHAPE@) and elevation. The for loop, iterates the feature which contains the point geometries and elevation values. Arcpy.PointGeometry helped to create geometry by extracting coordinates and then lastly we extracted elevation and inserted [geometry,elevation] into the feature class. The main aim was to transfer spatial and attribute data into the feature class

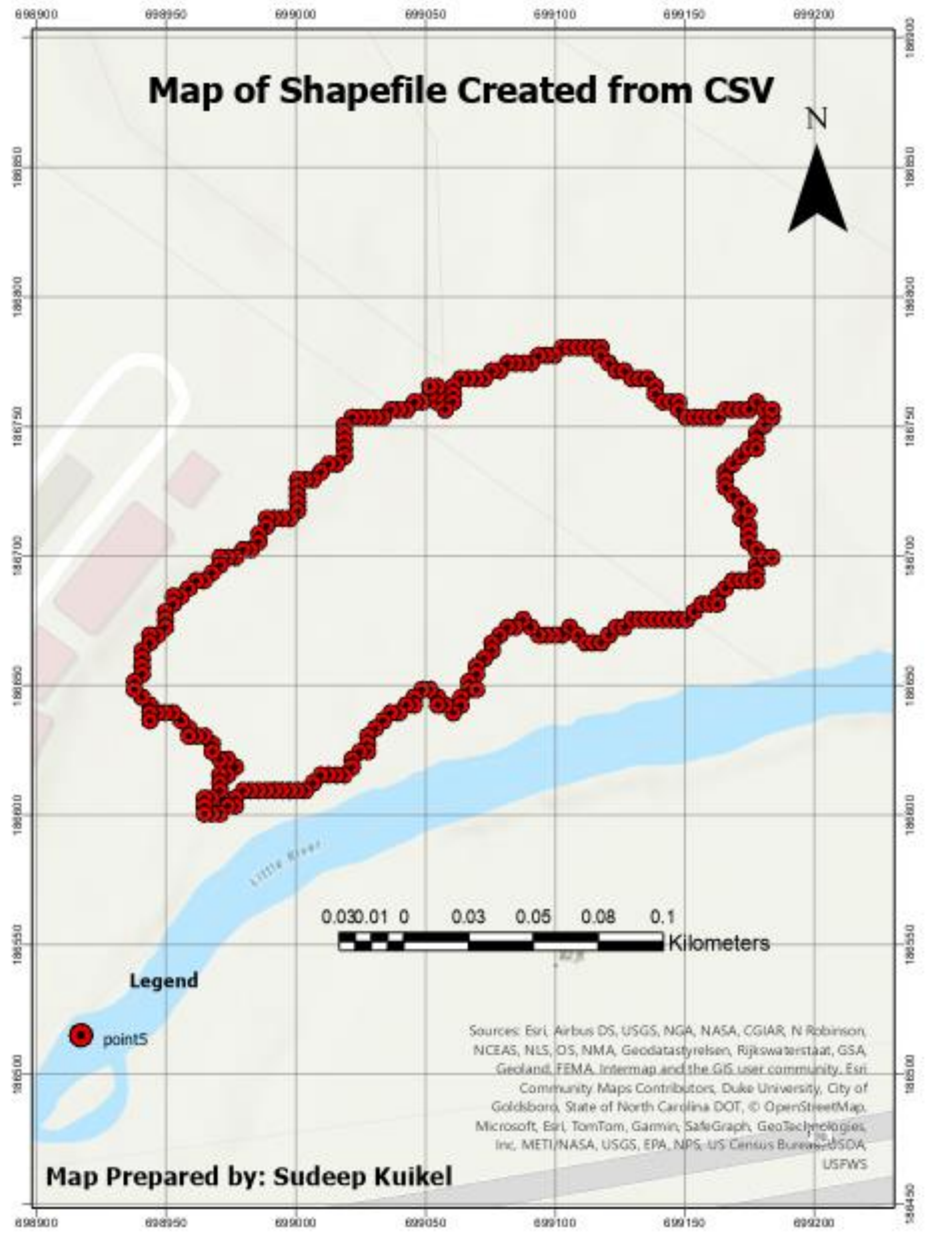
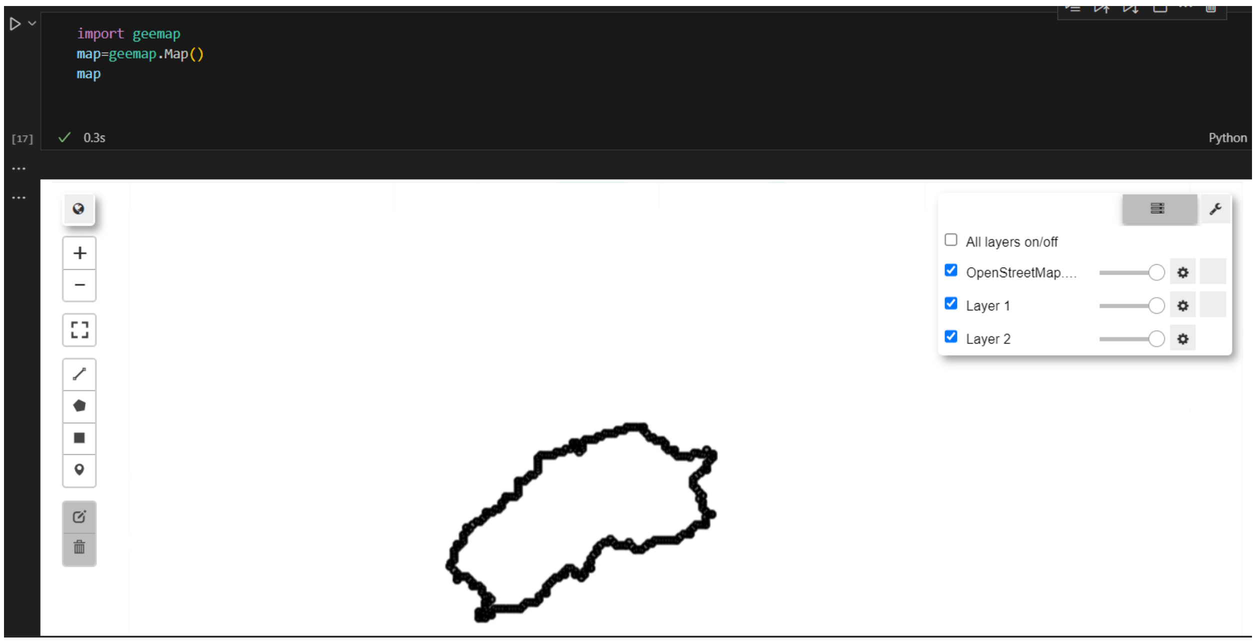

