CENTER for
DIGITAL
SCHOLARSHIP

Using Python in ArcGIS

Matthew L. Sisk
CLIR postdoctoral fellow in data curation and GIS services
Center for Digital Scholarship
Hesburgh Library, University of Notre Dame

library.nd.edu/cds/

UNIVERSITY OF
NOTRE DAME
Hesburgh Libraries

Data download

- http://tinyurl.com/l6bm8k4
- http://www3.nd.edu/~msisk1/PythonWorkshop_Data.zip
- Both links are the same data
- Extract to C:\Temp

What is Python?

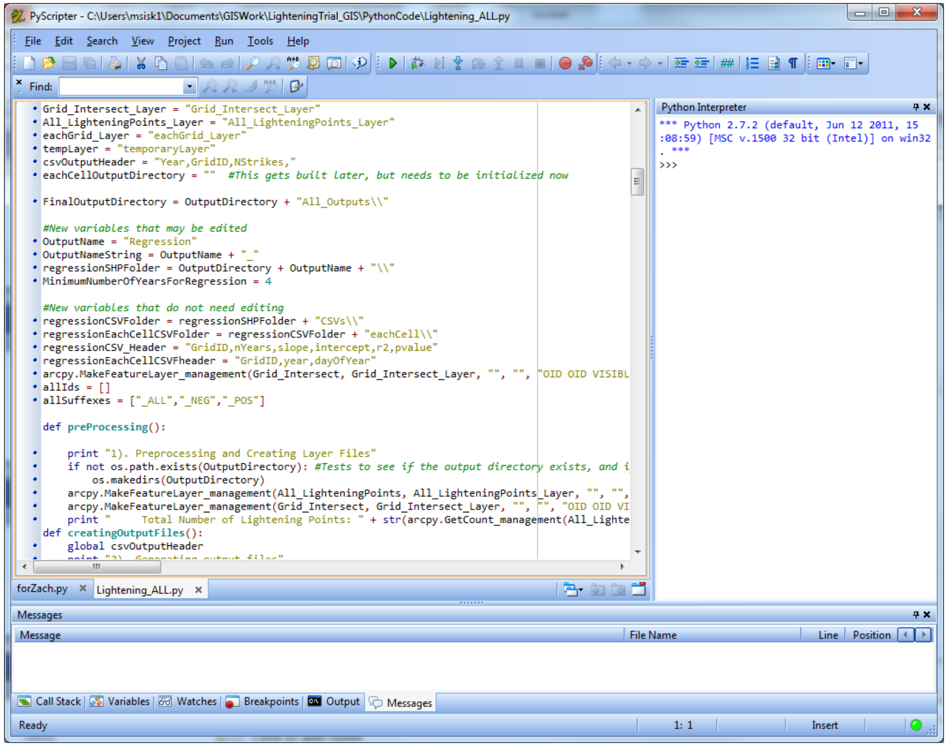
- *“Python is an easy to learn, powerful language…(with) high-level data structures and a simple but effective approach to object-oriented programming. Python’s elegant syntax and dynamic typing…make it an ideal language for scripting…in many areas and on most platforms.”*

  –python.org

- Scripting language of ArcGIS (and several other GIS packages)
- Free, cross-platform, easy to learn

Why use Python scripting?

- Multiple ways to run the same tool
- Excellent way to document, share and automate workflows
  - GIS often has a metadata problem
- Improves productivity
  - Increased modularity

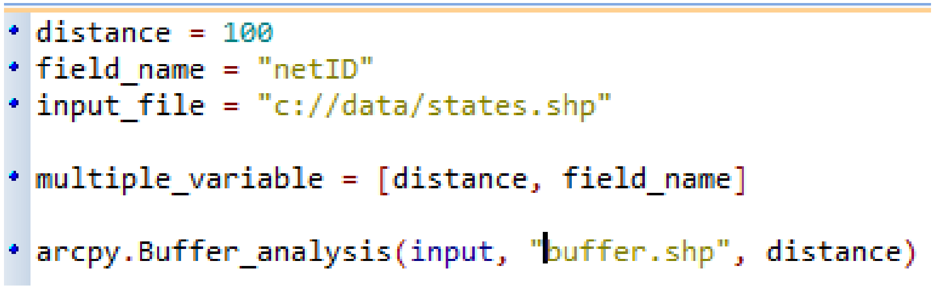


```
Grid_Intersect_Layer = "Grid_Intersect_Layer"
All_LighteningPoints_Layer = "All_LighteningPoints_Layer"
eachGrid_Layer = "eachGrid_Layer"
tempLayer = "temporaryLayer"
csvOutputHeader = "Year,GridID,NStrikes,"
eachCellOutputDirectory = ""  #This gets built later, but needs to be initialized now

FinalOutputDirectory = OutputDirectory + "All_Outputs\\"

#New variables that may be edited
OutputName = "Regression"
OutputNameString = OutputName + "_"
regressionSHPFolder = OutputDirectory + OutputName + "\\"
MinimumNumberOfYearsForRegression = 4

#New variables that do not need editing
regressionCSVFolder = regressionSHPFolder + "CSVs\\"
regressionEachCellCSVFolder = regressionCSVFolder + "eachCell\\"
regressionCSV_Header = "GridID,nYears,slope,intercept,r2,pvalue"
regressionEachCellCSVFheader = "GridID,year,dayOfYear"
arcpy.MakeFeatureLayer_management(Grid_Intersect, Grid_Intersect_Layer, "", "", "OID OID VISIBL
allIds = []
allSuffexes = ["_ALL","_NEG","_POS"]

def preProcessing():

    print "1). Preprocessing and Creating Layer Files"
    if not os.path.exists(OutputDirectory): #Tests to see if the output directory exists, and i
        os.makedirs(OutputDirectory)
    arcpy.MakeFeatureLayer_management(All_LighteningPoints, All_LighteningPoints_Layer, "", "",
    arcpy.MakeFeatureLayer_management(Grid_Intersect, Grid_Intersect_Layer, "", "", "OID OID VI
    print "     Total Number of Lightening Points: " + str(arcpy.GetCount_management(All_Lighte
def creatingOutputFiles():
    global csvOutputHeader
    print "2). Generating output files"
```

```
*** Python 2.7.2 (default, Jun 12 2011, 15
:08:59) [MSC v.1500 32 bit (Intel)] on win32
. ***
>>>
```

Python basics

- Where do I write Python code?
  - IDE like IDLE
    - PythonWin or PyScripter
  - Python window in ArcGIS
- How do I run a Python script?
  - Double-click or command prompt
  - Within IDE
  - As a tool in the ArcToolBox
- What are variables?
  - A name that stores a value
  - Acts as a substitute for a real value
  - Assigned using =



```
distance = 100
field_name = "netID"
input_file = "c://data/states.shp"

multiple_variable = [distance, field_name]

arcpy.Buffer_analysis(input, "buffer.shp", distance)
```

