



Esri International User Conference | San Diego, CA
Technical Workshops | July 12 - July 14, 2011

Python – Getting Started

Drew Flater, Ghislain Prince

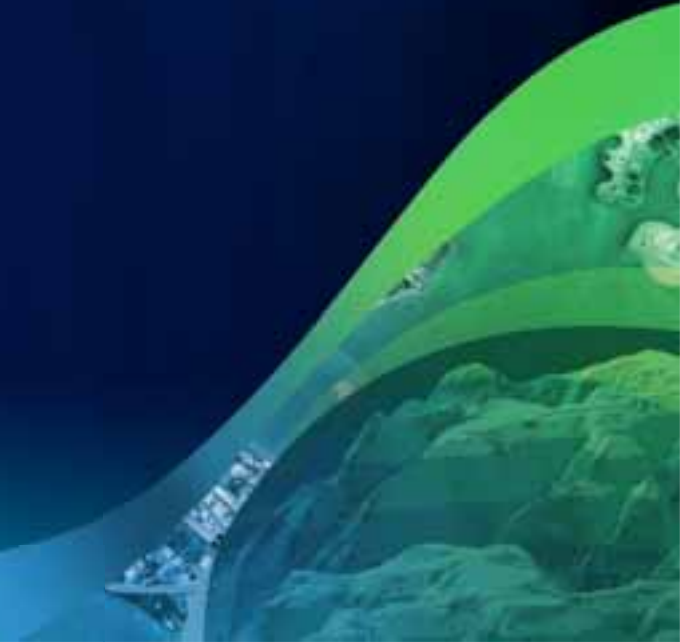
Does this describe you?

- **New to Python scripting**
- **Comfortable using ArcGIS but want to become more efficient**
- **Moving to Python from other scripting language**
- **Interested in what's new in ArcGIS 10**

Agenda

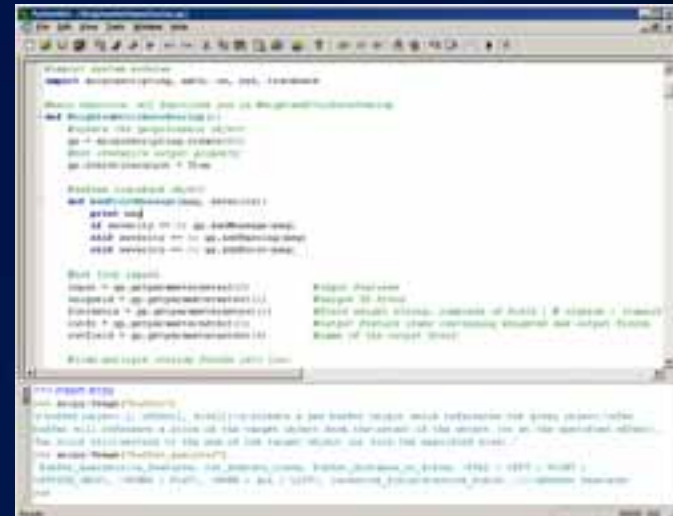
- **Python scripting essentials**
 - Why use Python scripting?
 - Python 101
 - What is ArcPy?
 - Executing geoprocessing tools
 - Messages and error handling
- **Geoprocessing tasks**
 - ArcPy functions
 - Batch processing
 - Receiving arguments

Python Scripting Essentials



Why use Python scripting?

- Scripting language of ArcGIS
- Free, cross-platform, easy to learn, great community
- But why? Other ways to run tools
- Develop, execute, and share geoprocessing workflows
- Improve productivity



Python 101

- Where do I write Python code?
 - IDE like PythonWin; Python window in ArcGIS
- Which lines will run?

```
# I am a comment, I will not execute  
import arcpy
```

- What are variables?
 - A name that stores a value; assigned using =

```
input = "C:/Data/Roads.shp"  
distance = 50  
both = [input, distance]  
  
# Variables act as substitutes for raw values  
arcpy.Buffer_analysis(input, "Roads_buffer.shp", distance)
```

Python 101

- Python has logic for testing conditions
 - **if, else** statement
 - Colon at end of each condition
 - Indentation determines what is executed
 - **==** tests equality; other operators like **>**, **<**, **!=**

```
var = "a"
if var == "a":
    # Execute indented lines
    print "variable is a"
else:
    print "variable is not a"
```

Python 101

- **Techniques for iterating or looping**
 - While loops, counted loops, list loops
 - Colon at end of statement
 - Indentation determines what is executed

```
x = 1
while x < 5:
    print x
    x = x + 1

for num in range(1,5):
    print num

x = [1, 2, 3, 4]
for num in x:
    print num
```


