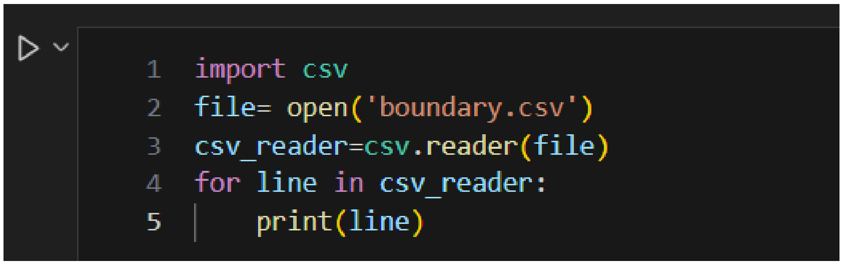
1. Importing and opening the csv file
   - The csv module is used for reading and writing CSV files efficiently.
   - The open function opens the boundary.csv file in read mode. This creates a file object.
   - The csv.reader function reads the CSV file and returns each row as a list of values.
   - The for loop iterates through each row in the CSV file, printing its content.

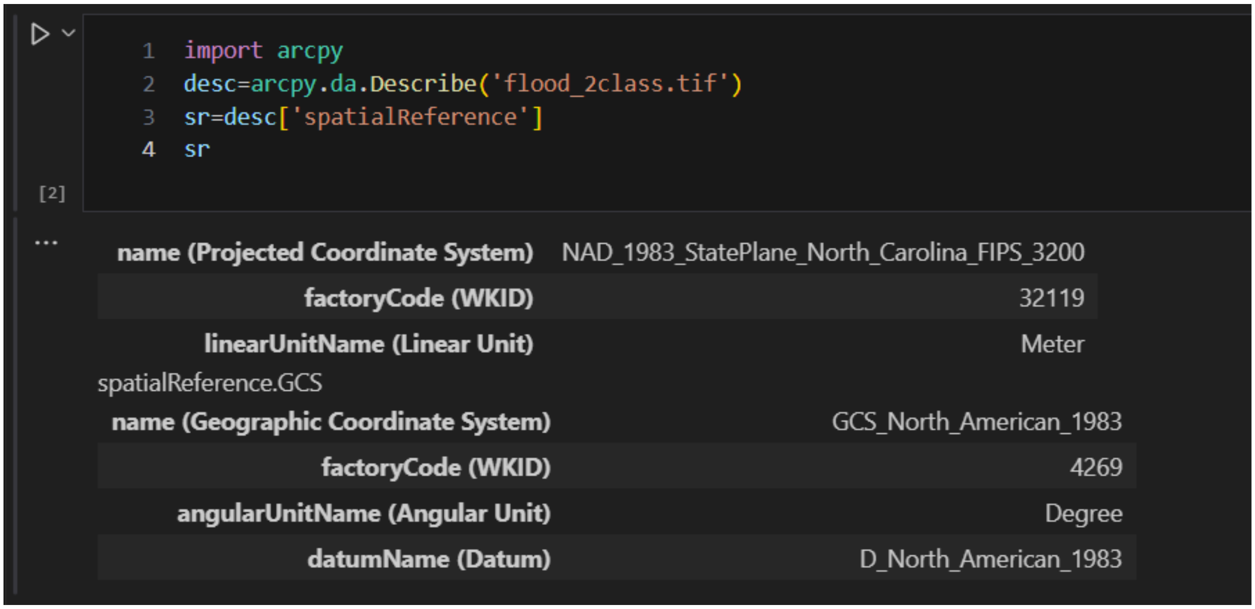


```
import csv
file= open('boundary.csv')
csv_reader=csv.reader(file)
for line in csv_reader:
    print(line)
```

2. Retrieve Raster Spatial Reference
   - The script imports the arcpy library, used for geographic data analysis and manipulation.
   - The arcpy.da.Describe function retrieves metadata about the raster file flood_2class.tif. This metadata includes properties like spatial reference, extent, pixel size, etc.
   - The 'spatialReference' key from the description dictionary contains the spatial reference information, such as coordinate system and projection.