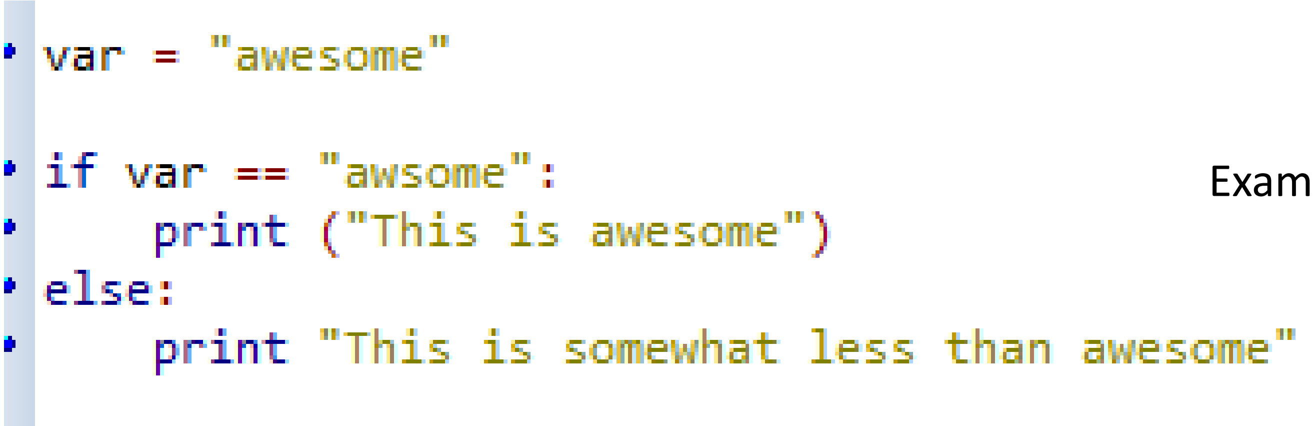
Python basics

- Python is a high-level computer language
  - Designed to be human readable
    - Indentation controls flow logic and code blocks
  - Python is also very forgiving
    - Strings can be defined using “ or ‘
    - Parentheticals are not always required
      - print (“Hello”) is the same as print “hello”
    - Comments can be defined multiple ways
      - #Comment
      - ##Comment
      - ‘”Comment’”
  - Most things are case-sensitive
    - Variables, functions, etc. are case sensitive
    - Print is not the same as print

Example 1

Python basics

- 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 >, <, !=

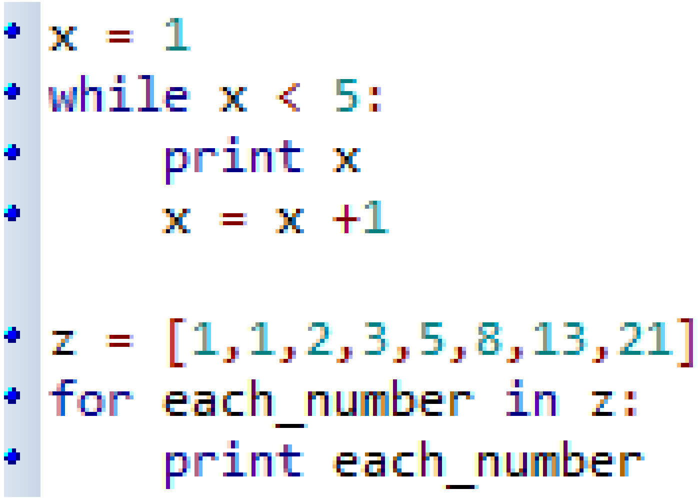


```
var = "awesome"

if var == "awsome":
    print ("This is awesome")
else:
    print "This is somewhat less than awesome"
```

Example 2

Python Basics

- Techniques for iterating or looping
  - While loops
    - Continue to loop as long as a condition is met
  - for loops
    - Step through a predefined list of values
  - Colon at end of statement
  - Indentation determines what is executed

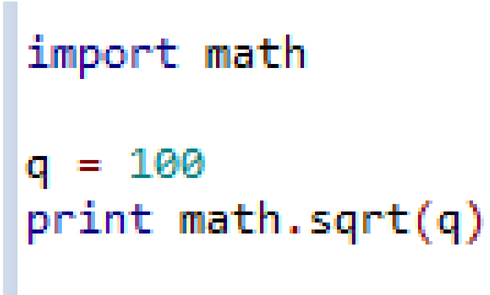


```
x = 1
while x < 5:
    print x
    x = x +1

z = [1,1,2,3,5,8,13,21]
for each_number in z:
    print each_number
```

Example 3

Python basics

- Building blocks
  - – *Function*: a defined piece of functionality that performs a specific task; requires arguments ()
  - – *Module*: a Python file where functions live
    - Should be imported at the start of the script
  - – Package: a collection of modules
  - – math.sqrt(100) … 10.0



```
import math

q = 100
print math.sqrt(q)
```

- Python packages often have to be manually downloaded and installed
  - – Not like r

Example 4

ArcPy Module

- The access point to geoprocessing tools

- A package of functions, classes and modules, all related to scripting in ArcGIS
  - Helper functions that enable workflows
    - ListFeatureClasses, Describe, SearchCursor, etc
  - Classes that can be used to create complex objects
    - SpatialReference, FieldMap objects
  - Modules that provide extended functionality
    - Mapping, SpatialAnalyst modules

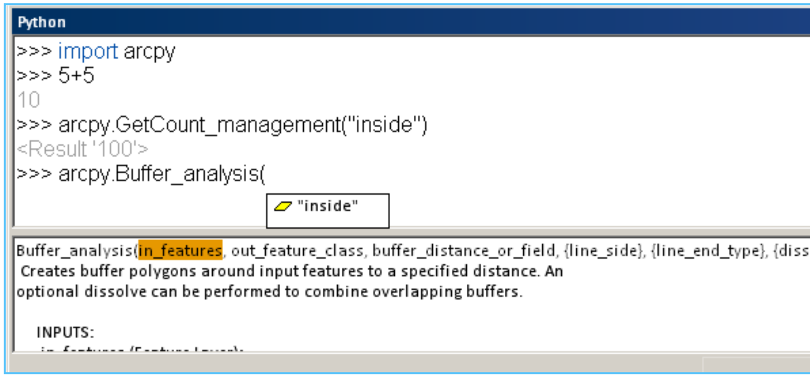
ArcPy Module

- Works with Python version 2.7x
  - Python 3.x is not currently compatible
- Enhancement of arcgisscripting module (pre-10.0)
  - Older Python scripts will work
- Every ArcGIS 10.x install also has Python 2.7
  - Typically in C:\Python27\ArcGIS10.1

ArcGIS Python window

- Embedded, interactive Python window within ArcGIS
  - – Access to ArcPy and any Python functionality

