Aleta March
amarch2u@hotmail.com
GEOG 375, American River College
Final Project, Spring 2020

Summary
This project demonstrates the use of a stand-alone Python script to retrieve real-time GeoJSON data from the USGS Earthquake Hazards website and utilize the data to create and export maps showing the location and magnitude of significant earthquakes worldwide.

Purpose
The purpose of this project was to utilize Python and ArcPy scripting to update a cumulative database of global earthquake events using data retrieved from the USGS Earthquake Hazards Program real-time GeoJSON Summary Format Significant Earthquakes feeds and produce and export to PDF format a series of maps comprising:

1) a global map of the location of earthquake events that occurred in the time frame of the real-time feed and the time frame of the cumulative database
2) local maps of the location of each individual earthquake event that occurred during the time frame (day, week, month) covered by the GeoJSON feed selected by the user.

Data Source
Raw data for this project were obtained from the USGS Earthquake Hazards website https://earthquake.usgs.gov/earthquake/feed/v1.0/geojson.php which provides three real-time Significant Earthquake Real-time Feeds updated every minute:

"USGS Significant Earthquakes, Past Day"
https://earthquake.usgs.gov/earthquakes/feed/v1.0/summary/significant_day.geojson
"USGS Significant Earthquakes, Past Week"
https://earthquake.usgs.gov/earthquakes/feed/v1.0/summary/significant_week.geojson
"USGS Significant Earthquakes, Past Month"
https://earthquake.usgs.gov/earthquakes/feed/v1.0/summary/significant_month.geojson

These "GeoJSON Summary Format" feeds use the GeoJSON format, based on the JavaScript Object Notation (JSON) standard, to encode point geographic data structures (Crockford; "GeoJSON").

Methods
Pre-script Requirements
In order for the Python script to function, a file geodatabase must be available to store, query, and manage spatial and non-spatial data ("What is a File Geodatabase?"). Two file geodatabases were created on disk from an ArcGIS Pro 2.4 project: one to handle the cumulative database and feature class located at C:\temp\Final_Project\Data\eq_base.gdb and one to handle in real-time table and feature class located at C:\temp\Final_Project\MyData\eq_output.gdb ("Create a File Geodatabase").

Geoprocessing

Data transformation and information flow from one geoprocess to another was a particularly important aspect of the script. Part I of the script governs the retrieval and extraction of data and management of the tables and feature classes. Part II controls map creation and export.

Part I: Data Management

The information flow begins with the retrieval of the GeoJSON data from the USGS website (Severance 158-160, 163-165; “JSON Encoding”)

```
import json
import urllib.request, urllib.parse, urllib.error
#  Open connection "url handle" to URL
urlhand = urllib.request.urlopen(url)
# Read data as bytes and convert to JSON string
data = urlhand.read().decode()
print('Retrieved ', len(data), 'characters')
# Convert (deserialize) JSON string to Python dictionary
jsdata = json.loads(data)
# Convert to JSON string, then "print it pretty" to examine data structure
print(json.dumps(jsdata, indent=4))
```

The GeoJSON data is printed out in an easy-to-read format similar to a Python dictionary:


```
{
  "features": [
    {
      "type": "Feature",
      "properties": {
        "mag": 6.3,
        "place": "132km W of Panguna, Papua New Guinea",

      },
      "geometry": {
        "type": "Point",
        "coordinates": [
          154.3008,
          -6.5105,
          16.85
        ]
      },
      "id": "us60009c06"
    },
```


Next, the data is extracted using indexing:

- magnitude = entry[[‘properties’][‘mag’]
- place = entry[‘properties’][‘place’]
- lon = entry[‘geometry’][‘coordinates’][0]
- lat = entry[‘geometry’][‘coordinates’][1]

The extracted data may have to be transformed if the format and type differ from that required by the table fields. The following is a good example of a complicated transformation of the earthquake event time to conventional date and time.

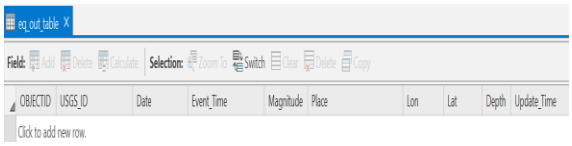
```
# Event origin time (UTC) expressed in milliseconds
# from midnight 01-01-1970
time = entry['properties']['time']
print('UNIX Epoch Time:', time)
print(type(time))

epochtime = datetime.datetime(1970, 1, 1) +
    datetime.timedelta(milliseconds = time)
print('Date and Time (UTC):', epochtime)
print(type(epochtime))

# Convert UNIX epoch time to string
str_epoch = str(epochtime)
#print('Epoch time as a string:', str_epoch)

# Strip date from epoch time string
date = str_epoch[:10]
print('Date (UTC):', date)
print(type(date))

# Strip event origin time from epoch time string
pos = str_epoch.find('')
event_time = str_epoch[pos + 11:]
print('Event Origin Time (UTC):', event_time)
print(type(event_time))
```

```
UNIX Epoch Time: 1587539027740
<class 'int'>
Date and Time (UTC): 2020-04-22 07:03:47.740000
<class 'datetime.datetime'>
Date (UTC): 2020-04-22
<class 'str'>
Event Origin Time (UTC): 07:03:47.740000
<class 'str'>
```

