Getting Started with Python and ArcPy in ArcGIS

Description:

Building on Python Programming Basics for GIS Professionals but open to anyone with some programming experience, this workshop will expand on those skills to further use Python in ArcGIS. The workshop will focus on the ArcPy Python site package to expand geoprocessing capabilities with Python scripts. Participants will learn to build multiple standalone geoprocessing scripts covering different GIS tasks and workflows. The workshop will also cover how to create scripting tools in ArcGIS toolboxes for reuse and sharing. Participants will finish with the skills to explore more resources and options for utilizing Python in ArcGIS.

Specific Topics Include:

- Work with the ArcPy Python site package for ArcGIS
- Convert ModelBuilder models to Python scripts
- Build and share stand-alone Python script tools for automation
- Learn tips and tricks for valid script syntax and error handling

Computing and Software Needs:

- Your own laptop; computers will not be provided.
- Python IDE
  - PyScripter (http://sourceforge.net/projects/pyscripter/files/) (v. 2.6.0 **32-bit**) *NOTE:* The zip files contain portable versions of PyScripter. No installation is needed. Just unzip the archive and start using PyScripter.
  - JetBrains PyCharm Community Edition (https://www.jetbrains.com/pycharm/download/) *NOTE:* This IDE requires admin privileges to install.
  - IDLE is also shipped with ArcGIS for Desktop, so it can also be used.
- OSM Nominatim API module (https://pypi.python.org/pypi/nominatim/0.1) *NOTE:* Windows users will need to download the Windows installer .exe file.
- ArcGIS 10.1+ for Desktop (Standard or Advanced)

Instructors:

James Whitacre, GIS Specialist, University Library, University of Illinois at Urbana-Champaign

As the GIS Specialist in the Main Library at the University of Illinois at Urbana-Champaign, Mr. Whitacre's primary role is to provide GIS consultation and research assistance for faculty, staff, and students. Additionally, he teaches a myriad of GIS workshops for beginner to advanced users and helps manage the Library's GIS data and software assets. He is also a central resource for the GIS community on campus to promote the use of GIS in research. Mr. Whitacre holds a Master of Science in Geography and was previously the GIS Manager for the Carnegie Museum of Natural History.

Zoe Zaloudek, GIS Specialist, Illinois State Water Survey, Prairie Research Institute

Zoe earned her B.A. in Geography at the University of Illinois-Urbana in 2005. Shortly afterward, she began working at the Illinois State Water Survey (ISWS). As a GIS Specialist there, her work is currently split between FEMA’s Risk Mapping, Assessment and Planning program (Risk MAP), and the Midwestern Regional Climate Center (MRCC). She started using Model Builder in 2007 to better catalog and streamline workflows. Upon starting with the MRCC in 2010, she realized the power of Python scripts and hasn’t looked back!

I. Review of *Python Programming Basics for GIS Professionals*

- Quick review of morning workshop from Zoe
- Review of minimal skills needed for those who didn't attend first session

Exercise: Calculate Fields Using Python

- Python and VB Script can be used to calculate fields
- VB Script is the default, so Python needs to be set as the parser
- Simple expressions use the Expression box
- More complex calculations can use the Pre-Logic Script Code (window) or Code Block (tool) boxes
- Can be difficult to debug and check for errors

Field Calculator Window

We see that there is a field named 'AREA', but we don't know the units.

1. Open the Attribute Table for the 'Illinois Counties' Layer

```
Note the fields
```

2. Add a field with the following parameters:

```
Name: AreaTest
Type: Double
```

3. Right-click on the new field header and select Field Calculator

4. Double click a field to add it to the expression box

```
Make sure the VB Script radio button is selected
```

5. Select the Python radio button and add the same field

```
What is different?
Clear the expression box when done
```

6. Click on the 'About calculating fields' link at the bottom left

```
Notice the different examples
Find the 'Code smaples--geometry' section
These expressions can be used in lieu of the 'Calculate Geometry' window accessed thro
ugh the attribute table
These expressions can also be used in the Calculate Field tool
```

6. Enter the following code in the expression box

*Note: The following code is intended to be used to calculate an attribute table field in ArcGIS using the Field Calculator window or Calculate Field tool.

