Python: Beyond the Basics
Brittney White, Esri
Jordan Douthitt, Pennsylvania State University
Topics Covered

- Cursors
- Geometry objects
- Leveraging other Python modules
- User presentation: Location-Based Analysis for Recruitment of United States Border Patrol Agents
Cursors

• Used to:
  - Iterate over the set of rows in a table
  - Insert new rows into a table

• Two varieties:
  - *arcpy.da cursors* (10.1 onwards; significantly faster performance)
  - “Classic” cursors (provided only for continuing backward compatibility)
Data Access Module Cursors

Search (da.SearchCursor)

Update (da.UpdateCursor)

Insert (da.InsertCursor)
Required Arguments

- Table
  - The feature class, layer, table, or table view

- Fields
  - Single field or list of field names
  - Index position in fields parameter defines value access

```
#          0     1     2
fields = ["Name", "Year", "Count"]
```
Used as shortcuts in place of field names

- **OID@** — The value of the ObjectID field.
- **SHAPE@** — A geometry object for the feature.

- **SHAPE@XY** — A tuple of the feature's centroid x,y coordinates.
- **SHAPE@TRUECENTROID** — A tuple of the feature's true centroid x,y coordinates.
- **SHAPE@X** — A double of the feature's x-coordinate.
- **SHAPE@Y** — A double of the feature's y-coordinate.
- **SHAPE@Z** — A double of the feature's z-coordinate.
- **SHAPE@M** — A double of the feature's m-value.
- **SHAPE@JSON** — The esri JSON string representing the geometry.
- **SHAPE@WKB** — The well-known binary (WKB) representation for OGC geometry. It provides a portable representation of a geometry value as a contiguous stream of bytes.
- **SHAPE@WKT** — The well-known text (WKT) representation for OGC geometry. It provides a portable representation of a geometry value as a text string.
- **SHAPE@AREA** — A double of the feature's area.
- **SHAPE@LENGTH** — A double of the feature's length.
arcpy.da.SearchCursor

```python
arcpy.da.SearchCursor(in_table, field_names, {where_clause}, {spatial_reference}, {explode_to_points}, {sql_clause})
```
arcpy.da.SearchCursor – optional where clause parameter
# Open a Search Cursor, print results

```python
table = "Crime_Incidents_2016"
fields = ["Shift", "Offense", "Method", "Ward"]
qry = " Ward = "6" "
with arcpy.da.SearchCursor(table, fields, qry) as cursor:
    for row in cursor:
        print("Shift: {} Offense:{} Method: {} Ward: {}".format(row[0], row[1], row[2], row[3]))
```

![Image of a table of crime incidents in 2016](image-url)
# Open a Search Cursor, print results

table = "Crime_Incidents_2016"
fields = ["Shift", "Offense", "Method", "Ward"]
qry = " Ward = '6' "
with arcpy.da.SearchCursor(table, fields, qry) as cursor:
    for row in cursor:
        print("Shift: {} Offense:{} Method: {} Ward: {}".format(row[0], row[1], row[2], row[3]))
arcpy.da.SearchCursor – optional spatial reference parameter
# Open a Search Cursor with a where clause, print results

table = "Crime_Incidents_2016"
fields = ["Offense", "SHAPE@XY"]
qry = " Ward = '6' "

# WKID: 2248 for NAD_1983_StatePlane_Maryland_FIPS_1900_Feet
sro = arcpy.SpatialReference(2248)

with arcpy.da.SearchCursor(table, fields, qry, sro) as cursor:
    for row in cursor:
        print("Offense:{} Coordinates: {}".format(row[0], row[1]))
arcpy.da.UpdateCursor

```
arcpy.da.UpdateCursor(in_table, field_names, {where_clause}, {spatial_reference}, {explode_to_points}, {sql_clause})
```

