

Federal GIS Conference

February 9–10, 2015 | Washington, DC



Python: Getting Started

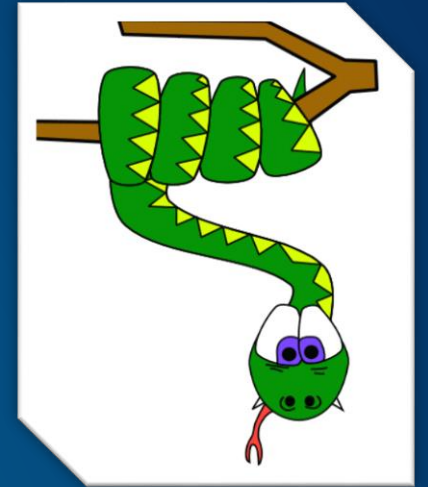
Jennifer Duerr

Agenda

- **Intro**
- **The what and why of Python & ArcGIS**
- **How to get started**
- **Demos**
- **Resources for getting started and moving forward**

Why Python?

- Free
- Stable and mature
- Cross-platform
- Open-source
- Powerful
- Scalable
- Widely used
- Easy to learn
- Well supported/documented
- Large user community



ArcGIS and Python

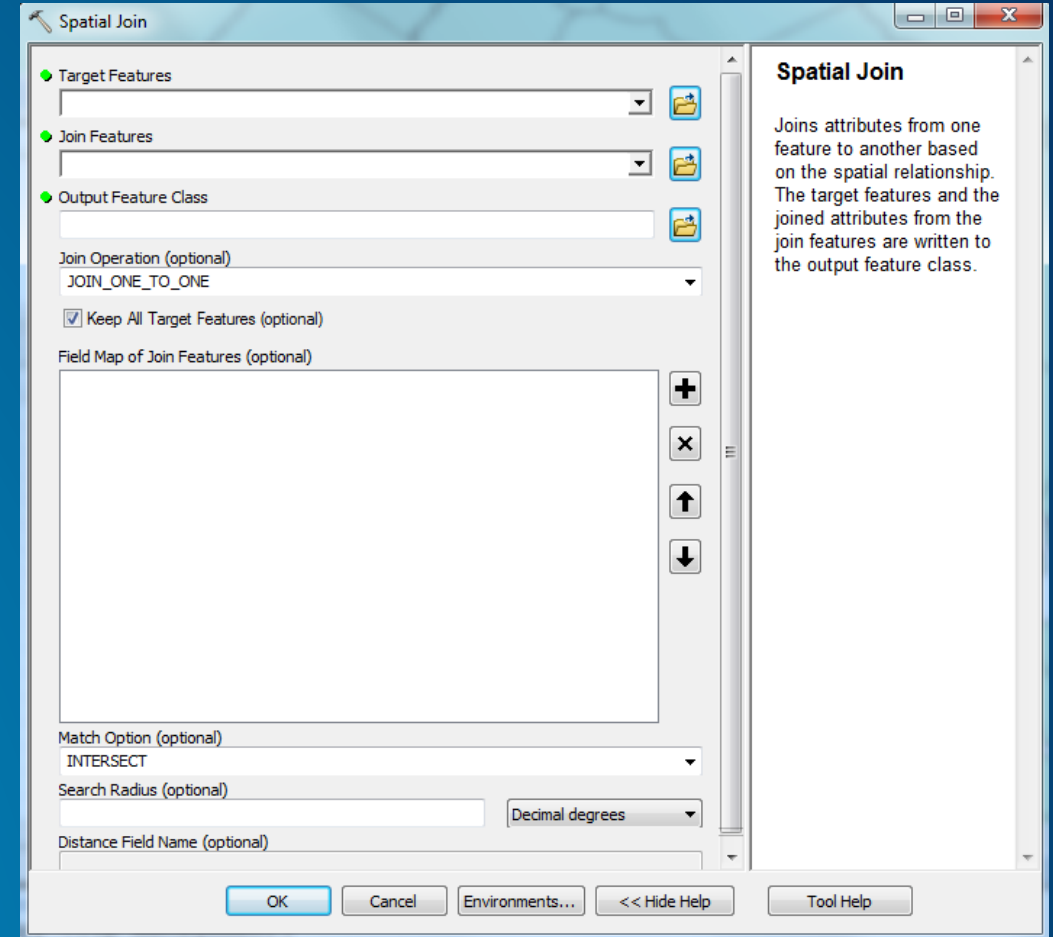
- **Cornerstone for achieving increased productivity in the realm of**
 - **Geographic data analysis**
 - **Data conversion**
 - **Data management**
 - **Map automation**
- **Introduced to the ArcGIS community at v9.0, arcgisscripting module.**
- **At v10.0, the ArcPy Site Package**
 - **Each release has furthered the Python experience, providing more capabilities and a richer, more Python-friendly experience.**
- **Currently → Integral, well established at this point as the scripting language used to further all aspects of geoprocessing customization and automation**