Python 101


- Case sensitivity
 - Variables, functions, etc. are case sensitive
 - name 'X' is not defined, function 'X' does not exist
- For paths, use forward-slash as separator
 - `"C:/Data/Roads.shp"`
- Functions & Modules
 - **Function**: a defined piece of functionality that performs a specific task; requires arguments ()
 - **Module**: a Python file where functions live; imported
 - `math.sqrt(100) ... 10.0`
 - *"There's a module for that"*

ArcPy

- The access point to geoprocessing tools
- A package of functions, classes and modules, all related to scripting in ArcGIS
 - Functions that enhance geoprocessing workflows (**ListFeatureClasses**, **Describe**, **SearchCursor**, etc)
 - Classes that can be used to create complex objects (**SpatialReference**, **FieldMap** objects)
 - Modules that provide additional functionality (**Mapping**, **SpatialAnalyst** modules)
- Builds on arcgisscripting module (pre-10.0)

ArcGIS Python window

- Embedded, interactive Python window within ArcGIS
 - Access to ArcPy, any Python functionality
- Great for experimenting with Python and learning tool syntax



The screenshot shows a Python window titled "Python" with a standard Windows-style title bar. The main area contains a Python interpreter session with the following text:

```
>>> import arcpy
>>> 5+5
10
>>> arcpy.GetCount_management("inside")
<Result '100">
>>> arcpy.Buffer_analysis(
```

Below the code, there is a small button with a yellow lightning bolt icon and the text "inside". Below that, the documentation for the `Buffer_analysis` function is displayed:

`Buffer_analysis(in_features, out_feature_class, buffer_distance_or_field, [line_side], [line_end_type], [dissolve_option], [dissolve_field,dissolve_field...])`
Creates buffer polygons around input features to a specified distance. An optional dissolve can be performed to combine overlapping buffers.

Below the documentation, the "INPUTS:" section is partially visible, showing the `in_features` parameter.

Executing a tool in Python

Demo

Executing a tool in Python

- ArcPy must be imported
- Follow syntax: **arcpy.toolname_toolboxalias()**
- Enter input and output parameters

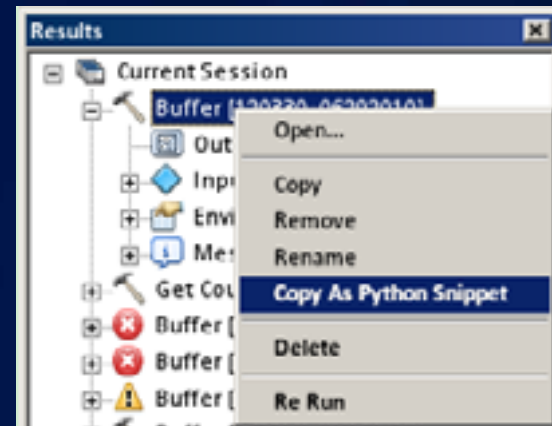
```
# Import ArcPy
import arcpy

# Set workspace environment
arcpy.env.workspace = "C:/Data"

# Execute Geoprocessing tool
arcpy.Buffer_analysis("Roads.shp", "Roads_buffer.shp",
    "50 Meters")
```

Getting tool syntax

- Results window, 'Copy as Python Snippet'
- Tool documentation
- Export Model to Python script
- Drag tool into Python window
- `arcpy.Usage("Buffer_analysis")`






Setting environments in Python

- Accessed from `arcpy.env`
- Provides finer control of tool execution; makes scripting easier
- Common environments:
 - Workspace, coordinate system, extent

```
arcpy.env.workspace = "C:/Data"  
arcpy.env.extent = "0 0 100 100"
```

Geoprocessing messages

- Tools return three types of messages:
 - Informative messages 
 - Warning messages 
 - Error messages 
- Displayed in the ArcGIS Python window
- `arcpy.GetMessages()`
 - `GetMessages()`: All messages
 - `GetMessages(0)`: Only informative messages
 - `GetMessages(1)`: Only warning messages
 - `GetMessages(2)`: Only error messages

arcpy.GetMessages

```
# Execute Geoprocessing tool
arcpy.Buffer_analysis("Roads.shp", "Roads_buffer.shp",
    "50 Meters")

# Print the execution messages
print arcpy.GetMessages()

>>>
Executing: Buffer Roads.shp Roads_buffer.shp '50 Meters'
Start Time: Tue July 12 08:52:40 2011
Executing (Buffer) successfully.
End Time: Tue July 12 03:52:45 2011(Elapsed Time: 5.00...
```