Let the Geoprocessing Begin!

arcpy.CreateTable_management
arcpy.AddField_management

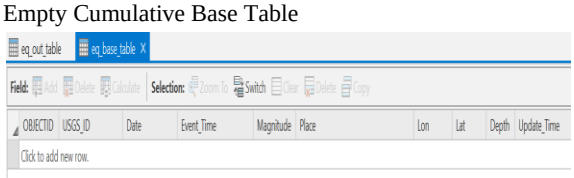
Empty Real-time Output Table




```
# USGS GeoJSON id value
arcpy.AddField_management(eq_out_tbl_path, 'USGS_ID','TEXT', '', '', 15)
# Date(string)derived from USGS GeoJSON time value
arcpy.AddField_management(eq_out_tbl_path, 'Date','TEXT', '', '', 10)
# USGS Event time(string)derived from USGS GeoJSON time value
arcpy.AddField_management(eq_out_tbl_path, 'Event_Time','TEXT', '', '', 15)
# USGS GeoJSON mag value
arcpy.AddField_management(eq_out_tbl_path, 'Magnitude','FLOAT', '4', '2', '')
# USGS GeoJSON place value
arcpy.AddField_management(eq_out_tbl_path, 'Place','TEXT', '', '', 50)
# USGS GeoJSON coordinates[1] value
arcpy.AddField_management(eq_out_tbl_path, 'Lon','FLOAT', 12, 9, '')
# USGS GeoJSON coordinates[2] value
arcpy.AddField_management(eq_out_tbl_path, 'Lat','FLOAT', 11, 9, '')
# USGS GeoJSON coordinates[3] value
arcpy.AddField_management(eq_out_tbl_path, 'Depth','FLOAT', 5, 2, '')
# USGS GeoJSON feed update time value(milliseconds since 1970.01.01 00:00:00)
arcpy.AddField_management(eq_out_tbl_path, 'Update_Time','FLOAT', 14, 0, '')
print('Created table and added fields')
```

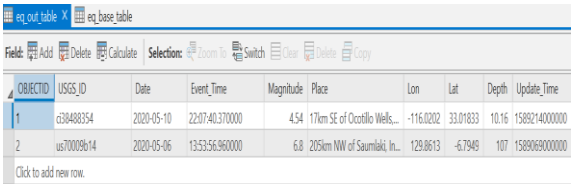
[Create base table if it does not exist using arcpy.Copy_management]

Empty Cumulative Base Table



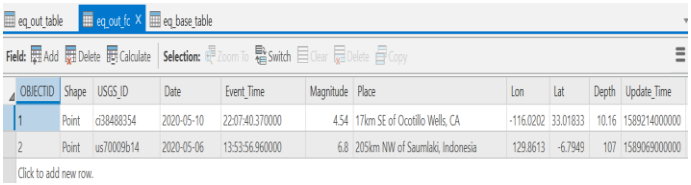
InsertCursor
InsertRow

Populated Real-time Output Table



arcpy.MakeTableView_management
arcpy.management.XYTableToPoint

Populated Real-time Feature Class



arcpy.da.SearchCursor
arcpy.da.UpdateCursor

Base Table with Updated Rows

```
with arcpy.da.SearchCursor(eq_out_tbl_path, field_list) as srows:
    with arcpy.da.UpdateCursor(base_table, field_list) as uprows:
        search_loop_counter = 0 #Shows current cycle of search loop through real-time feed table

        for srow in srows:      # Assign search row field values used for comparison
            rt_feed_id = srow[0]       # USGS_ID field value
            rt_feed_updated = srow[8]  # Update_Time field value
            print('Search loop cycle', search_loop_counter)

            if base_row_count > 0: # If base table filled, compare entries with real-time feed entries
                for uprow in uprows:
                    found = False
                    # Set field values to be compared
                    base_id = uprow[0]
                    base_updated = uprow[8]

                    if base_id == rt_feed_id: # If a match is found
                        found = True

                        # Check if more recent update available; update base table if so
                        if base_updated < rt_feed_updated:
                            uprows.updateRow(srow)
                            print(base_id, 'base table updated')

                            # Add event row values to updated_event_rows list
                            updated_event_rows.append(srow)
                            print('USGS_ID', rt_feed_id, 'row values added to updated_event_rows list')

                    # If match found, end search
                    if found == True:
                        uprows.reset() # Cursor reset to first row for next loop
                        print('Break out of loop')  # Avoid unnecessary further search
                        break

            if found == False or base_row_count == 0: # If match not found in permanent db or db empty
                print('Event not found in permanent database.')
                # Add to end of list
                new_event_rows.append(srow)
                print('USGS_ID', rt_feed_id, 'event row values added to new_event_rows list')
                print(srow)
                uprows.reset()
            search_loop_counter = search_loop_counter + 1
```

Produces Two Lists of Row Tuples

updated_event_rows

new_event_rows

These lists are key to the remainder script. They contain the values of rows that will be updated or added to the base table and feature class.

arcpy.da.InsertCursor
InsertRow

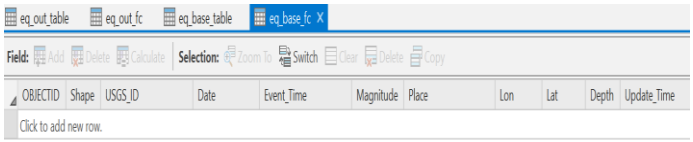
→ Base Feature Table with New Event Rows Added