<table>
<thead>
<tr>
<th>Method</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>deleteRow ()</td>
<td>Deletes the current row</td>
</tr>
<tr>
<td>updateRow (row)</td>
<td>Updates the current row in the table.</td>
</tr>
<tr>
<td>S</td>
<td>CLASS</td>
</tr>
<tr>
<td>---</td>
<td>-------</td>
</tr>
<tr>
<td>12400</td>
<td>00</td>
</tr>
<tr>
<td>26</td>
<td>401</td>
</tr>
<tr>
<td>11</td>
<td>URS/MHA</td>
</tr>
<tr>
<td>12</td>
<td>URS/MHA</td>
</tr>
<tr>
<td>23</td>
<td>PARKSIDE</td>
</tr>
<tr>
<td>46</td>
<td>RS-3.5</td>
</tr>
<tr>
<td>25</td>
<td>RS-3.5</td>
</tr>
<tr>
<td>23</td>
<td>RS-5</td>
</tr>
<tr>
<td>37</td>
<td>RS-5</td>
</tr>
<tr>
<td>73</td>
<td>RR-5</td>
</tr>
</tbody>
</table>

**arcpy.da.UpdateCursor**
arcpy.da.UpdateCursor

```python
# Add a new field named ACRES to the Parcel feature class
arcpy.AddField_management("Parcel", "ACRES", "Double")

# Update ACRES field using the SHAPE@AREA token
with arcpy.da.UpdateCursor("Parcel", ["SHAPE@AREA", "ACRES"]) as cursor:
    for row in cursor:
        row[1] = row[0] / 43560
    cursor.updateRow(row)
print ("Script completed")
```
```python
arcpy.AddField_management("Parcel", "ACRES", "Double")

# Update ACRES field using the SHAPE@AREA token
with arcpy.da.UpdateCursor("Parcel", ["SHAPE@AREA", "ACRES"]) as cursor:
    for row in cursor:
        row[1] = row[0] / 43560
    cursor.updateRow(row)

print ("Script completed")
```
arcpy.da.InsertCursor

\[
\text{arcpy.da.InsertCursor(in\_table, field\_names)}
\]

<table>
<thead>
<tr>
<th>Method</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>insertRow (row)</td>
<td>Inserts a row into a table.</td>
</tr>
<tr>
<td>A</td>
<td>B</td>
</tr>
<tr>
<td>--------</td>
<td>-----------</td>
</tr>
<tr>
<td>Latitude</td>
<td>Longitude</td>
</tr>
<tr>
<td>38.99583</td>
<td>-77.041</td>
</tr>
<tr>
<td>38.9855</td>
<td>-77.0277</td>
</tr>
<tr>
<td>38.97533</td>
<td>-77.0145</td>
</tr>
<tr>
<td>38.96517</td>
<td>-77.0017</td>
</tr>
<tr>
<td>38.95467</td>
<td>-76.9882</td>
</tr>
<tr>
<td>38.94433</td>
<td>-76.9748</td>
</tr>
<tr>
<td>38.93367</td>
<td>-76.9613</td>
</tr>
<tr>
<td>38.91357</td>
<td>-76.9355</td>
</tr>
<tr>
<td>38.90322</td>
<td>-76.9223</td>
</tr>
<tr>
<td>38.8929</td>
<td>-76.9092</td>
</tr>
<tr>
<td>38.90352</td>
<td>-77.1591</td>
</tr>
<tr>
<td>38.91407</td>
<td>-77.1459</td>
</tr>
<tr>
<td>38.9376</td>
<td>-77.1157</td>
</tr>
<tr>
<td>38.94442</td>
<td>-77.1067</td>
</tr>
<tr>
<td>38.95488</td>
<td>-77.0936</td>
</tr>
<tr>
<td>38.96517</td>
<td>-77.0803</td>
</tr>
<tr>
<td>38.97533</td>
<td>-77.0672</td>
</tr>
<tr>
<td>38.98567</td>
<td>-77.054</td>
</tr>
</tbody>
</table>

**boundaryStones**
import arcpy
import csv

arcpy.env.workspace = r"C:\demo\DC.gdb"
file = r"C:\demo\boundaryStones.csv"

# Create empty feature class
sr = arcpy.SpatialReference(4326)

# Create an empty feature class
arcpy.CreateFeatureclass_management(r"C:\demo\DC.gdb", "BoundaryStones", "Point", spatial_reference=sr)

