Name: Jeimy Martinez De La Hoz
Date: 12/04/25

Final Project: Project 2

URL: https://github.com/Jeimymdelahoz/geog4057_JeimyMartinez/tree/c9c8b8bd668996259c3d7d4e07073b66be93effb/Project2

1. Introduction

This project focuses on extracting elevation values for a set of boundary points derived from the flood_2class.tif dataset. Using Python, Google Earth Engine, and ArcGIS Pro, the workflow converts the provided CSV file into a shapefile, retrieves elevation values from the USGS 3DEP model, and writes those values back into the shapefile. A custom ArcGIS toolbox was developed to allow users to run the entire process through a graphical interface.

1.1 Project Summary

The goal of this project is to extract elevation values for water-boundary points obtained from the *flood_2class.tif* geotiff. The boundary coordinates were provided in a CSV file, which was first converted into a shapefile to enable geoprocessing in ArcGIS Pro. Elevation values for each point were then retrieved from the USGS 3DEP 10-meter digital elevation model using Google Earth Engine (GEE). A custom Python script was developed to automate the conversion, elevation extraction, and attribute writing process. To support repeatable use, the workflow was integrated into an ArcGIS Python Toolbox, allowing users to run the procedure through a simple interface. The final outputs include the elevation-enhanced shapefile and a map layout visualizing the results.

1.2 Data Format and Description:

The project data consists of a CSV file containing boundary points extracted from the *flood_2class.tif* raster. The file includes four fields, *row*, *column*, *X*, and *Y*, where the *X* and *Y* values represent the point locations in the same coordinate system as the original raster. These coordinates serve as the basis for generating point geometries in ArcGIS Pro. Elevation information is obtained separately from the USGS 3DEP 10-meter digital elevation model, available through Google Earth Engine.

2. Methodology

2.1 Explanation of data sources

The CSV file provides the spatial locations needed to reconstruct water-boundary points as a shapefile for further geoprocessing. Once converted into point features, these locations serve as inputs to Google Earth Engine, where the USGS 3DEP DEM is used to retrieve elevation values.

The combination of locally supplied boundary coordinates and cloud-based elevation data enables the integration of terrain information into the final GIS dataset.

2.2 processing steps

The workflow was implemented in three stages using a Jupyter Notebook, a standalone Python script, and an ArcGIS Python Toolbox. The notebook was first used to inspect the CSV structure, verify coordinate behavior, test Earth Engine authentication, and confirm that elevation values could be successfully retrieved from the USGS 3DEP dataset. After validating the logic, the functionality was consolidated into a Python script that automated the complete process: reading the CSV, converting the coordinates into a shapefile, querying Google Earth Engine for elevation, and writing the extracted values back into the attribute table. To make the workflow accessible within ArcGIS Pro, the script was integrated into a Python Toolbox (.pyt) that allowed users to execute the procedure through a graphical interface by selecting the required input files and providing the Earth Engine project name.

2.3 Analysis Approach

The analysis centered on integrating elevation information into the boundary-point dataset to enhance its spatial characterization. Once the point geometries were generated, each location was converted into a Google Earth Engine geometry object and used to sample elevation values from the USGS 3DEP DEM. This approach ensured consistent elevation retrieval across all points while leveraging GEE's ability to process large-scale geospatial data efficiently. The resulting elevation values were written directly into the shapefile's attribute table, allowing the enhanced dataset to be visualized and symbolized in ArcGIS Pro. The final map layout was used to examine the spatial pattern of elevation along the water boundary and to validate the accuracy of the processed output.

3. Code Documentation

This section provides a high-level overview of the Python scripts used to convert boundary coordinates from a CSV file into a shapefile, extract elevation values from Google Earth Engine (GEE), and write those values back into the shapefile. The workflow is implemented across three components: a primary Python script (project2.py), an exploratory Jupyter notebook, and a custom ArcGIS Pro toolbox (project2.pyt) that provides a user interface for running the process.