In [ ]:
```
# First try

!Shape.area@SQUAREMILES!

# How clase are the vales?

# Now try

!Shape.geodesicArea@SQUAREMILES!
```

Field Calculator Window

Now we know the units, but we want to calculate the population density

1. Delete the 'AreaTest' field

Click 'Yes' on the warning

2. Add a field with the following parameters:

Name: DensityPerSqmi
Type: Double

3. Calculate the field using Python

Clear the expression box
Make sure the Python radio button is selected
Enter the code below

In [ ]:
```
!ACSTOTPOP! / !AREA!
```

Field Calculator Window

Now we would like to have two more fields for a short name and a long name

1. Add two fields with the following parameters:

```
Field 1:
    Name: NameLong
    Type: Text
    Length: 100

Field 2:
    Name: NameShort
    Type: Text
    Length: 50
```

2. Calculate the fields using Python

```
Clear the expression box
Make sure the Python radio button is selected
Enter the code below
```

In [ ]:
```
# NameLong calculation

!NAME! + ", " !STATE_NAME!

# NameShort calculation
# Note the first parameter...what do you notice?

!NAME!.replace(" County", "")
```

Field Calculator Window

Now we would like to classify the median income levels

1. Add a field with the following parameters:

```
Name: IncomeLevels
Type: Text
Length: 50
```

2. Calculate the fields using Python

```
Clear the expression box
Make sure the Python radio button is selected
Check the 'Show Codeblock' box
Enter the code below
```

3. When you have completed the calculation in the Field Calculator window, open the result from the Results window

```
How does this look different?
```

In [ ]:
```
# Pre-Logic Script Code:

def income_level(medincome):
    if medincome < 40000:
        return "Low Income"
    elif 40000 <= medincome < 50000:
        return "Medium Income"
    else:
        return "High Income"


# Expression:

income_level(!ACSMEDHINC!)
```

II. Introduction to ArcPy

What is ArcPy?

> ArcPy is a site package that builds on (and is a successor to) the successful arcgisscripting module. Its goal is to create the cornerstone for a useful and productive way to perform geographic data analysis, data conversion, data management, and map automation with Python.
>
> This package provides a rich and native Python experience offering code completion (type a keyword and a dot to get a pop-up list of properties and methods supported by that keyword; select one to insert it) and reference documentation for each function, module, and class.
>
> The additional power of using ArcPy within Python is the fact that Python is a general-purpose programming language. It is interpreted and dynamically typed and is suited for interactive work and quick prototyping of one-off programs known as scripts while being powerful enough to write large applications in. ArcGIS applications written with ArcPy benefit from the development of additional modules in numerous niches of Python by GIS professionals and programmers from many different disciplines.

From http://desktop.arcgis.com/en/desktop/latest/analyze/arcpy/what-is-arcpy-.htm (http://desktop.arcgis.com/en/desktop/latest/analyze/arcpy/what-is-arcpy-.htm)

What is a Site Package?

- Like a module, but contains a collections of modules, functions, and classes
- Site packages and modules both add functionality to Python

Importing ArcPy

- To work with ArcPy, you must import in your code
- Once imported, you can now use all the modules, functions, tools, and classes

In [ ]:
```
import arcpy

# Always import the ArcPy site package and other modules first

# this may take a while if you are working outside of ArcGIS
```

Utilizing the ArcPy ArcGIS for Desktop Help Documentation

- Desktop vs. Python window vs. Online documentation
  - Desktop help can be opened through the Help menu or from the tool interface
  - Python window help and syntax panel on the right (more on this later)
  - http://desktop.arcgis.com/en/documentation/ (http://desktop.arcgis.com/en/documentation/)
- Understanding geoprocessing tool help with Python and ArcPy
  - Includes tool syntax and sample codes for the Python window and stand-alone scripts
  - Includes detailed explanations of each parameter
- ArcPy help: http://desktop.arcgis.com/en/desktop/latest/analyze/arcpy/what-is-arcpy-.htm (http://desktop.arcgis.com/en/desktop/latest/analyze/arcpy/what-is-arcpy-.htm)
  - Contains help on functions and class not listed in the tool documentation
  - Note the additional modules