# Add fields to feature class
arcpy.AddField_management("BoundaryStones", "Description", "Text")

# Create Insert Cursor
cursor = arcpy.da.InsertCursor("BoundaryStones", ["SHAPE@XY", "Description"])

with open(file, newline='') as csvfile:
    reader = csv.reader(csvfile)
    count = 0
    for line in reader:
        count += 1
        if count > 0:
            # SHAPE@XY (X,Y) passed as a tuple
            cursor.insertRow([[float(line[1]), float(line[0])], line[2]])
        count = +1

deleted cursor
import arcpy
import csv

DCBoundaryStones

BoundaryStones

<table>
<thead>
<tr>
<th>OBJECTID</th>
<th>Shape</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Point</td>
<td>North Corner</td>
</tr>
<tr>
<td>2</td>
<td>Point</td>
<td>Northeast Mile 1</td>
</tr>
<tr>
<td>3</td>
<td>Point</td>
<td>Northeast Mile 2</td>
</tr>
<tr>
<td>4</td>
<td>Point</td>
<td>Northeast Mile 3</td>
</tr>
<tr>
<td>5</td>
<td>Point</td>
<td>Northeast Mile 4</td>
</tr>
<tr>
<td>6</td>
<td>Point</td>
<td>Northeast Mile 5</td>
</tr>
<tr>
<td>7</td>
<td>Point</td>
<td>Northeast Mile 6</td>
</tr>
<tr>
<td>8</td>
<td>Point</td>
<td>Northeast Mile 7</td>
</tr>
<tr>
<td>9</td>
<td>Point</td>
<td>Northeast Mile 8</td>
</tr>
<tr>
<td>10</td>
<td>Point</td>
<td>East Corner</td>
</tr>
</tbody>
</table>
Working with Geometry Objects

• Reading geometries
  - arcpy.da.SearchCursor
  - arcpy.da.UpdateCursor

• Writing geometries
  - arcpy.da.UpdateCursor
  - arcpy.da.InsertCursor

• Work with geoprocessing tools
Using geometry as input with geoprocessing tools

```python
import arcpy

# List of coordinates.
coordinates = [
    [2365000, 7355000],
    [2365000, 7455000],
    [2465000, 7455000],
    [2465000, 7355000]]

# Create an array with a point object for each coordinate pair
array = arcpy.Array([arcpy.Point(x, y) for x, y in coordinates])

# Create a polygon geometry object using the array object
boundary = arcpy.Polygon(array, arcpy.SpatialReference(2953))

# Use the geometry to clip an input feature class
arcpy.Clip_analysis('c:/data/rivers.shp',
                    boundary,
                    'c:/data/rivers_clipped.shp')
```
Demo: Which buffers overlap?
Which buffers overlap?

```python
for row in arcpy.da.SearchCursor('BuffHalfMi', ['OID@', 'SHAPE@', 'NAME']):
    for row2 in arcpy.da.SearchCursor('BuffHalfMi', ['OID@', 'SHAPE@', 'NAME']):
        if row2[1].overlaps(row[1]):
            print ('{0} (oid:{1}) overlaps {2} (oid:{3})'.format(row2[2], row2[0], row[2], row[0]))
```

Federal Center SW (oid:6) overlaps Archives-Navy Mem'1 (oid:1)
Judiciary Sq (oid:7) overlaps Archives-Navy Mem'1 (oid:1)
Metro Center (oid:9) overlaps Archives-Navy Mem'1 (oid:1)
Gallery Pl-Chinatown (oid:10) overlaps Archives-Navy Mem'1 (oid:1)
L'Enfant Plaza (oid:88) overlaps Archives-Navy Mem'1 (oid:1)
Leveraging other Python modules

- Python Standard Library
  - *Read a csv with the csv module*
  - *Unzip a .zip file with the zipfile module*
  - *Download data from the internet with the urllib module*
  - *Automatically open an output with the os module*
  - … *many more*

- Can install other pre-existing Python packages with ArcGIS Pro Python Package Manager
Location-Based Analysis for Recruitment of United States Border Patrol Agents