Sample Row Tuple

('us70008jr5', '2020-03-31', '23:52:31.094000', 6.5,
'72km W of Challis, Idaho', -115.1355972290039,
44.46030044555664, 14.529999732971191, 1587859423232.0)

[Create base feature class
if it does not exist using
arcpy.management.XYTableToPoint]

→ Base Feature Class



arcpy.da.UpdateCursor
UpdateRow

→ Base Feature Class with Updated Rows

Sample Row Tuple

((-115.1355972290039, 44.46030044555664), 'us70008jr5', '2020-03-31',
'23:52:31.094000', 6.5, '72km W of Challis, Idaho', -115.1355972290039,
44.46030044555664, 14.529999732971191, 1587859423232.0)

NOTE: inserting or updating a row in a feature class is different than in a table because a geometry must be written.

1) The field list must include the 'Shape' field in the initial position
2) The 'SHAPE@XY' geometry token may be used in place of 'Shape' in the field list
3) The row tuple to be inserted must have a Point value tuple (lon, lat) in the initial position to populate the 'Shape' field (see highlighted tuple above).

```
# Create update cursor for base feature class; replace 'Shape' field with 'SHAPE@XY' to add 'Point' geometry

with arcpy.da.UpdateCursor(base_fc, ['SHAPE@XY', 'USGS_ID', 'Date', 'Event_Time', 'Magnitude', 'Place', 'Lon', 'Lat', 'Depth', 'Update_Time']) as uprows:
    index = 0  #Counter for rows to be updated

    for uprow in uprows:

        for index in range(0,total_updates):
            # If USGS_ID in base_fc matches the ID in the tuple of the updated_event_rows list, then update row
            if uprow[1] == updated_event_rows[index][0]:
                # Create new tuple with xy point value as first element (needed to add Point geometry to 'Shape' field)
                xy = (updated_event_rows[index][5], updated_event_rows[index][6])  # tuple represents (Lon, Lat) values

                # Create new update row tuple by appending the row elements of updated_event_rows to xy geometry tuple
                new_tup = (xy,)

                for i in range(0,9):
                    new_tup = new_tup + (updated_event_rows[index][i],)

                # Update row of base_feature class with element (tuple) of new_tup
                uprows.updateRow(new_tup)
                print('Updated row in base feature class', new_tup, '\n')

    del uprow
del uprows
```

arcpy.da.InsertCursor
InsertRow

Base Feature Class with New Event Rows Added

((-85.70989990234375, 16.93320083618164), 'us70008xeb', '2020-04-16', '08:04:37.597000', 6.0, '55km NNE of Savannah Bight, Honduras', -85.70989990234375, 16.93320083618164, 10.0, 1587751813120.0)

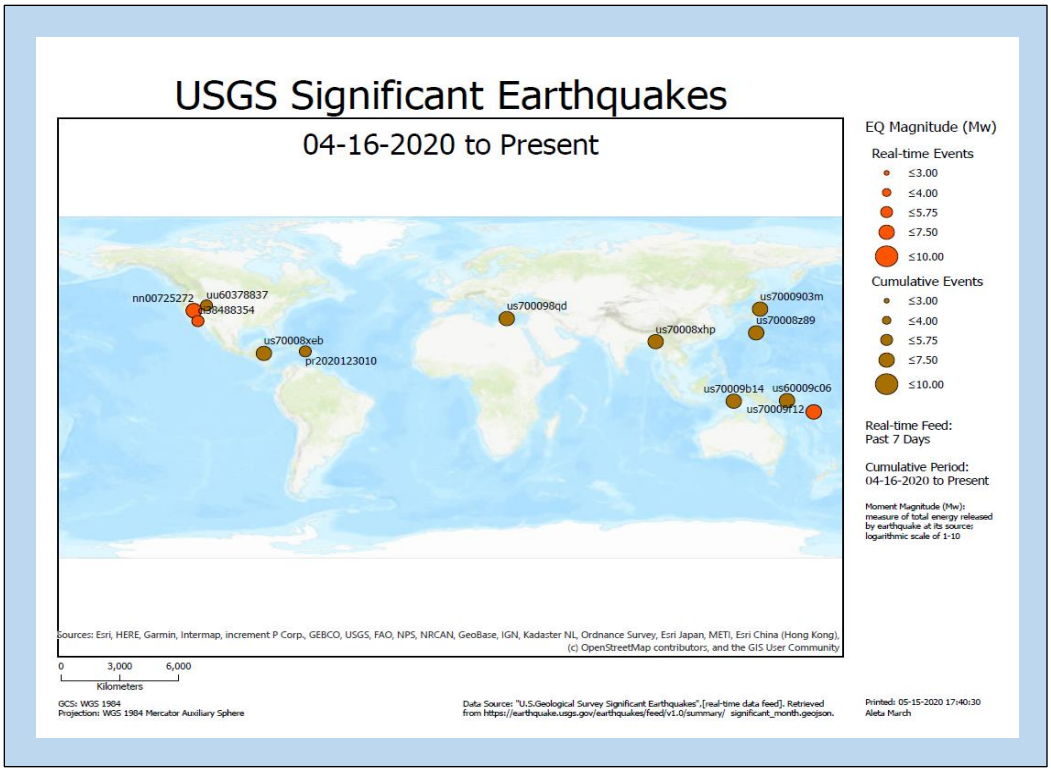
Part II: Map Creation and Export

This segment of the script is responsible for the production and export of maps showing the location of earthquake events.