Error handling basics

- Why do errors occur?
 - Incorrect tool use
 - Typos
 - Syntax errors
- Python error handling
 - Try...Except...

```
try:
```

```
    Pass Go
```

```
except:
```

```
    Do not Pass Go, Do not Collect $200
```

Try, Except Statement

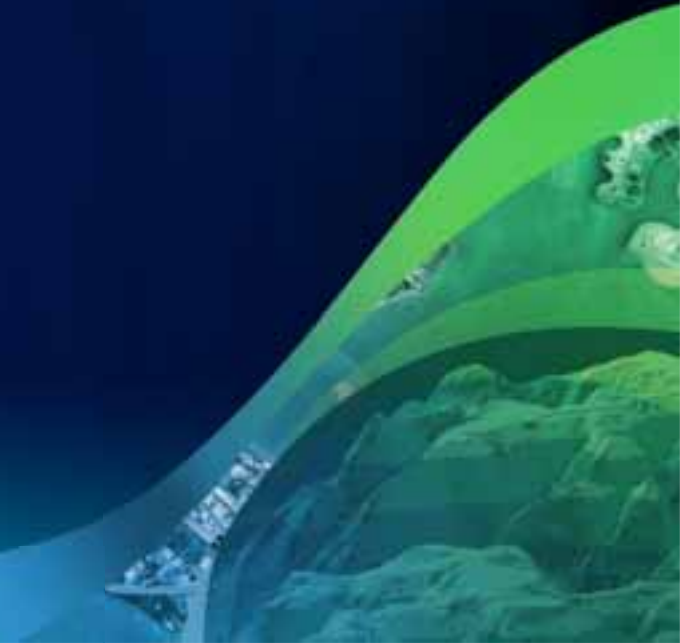
```
# Start Try block
try:
    arcpy.Buffer_analysis("Roads.shp", "Roads_buffer.shp",
        "50 Meters")

# If an error occurs
except:
    # Print that Buffer failed and why
    print "Buffer failed"
    print arcpy.GetMessages(2))
```

Error handling & messaging

Demo

ArcPy Functions



ArcGIS 10 Help

Hide

Locate

Back

Home

Options

Resource Center

Contents | Favorites | Search

- Welcome to ArcGIS Help
- What's new in ArcGIS
- Essentials Library
- Professional Library
 - What's in the Professional Library
 - Data Management
 - Mapping and Visualization
 - Geoprocessing
 - What is geoprocessing?
 - A quick tour of geoprocessing
 - Essential geoprocessing vocabulary
 - Geoprocessing tools
 - The geoprocessing framework
 - Commonly used tools
 - Finding tools
 - Executing tools
 - Managing tools and toolboxes
 - Creating tools
 - Sharing tools
 - Geoprocessing with ModelBuilder
 - Geoprocessing with Python
 - Geoprocessing with ArcGIS Server
 - The ArcPy site package
 - What is ArcPy?
 - Essential ArcPy vocabulary
 - A quick tour of ArcPy
 - Functions
 - Alphabetical list of ArcPy functions**
 - Cursors
 - Describing data
 - Environments and settings
 - Fields
 - General
 - General data functions
 - Getting and setting parameters
 - Licensing and installation
 - Listing data
 - Messaging and error handling

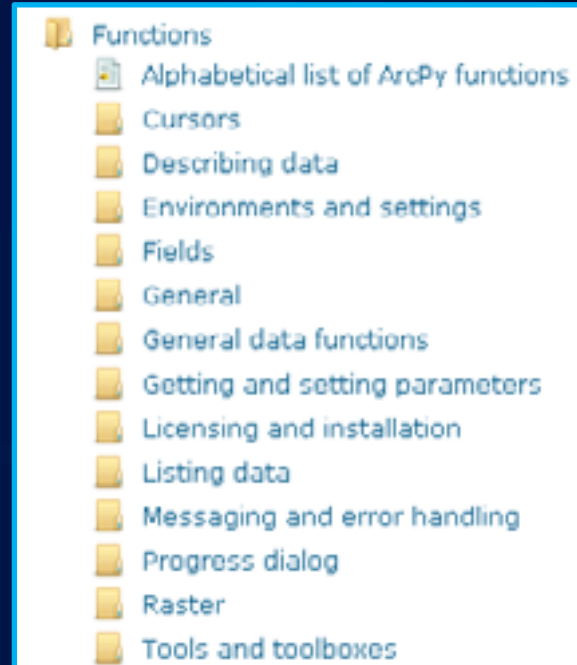
Alphabetical list of ArcPy functions

ArcGIS 10

Function name	Category
AddError	Messages and error handling
AddFieldDelimiters	Fields
AddIDMessage	Messages and error handling
AddMessage	Messages and error handling
AddReturnMessage	Messages and error handling
AddToolbox	Tools and toolboxes
AddWarning	Messages and error handling
AsShape	General
CheckExtension	Licensing and installation
CheckInExtension	Licensing and installation
CheckOutExtension	Licensing and installation
CheckProduct	Licensing and installation
ClearEnvironment	Environments and settings
Command	General
CopyParameter	Getting and setting parameters
CreateObject	General
CreateRandomValueGenerator	General
CreateScratchName	General data functions