- Great for experimenting with Python and learning tool syntax
  - – Help pane

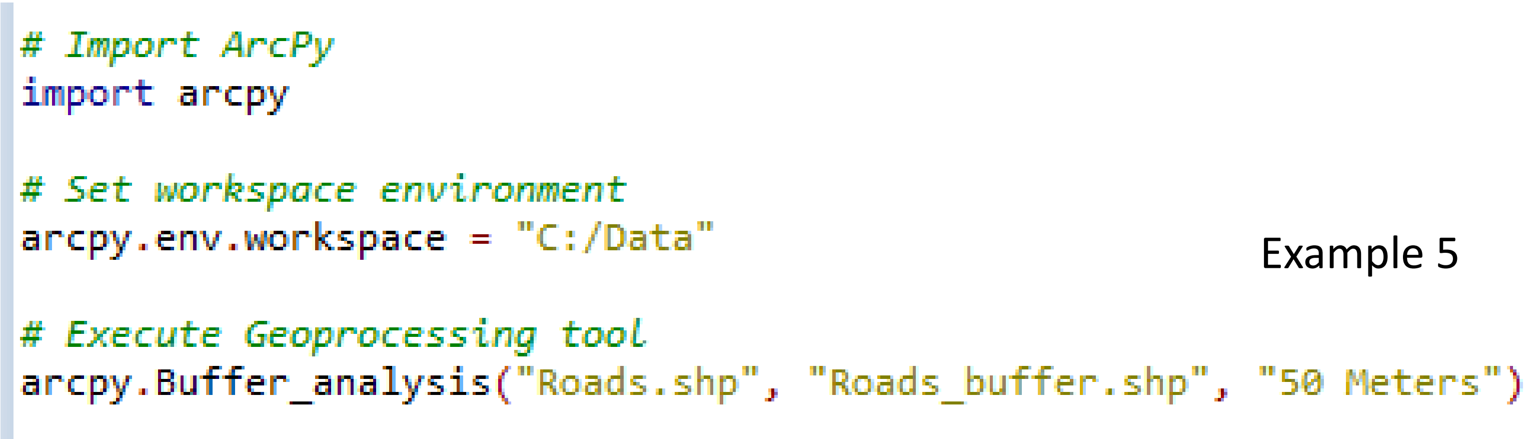


```
>>> import arcpy
>>> 5+5
10
>>> arcpy.GetCount_management("inside")
<Result '100'>
>>> arcpy.Buffer_analysis(
                        "inside"
```

```
Buffer_analysis(in_features, out_feature_class, buffer_distance_or_field, {line_side}, {line_end_type}, {diss
 Creates buffer polygons around input features to a specified distance. An
optional dissolve can be performed to combine overlapping buffers.

   INPUTS:
    in_features (Feature Layer):
```

Executing tools in Python

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

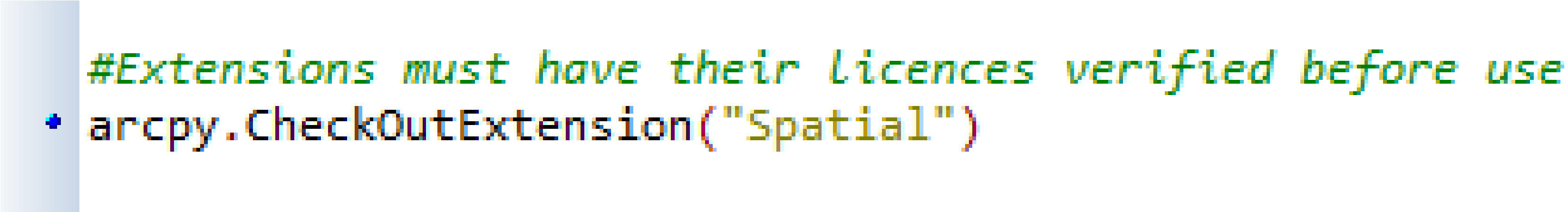


```
# Import ArcPy
import arcpy

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

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

Example 5

Executing tools in Python

- Tools from extensions must have their license verified prior to running
  - – Spatial analyst, 3D analyst, etc
  - – Notre Dame’s site license includes most of these

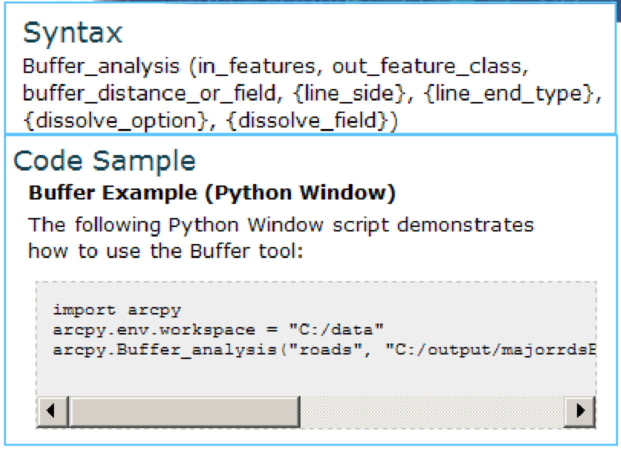


```
#Extensions must have their licences verified before use
arcpy.CheckOutExtension("Spatial")
```

Getting tool syntax

CENTER for DIGITAL SCHOLARSHIP

- Tool help page
- *Copy as Python Snippet*
- Export Model to Python script
- Drag tool into Python window
- help(“arcpy.Buffer_analysis”)



Syntax

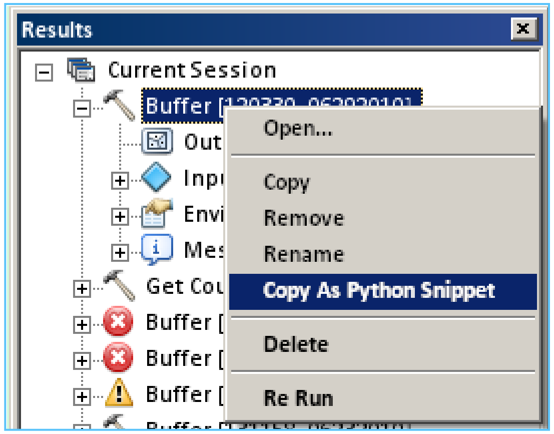
Buffer_analysis (in_features, out_feature_class, buffer_distance_or_field, {line_side}, {line_end_type}, {dissolve_option}, {dissolve_field})

Code Sample

Buffer Example (Python Window)

The following Python Window script demonstrates how to use the Buffer tool:

```
import arcpy
arcpy.env.workspace = "C:/data"
arcpy.Buffer_analysis("roads", "C:/output/majorrdsB
```

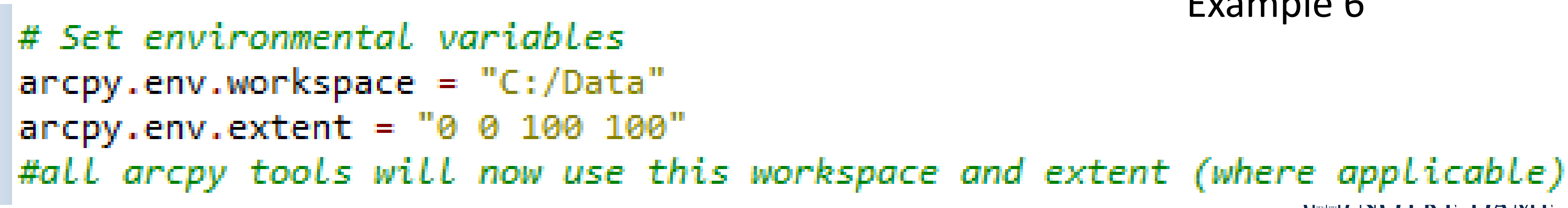


UNIVERSITY OF NOTRE DAME
Hesburgh Libraries

Setting environment variables

- Accessed from arcpy.env
  - – Workspace, coordinate system, extent, etc.
- Global parameter
  - – See tool help for honored environments
- Provides finer control of tool execution
- Makes scripting easier

Example 6

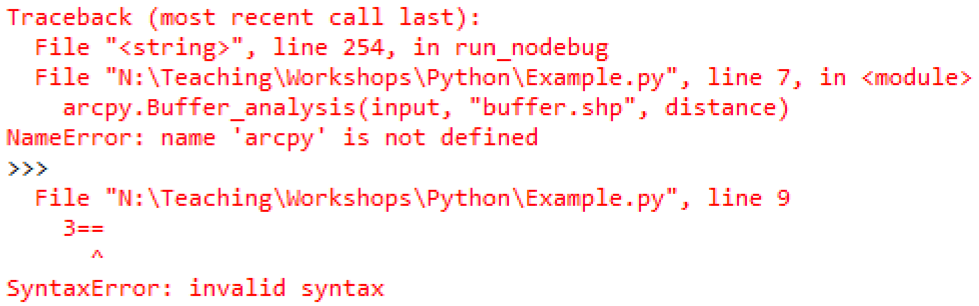


```
# Set environmental variables
arcpy.env.workspace = "C:/Data"
arcpy.env.extent = "0 0 100 100"
#all arcpy tools will now use this workspace and extent (where applicable)
```

Error Handling

CENTER for DIGITAL SCHOLARSHIP

- Why do errors occur?
  - Incorrect tool use
  - Typos
  - Syntax errors



```
Traceback (most recent call last):
  File "<string>", line 254, in run_nodebug
  File "N:\Teaching\Workshops\Python\Example.py", line 7, in <module>
    arcpy.Buffer_analysis(input, "buffer.shp", distance)
NameError: name 'arcpy' is not defined
>>>
  File "N:\Teaching\Workshops\Python\Example.py", line 9
    3==
      ^
SyntaxError: invalid syntax
```

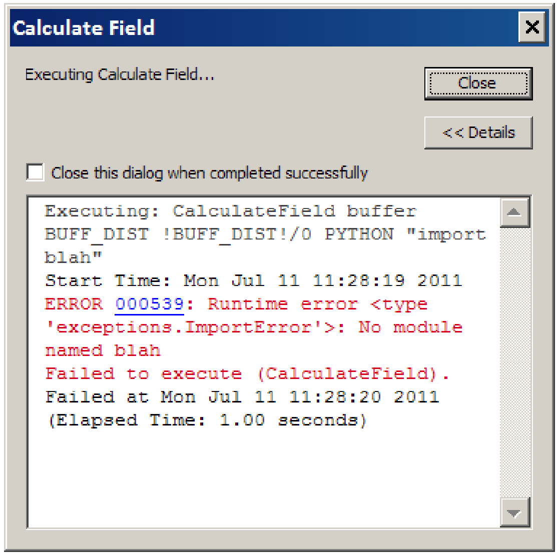
- What can I do if my script doesn't work?
  - View the geoprocessing messages
  - Use Python error handling
  - Debug the script in an IDE

Example 7

UNIVERSITY OF NOTRE DAME
Hesburgh Libraries

Geoprocessing messages

- Three types of messages:
  - Informative messages ℹ
  - Warning messages ⚠
  - Error messages ✖
- Displayed in the Python window
- arcpy.GetMessages()
  - (): All messages
  - (0): Only informative messages
  - (1): Only warning messages
  - (2): Only error messages

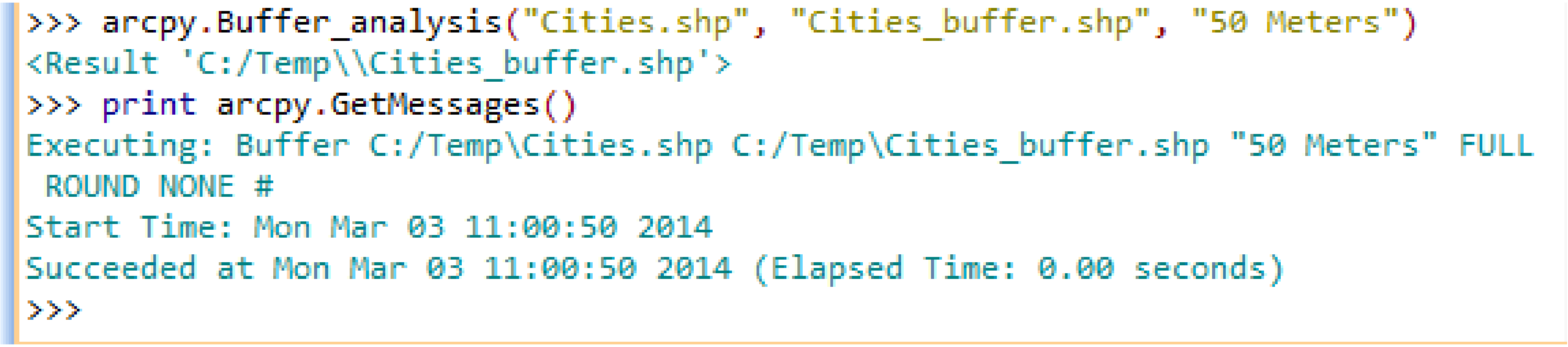


Executing Calculate Field...

□ Close this dialog when completed successfully