ArcPy Site Package

- **Introduced at v10.0, continual improvement since**
- **Help → <http://desktop.arcgis.com/en/desktop/latest/analyze/arcpy/what-is-arcpy-.htm>**
- **Python access to**
 - **Geoprocessing tools**
 - **Extensions**
 - **Functions and classes to work with and interrogate GIS data**
- **Special modules for data access (improved speed), mapping automation, SA, NA**
- **Rich python experience**
- **Code completion, syntax guidance**
- **Well-documented reference help**

What Is Needed

- ArcGIS installed! Python v2.7 with it
- Comfort with executing geoprocessing tools
 - understanding of inputs, outputs, parameters
- Some familiarity with basic scripting
 - Variables
 - If, elif, else statement... conditions
 - Looping... while/for
- IDE... gotta write the code somewhere!



Python IDEs



- **ArcGIS Desktop's Python Window**
 - simple and intuitive interface
 - Provides Python functionality for desktop
 - Syntax help automatically displays for python and gp tools
 - Code completion for imported modules, ArcPy functions, gp tools
 - Allows you to work directly with layers in ArcMap
 - Test and verify workflows easily
- **Other IDEs --Provides a rich debugging environment (free, cost)**
 - PythonWin
 - PyCharm
 - IDLE
 - Notepad++
 - Wing IDE
 - Visual Studio Python Plug In
 - Komodo IDE

Variables

- Set with one equal sign =
- Substitute for raw values
- Name that stores a value
- Names are case-sensitive (Scale ≠ scale)
- Holds data types of String (numbers or text), Number, Lists, Files, Dictionaries, etc

```
>>> x = 43560
>>> airportsShapefile = r"C:\Data\airports.shp"
>>> expression = '[score] * 0.5'
```

- Note: the **r** in front of the path is a string literal, meaning tell python to treat the string as-is and not treat anything as an escape sequence, **** is an escape sequence, see https://docs.python.org/2/reference/lexical_analysis.html#string-literals

Types

- Strings

- Enclosed in ' or "
- Pathnames options
- Indexed, start with 0
- Stores numbers and/or characters

- Numbers

- Store numbers, from simple and small to large and complicated

```
>>> x = 481516
>>> y = 23.42
>>> multiple = x * y
>>> multiple
11277104.72
>>>
>>> import math
>>> math.sqrt(x)
693.913539282813
>>> round(math.sqrt(x), 3)
693.914
```

```
>>> fgdbPath = "C:\\ProjectX\\Data.gdb"
>>> fgdbPath = r'C:\ProjectX\Data.gdb'
>>>
>>> "forests.shp"[0:7]
'forests'
>>> "forests.shp"[:-4]
'forests'
>>> "forests.shp"[2:5]
'res'
>>>
>>> xNumber = 108
>>> yStringNumber = "108"
>>> xNumber == yStringNumber
False
>>> str(xNumber) == yStringNumber
True
```

Types

- Lists

- Initiated as bracketed list
- Each element is indexed, accessible
- Ability to sort, insert, remove, etc

```
>>> countiesList = ["Prince William", "Loudoun", "Fairfax"]
>>> countiesList[2]
'Fairfax'
>>> countiesList.sort()
>>> countiesList
['Fairfax', 'Loudoun', 'Prince William']
>>> countiesList.insert(1, "King George")
>>> countiesList
['Fairfax', 'King George', 'Loudoun', 'Prince William']
```

- Dictionaries

- Initiated as curly bracket
- Key:Value pairs
- Unordered collection of data elements

```
>>> countiesPopulationDict = {"Fairfax": 969749, "Loudoun": 169599, "Prince William": 280813}
>>> countiesPopulationDict["Prince William"]
280813
>>> countiesPopulationDict.keys()
['Loudoun', 'Prince William', 'Fairfax']
```

- Tuples

- Like lists but unchangeable

Decision making, if...elif...else

- Colon at the end of each condition :
- Indentation important, determines what gets executed, adds readability
- Test equality with two equal signs == , use other operators too >, <, !=
- Use operands for testing multiple conditions, and/or

```
>>> todaysDate = "2/9/2015"
>>> theWeatherIs = "Sunny and Warm" #lies
>>> if todaysDate == "2/9/2015" and theWeatherIs == "Sunny and Warm":
...     print("Welcome to FedGIS2015 and enjoy this great weather Esri ordered up for you! (... you can wake from your dream now)")
... elif todaysDate == "2/9/2015" and theWeatherIs == "Cold":
...     print("Welcome to FedGIS2015, Stay warm!")
... else:
...     print("Just another day at the office")
...
Welcome to FedGIS2015 and enjoy this great weather Esri ordered up for you! (... you can wake from your dream now)
```