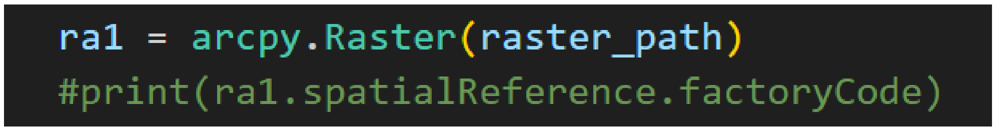
➢ **Function Overview: project2()**

The core processing logic is encapsulated in the project2() function defined in *project2.py*. This function automates the entire workflow and accepts four parameters:

- **project_name**: GEE project identifier used for initializing Earth Engine
- **csv_path**: path to the input CSV file containing point coordinates
- **raster_path**: path to the reference raster (flood_2class.tif) used to determine spatial reference
- **shapefile_path**: path where the output shapefile will be created

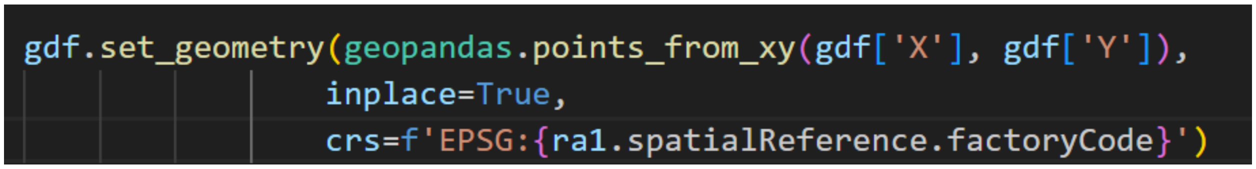
The function performs all major tasks: reading inputs, creating the shapefile, extracting GEE elevation, and updating the attribute table.

- **Reading and Preparing the Input Data:** The CSV is first loaded using pandas, then converted into a GeoDataFrame. Because the coordinates must match the CRS of the original geotiff, the script loads the raster with:



```
ra1 = arcpy.Raster(raster_path)
#print(ra1.spatialReference.factoryCode)
```

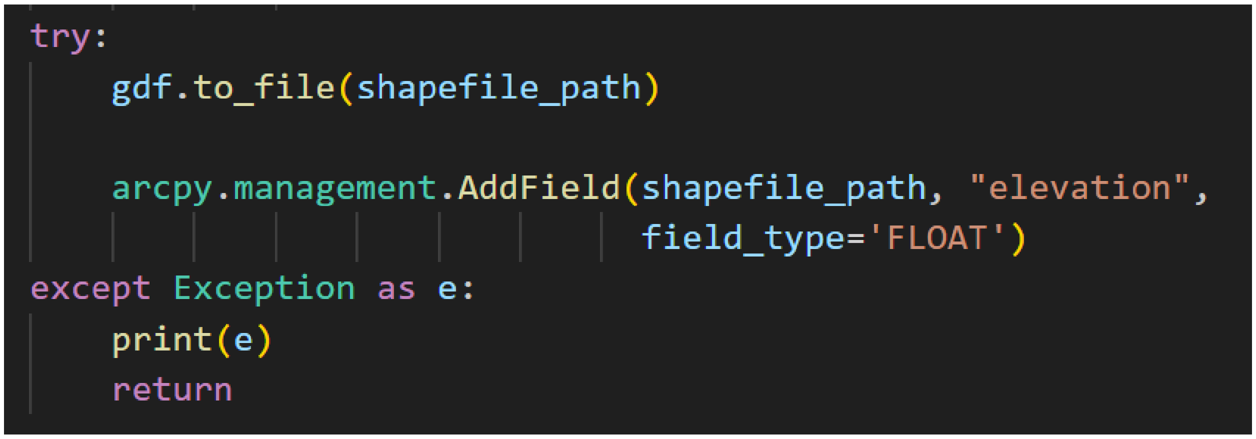
The raster's spatial reference (factoryCode) is then applied to the point geometries:



```
gdf.set_geometry(geopandas.points_from_xy(gdf['X'], gdf['Y']),
                 inplace=True,
                 crs=f'EPSG:{ra1.spatialReference.factoryCode}')
```

This ensures that the output shapefile inherits the correct projection.