   The variable `sr` will store the spatial reference information as an ArcPy `SpatialReference` object. This object can provide details like:

   - Coordinate system name.
   - EPSG code.
   - Linear and angular units.
   - Datum and projection type.

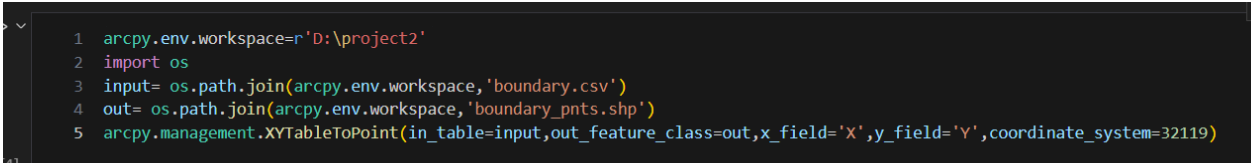


```
import arcpy
desc=arcpy.da.Describe('flood_2class.tif')
sr=desc['spatialReference']
sr
```

```
[2]
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
```

3. Convert CSV to Point Shapefile

- The workspace is defined as D:\project2, where input and output files are located.
- **Define Input and Output**
  - input: Path to the CSV file containing the X and Y coordinate fields.
  - out: Path for the output shapefile.
- **Convert CSV to Shapefile**
  - The XYTableToPoint tool creates a point feature class (shapefile) using the X and Y fields as coordinates.
  - The spatial reference is defined using EPSG code 32119 (NAD83 / Louisiana South).

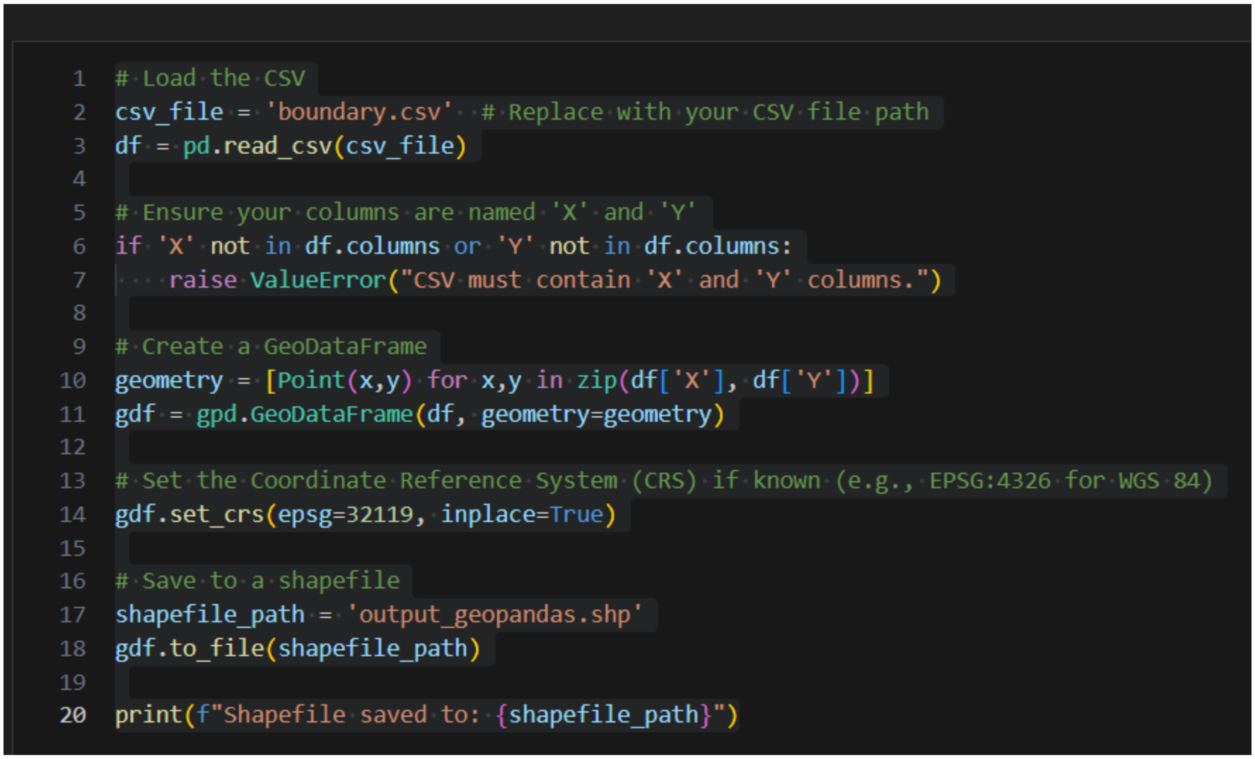


```
arcpy.env.workspace=r'D:\project2'
import os
input= os.path.join(arcpy.env.workspace,'boundary.csv')
out= os.path.join(arcpy.env.workspace,'boundary_pnts.shp')
arcpy.management.XYTableToPoint(in_table=input,out_feature_class=out,x_field='X',y_field='Y',coordinate_system=32119)
```