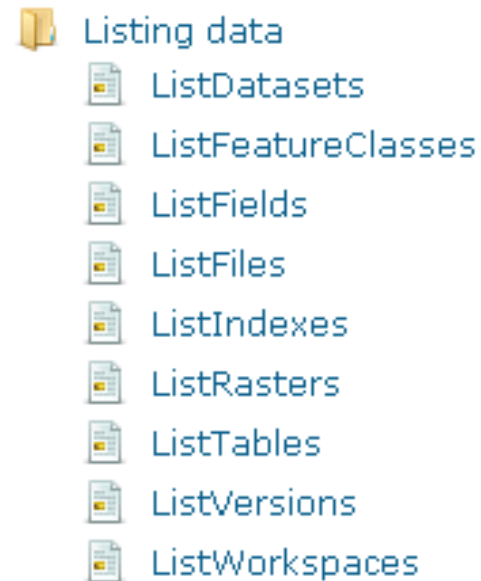
ArcPy functions

- Perform useful scripting tasks
 - Print messages (**GetMessages**)
 - List data to aid batch processing (**ListFeatureClasses**, 12 total List functions)
 - Getting data properties (**Describe**)
 - Etc.
- Supports **automation** of manual tasks



Batch processing

- **Run a geoprocessing operation multiple times with some automation**
 - **Example: Using the Clip tool to clip every feature class in a workspace to a boundary**
- **List functions used in Python to perform batch processing**

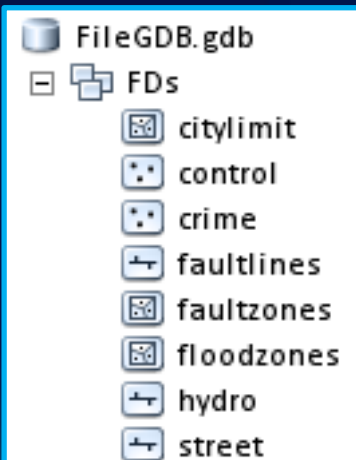


arcpy.ListFeatureClasses

```
# Set the workspace
arcpy.env.workspace = "C:/Data/FileGDB.gdb/FDs"

# Get a list of all feature classes
fcList = arcpy.ListFeatureClasses()

# Print the list of feature classes one at a time
for fc in fcList:
    print fc
```



Batch Processing

Demo

ArcPy functions -- Describe

- Use the Describe function to read data properties
 - Returns an object with properties



- Allows script to determine properties of data
 - Data type (shapefile, coverage, network dataset, etc.)
 - Shape type (point, polygon, line, etc.)
 - Spatial reference
 - Etc.

```
# Describe a feature class
desc = arcpy.Describe("C:/Data/Roads.shp")

print desc.shapeType
>>> "Polyline"
```

Receiving arguments

- Arguments are user-defined inputs to a script
 - Values passed to script from user, instead of hard-coded
- Use **GetParameterAsText** to read arguments
- Connect script to an ArcGIS script tool
 - Best way to create and share custom workflows
 - More accessible than stand-alone Python script

```
# Create variables from input arguments
inputFC = arcpy.GetParameterAsText(0)
outputFC = arcpy.GetParameterAsText(1)

# First and third parameters come from arguments
arcpy.Clip_analysis(inputFC, "C:/Data/boundary.shp", outputFC)
```

Describe function & Arguments

Demo

Python scripting resources

- ArcGIS Resource Centers
 - resources.arcgis.com
 - Online documentation
 - Geoprocessing: script gallery, blog, tutorials, presentations
- Python Organization
 - python.org
- Python Reference Books
 - *Learning Python* by Lutz, et al
 - *Core Python Programming* by Chun



Esri Training for Python

esri.com/training



- **Instructor-Led Course**
 - [Introduction to Geoprocessing Scripts Using Python](#)
- **Web Course (free)**
 - [Using Python in ArcGIS Desktop 10](#)