Output maps are variations of two types, global and local:

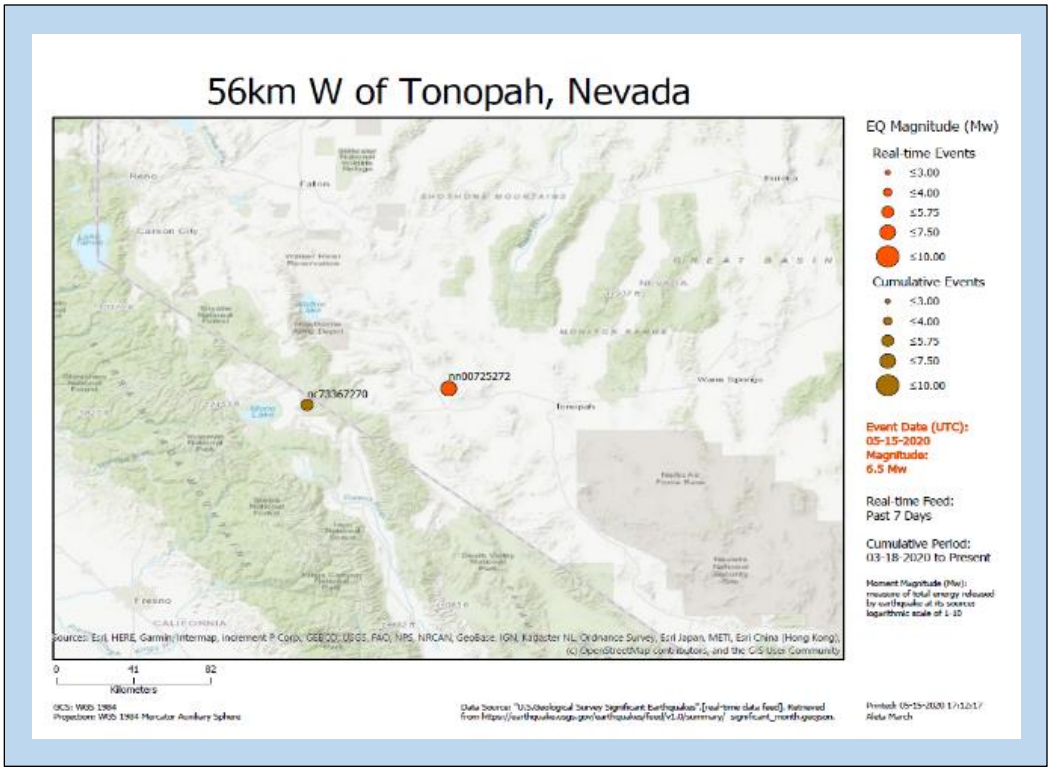
1) one global map at full map extent showing the cumulative base layer overlaid by the real-time layer produced by the 1, 7, or 30-day real-time feed) with graduated point symbols based on earthquake event magnitude
2) a group of local maps equivalent to the number of real-time events. Individual maps each show a zoomed-in view of one real-time event, along with any previous events within the map frame extent.

Geoprocessing is scant in this section. The arcpy.mp module is used to modify existing layout elements, visibility of map frame layers, and map frame extents prior to the export of maps in PDF format. The global map is printed at the full geographic extent of the base map.