4. CSV to Shapefile Conversion with GeoPandas

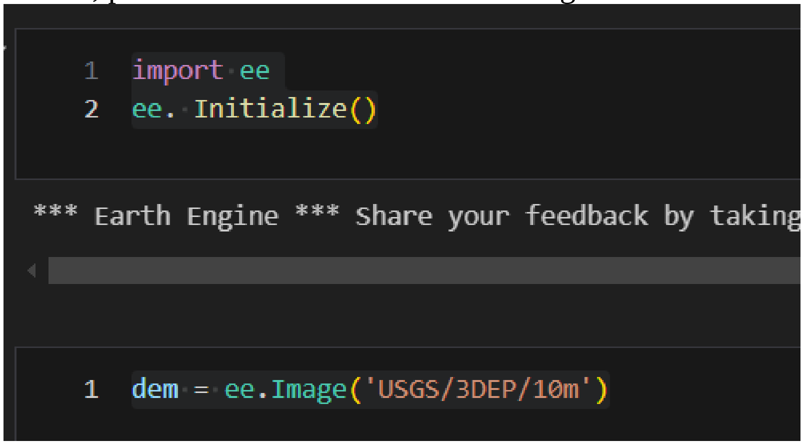
- Reads a CSV file (boundary.csv) using pandas.
- Ensures the CSV contains X and Y columns for the coordinates.
- Uses geopandas to construct a GeoDataFrame from the CSV data.
- Converts X and Y coordinates into Point geometries.
- Assigns a coordinate reference system (CRS) to the GeoDataFrame (EPSG:32119 for NAD83 Louisiana South).
- Writes the GeoDataFrame to a shapefile (output_geopandas.shp) for use in GIS applications

A shapefile (output_geopandas.shp) containing point features based on the CSV's coordinates.



```
# Load the CSV
csv_file = 'boundary.csv'  # Replace with your CSV file path
df = pd.read_csv(csv_file)

# Ensure your columns are named 'X' and 'Y'
if 'X' not in df.columns or 'Y' not in df.columns:
    raise ValueError("CSV must contain 'X' and 'Y' columns.")

# Create a GeoDataFrame
geometry = [Point(x,y) for x,y in zip(df['X'], df['Y'])]
gdf = gpd.GeoDataFrame(df, geometry=geometry)

# Set the Coordinate Reference System (CRS) if known (e.g., EPSG:4326 for WGS 84)
gdf.set_crs(epsg=32119, inplace=True)

# Save to a shapefile
shapefile_path = 'output_geopandas.shp'
gdf.to_file(shapefile_path)

print(f"Shapefile saved to: {shapefile_path}")
```

5. Importing and Initializing the google earth engine
    - The ee module is imported to interact with Google Earth Engine resources.
    - The ee.Initialize() function connects your script to GEE, requiring authentication (if not already authenticated).
    - The ee.Image('USGS/3DEP/10m') function accesses the 10m resolution elevation dataset, part of the USGS 3D Elevation Program.