```
Executing: CalculateField buffer
BUFF_DIST !BUFF_DIST!/0 PYTHON "import
blah"
Start Time: Mon Jul 11 11:28:19 2011
ERROR 000539: Runtime error <type
'exceptions.ImportError'>: No module
named blah
Failed to execute (CalculateField).
Failed at Mon Jul 11 11:28:20 2011
(Elapsed Time: 1.00 seconds)
```

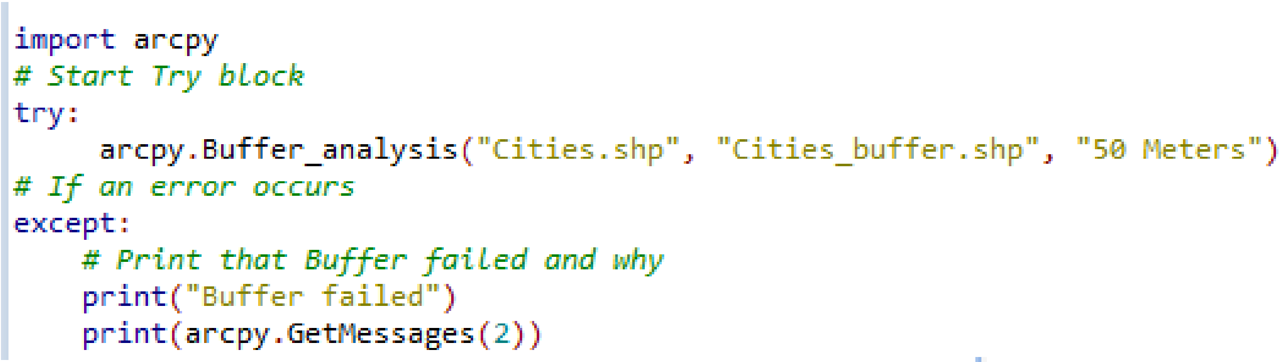
arcpy.GetMessages



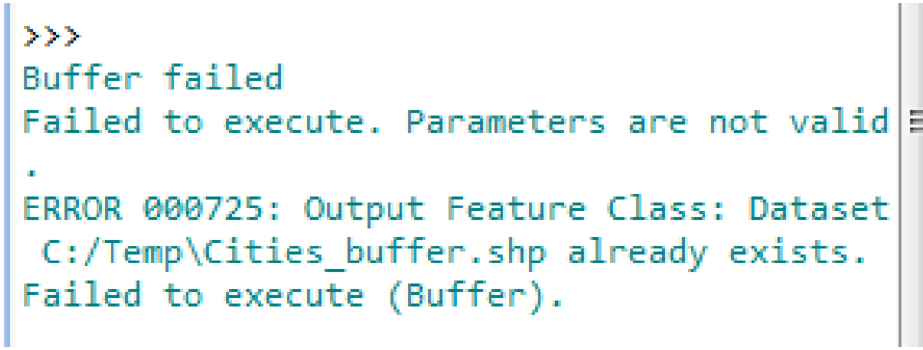
```
>>> arcpy.Buffer_analysis("Cities.shp", "Cities_buffer.shp", "50 Meters")
<Result 'C:/Temp\\Cities_buffer.shp'>
>>> print arcpy.GetMessages()
Executing: Buffer C:/Temp\Cities.shp C:/Temp\Cities_buffer.shp "50 Meters" FULL
 ROUND NONE #
Start Time: Mon Mar 03 11:00:50 2014
Succeeded at Mon Mar 03 11:00:50 2014 (Elapsed Time: 0.00 seconds)
>>>
```

Python error handling

- Try…Except…
  - *Try to do something, and if an error occurs, do something else*

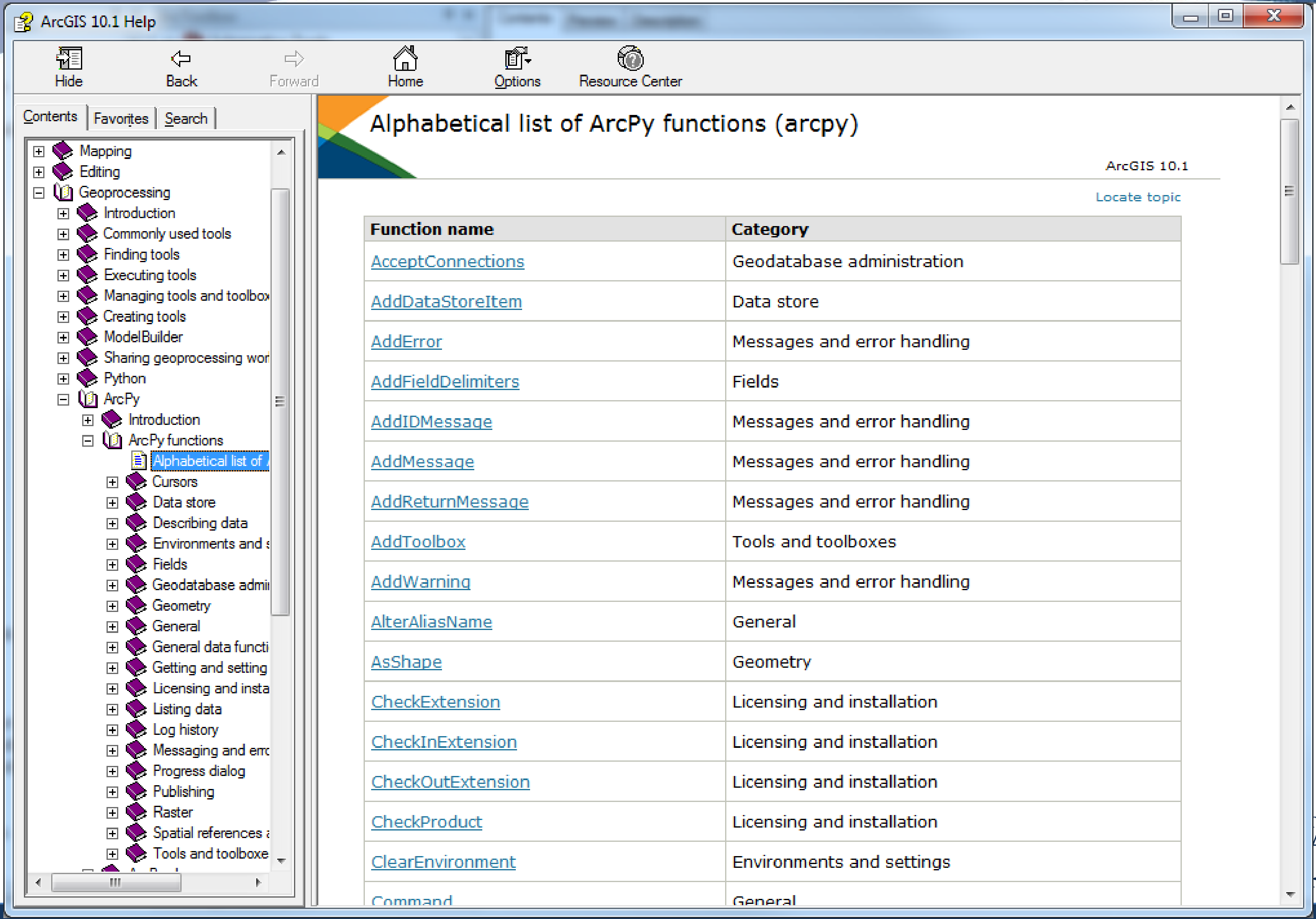


```
import arcpy
# Start Try block
try:
     arcpy.Buffer_analysis("Cities.shp", "Cities_buffer.shp", "50 Meters")
# If an error occurs
except:
    # Print that Buffer failed and why
    print("Buffer failed")
    print(arcpy.GetMessages(2))
```