III. Working with the Python Window in ArcMap

Download exercise data from: https://uofi.box.com/ILGISA-Python (https://uofi.box.com/ILGISA-Python)

Open the ArcMap Document 'Illinois.mxd'

Open the Python Window in ArcMap

- Replaces the Command Line for earlier releases of ArcGIS
- Allows for running geoprocessing tools while also taking advantage of other Python modules and librarires
- Can be used for single-line code (e.g. Used instead of ArcToolbox to access tools)
- Can be used for testing syntax and longer, more complex scripts that might be used for automating workflows

Notes About the Python Window

- Left side is the interactive Python interpreter panel
  - This is where code is entered
  - Note the three greater-than symbols (>>>)
- Right side is the help and syntax panel
- Code is generally executed one line at a time and displayed immediately. Exceptions include:
  - Multi-line constucts (such as IF statements)
  - Pressing SHIFT or CTRL + ENTER at the end of a line of code (you may need to hit ENTER a few times when you are ready to run the code)
- Other Advanatages of the Python window
  - Autocompletion
  - Conditional and Iteration executuion
  - Scripts can be saved and reused or opened with another Python IDE

Exercise: Describing Data in the ArcMap Python Window

Describing Data with ArcPy

- Describing data allows to learn more about the data we are working with
- Describe functions are useful when scripts may be dependent on the type of data being used
- Good for controlling script flow and valdating parameters
- Many property groups and some prperties exist for only some types

See ArcGIS Documentation: http://desktop.arcgis.com/en/desktop/latest/analyze/arcpy-functions/describe.htm (http://desktop.arcgis.com/en/desktop/latest/analyze/arcpy-functions/describe.htm)

System Paths vs. Catalog paths

- System paths are recognized by the Windows operating system
  - Files in folders
  - Ex. Shapefiles and rasters
- Catalog paths are only recognized by ArcGIS
  - Used for feature classes and other data in geodatabases
  - Geodatabases could be file (.gdb), personal (.mdb) or enterprise (.sde)
  - Contain 2 parts:
    - Workspace - could be the geodatabase root or a feture dataset
    - Base name - the feature class, raster, or other file types that can be saved in geodatabases
  - Requires the programmer to be aware of the context inwhich the path is being used

Type, do not copy, the following code into the Python window

Note: This code is intended to only be run in the Python window of ArcMap

In [ ]:
```
# Step 1:

import arcpy

file_name = r'C:\Data\CSV\JeopardyContestants_LatLon.csv'
print arcpy.Exists(file_name)

# What kind of path is this? System or Catalog?
# What is the result?

# Try again, this time, but type the up arrow to reload the last code
# Replace the '...' with the location where you placed your downloaded data
# I also have a cool trick...drag, drop, and roll!

file_name = r'C:\...\Data\CSV\JeopardyContestants_LatLon.csv'
print arcpy.Exists(file_name)

# Does it exist? Keep trying until you get the path correct and the result return
s TRUE

print arcpy.Describe(file_name).dataType
# What type of data is this?
```

In [ ]:
```
# Step 2:

# Replace the '...' with the location where you placed your downloaded data

feature_class = r'C:\...\Data\Illinois.gdb\Illinois_Counties'
# What kind of path is this?

print arcpy.Exists(feature_class)

print arcpy.Describe(feature_class).dataType
# What type of data is this?
```

In [ ]:
```
# Step 3:

layer_name = 'Illinois Counties'

# What kind of path is this? System or Catalog?

print arcpy.Exists(layer_name)

lyr_desc = arcpy.Describe(layer_name)
# Note how this Describe function and variable look different than before...
# We can have describe objects become variables to be used later

print lyr_desc.dataType
print lyr_desc.dataElement.dataType

# What is the different between the two print statements?
# Lets look at the help:
# http://desktop.arcgis.com/en/desktop/latest/analyze/arcpy-functions/describe-object-properties.htm
# http://desktop.arcgis.com/en/desktop/latest/analyze/arcpy-functions/layer-properties.htm
```