Iteration

- for loop

```
>>> colorList = ['red', 'blue', 'yellow', 'purple', 'green']
>>> for color in colorList:
...     if color == 'purple':
...         print(color + " is the best color")
...     else:
...         print(color)
...
red
blue
yellow
purple is the best color
green
```

- while loop

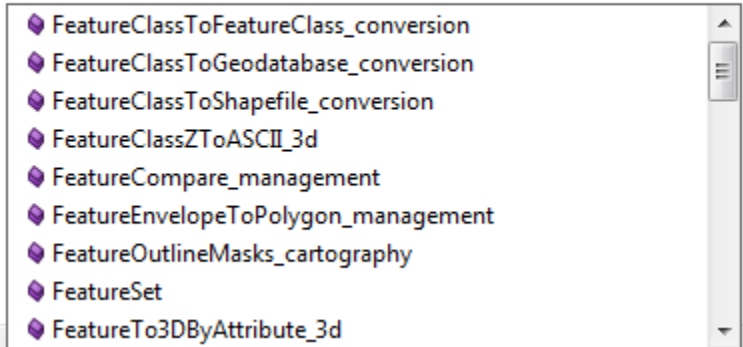
```
>>> x = 1
>>> while x < 5:
...     print(x)
...     x = x + 1
...
1
2
3
4
```

Python Building Blocks

- **Function** – defined bit of functionality that performs a task
- **Classes** – a framework for how to create something, a blueprint
- **Module** – where functions reside, includes functions and classes
- **Package** – group of related modules, ArcPy!
- **Standard library with Python:**
 - **os**
 - **sys**
 - **math**
 - **datetime**
 - **urllib2**

```
>>> import os
>>> fcName = os.path.basename(r"C:\Data\Landmarks.gdb\schools")
>>> fcName
'schools'
```

```
>>> import arcpy
>>> arcpy.F
```



- ◆ FeatureClassToFeatureClass_conversion
- ◆ FeatureClassToGeodatabase_conversion
- ◆ FeatureClassToShapefile_conversion
- ◆ FeatureClassZToASCII_3d
- ◆ FeatureCompare_management
- ◆ FeatureEnvelopeToPolygon_management
- ◆ FeatureOutlineMasks_cartography
- ◆ FeatureSet
- ◆ FeatureTo3DByAttribute_3d

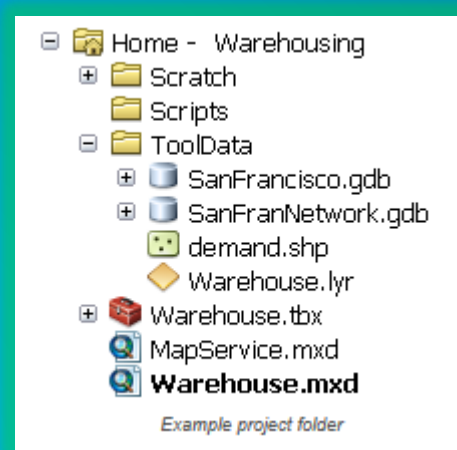
List Functions

- Important functions outside of core geoprocessing tools
- Lists, in a list, the items in a workspace... workspace environment must be declared
- ListDatasets, ListFeatureClasses, ListTables, ListRasters, ListFiles, ListWorkspaces

```
>>> ptFCs = arcpy.ListFeatureClasses(feature_type='Point')
>>> for fc in ptFCs:
...     print(fc)
...
NursingHomes
Hospitals
Pharmacies
PublicHealthDepartments
UrgentCare
>>> ptFCs
[u'NursingHomes', u'Hospitals', u'Pharmacies', u'PublicHealthDepartments', u'UrgentCare']
```

Best Practices

- **Error handling** – see blog with a template <http://blogs.esri.com/esri/arcgis/2011/08/04/pythontemplate/>
- **Document!**
 - Provide script comments and header info -- use `### '''`
 - Script tools, provide metadata (title, abstract, key words, parameter description)
- **Keep clean** – clean up code as you go, remove unused lines
- **Message feedback** – hand back messages to the user about status, success, info... timers for finding code that is slowing progress, some other method may be able to help improve performance
- **Organize** – organize the code, standardize the project directory structure



How To Get Started

- **Look at system GP tools, help, see the syntax, copy samples**
- **Look at existing core gp tools that are scripts, see how they work (especially Spatial Statistics Toolbox)**
- **ModelBuilder – Export to Python Script**
- **Run gp tools, see Results, Copy as Python Snippet**
- **Use python window**