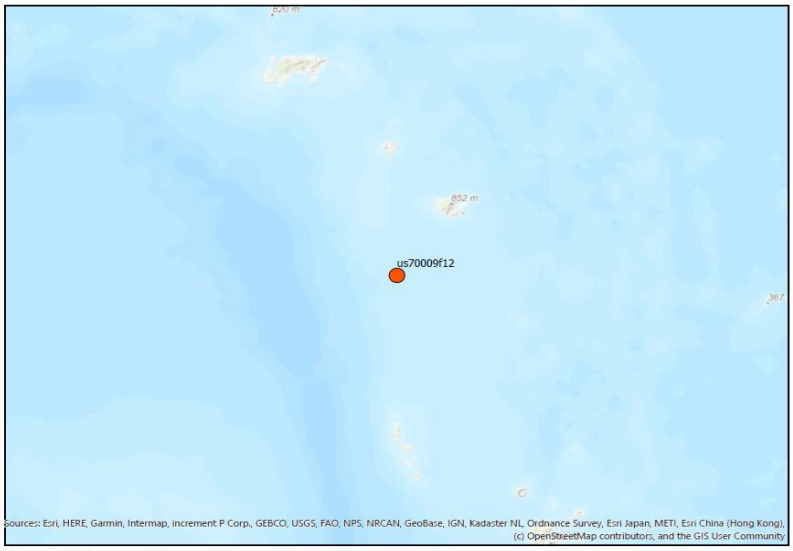


A multi-step process is required to perform a zoom-in effect on an individual real-time event. First, SelectByAttribute with a selection query produces a subset of the real-time layer and GetExtent returns a derived extent ("Zoom Map"). Extent and scale of the map frame is then controlled by the Camera object, mimicking zooming-in to a feature ('Camera").

```
# Select current row for definition query
query = """"USGS_ID" = '""" + rt_USGS_ID +"""'"""
arcpy.SelectLayerByAttribute_management(rt_lyr, "NEW_SELECTION", query)
rt_lyr.definitionQuery = query
print(rt_USGS_ID, 'selected to map')

# Get/Set layerExtent for selected row
layer_extent = mapframe.getLayerExtent(rt_lyr, True, True)
print('Layer extent rt_layer:'  + str(layer_extent))

# Set mapframe camera to layer extent
mapframe.camera.setExtent(layer_extent)
# Set mapframe scale (create zoom-in effect)
mapframe.camera.scale = 2400000.00
```



173km SSE of Lata, Solomon Islands



EQ Magnitude (Mw)

Real-time Events

- ≤3.00
- ≤4.00
- ≤5.75
- ≤7.50
- ≤10.00

Cumulative Events

- ≤3.00
- ≤4.00
- ≤5.75
- ≤7.50
- ≤10.00

Event Date (UTC):
05-12-2020
Magnitude:
6.6 Mw

Real-time Feed:
Past 7 Days

Cumulative Period:
04-16-2020 to Present

Moment Magnitude (Mw): measure of total energy released by earthquake at its source; logarithmic scale of 1-10

GCS: WGS 1984
Projection: WGS 1984 Mercator Auxiliary Sphere

Data Source: "U.S.Geological Survey Significant Earthquakes",[real-time data feed]. Retrieved from https://earthquake.usgs.gov/earthquakes/feed/v1.0/summary/ significant_month.geojson.

Printed: 05-15-2020 17:40:30
Aleta March

38km ESE of Falam, Burma

EQ Magnitude (Mw)

Real-time Events

- ≤3.00
- ≤4.00
- ≤5.75
- ≤7.50
- ≤10.00

Cumulative Events

- ≤3.00
- ≤4.00
- ≤5.75
- ≤7.50
- ≤10.00

Event Date (UTC):
04-16-2020
Magnitude:
5.9 Mw

Real-time Feed:
Past 30 Days

Cumulative Period:
04-16-2020 to Present

Moment Magnitude (Mw): measure of total energy released by earthquake at its source; logarithmic scale of 1-10

GCS: WGS 1984
Projection: WGS 1984 Mercator Auxiliary Sphere

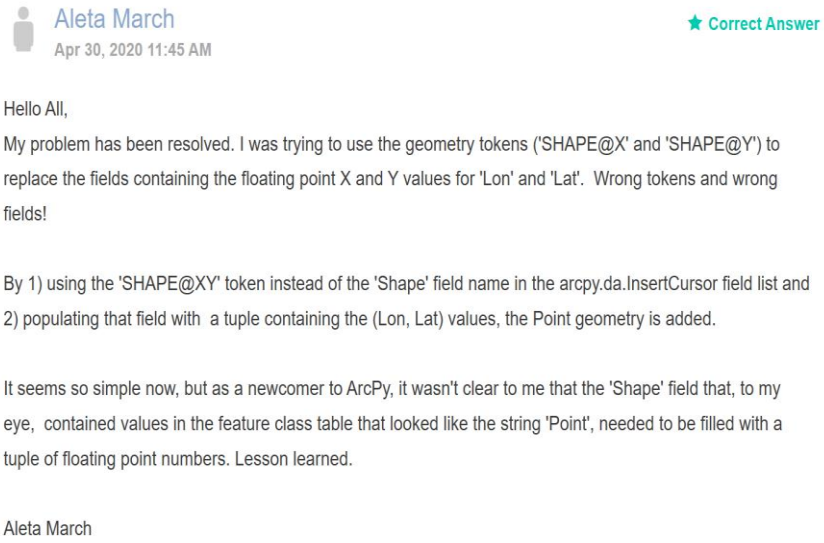
Data Source: "U.S.Geological Survey Significant Earthquakes",[real-time data feed]. Retrieved from https://earthquake.usgs.gov/earthquakes/feed/v1.0/summary/ significant_month.geojson.

Printed: 05-15-2020 17:27:13
Aleta March

Challenges / Solutions

Much of the basic geoprocessing code related to the use of insert, search, and update cursors, selection queries, and map production was taken directly from the video lectures given by Professor Jennings this semester (Jennings, Video Lectures) and from the workbooks he authored (Jennings, *Python Primer 1, 2, 3*). Early in the process of creating the script, I relied heavily on these coding templates, to the point of using some of the same variable names, until I gained familiarity with the concepts. As I progressed in my learning, I was able to customize and build on these scripts, incorporating additional processes using original code.

Many programming issues resulted from my inexperience using Python and were eventually overcome. One problem that was resolved after considerable effort pertained to the use of the geometry token 'SHAPE@XY' when adding a feature to a feature class ("How To: Add Features"). Silver lining: it gave me the opportunity to post something on GeoNet ("Add Row").



Aleta March

Apr 30, 2020 11:45 AM

★ Correct Answer

Hello All,

My problem has been resolved. I was trying to use the geometry tokens ('SHAPE@X' and 'SHAPE@Y') to replace the fields containing the floating point X and Y values for 'Lon' and 'Lat'. Wrong tokens and wrong fields!