Map of Shapefile Created from CSV
N
0.030.01 0 0.03 0.05 0.08 0.1
Kilometers
Legend
point5
Little River
Sources: Esri, Airbus DS, USGS, NGA, NASA, CGIAR, N Robinson, NCEAS, NLS, OS, NMA, Geodatastyrelsen, Rijkswaterstaat, GSA, Geoland, FEMA, Intermap and the GIS user community, Esri Community Maps Contributors, Duke University, City of Goldsboro, State of North Carolina DOT, © OpenStreetMap, Microsoft, Esri, TomTom, Garmin, SafeGraph, GeoTechnologies, Inc, METI/NASA, USGS, EPA, NPS, US Census Bureau, USDA, USFWS
Map Prepared by: Sudeep Kuikel

Figure 1: Final Layout of Map prepared as part of Guided Project2

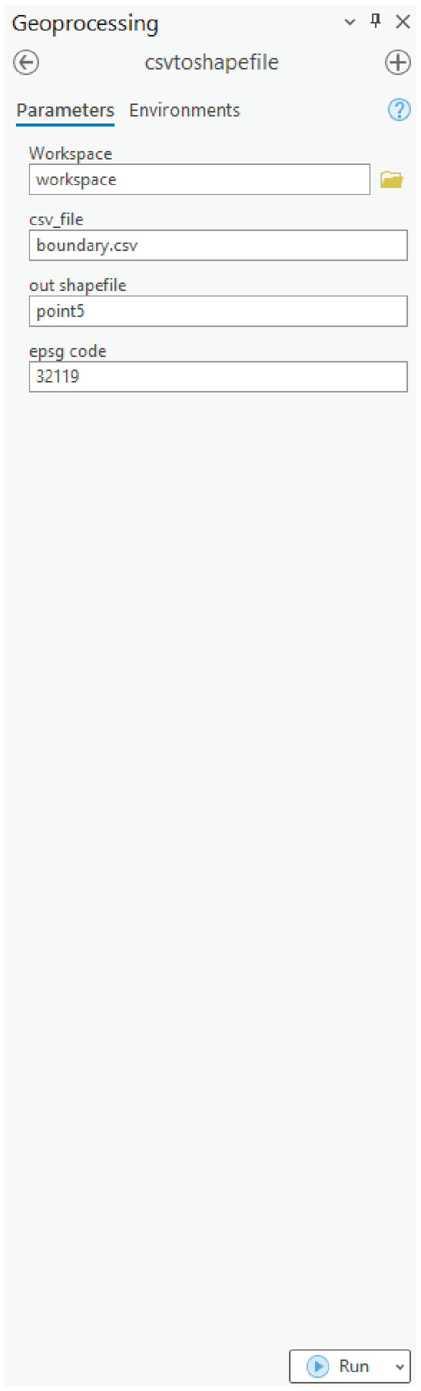
Some additional snapshots for the reference to our work are provided below:



```
import geemap
map=geemap.Map()
map
```

```
[17]  ✓ 0.3s                                                    Python
```

