Rapid Geodata Automation with Python Multiprocessing, ArcPy & NumPy
"Building relationships one project at a time."

Sacramento -- Pleasanton -- Oakland -- San Dimas
Reno -- Las Vegas
Concepts

- ArcPy Geodata Automation
- Python Multiprocessing
- NumPy with ArcPy
Use Case

- Water Resources
  - Predictive flood modeling
  - The models require geometry for storm drain infrastructure
  - Cross Sections of the Channels are needed for the flood modeling
Automated Channel Cross Sections from Survey Data Points

1. Input Survey Points from field data collection.

Program: Create Best Fit Line for each cross section using linear regression. Best fit lines are then rotated 90 degrees from the given channel.

2. Program: Create Cross Section Profiles to allow the engineer to compare survey vs. DEM elevations. Engineer chooses which points to bring into the geodatabase.

A PDF with the black line indicating the final profile is generated.

3. The cross section line feature class and normalized point data is output for review by the engineer.
Concepts

ArcPy Geodata Automation
ArcPy GeoData Automation

- For Cross Section Tool:

  - Get x,y of all Survey Points
  - Use Linear Regression to generate a Best Fit Line
  - Snap/Normalize Survey Points to Best Fit Line
ArcPy GeoData Automation

- **ArcPy Geometry Classes**
  - Point, Polyline, Polygon

- **Data Access Cursor**
  - Can access and update all fields, including geometry using arcpy.da cursors
  - Search, Insert, Update
ArcPy GeoData Automation

- ArcPy Data Access Cursor
  - General structure:
    ```python
    rows = arcpy.da.UpdateCursor(os.path.join(gdb, "temp_BestFitLine"), field_names = [txtfld, "SHAPE\", "Rotation"], where_clause = "")
    for row in rows:
        print(row)
    ```
  - row[1] = `<geoprocessing describe geometry object object at 0x022C3760>`
  - row[1].firstPoint will return a Point (not PointGeometry)

- Fields: 0 1 2
  - Cursor Type
  - Feature class, Layer, Table
  - Fields
  - Properties:
ArcPy GeoData Automation

- ArcPy Geometry – the easy way

```python
rows = arcpy.da.UpdateCursor(os.path.join(gdb, "temp_BestFitLine"), field_names = [txtfld, "SHAPE@", "Rotation"], where_clause = "")
for row in rows:
    arrayXY = arcpy.Array()
    arrayXY.append(arcpy.Point(extFirstPointX, extFirstPointY))
    arrayXY.append(arcpy.Point(extLastPointX, extLastPointY))
    pl = arcpy.Polyline(arrayXY)
    row[1] = pl
    rows.updateRow(row)
```
ArcPy GeoData Automation

- Gotchas
  - Cursors
    - SQL/Where clause and fields
    - Use with statement for automatic cleanup
    - Some situations require you to start an edit session
  - Arcpy.Point vs. Arcpy.PointGeometry
Python Multiprocessing
Python Multiprocessing

- Parallel processing
  - Multithreading
    - Runs multiple functions at the same time, using the same process
  - Multiprocessing
    - Utilizes multiple processors on your machine
    - Good for functions that must be iterated over a large dataset
Python Multiprocessing

- Code Structure
Python Multiprocessing

- **Gotchas**
  - Anything outside of the main() function will be run each time a process is started
  - In Script tool must uncheck Run Python Script in process
- **Global vars**
  - Copied to own memory space for process
  - Use Multiprocessing Array, Variable, or Queue to use shared memory space
Time Savings for Best Fit Lines

Small Dataset (17,000 Points)

Large Dataset (68,000 Points)
Time Savings for Normalizing Points

Small Dataset (17,000 Points)

Large Dataset (68,000 Points)
Concepts

Numpy with ArcPy
NumPy with ArcPy

- Use arcpy.da.FeatureClassToNumPyArray
- Faster processing of x,y geometry versus arcpy.da.SearchCursor
Time Savings with NumPy

- Uniformly 50% time savings to sort and identify First/Last points

<table>
<thead>
<tr>
<th>Dataset Size</th>
<th>Time with NumPy (in minutes)</th>
<th>Time with ArcPy Cursors (in minutes)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Small Dataset (17,000 Points)</td>
<td>0.4</td>
<td>1</td>
</tr>
<tr>
<td>Medium Sized Dataset (34,000 Points)</td>
<td>1.4</td>
<td>2.7</td>
</tr>
<tr>
<td>Large Dataset (68,000 Points)</td>
<td>2.5</td>
<td>5.3</td>
</tr>
</tbody>
</table>
Conclusion

- **ArcPy Geodata Automation**
  - Saves you hours of manual digitizing
  - Creates standardized output

- **Python Multiprocessing**
  - Up to 97% time savings for larger datasets
  - Overhead involved with multiprocessing

- **NumPy with ArcPy**
  - 50% time savings
Questions?

Solana Foo
GIS Analyst
sfoo@woodrodgers.com