- **Creating the Output Shapefile:** The GeoDataFrame is exported directly to a shapefile, and an empty field named elevation is then added using ArcPy:

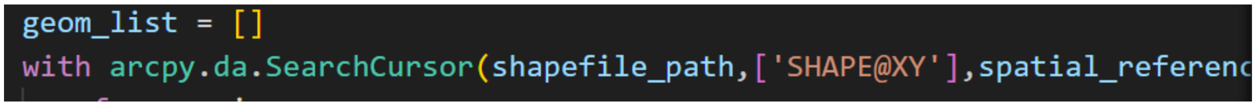


```
try:
    gdf.to_file(shapefile_path)

    arcpy.management.AddField(shapefile_path, "elevation",
                              field_type='FLOAT')
except Exception as e:
    print(e)
    return
```

This prepares the attribute table to receive the elevation values extracted from Google Earth Engine.

- **Extracting Coordinates and Building a GEE FeatureCollection:** The script reads every point in the shapefile using:



```
geom_list = []
with arcpy.da.SearchCursor(shapefile_path,['SHAPE@XY'],spatial_referenc
```

Each coordinate pair (X, Y) is converted into an Earth Engine point geometry:

```
        geom = ee.Geometry.Point([X,Y])
        geom_list.append(geom)
geom_col = ee.FeatureCollection(geom_list)
elev = dem.sampleRegions(geom_col).getInfo().get('features')
```

All points are aggregated into an ee.FeatureCollection, which is required for batch sampling of the elevation model.

- **Accessing the USGS 3DEP Elevation Model in Google Earth Engine:** The script initializes Earth Engine using the project name provided, the elevation model is then loaded. Elevation values for each boundary point are retrieved; the resulting list contains one elevation value per input geometry.
- **Writing Elevation Values Back to the Shapefile:** An UpdateCursor is used to populate the elevation field:

```
with arcpy.da.UpdateCursor(shapefile_path,['elevation']) as cursor:
    for row in cursor:
        elevation = elev[i]['properties']['elevation']
        row[0] = elevation
        cursor.updateRow(row)
        i += 1
```

This step ensures that each point in the shapefile receives the correct elevation value corresponding to its location.

- **Toolbox Integration (project2.pyt):** A custom toolbox was developed to allow ArcGIS Pro users to run the script through a graphical interface. The toolbox:

  - Defines four parameters: CSV input, shapefile output, raster dataset, and GEE project name
  - Passes these values into `project2()` inside the `execute ()` method
  - Allows the workflow to operate like a standard geoprocessing tool

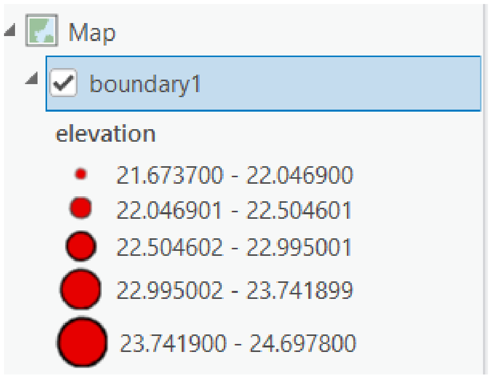
Example parameter extraction inside the toolbox:

```
csv_path = parameters[0].valueAsText
shapefile = parameters[1].valueAsText
raster_path = parameters[2].valueAsText
project_name = parameters[3].valueAsText
project2(project_name,csv_path,raster_path,shapefile)
```

This design enables full integration with ArcGIS Pro, eliminating the need for manual scripting once the tool is published.