Jordan Douthitt
Advisor: Stephen A. Matthews

March 20 & 21, 2018
Executive Order 13767 on January 25th, 2017

• Need to hire 5,000 additional U.S. Border Patrol Agents

Historical difficulty hiring Agents

>10% loss in Southwest Agents 2011-2017
Need to strategically recruit a large amount of skilled employees (United States Border Patrol Agents) to hard to fill positions in undesirable locations.
Analysis answering two questions

1. Which Border Patrol Sectors have the greatest recruitment support need?

2. Which areas outside of these Border Patrol Sectors can additional recruiting focus on?
Factors Affecting Recruitment

Border Patrol Agent Factors
- Starting age under 39
- Degree/Background in Law Enforcement
- Ability to Speak/Learn Spanish
- Workload & Location

Labor Pool & Migration Factors
- Total Population
- Age
- Unemployment Rate
- Salary
- Community Size & Demography
- Distance from Community of Origin
- Previous Migration/Social Ties
Methodology: Sector Profiles

**Sector Information**
- Agent Count
- Agent Turnover
- Station Count

**Workload**
- Border crossing locations
- Apprehensions per agent
- Border entries per agent

**Labor Pool**
- Percent of stations in nonmetropolitan areas
- Population of labor pools

**DECISION:**
Which sectors are most in need of additional recruitment assistance
Methodology: Labor Pool Analysis

**Sensitivity Testing**
- $R^2$ correlation between factor values

**Measure Factors**
- 1-5 quantile rating
- Final Score: sum of individual ratings

**Results Testing**
- $R^2$ correlation between factor scores
- & total score

**Spatial Cluster Analysis**
- Univariate Local Moran's I
- Getis-Ord Gi*

DECISION:
Which outside labor pools are most likely to provide more viable recruits
Results: Sector Profiles

**Sector Information**
- 2017 Agent Count: 500
- 2011-2017 Agent Change: 25% loss
- Station Count: 13

**Workload**
- Border Crossing Count: 2
- 2011-2017 Agent Change: 25% loss
- Station Count: 13
- Border Entries per Agent: 2,990
- Apprehensions per Agent: 12

**Labor Pool**
- Percent of Stations in Nonmetropolitan Areas: 69.2%
- Population of Surrounding Labor Pools: 2,142,736
Results: Labor Pool Analysis

Total Rating by Metropolitan and Nonmetropolitan Area

Legend
- Total Factor Rating
- 10 Quantiles
  - 14 - 24
  - 25 - 27
  - 28 - 30
  - 31 - 33
  - 34 - 41

Projection: USA Contiguous Albers Equal Area Conic USGS

Hot and Cold Spot Clusters by Metropolitan and Nonmetropolitan Area Using Getis-Ord Gi* Statistic

Legend
- Hot and Cold Spots
  - Cold Spot - 95% Confidence
  - Cold Spot - 99% Confidence
  - Not Significant
  - Hot Spot - 10% Confidence
  - Hot Spot - 15% Confidence
  - Hot Spot - 19% Confidence

Projection: USA Contiguous Albers Equal Area Conic USGS
Conclusions

Value to Human Resources Field:

• Previously minimal demographic or location-based research performed

• Model for focused recruitment of hard to fill positions that can be used in other similar scenarios
ArcPy

Need:
• Join a large amount of Census CSV files to a feature class to visualize factors within
• Tell which areas are within a desired distance of the border between the United States and Mexico
• Easily allow for changes needed as data is researched

Python:
• Automate Repetitive Tasks
• Changeable Parameters
• Ease of Use with ArcGIS Script Tool
• Sharable
ArcPy

# Import both arcpy and csv modules for use in script
import arcpy
import csv
import os

# Set overwriteOutput to true, so that files can be overwritten if an error happens
arcpy.env.overwriteOutput = True

# Set needed input and output parameters
inputTablesFolder = arcpy.GetParameterAsText(0)
inputJoinShapefile = arcpy.GetParameterAsText(1)
inputBufferShapefile = arcpy.GetParameterAsText(2)
inputBufferDistance = arcpy.GetParameterAsText(3)
outputFolder = arcpy.GetParameterAsText(4)