In [ ]:
```
# Step 4:

lyr_datatype = lyr_desc.dataElement.dataType

lyr_path = lyr_desc.path

lyr_basename = lyr_desc.baseName

lyr_spatialref = lyr_desc.spatialReference.name

lyr_count = arcpy.GetCount_management(layer_name)

print "{0} is a {1} stored at {2} with a file name of {3} with the {4} coordinate system and contains {5} features.".format(layer_name, lyr_datatype, lyr_path, lyr_basename, lyr_spatialref, str(lyr_count))
```

Exercise: Listing Data in the ArcMap Python Window

- Helps with batch processing (primary reason for developing scripts)
- Lists are good for interating processes (another primary reason for developing scripts)
- Easy inventory of data (e.g. list of fields in a table; feature classes in a geodatabase)

Type, do not copy, the following code into the Python window

Note: This code is intended to only be run in the Python window of ArcMap

In [ ]:
```
# Step 1:

# More on environmental settings later
# Replace the '...' with the location where you placed your downloaded data
arcpy.env.workspace = r'C:\...\Data\CSV'

print arcpy.ListFiles()

arcpy.env.workspace = r'C:\...\Data\Illinois.gdb'

print arcpy.ListFiles()

print arcpy.ListFeatureClasses()

# What is the difference between the two list functions?
```

In [ ]:
```
# Step 2:

print arcpy.ListFields("Illinois Counties")

# What is the result? Can you understand it?
# What is actually being output in this case? (Hint: What type of object?)

fields = arcpy.ListFields("Illinois Counties")
for field in fields:
    print field.name
```

In [ ]:
```
# Step 3:

field_list = [field.name for field in arcpy.ListFields("Illinois Counties")]

print field_list

# How does this result look differently than above?

test_field = "OBJECTID"

if test_field in field_list:
    print "{0} exists".format(test_field)
else:
    print "{0} does not exist".format(test_field)

# Repeat the above code using a test field that you know doesn't exist to test th
e 'else:' statement code
# Does it work?
```

Exercise: Working with Geoprocessing Tools in the ArcMap Python Window

- ArcPy gives access to all tools in ArcToolbox in addition to nontool functions
- Working in the Python window allows for testing ideas and seeing how things work
- Tested scripts can also be saved for reuse in the Python window or in an IDE.

Type, do not copy, the following code into the Python window

Note: This code is intended to only be run in the Python window of ArcMap

In [ ]:
```
# Step 1:
import arcpy

layer_name = "Illinois Counties"

expression = "ACSMEDHINC <= 40000"

arcpy.SelectLayerByAttribute_management(layer_name, "NEW_SELECTION", expression)

arcpy.SelectLayerByAttribute_management(layer_name, "CLEAR_SELECTION")
```

In [ ]:
```
# Step 2:

# Ensure that a default geodatabase is set

disolve_felds = ["STATE_NAME", "ST_ABBREV"]

# New drag, drop, and roll trick...

arcpy.Dissolve_management(layer_name, "Illinois", disolve_felds, "", "SINGLE_PAR
T", "DISSOLVE_LINES")

arcpy.PolygonToLine_management("Illinois", "Illinois_Boundary")
```

In [ ]:
```
# Step 3:

# Replace the '...' with the location where you placed your downloaded data

csv_file = r"C:\...\Data\CSV\JeopardyContestants_LatLon.csv"

arcpy.CopyRows_management(csv_file, "JeopardyContestants_Table")

arcpy.MakeXYEventLayer_management("JeopardyContestants_Table", "lon", "lat", "Jeo
pardy Contestants")

arcpy.Select_analysis("Jeopardy Contestants", "JeopardyContestants", '"lat" IS NO
T  NULL OR "lon" IS NOT NULL')

arcpy.Buffer_analysis("JeopardyContestants", "JeopardyContestants_Buffer", "5 Mil
es", "FULL", "ROUND", "ALL", "", "GEODESIC")
```

Your turn:

1: Write code to clip the buffer to the Illinois state layer

Be careful with what tool you choose...