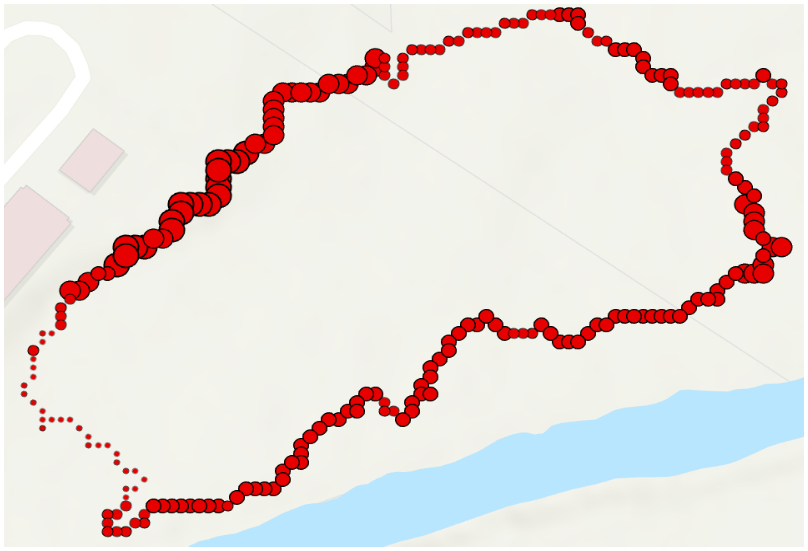
4. Results and Visualization

The workflow produced a shapefile in which every boundary point from the input CSV was successfully attributed with an elevation value extracted from the USGS 3DEP 10-meter digital elevation model via Google Earth Engine. The resulting dataset, *boundary1.shp*, contains the original coordinate fields along with a new numeric field, elevation, populated through automated sampling of the terrain model. A review of the attribute table confirmed one elevation value per point, with no missing or null records. The extracted elevations ranged approximately from 21.6 meters to 24.7 meters, reflecting modest topographic variation along the water boundary.