- **Build your script!**

Demo

- **Model in Model Builder**
- **Export to Python script**
- **Do necessary Edits**
- **Add new code for more functionality**

Demo'd Script... From Model to Script... Simple Looping... Copy-n-Paste-n-Resize Text Below

```
# Import arcpy module

import arcpy

import os

# allow overwrites

arcpy.env.overwriteOutput = True

# Local variables:

US_counties = r"C:\GIS_Data\Baso_GIS_Data\BasoGISdata.gdb\counties_us"

output_location = "C:\FedGIS_2015\GettingStarted\Data.gdb"

schools_points = r"C:\FedGIS_2015\GettingStarted\Data.gdb\schoools"

Expression = "STATE_NAME = 'Virginia'"

counties_stateSubset = "C:\FedGIS_2015\GettingStarted\Data.gdb\counties_VA"

#output_polyCounts = "C:\FedGIS_2015\GettingStarted\Data.gdb\virginia_schools_by_county"

# Process: Feature Class to Feature Class

arcpy.FeatureClassToFeatureClass_conversion(US_counties, output_location, "counties_VA", Expression)

# Declare workspace, create list of FCs

arcpy.env.workspace = r"C:\FedGIS_2015\GettingStarted\Data.gdb"

pointFClist = arcpy.ListFeatureClasses(feature_type='Point')

# Loop thru list of FCs in workspace

for ptFC in pointFClist:

    # Get directory and FC name for input into spatial join tool

    ptFC2 = os.path.join(output_location, ptFC)

    print(ptFC2)

    # Declare output FC name for each SJ output

    output_polyFC = os.path.join(output_location, ptFC + ".byCounty")

    # Process: Spatial Join, point FC to counties, to get at count of features per county

    arcpy.SpatialJoin_analysis(counties_stateSubset, ptFC2, output_polyFC)

    print(output_polyFC + " has been created")

print("Finished!")
```

Help Resources

- **ArcGIS Resources** -- <http://resources.arcgis.com/en/home/>
 - Help Documentation
 - Forums
 - Videos
 - Blogs
- **Python Community for ArcGIS** -- <http://resources.arcgis.com/en/communities/python/>
- **ArcPy Café** -- <https://arcpy.wordpress.com/>
- **Esri's Geography Channel, Tech Workshops** -- <http://geochannel.esri.com/index.cfm?event=video.home>
- **Python.org** -- also see style guide here <https://www.python.org/dev/peps/pep-0008/>
- **Esri's GeoNet** -- <https://geonet.esri.com/welcome>
- **Google Python group** -- <https://groups.google.com/forum/?fromgroups#!forum/comp.lang.python>
- **GitHub** -- <https://github.com>, <https://github.com/Esri>

Training Resources

- Previous slide
- Esri training...
 - Instructor led, 3-day course, \$
 - Several 1-3 hour web courses
- Coursera – free online learning, university-partnered courses, a few for Python
- Dive Into Python – free, online Python book
- Code Academy Course -- <http://www.codecademy.com/tracks/python>
- Several hardcopy books available, varying levels of experience... explore at will!
 - *Beginners recommendations:*
 - *Learning Python* by Mark Lutz and David Ascher
 - *Core Python Programming* by Wesley J. Chun
 - Esri Press books:
 - *GIS Tutorial for Python Scripting* by David Allen
 - *Python Scripting for ArcGIS* by Paul Zandbergen

Slides for this presentation....

- Will be available online in 2-3 weeks
- View and download at <http://proceedings.esri.com/library/userconf/index.html>
- Or just search 'esri conference proceedings', first returned link will take you to a list of recent proceedings
- Click on the event of interest – PDF or PPTX or video possibly available

The Zen of Python

by Tim Peters

Easter Egg, `>>> import this`

Beautiful is better than ugly.
Explicit is better than implicit.
Simple is better than complex.
Complex is better than complicated.
Flat is better than nested.
Sparse is better than dense.
Readability counts.
Special cases aren't special enough to break the rules.
Although practicality beats purity.
Errors should never pass silently.
Unless explicitly silenced.
In the face of ambiguity, refuse the temptation to guess.
There should be one-- and preferably only one --obvious way to do it.
Although that way may not be obvious at first unless you're Dutch.
Now is better than never.
Although never is often better than **right** now.
If the implementation is hard to explain, it's a bad idea.
If the implementation is easy to explain, it may be a good idea.
Namespaces are one honking great idea -- let's do more of those!

Python: Getting Started

Jennifer Duerr

jduerr@esri.com



Understanding our world.