2: Write code to intersect the new buffer to the Illinois Counties layer

You might need to check the tool's help documentation...

3: Save your work as a text file (.txt) in the 'Scripts' folder

Open the text file and view it in a text editor (like Notepad)

4: Save your work as a Python file (.py) in the 'Scripts' folder

Clear the Python window in ArcGIS and load the Python file back into the Python window
Try running the whole script at once

Break!!

IV. Python vs. ModelBuilder

Scenario:

You developed a model in ModelBuilder for a common geoprocessing workflow that many people use in your organization. However, your fellow GIS users are asking for some more sofisticated functionality that ModelBuilder has a tough time handling. Therefore, you think Python will do the trick! How can you assess this?

Python	**ModelBuilder**
Textual programming language	Visual Programming language
Easy to learn with very flexible structure	Relativiely easier to learn for GIS beginners, but restrictive structure
Python scripts can be used in ModelBuilder	ModelBuilder can export to Python (but not vice versa)
Lower-level geoprocessing tasks (e.g. cursors)	Lower-level geoprocessing tasks may not be possible
More advanced error handling	Errors handled by the tools in the model
Better for iterations and loops	Can do iterations and loops, but can be difficult to setup properly
Can use other Python modules and wrap other software (e.g. R)	Restricted by ArcToolbox tools
Can run Python scripts outside of ArcGIS	ModelBuilder works when ArcGIS is running

Exercise: ModelBuilder to Python Script

Open the Model Named 'Illinois in Jeopardy' in edit mode (right-click > Edit...)

- Note the workflow...look familiar?
- Double check the outputs to ensure they are pointing to the correct location
- If not, clisk the checkmark icon in the toolbar

Convert the Model to a Python Script

- Click Model > Export > To Python Script...
- Save it in the 'Scripts' folder

Open the new script in the IDE

- What do you notice with how it is laid out and written?
- How is any of the code different/similar to how we have written code up to now?
- Can you run it from the IDE? Go ahead and try...
- What modifications would you need to make for it to work as a stand-alone script?

ModelBuilder to Python Script Example:

In [ ]:
```
# -*- coding: utf-8 -*-
# ---------------------------------------------------------------------------
# IllinoisInJeopardy.py
# Created on: 2015-09-11 12:14:22.00000
#   (generated by ArcGIS/ModelBuilder)
# Usage: IllinoisInJeopardy <Jeopardy_CSV_File>
# Description:
# Illinois is in Jeopardy!
# ---------------------------------------------------------------------------

# Import arcpy module
import arcpy

# Script arguments
Jeopardy_CSV_File = arcpy.GetParameterAsText(0)
if Jeopardy_CSV_File == '#' or not Jeopardy_CSV_File:
    Jeopardy_CSV_File = "C:\\...\\Data\\CSV\\JeopardyContestants_LatLon.csv" # pr
ovide a default value if unspecified

# Local variables:
JeopardyContestants_Table = "C:\\...\\Data\\Illinois.gdb\\JeopardyContestants_Tab
le"
Jeopardy_Contestants = "Jeopardy Contestants"
JeopardyContestants = "C:\\...\\Data\\Illinois.gdb\\JeopardyContestants"
JeopardyContestants_Buffer = "C:\\...\\Data\\Illinois.gdb\\JeopardyContestants_Bu
ffer"

# Process: Copy Rows
arcpy.CopyRows_management(Jeopardy_CSV_File, JeopardyContestants_Table, "")

# Process: Make XY Event Layer
arcpy.MakeXYEventLayer_management(JeopardyContestants_Table, "lon", "lat", Jeopar
dy_Contestants, "GEOGCS['GCS_WGS_1984',DATUM['D_WGS_1984',SPHEROID['WGS_1984',637
8137.0,298.257223563]],PRIMEM['Greenwich',0.0],UNIT['Degree',0.017453292519943
3]];-400 -400 1000000000;-100000 10000;-100000 10000;8.98315284119522E-09;0.00
1;0.001;IsHighPrecision", "")

# Process: Select
arcpy.Select_analysis(Jeopardy_Contestants, JeopardyContestants, "\"lat\" IS NOT
NULL OR \"lon\" IS NOT NULL")

# Process: Buffer
arcpy.Buffer_analysis(JeopardyContestants, JeopardyContestants_Buffer, "5 Miles",
"FULL", "ROUND", "NONE", "", "GEODESIC")
```