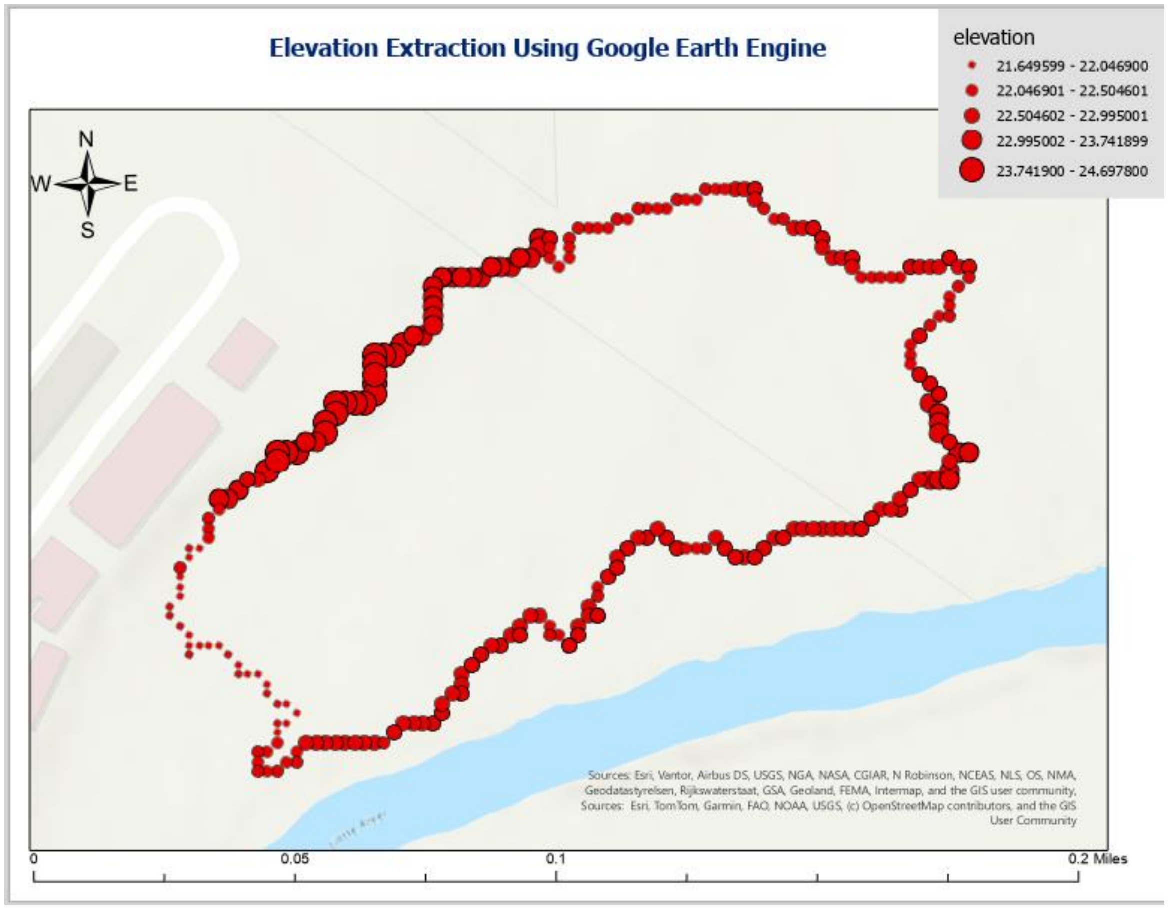

Map
boundary1
elevation

●	21.673700 - 22.046900
●	22.046901 - 22.504601
●	22.504602 - 22.995001
●	22.995002 - 23.741899
●	23.741900 - 24.697800

To visualize the results, the shapefile was added to an ArcGIS Pro map and symbolized using graduated symbols based on the elevation field. This representation effectively highlighted the relative elevation differences around the perimeter. Larger, brighter symbols corresponded to higher elevations, allowing a clear interpretation of spatial patterns. The symbology showed a gradual increase in elevation toward the northern portion of the boundary and lower values near the southern and eastern segments.



A formal layout was then created following standard cartographic practice. The layout included a title ("Elevation Extraction Using Google Earth Engine"), legend, north arrow, and scale bar. The map frame was adjusted to center the boundary points and provide appropriate visual balance. The final layout was exported as **Project2_Layout.pdf**, which is included in the GitHub repository as part of the project deliverables. Together, these visual outputs demonstrate that the script, toolbox interface, and Google Earth Engine integration performed as intended, producing an accurate and interpretable elevation-enhanced dataset.


Elevation Extraction Using Google Earth Engine
elevation
21.649599 - 22.046900
22.046901 - 22.504601
22.504602 - 22.995001
22.995002 - 23.741899
23.741900 - 24.697800
N
W
E
S
Sources: Esri, Vantor, Airbus DS, USGS, NGA, NASA, CGIAR, N Robinson, NCEAS, NLS, OS, NMA, Geodatastyrelsen, Rijkswaterstaat, GSA, Geoland, FEMA, Intermap, and the GIS user community, Sources: Esri, TomTom, Garmin, FAO, NOAA, USGS, (c) OpenStreetMap contributors, and the GIS User Community
0
0.05
0.1
0.2 Miles

5. Challenges and Future Work.

Several challenges emerged during the development and execution of the elevation-extraction workflow. A primary difficulty involved coordinating spatial references across different environments. The coordinates provided in the CSV file were based on the projection of the flood_2class.tif raster, requiring careful validation to ensure that the shapefile inherited the correct coordinate system. Any mismatch in CRS would have prevented the Google Earth Engine sampling step from returning accurate elevation values. Another challenge involved Earth Engine initialization inside the ArcGIS Pro Python environment. Because ArcGIS relies on a cloned conda environment, installing and authenticating the Earth Engine API required additional configuration before the tool could run successfully.

A second difficulty involved converting exploratory notebook code into a fully automated script and then integrating that script into a Python Toolbox. Each environment behaved slightly differently, for example, attribute-writing functions and cursor operations in ArcPy required adjustments when transitioning from the notebook to the .py file and then to the .pyt interface. Ensuring that the tool accepted user inputs and produced valid shapefiles without manual intervention required iterative debugging.

Future improvements could make the tool more flexible and user-friendly. One enhancement would be to support additional elevation sources or allow the user to select the Earth Engine dataset directly within the toolbox interface.

6. References

- Wang, L. (2025). GEOG 4057 – GIS Programming Course Repository. GitHub. https://github.com/leiwanglsu/geog4057
- Google Earth Engine Developers. (2024). *Earth Engine Python API Documentation.* https://developers.google.com/earth-engine/guides/python_install
- U.S. Geological Survey. (2024). *USGS 3D Elevation Program (3DEP) 10m Digital Elevation Model.* https://www.usgs.gov/3d-elevation-program