Figure 2: Visualization of point data using geemap



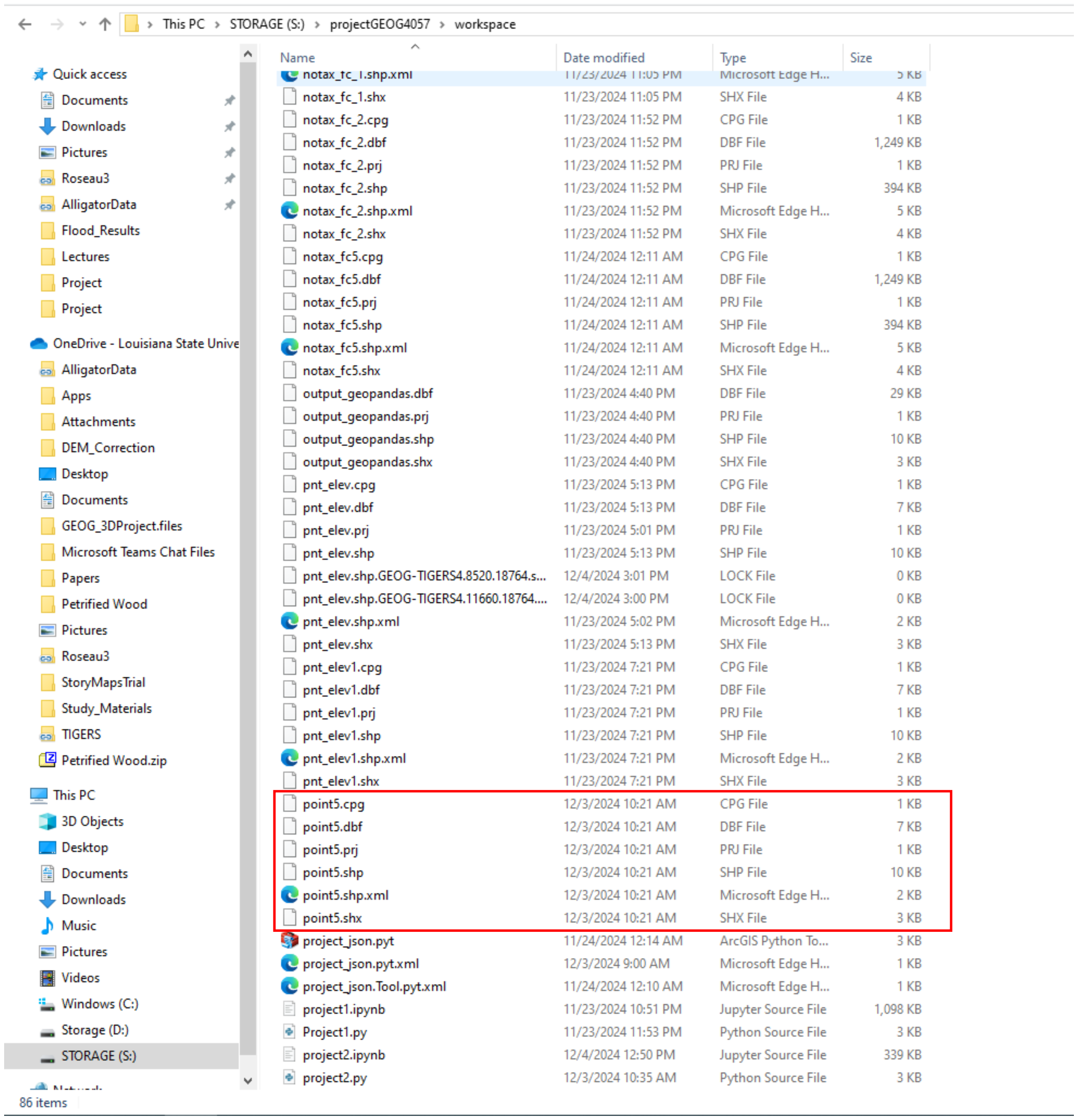
Workspace: workspace

csv_file: boundary.csv

out shapefile: point5

epsg code: 32119

Figure 3: Parameters used for the Tool Created for ArcGIS Pro



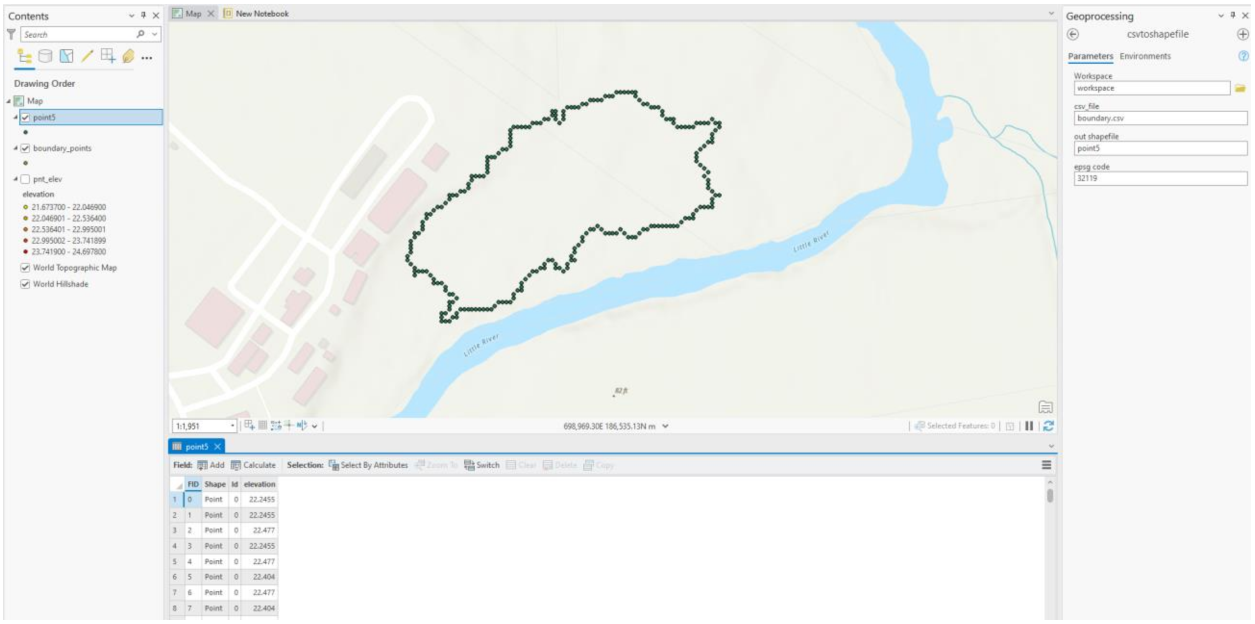
This PC › STORAGE (S:) › projectGEOG4057 › workspace

- Quick access
  - Documents
  - Downloads
  - Pictures
  - Roseau3
  - AlligatorData
  - Flood_Results
  - Lectures
  - Project
  - Project
- OneDrive - Louisiana State Unive
  - AlligatorData
  - Apps
  - Attachments
  - DEM_Correction
  - Desktop
  - Documents
  - GEOG_3DProject.files
  - Microsoft Teams Chat Files
  - Papers
  - Petrified Wood
  - Pictures
  - Roseau3
  - StoryMapsTrial
  - Study_Materials
  - TIGERS
  - Petrified Wood.zip
- This PC
  - 3D Objects
  - Desktop
  - Documents
  - Downloads
  - Music
  - Pictures
  - Videos
  - Windows (C:)
  - Storage (D:)
  - STORAGE (S:)