V. Automating Geoprocesses with Stand-alone Scripts

Senario:

Your supervisor is a Jeopardy! nut job and he watches it every night! To make matters worse, he has compiled a list of all of the Jeopardy! contestants on his nerdy website (It's probably a Blogspot site from 2003!). He's pretty good at Python (but is awful at GIS!), so he used the Nominatim module to find latitude and longitude values for contestants' home towns. He shared his CSV file with you containing all the contestants from the state of Illinois and has asked you to map them out and determine how wealthy their home county is and how many contestants came from each county.

Remember that time he asked you last year to do this? Well, he's asking again becasue he wants an updated list! This time he's discovered the beauty of geodatabases, so he definitely wants a feature class this time, unlike the shapefile from before (but, next time he might forget about geodatabases...). And you suspect he will ask you again, only he will want all of his contestants mapped!

Since you took that GIS Python course, you decide it might best for you write a script...and one that allows him to choose which format, shapefile or a FGDB feature class. You also need to impressive your boss so your job isn't in Jeopardy! (Is that going too far...?).

Stand-alone Scripts Exercises

- Copying Python code snippets from ArcMap
- Convert CSV files to Feature Class/Shapefile. (Data Management process)
  - Data preparation for analysis
  - Dataset mashups (??)
  - Iterating processes

Exercise: Planning a Stand-alone Script (i.e. Pseudo Code)

What is Pseudo Code?

- A plain-language explanation or outline of the script workflow
- Helps to organize and plan the script workflow
- Allows others reading the code to understand more clearly what is being done
- Can be written directly in the code using comments (e.g. '#')

Note: This is just one of many methodologies...

- You can modify this as much as you want
- Find a stucture that works for you
- Keep it simple

General Components of a Script

In [ ]:
```
# Script Info and Metadata

    #-------------------------------------------------------------------------
---
    # Name:        <ScriptName>.py
    # Purpose:     <Some important purpose.>
    #
    # Author:      <Name, Your>
    #
    # Created:     11/09/2015
    # Copyright:   (c) <Your Name> 2015
    # Licence:     <your licence>
    #-------------------------------------------------------------------------
---

# Import Modules
    # arcpy and other modules that may help you complete your code

# Parameters
    # What inputs and outputs will the user have contorl over?

# Environments
    # Gets the house in order for how data and outputs will be dealt with

# Variables
    # Anything that is not a parameter thay may be used multiple times
    # These may also be dispersed throughout the script

# Processes
    # The code that dictates the workflow
```

Pseudo Code for the Stand-alone Script

Create a new script in you IDE and save it to the 'Scripts' folder

Add the pseudo code below as comments int he script

In [ ]:
```
# Processes

# Copy CSV table to geodatabase stand-alone table

# Convert table to XY Layer

# Make the XY Layer a permanent point feature class or shapefile based on user in
put

# Intersect the points with counties layer and include all fields from both layer
s

# Sum the number of contestants from each county and total winnings and output as
a table
```

What are the modules and user parameter inputs and outputs?