```
>>>
Buffer failed
Failed to execute. Parameters are not valid
.
ERROR 000725: Output Feature Class: Dataset
 C:/Temp\Cities_buffer.shp already exists.
Failed to execute (Buffer).
```

ArcPy Functions



Alphabetical list of ArcPy functions (arcpy)

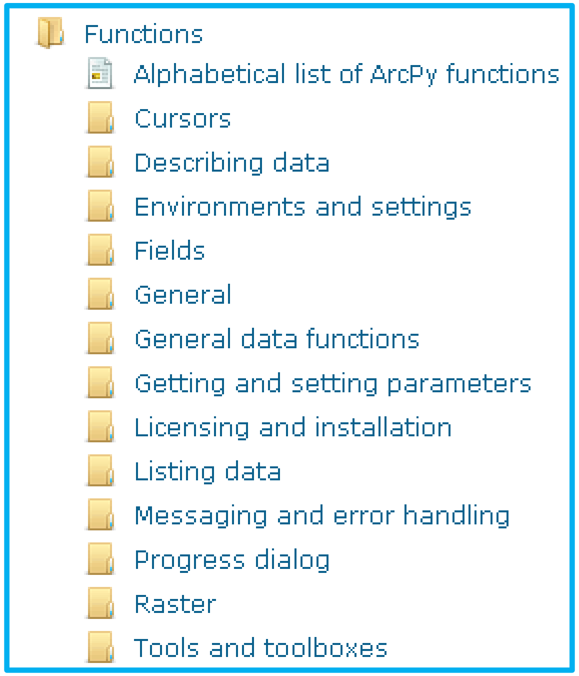
ArcGIS 10.1

Locate topic

Function name	Category
AcceptConnections	Geodatabase administration
AddDataStoreItem	Data store
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
AlterAliasName	General
AsShape	Geometry
CheckExtension	Licensing and installation
CheckInExtension	Licensing and installation
CheckOutExtension	Licensing and installation
CheckProduct	Licensing and installation
ClearEnvironment	Environments and settings
Command	General

ArcPy functions

- Helper functions
- Perform useful scripting tasks
  - Print messages (GetMessages)
  - List data to perform batch processing (ListFeatureClasses, 12 total List functions)
  - Getting data properties (Describe)
  - Etc.



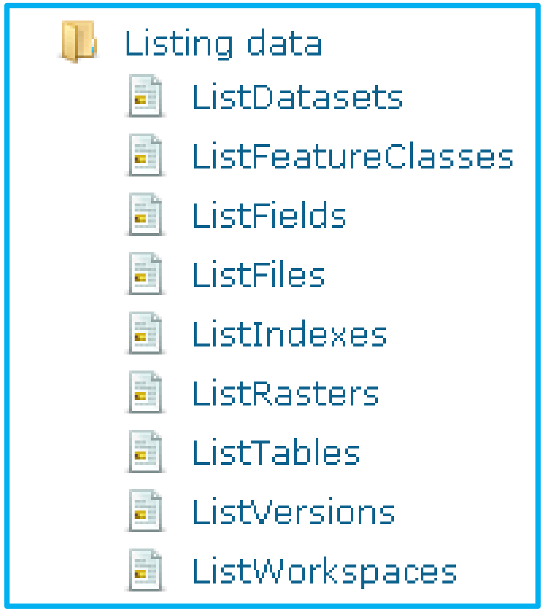
- Functions
  - Alphabetical list of ArcPy functions
  - Cursors
  - Describing data
  - Environments and settings
  - Fields
  - General
  - General data functions
  - Getting and setting parameters
  - Licensing and installation
  - Listing data
  - Messaging and error handling
  - Progress dialog
  - Raster
  - Tools and toolboxes

- Supports automation of manual tasks

Batch processing

- Run a geoprocessing operation multiple times with some automation
  - Clip every feature class in a geodatabase to a boundary
  - Calculate statistics for every raster in a folder
- List functions used in Python to perform batch processing
- One of the highest values in scripting

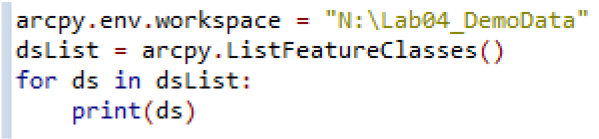
Example 8



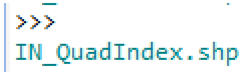
Listing data

- ListDatasets
- ListFeatureClasses
- ListFields
- ListFiles
- ListIndexes
- ListRasters
- ListTables
- ListVersions
- ListWorkspaces

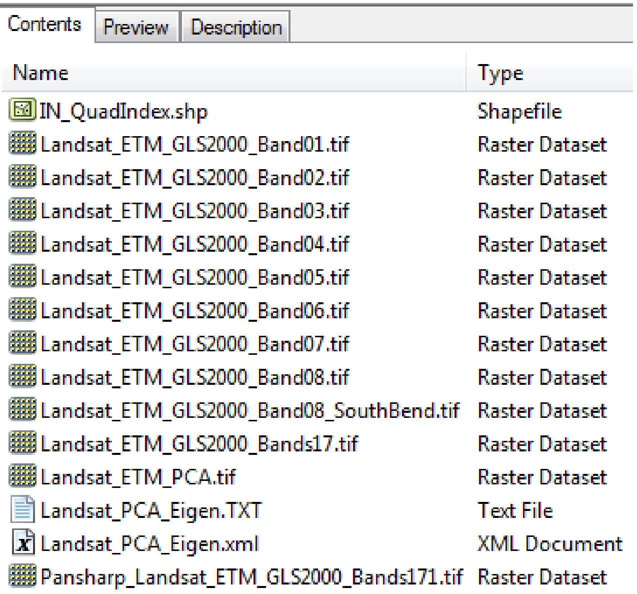
arcpy.ListFeatureClasses



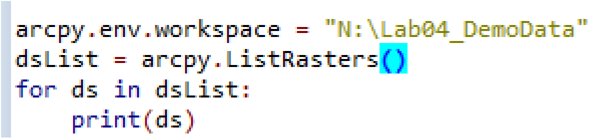
```python
arcpy.env.workspace = "N:\Lab04_DemoData"
dsList = arcpy.ListFeatureClasses()
for ds in dsList:
    print(ds)
```



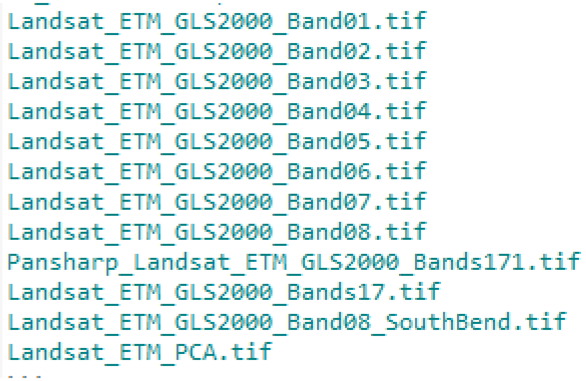
```
>>>
IN_QuadIndex.shp
```



Name	Type
IN_QuadIndex.shp	Shapefile
Landsat_ETM_GLS2000_Band01.tif	Raster Dataset
Landsat_ETM_GLS2000_Band02.tif	Raster Dataset
Landsat_ETM_GLS2000_Band03.tif	Raster Dataset
Landsat_ETM_GLS2000_Band04.tif	Raster Dataset
Landsat_ETM_GLS2000_Band05.tif	Raster Dataset
Landsat_ETM_GLS2000_Band06.tif	Raster Dataset
Landsat_ETM_GLS2000_Band07.tif	Raster Dataset
Landsat_ETM_GLS2000_Band08.tif	Raster Dataset
Landsat_ETM_GLS2000_Band08_SouthBend.tif	Raster Dataset
Landsat_ETM_GLS2000_Bands17.tif	Raster Dataset
Landsat_ETM_PCA.tif	Raster Dataset
Landsat_PCA_Eigen.TXT	Text File
Landsat_PCA_Eigen.xml	XML Document
Pansharp_Landsat_ETM_GLS2000_Bands171.tif	Raster Dataset



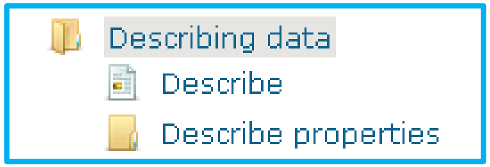
```python
arcpy.env.workspace = "N:\Lab04_DemoData"
dsList = arcpy.ListRasters()
for ds in dsList:
    print(ds)
```