By 1) using the 'SHAPE@XY' token instead of the 'Shape' field name in the arcpy.da.InsertCursor field list and 2) populating that field with a tuple containing the (Lon, Lat) values, the Point geometry is added.

It seems so simple now, but as a newcomer to ArcPy, it wasn't clear to me that the 'Shape' field that, to my eye, contained values in the feature class table that looked like the string 'Point', needed to be filled with a tuple of floating point numbers. Lesson learned.

Aleta March

At the conclusion of the project, three problems remained unresolved. The first two issues are related to basic ArcPy programming concepts and the third may be specific to either the input data format or the program logic. In addition, a deficiency with respect to exception code was noted.

PROBLEM 1**:** Unable to Control the Position of Rows Inserted into a Table

As a consequence of the manner in which the arcpy.da.InsertCursor and InsertRow functions operate, new rows are inserted at the bottom of a table. This results in the creation of the cumulative base table with the oldest event at the top of the table and the newest event at the bottom, preserving the order of the table as it accumulates rows. The disadvantage is that it increases the time it takes to search the existing base table, to determine the necessity of an event addition or update. As the base table accumulates more rows, the search time will become increasingly longer. If the order were reversed, the comparison of the real-time events with the most recent events in the base table would be more efficient.

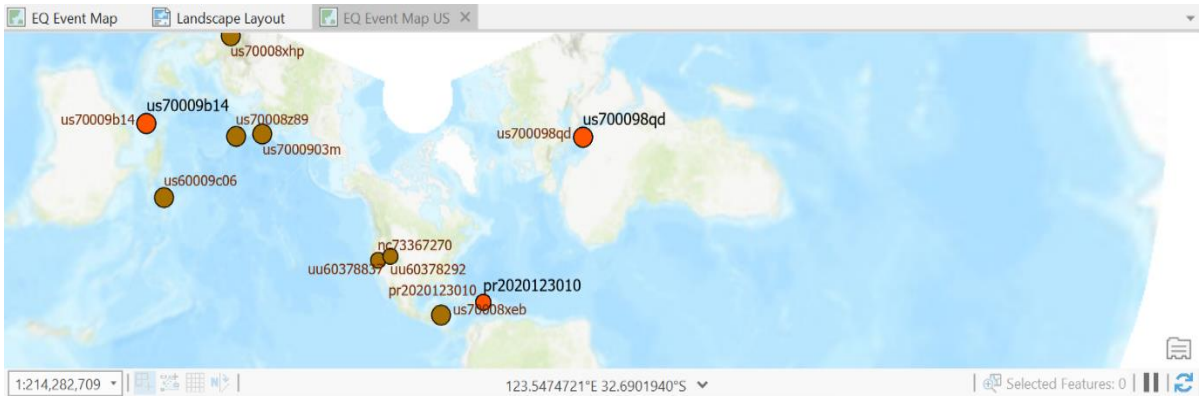
In my hunt for a solution, I tried to find a way to add a row to the top of the table or to begin a search at the bottom of the table. My attempts to find an elegant solution were futile. Perhaps, due to inexperience, my viewpoint was so narrow that I could not recognize the alternatives.

PROBLEM 2: Unable to Change Spatial Reference of Mapframe
It was my original intent to customize the spatial reference of the exported maps of individual events according to their geographic location and extent. Unfortunately, I was unable to do so, resulting in all maps being produced with WGS 1984 geographic coordinate system and WGS 1984 Mercator Auxiliary Sphere projection.