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

```
*** Earth Engine *** Share your feedback by taking
```

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

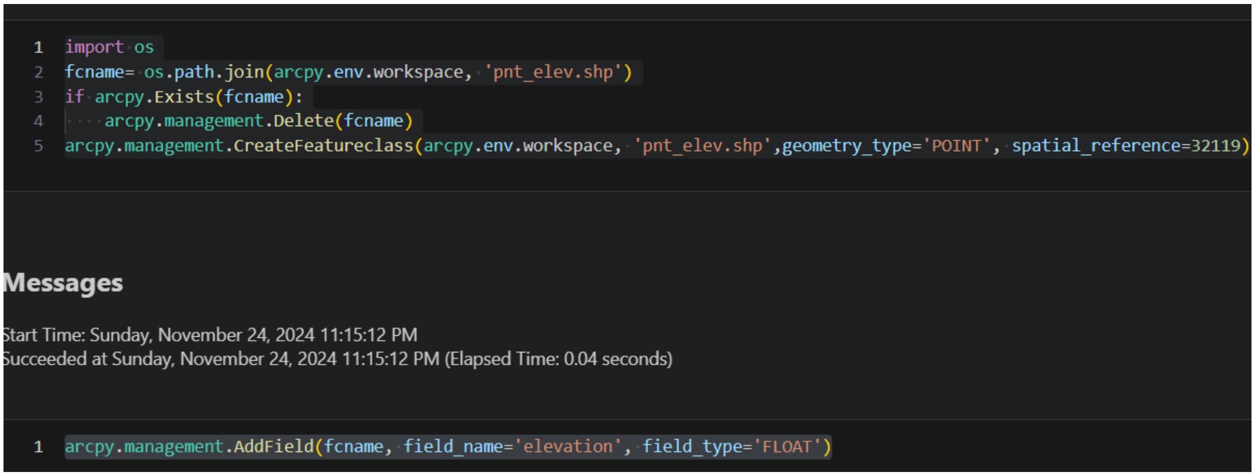
6. Create Shapefile for Point Features with Elevation

- fcname: Specifies the path for the new shapefile (pnt_elev.shp) within the workspace.
- Deletes the shapefile if it already exists to avoid conflicts.

- Uses CreateFeatureclass to create a new shapefile with:
    - Geometry type: POINT.

- Spatial reference: EPSG:32119 (NAD83 Louisiana South).

- Adds a new field named elevation with a FLOAT data type to store numerical elevation values.



```
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