```
Landsat_ETM_GLS2000_Band01.tif
Landsat_ETM_GLS2000_Band02.tif
Landsat_ETM_GLS2000_Band03.tif
Landsat_ETM_GLS2000_Band04.tif
Landsat_ETM_GLS2000_Band05.tif
Landsat_ETM_GLS2000_Band06.tif
Landsat_ETM_GLS2000_Band07.tif
Landsat_ETM_GLS2000_Band08.tif
Pansharp_Landsat_ETM_GLS2000_Bands171.tif
Landsat_ETM_GLS2000_Bands17.tif
Landsat_ETM_GLS2000_Band08_SouthBend.tif
Landsat_ETM_PCA.tif
```

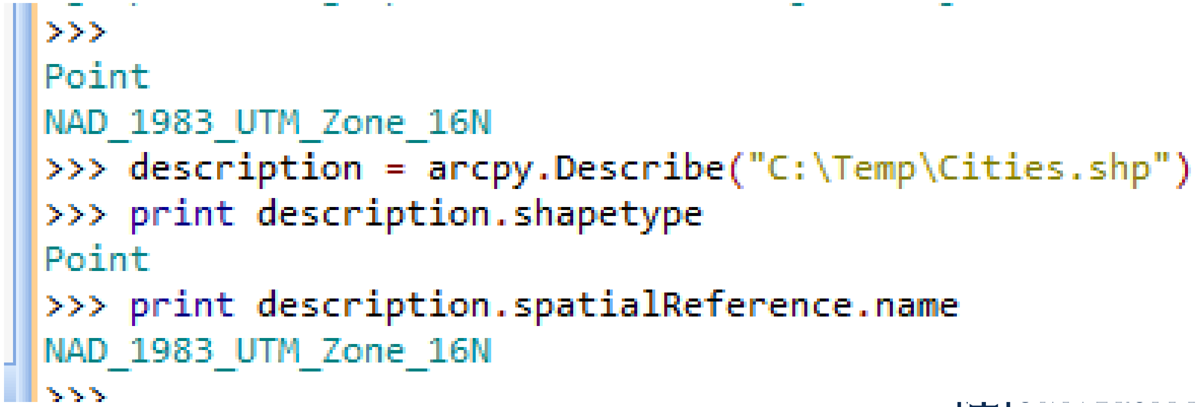
ArcPy functions -- Describe

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



Describing data
- Describe
- Describe properties

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



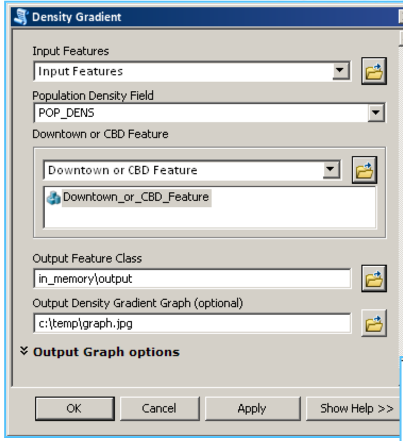
```
>>>
Point
NAD_1983_UTM_Zone_16N
>>> description = arcpy.Describe("C:\Temp\Cities.shp")
>>> print description.shapetype
Point
>>> print description.spatialReference.name
NAD_1983_UTM_Zone_16N
>>>
```

Python script tools

- Connects Python to ArcGIS

- Best way to create and share custom workflows
  - More accessible than stand-alone Python script
  - Extends ArcGIS

- Integrated with geoprocessing framework
  - Inherits geoprocessing properties and environments from application
  - Can be used in ModelBuilder
  - Works with map layers



Density Gradient

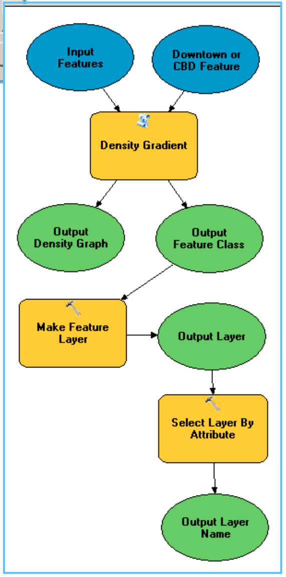
Input Features
Input Features

Population Density Field
POP_DENS

Downtown or CBD Feature
Downtown or CBD Feature
Downtown_or_CBD_Feature

Output Feature Class
in_memory\output

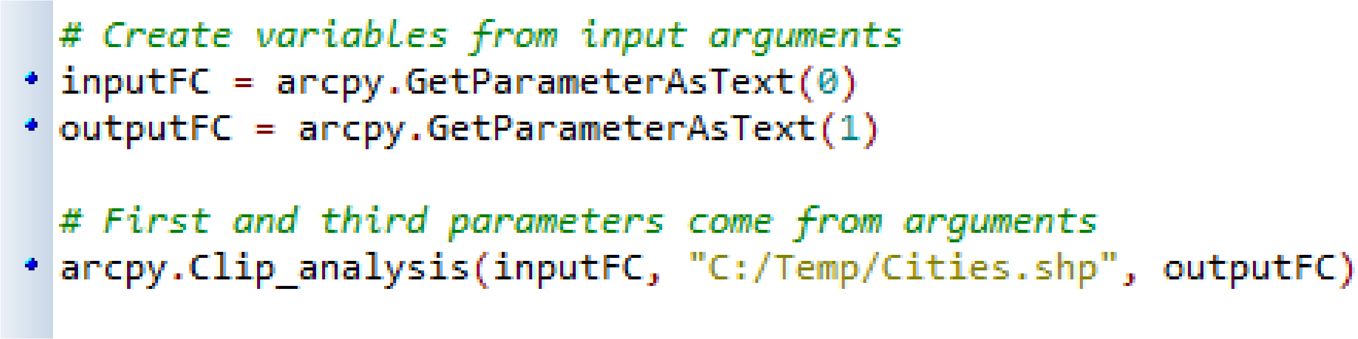
Output Density Gradient Graph (optional)
c:\temp\graph.jpg



Receiving arguments

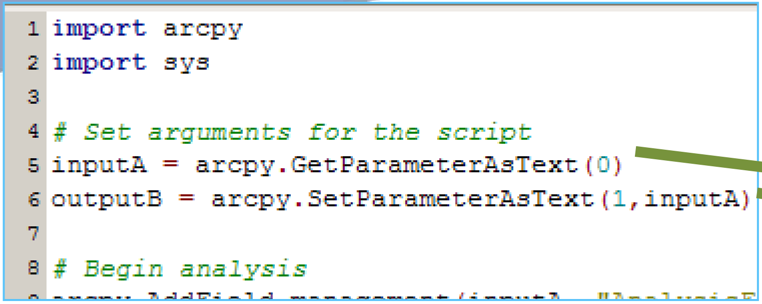
- Arguments are user-defined inputs to a script
  - –Values passed to script from user, instead of hard-coded
- Use GetParameter or GetParameterAsText to read arguments

Example 9

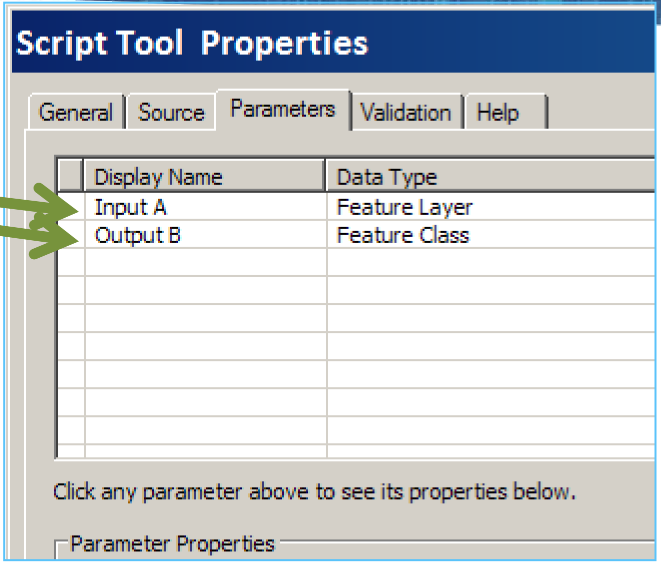