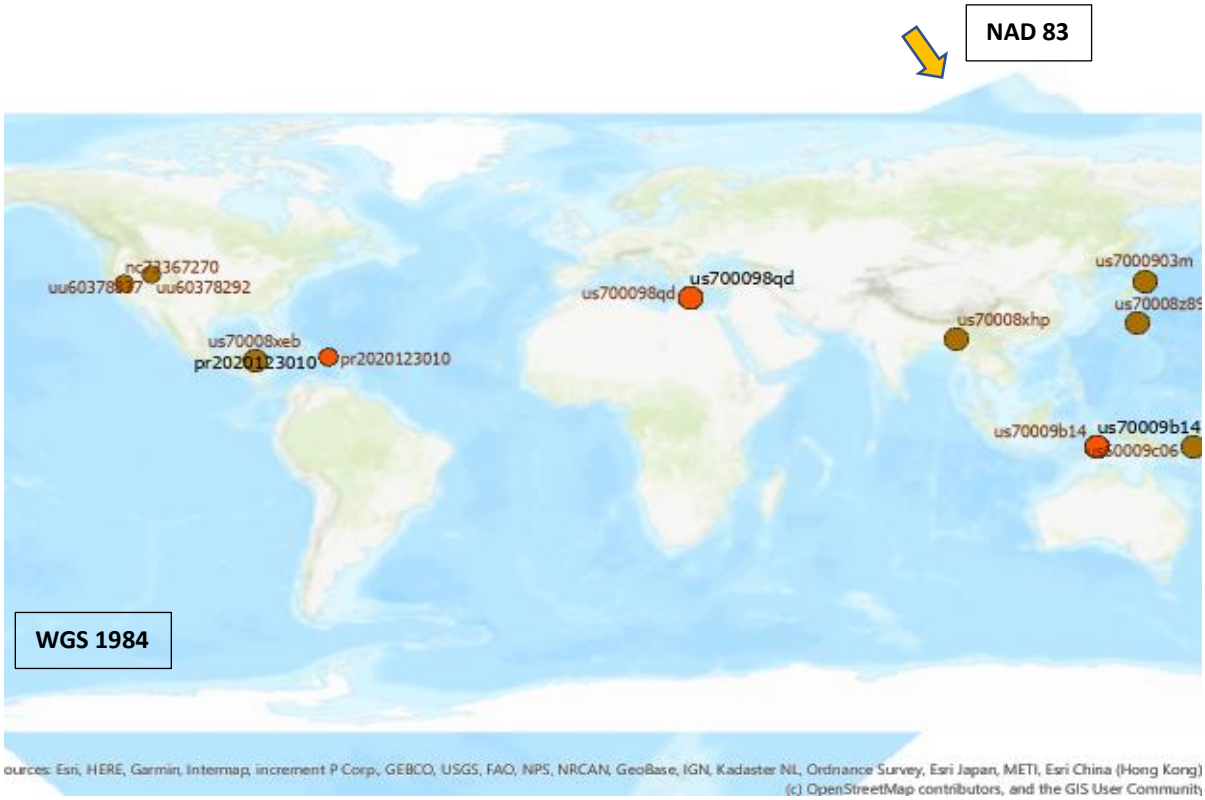
It is my understanding that arcpy.mp does not yet provide a way to change the spatial reference of mapframes or maps ("Allow arcpy.mp to Modify"). According to Jeff Barrette of the ESRI arcpy.mp team, that functionality will be available with ArcGIS Pro 2.6.

A potential workaround would be to make several maps within the ArcGIS Pro project, each with a different spatial reference and set up one layout to contain multiple mapframes. The Python script would then be used to select and make visible only the mapframe with the appropriate spatial reference for the extent of the exported map.

Pursuing this workaround proved beyond the time constraints of this project. I was able, however, to create one additional test map with a spatial reference of NAD 83 / USA Contiguous Lambert Conformal Conic, as shown below.

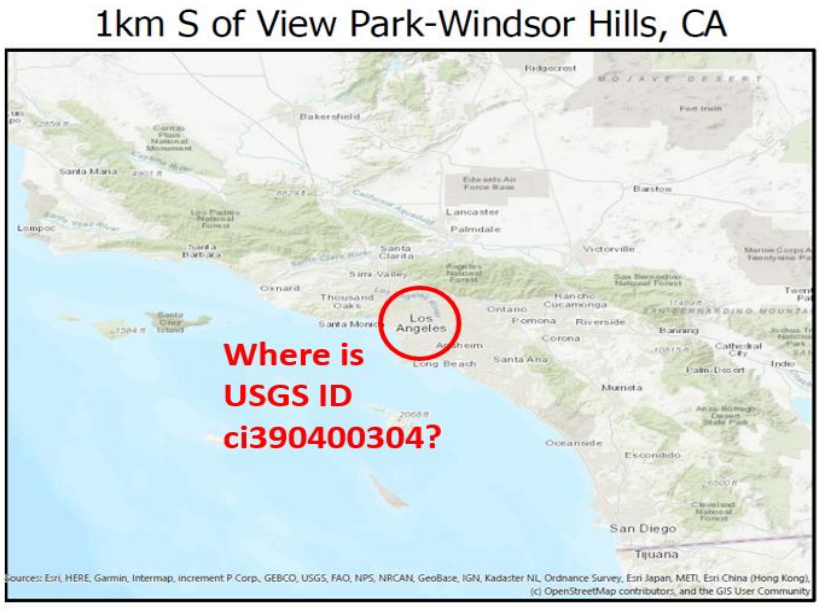


The corresponding mapframe was then added as a mapframe to the common layout, stacked below the original mapframe.



PROBLEM 3: Point Features Occasionally Missing from a Map

I have been unable to explain the infrequent (two occurrences) of a point feature missing from a map. The data format of the GeoJSON input data matches that of the extracted data output. There were no discernable differences between the rows of the faulty point features and the others.



DEFICIENCY: Custom Exception Code Not Included

Other than the Try:/Except statement surrounding the script, custom exception code was not created for the script (Jennings, "Demo 6D"). Custom exception code designed to handle errors caused by a faulty Internet connection or a request timing out would have been beneficial. The rare instances that triggered the runtime error, "Not signed into Portal", were due to an unstable or absent Internet connection. This was verified by intentionally turning off the Internet hotspot and running the script to re-create the error message. Although I did some preliminary research on the urllib.request and urllib.error submodules, I did not feel that I had sufficient knowledge to create the code required to raise a connection or timeout exception ("urllib.error"; "urllib.request").

Conclusion

This project has helped to change my perspective on programming for GIS. Prior to taking this course, I thought using Python was all about coding syntax and logic and troubleshooting and frustration. Well, it is, but it is also so much more! It involves knowing your data and understanding how the software for which you are coding functions "under the hood". I know that I have barely scratched the surface, but even this little bit of knowledge has given me a lot of power to learn more. It has been a long journey from printing "Hello World!" back in January.