```
tart Time: Sunday, November 24, 2024 11:15:12 PM
ucceeded at Sunday, November 24, 2024 11:15:12 PM (Elapsed Time: 0.04 seconds)
```

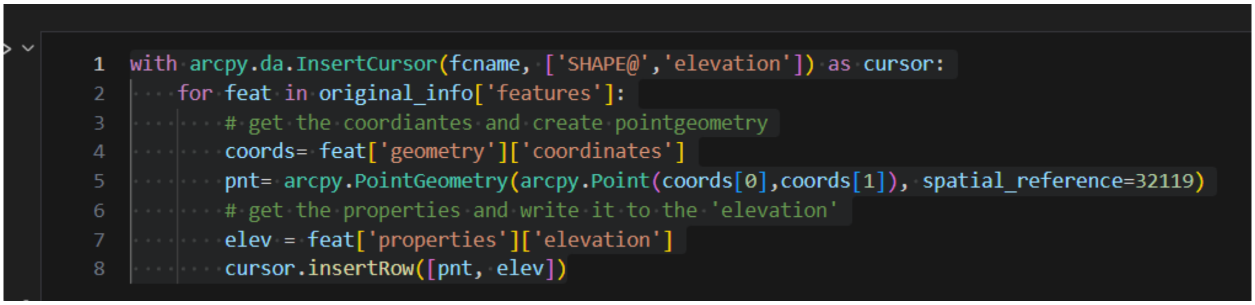
```
arcpy.management.AddField(fcname, field_name='elevation', field_type='FLOAT')
```

7. Insert Point Features with Elevation into Shapefile

- The cursor targets the shapefile (fcname) and includes the SHAPE@ (geometry) and elevation fields.
- Loops over features in the original_info dictionary (presumably GeoJSON-like data).

- Retrieves the coordinates of each feature to create an ArcPy PointGeometry object with the specified spatial reference (EPSG:32119).
- Reads the elevation value from the properties of the feature.

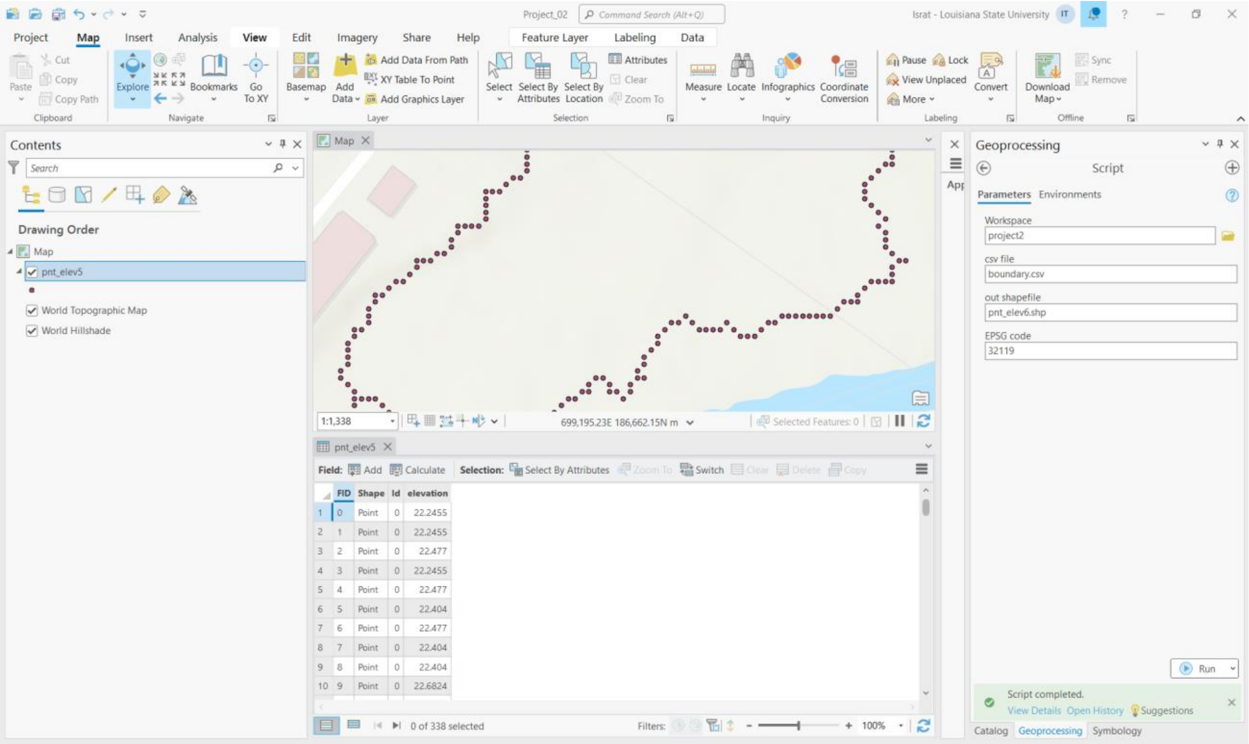
- Adds each point and its corresponding elevation value into the shapefile.



```
with arcpy.da.InsertCursor(fcname, ['SHAPE@','elevation']) as cursor:
    for feat in original_info['features']:
        # get the coordiantes and create pointgeometry
        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])
```

8. Finally running the tool

9.



Workspace
project2

csv file
boundary.csv

out shapefile
pnt_elev6.shp

EPSG code
32119

Script completed.
View Details Open History Suggestions

	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
9	8	Point	0	22.404
10	9	Point	0	22.6824