```
# Create variables from input arguments
inputFC = arcpy.GetParameterAsText(0)
outputFC = arcpy.GetParameterAsText(1)

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

Connecting parameters



```
import arcpy
import sys

# Set arguments for the script
inputA = arcpy.GetParameterAsText(0)
outputB = arcpy.SetParameterAsText(1,inputA)

# Begin analysis
arcpy.AddField_management(inputA, "AnalysisF
```

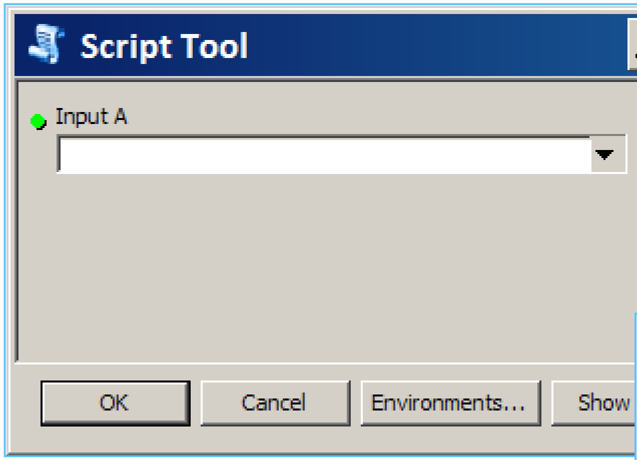


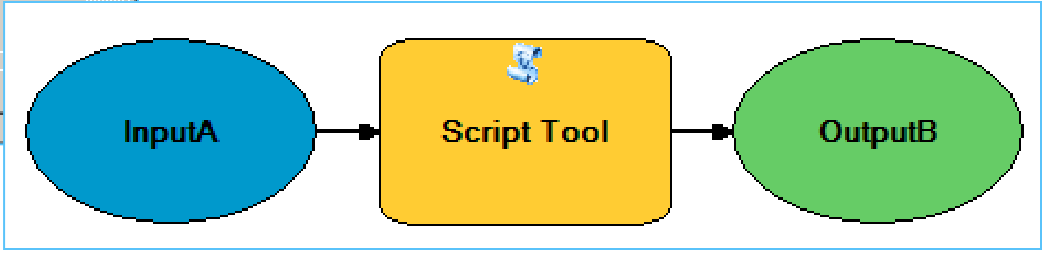
Script Tool Properties

Display Name	Data Type
Input A	Feature Layer
Output B	Feature Class

Click any parameter above to see its properties below.

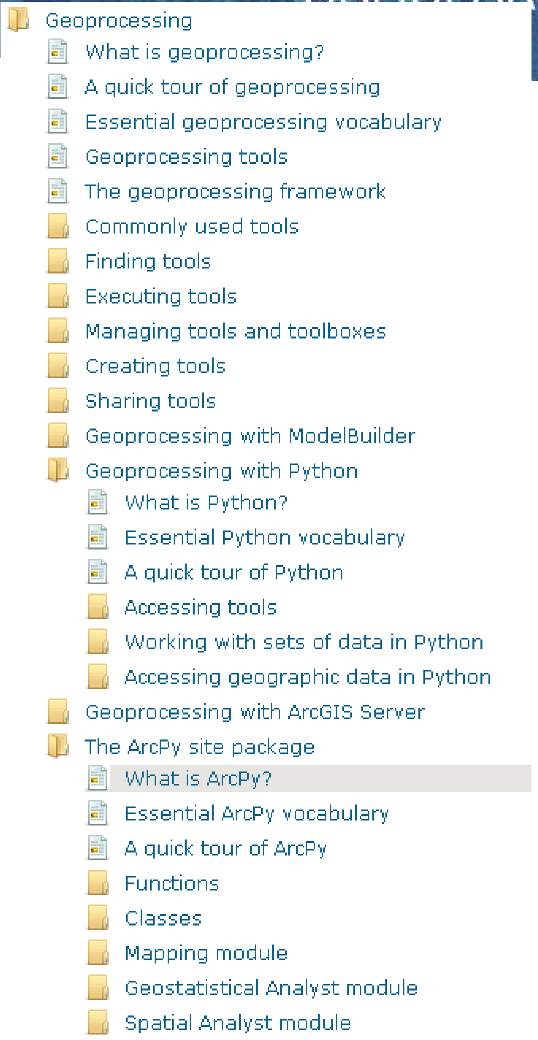
Parameter Properties





Python scripting resources

- ArcGIS Resource Center
  - resources.arcgis.com
  - Online documentation
  - Geoprocessing: script gallery, blog, tutorials, presentations
  - Web Course (free)
    - Using Python in ArcGIS Desktop 10
  - Python Scripting for ArcGIS
    - PA Zandbergen
- Python Resources
  - python.org
  - Learning Python by Lutz, et al.
  - The Python Standard Library by Example by Hellmann
  - diveintopython.org



- 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
    - What is Python?
    - Essential Python vocabulary
    - A quick tour of Python
    - Accessing tools
    - Working with sets of data in Python
    - Accessing geographic data in Python
  - Geoprocessing with ArcGIS Server
  - The ArcPy site package
    - What is ArcPy?
    - Essential ArcPy vocabulary
    - A quick tour of ArcPy
    - Functions
    - Classes
    - Mapping module
    - Geostatistical Analyst module
    - Spatial Analyst module