References

"Add Row to Existing Point Feature Class". *GeoNet*, ESRI, community.esri.com/thread/252156-add-row-to-existing-point-feature-class.

"Allow arcpy.mp to Modify Spatial Reference Property of Map/MapFrame". *GeoNet*, ESRI, community.esri.com/ideas/12767-allow-arcpymp-to-modify-spatial-reference-property-of-mapmapframe.

"Camera". *ArcGIS Pro – ArcPy | Documentation*, ESRI, pro.arcgis.com/en/pro-app/arcpy/mapping/camera-class.htm

"Convert Python Datetime to Timestamp in Milliseconds." *Stack Overflow*, 20 Dec. 2017, stackoverflow.com/questions/41635547/. Accessed 12 May 2020.

"Create a File Geodatabase." *ArcGIS Pro Help*, ESRI, pro.arcgis.com/en/pro-app/help/data/geodatabases/manage-file-gdb/create-file-geodatabase.htm. Accessed 8 May 2020.

Crockford, Douglas. "Introducing JSON." *JSON*, json.org/json-en.html.

"Datetime — Basic Date and Time Types — Python 3.7.2 Documentation." *Python.Org*, Python Software Foundation, 2002, docs.python.org/3/library/datetime.html.

"GeoJSON." *GeoJSON*, geojson.org/.

"GeoJSON Summary Format." *USGS Earthquake Hazards*, United States Geological Survey, earthquake.usgs.gov/earthquakes/feed/v1.0/geojson.php.

"Guidelines for arcpy.mp." *ArcGIS Pro – ArcPy | Documentation*, ESRI, pro.arcgis.com/en/pro-app/arcpy/mapping/ guidelines-for-arcpy-mapping.htm#ESRI_SECTION1_ B482E78268684B728BFF0636AAFF40FD. Accessed 13 April 2020.

"How To: Add Features to a Feature Class Using Python." *ESRI Technical Support*, ESRI, 26 Nov. 2018, support.esri.com/en/technical-article/000018755.

Jennings, Nathan. *A Python Primer for ArcGIS® Workbook I*. 2015, *LRCCD Canvas ARC SP20 GEOG 375 Lecture 11189 Jennings*, lrccd.instructure.com/courses/74982.

Jennings, Nathan. *A Python Primer for ArcGIS® Workbook II*. 2015, *LRCCD Canvas ARC SP20 GEOG 375 Lecture 11189 Jennings*, lrccd.instructure.com/courses/74982.

Jennings, Nathan. *A Python Primer for ArcGIS® Workbook III*. 2015, *LRCCD Canvas ARC SP20 GEOG 375 Lecture 11189 Jennings*, lrccd.instructure.com/courses/74982.

Jennings, Nathan. "Demo 6D: Create and Use Table Joins." 21 Feb. 2020. GEOG 375, Spring 2020, American River College, Sacramento. Lecture materials, Module 3.

Jennings, Nathan. Video lectures, Modules 1-4. Geography 375. American River College, Sacramento, California. Spring 2020.

"JSON Encoding and Decoding with Python." *Pythonspot*, pythonspot.com/json-encoding-and-decoding-with-python/. Accessed 27 March 2020.

"7 Examples to Understand Python Strftime() [Datetime and Time Modules]." *A-Z Tech*, 22 Dec. 2018, www.jquery-az.com/python-strftime/. Accessed 12 May 2020.

Severance, Charles. *Python for Everybody: Exploring Data Using Python 3*. CC ShareAlike 3.0, 5 July 2016, do1.dr-chuck.com/pythonlearn/EN_us/pythonlearn.pdf.

"urllib.error - Exception Classes Raised by Urllib.request." *Python 3.8.3 Documentation*, Python Software Foundation, docs.python.org/3/library/urllib.error.html.

"urllib.request – Extensible Library for Opening URL's." *Python 3.8.3 Documentation*, Python Software Foundation, docs.python.org/3/library/urllib.request.html#module-urllib.request.

"USGS Significant Earthquakes, Past Day" (Version 1.0)[real-time data feed]. US Geological Survey. Retrieved from https://earthquake.usgs.gov/earthquakes/feed/v1.0/summary/ significant_day.geojson. Accessed May 2020.

"USGS Significant Earthquakes, Past Week" (Version 1.0)[real-time data feed]. U.S. Geological Survey. Retrieved from https://earthquake.usgs.gov/earthquakes/feed/v1.0/summary/significant_week.geojson. Accessed May 2020.

"USGS Significant Earthquakes, Past Month" (Version 1.0)[real-time data feed]. U.S. Geological Survey. Retrieved from https://earthquake.usgs.gov/earthquakes/feed/v1.0/summary/significant_month.geojson. Accessed April, May 2020.

“What Is a File Geodatabase?” *ArcGIS Pro Help*, ESRI, 13 Dec. 2019, pro.arcgis.com/en/pro-app/help/data/geodatabases/manage-file-gdb/file-geodatabases.htm. Accessed 8 April 2020.

“Zoom Map to the Extent of a Layer in arcpy.mp”. *GeoNet*, ESRI, community.esri.com/thread/194310-zoom-map-to-an-layer-in-arcpymp-arcgis-pro-project. Accessed 13 April 2020.