Name	Date modified	Type	Size
notax_fc_1.shp.xml	11/23/2024 11:05 PM	Microsoft Edge H...	5 KB
notax_fc_1.shx	11/23/2024 11:05 PM	SHX File	4 KB
notax_fc_2.cpg	11/23/2024 11:52 PM	CPG File	1 KB
notax_fc_2.dbf	11/23/2024 11:52 PM	DBF File	1,249 KB
notax_fc_2.prj	11/23/2024 11:52 PM	PRJ File	1 KB
notax_fc_2.shp	11/23/2024 11:52 PM	SHP File	394 KB
notax_fc_2.shp.xml	11/23/2024 11:52 PM	Microsoft Edge H...	5 KB
notax_fc_2.shx	11/23/2024 11:52 PM	SHX File	4 KB
notax_fc5.cpg	11/24/2024 12:11 AM	CPG File	1 KB
notax_fc5.dbf	11/24/2024 12:11 AM	DBF File	1,249 KB
notax_fc5.prj	11/24/2024 12:11 AM	PRJ File	1 KB
notax_fc5.shp	11/24/2024 12:11 AM	SHP File	394 KB
notax_fc5.shp.xml	11/24/2024 12:11 AM	Microsoft Edge H...	5 KB
notax_fc5.shx	11/24/2024 12:11 AM	SHX File	4 KB
output_geopandas.dbf	11/23/2024 4:40 PM	DBF File	29 KB
output_geopandas.prj	11/23/2024 4:40 PM	PRJ File	1 KB
output_geopandas.shp	11/23/2024 4:40 PM	SHP File	10 KB
output_geopandas.shx	11/23/2024 4:40 PM	SHX File	3 KB
pnt_elev.cpg	11/23/2024 5:13 PM	CPG File	1 KB
pnt_elev.dbf	11/23/2024 5:13 PM	DBF File	7 KB
pnt_elev.prj	11/23/2024 5:01 PM	PRJ File	1 KB
pnt_elev.shp	11/23/2024 5:13 PM	SHP File	10 KB
pnt_elev.shp.GEOG-TIGERS4.8520.18764.s...	12/4/2024 3:01 PM	LOCK File	0 KB
pnt_elev.shp.GEOG-TIGERS4.11660.18764....	12/4/2024 3:00 PM	LOCK File	0 KB
pnt_elev.shp.xml	11/23/2024 5:02 PM	Microsoft Edge H...	2 KB
pnt_elev.shx	11/23/2024 5:13 PM	SHX File	3 KB
pnt_elev1.cpg	11/23/2024 7:21 PM	CPG File	1 KB
pnt_elev1.dbf	11/23/2024 7:21 PM	DBF File	7 KB
pnt_elev1.prj	11/23/2024 7:21 PM	PRJ File	1 KB
pnt_elev1.shp	11/23/2024 7:21 PM	SHP File	10 KB
pnt_elev1.shp.xml	11/23/2024 7:21 PM	Microsoft Edge H...	2 KB
pnt_elev1.shx	11/23/2024 7:21 PM	SHX File	3 KB
point5.cpg	12/3/2024 10:21 AM	CPG File	1 KB
point5.dbf	12/3/2024 10:21 AM	DBF File	7 KB
point5.prj	12/3/2024 10:21 AM	PRJ File	1 KB
point5.shp	12/3/2024 10:21 AM	SHP File	10 KB
point5.shp.xml	12/3/2024 10:21 AM	Microsoft Edge H...	2 KB
point5.shx	12/3/2024 10:21 AM	SHX File	3 KB
project_json.pyt	11/24/2024 12:14 AM	ArcGIS Python To...	3 KB
project_json.pyt.xml	12/3/2024 9:00 AM	Microsoft Edge H...	1 KB
project_json.Tool.pyt.xml	11/24/2024 12:10 AM	Microsoft Edge H...	1 KB
project1.ipynb	11/23/2024 10:51 PM	Jupyter Source File	1,098 KB
Project1.py	11/23/2024 11:53 PM	Python Source File	3 KB
project2.ipynb	12/4/2024 12:50 PM	Jupyter Source File	339 KB
project2.py	12/3/2024 10:35 AM	Python Source File	3 KB

86 items

Figure 4: Reference of the shapefile created using the Tool we created



Workspace: workspace
csv_file: boundary.csv
out shapefile: point5
epsg code: 32119

	FID	Shape	Id	elevation
1	0	Point	0	22.2455
2	1	Point	0	22.2455
3	2	Point	0	22.477
4	3	Point	0	22.2455
5	4	Point	0	22.477
6	5	Point	0	22.404
7	6	Point	0	22.477
8	7	Point	0	22.404

Figure 5: Loaded the point on the ArcGIS pro