# Start try statement for bulk of script
try:

    # Check if this geodatabase already exists in the output folder
    # Create name and location to check for geodatabase
    outputGDBCheck = os.path.join(outputFolder, "Output_GDB.gdb")
    # Start if statement, if geodatabase exists, set output geodatabase path
    if arcpy.Exists(outputGDBCheck):
        arcpy.AddMessage("FileGDB Already Exists")
        outputGDB = outputGDBCheck
    # If the output file geodatabase does not already exist in the output folder, create it
    else:
        # Create file geodatabase to output new shapefiles into
        outputGDB = arcpy.CreateFileGDB_management(outputFolder, "Output_GDB")
ArcPy

```python
# Sets workspace for the arcpy.ListFiles() function
arcpy.env.workspace = inputTablesFolder

# Lists all files from input folder that end in ".csv"
csvList = arcpy.ListFiles("*.csv")

# Start for loop for csv files
for table in csvList:
    # Create output feature layer name from csv name
    fName = table.replace(".csv", "")

    # Make layer from input shapefile to allow for select layer by location later in script
    arcpy.MakeFeatureLayer_management(inputJoinShapefile, fName)

    # Create buffer distance name
    # This removes all spaces from the buffer distance string
    fieldNameString = inputBufferDistance.replace(" ", ",")
    # Length of field name is kept to 10 characters with 'bf_' and the first 7 characters of the buffer distance
    fieldName = "bf_" + fieldNameString[:6]
    # Adds field
    arcpy.AddField_management(fName, fieldName, "TEXT")

    # Select layer by location based on input buffer shapefile and distance
    arcpy.SelectLayerByLocation_management(fName, "WITHIN A DISTANCE", inputBufferShapefile, inputBufferDistance)

    # Update cursor to add whether or not each record falls in the specified distance of the selected feature
    # If record falls within buffer distance (select by location), it prints "YES" in the field, otherwise a null value is left
    with arcpy.da.UpdateCursor(fName, (fieldName,)) as cursor:
        for row in cursor:
            row[0] = "YES"
            cursor.updateRow(row)
```

### ArcPy

- **Select by Buffer Distance**
- **Create New Field**
- **Update Field Values**
# Clear selected features before join to csv, so that all features are joined, not just selected features
arcpy.SelectLayerByAttribute_management(fcName, "CLEAR_SELECTION")

# Join csv to new feature layer based on the GEOID2 field in each
arcpy.AddJoin_management(fcName, "GEOID2", table, "GEOID2")

# Create shapefile from layer and joined table
# Set name of output shapefile
lyrName = fcName
lyrName2 = lyrName + "_fc"
# Create path for output shapefile
outPath = str(outputGDB)
fcPath = os.path.join(outPath, lyrName2)

# Create shapefile
arcpy.CopyFeatures_management(lyrName, fcPath)

arcpy.AddMessage(arcpy.GetMessages())

try:
    # Report error message
    arcpy.AddMessage("Could Not Complete Process")
    # Report any error messages that tools in this script might have generated
    arcpy.AddMessage(arcpy.GetMessages(2))
except:
    # Delete the built layer, even if the rest of the script does not run
finally:
    # Delete layer
    arcpy.Delete_management(lyrName)
ArcPy

Python Script Tool

- Use script in other projects
- Rework process
- Change input files
- Change buffer distance

Background
Problem & Goal
Data
Methodology
Results
ArcPy
Thank You
Additional Python Sessions

- Python: Building Geoprocessing Tools (145 B)
  - Wednesday 4:00 pm
- ArcGIS API for Python: Scripting Your Web GIS (145 B)
  - Wednesday 8:15 am
Print Your Certificate of Attendance
Print stations located in the 140 Concourse

**Tuesday**
12:30 pm - 6:30 pm  
GIS Solutions Expo  
Hall B

5:00 pm - 6:30 pm  
GIS Solutions Expo Social  
Hall B

**Wednesday**
10:30 am - 5:15 pm  
GIS Solutions Expo  
Hall B

6:30 pm - 9:00 pm  
Networking Reception  
Smithsonian National Portrait Gallery
Please Take Our Survey in the Esri Events App

1. Download the Esri Events app and find your event
2. Select the session you attended
3. Scroll down to find the feedback section
4. Complete answers and select “Submit”