Add the following pseudo code and parameter variables to your script ***above the previous pseudo code***

In [ ]:
```
# Import Modules
import arcpy


# Parameters

# Inputs:
# CSV File with XY coordinates
csv_file = r'C:\...\Data\CSV\JeopardyContestants_LatLon.csv' # Replace the '...'
with the location where you placed your downloaded data

# Fields that holds the lat and long values
lon = 'lon' # this is the x value
lat = 'lat' # this is the y value

# Spatial reference CSV XY coordinates
csv_spatialref = arcpy.SpatialReference('WGS 1984')

# Counties feature class
counties = 'Illinois_Counties'

# Outputs:
# Contestants Intersect Feature Class or Shapefile
fc_output = 'JeopardyContestants_Income'

# Sum of Contestants by County Table
tbl_output = 'JeopardyContestants_ByCounty'


# Environments
```

Exercise: Using Environment Settings (See pg. 110 in Python Scripting for ArcGIS)

Now we're ready to write our script! But first, we need to do some house keeping!

What are Environmental Settings?

http://desktop.arcgis.com/en/desktop/latest/tools/environments/what-is-a-geoprocessing-environment.htm (http://desktop.arcgis.com/en/desktop/latest/tools/environments/what-is-a-geoprocessing-environment.htm)

- Hidden parameters that influence how tools run
- Fundamental in controlling the geoprocessing workflow
- Exposed as properties and set using the **env** class
- Corresponds to the 'Environments...' found in tools
- There is a hierarchy with how ArcGIS deals with environmental settings. See http://desktop.arcgis.com/en/desktop/latest/tools/environments/environment-levels-and-hierarchy.htm (http://desktop.arcgis.com/en/desktop/latest/tools/environments/environment-levels-and-hierarchy.htm)

In [ ]:
```
# Try this code in the Python window to see the result...

print arcpy.ListEnvironments()
```

Add a workspace as an environment

Setting the workspace environment allows us to shorten some of the input parameter variables. For example, by having the workspace set, arcpy automatially knows to save the **fc_output** parameter of **'JeopardyContestants_Income'** in the defined workspace. If we didn't define the workspace, we would need to set the parameter as **fc_output = r'C:\...\Data\Illinois.gdb\JeopardyContestants_Income'**. Therefore, this is a shortcut!

We also may be overwriting our outputs a lot, so we will add the evnvironment also.

Add the following code below the **# Environments** comment

In [ ]:
```
# Environments

arcpy.env.workspace = r'C:\...\Data\Illinois.gdb' # Replace the '...' with the lo
cation where you placed your downloaded data
arcpy.env.overwriteOutput = True
```

Exercise: Adding geoprocessing tools to your script

1. Typing in the IDE

- Many IDEs will auto complete
- Type the following code in the IDE script after the matching pseudo code:

In [ ]:
```
# Copy CSV table to geodatabase stand-alone table

xy_table = 'JeopardyContestants_Table'

arcpy.CopyRows_management(csv_file, xy_table)
```

2. Why type when you can copy from an exisitng script!

- Scipts can be copied, but make sure you check the variables and parameters to enuser they are correct (I took care of this for you)
- Copy the following code to the IDE script after the matching pseudo code:

In [ ]:
```
# Convert table to XY Layer

xy_layer = 'Jeopardy Contestants'

arcpy.MakeXYEventLayer_management(xy_table, lon, lat, xy_layer, csv_spatialref)
```

3. Test the tool in ArcMap, then 'Copy as Python Snippet'

- Watch the demonstration for **`# Make the XY Layer a permanent point feature class or shapefile based on user input`**
- Note how the code looks
- Clean the code to make it more readable and useable
- Add any variables that may be useful later

4. On your Own, add the code for last two processes using any method

- Have you created code for either process?
- Which tools will you use?
- If you are stuck, an example of the final script is in the 'Answers\Scripts' folder
- Run your script to test it

Exercise: Handling Errors and Debugging

Three types of errors in Python

- Syntax errors - prevent code from running
- Exceptions - code will stop running midprocess
- Logic errors - code will run, but produces undesired results

What is debugging?

- Methodological process for finding errors in the script
- Debugging methods don't usually tell you why, but rather where the issue is occurring
- Examples include:
  - Carefully reviewing error messages (we have done this a little bit...)
  - Adding **`print`** statements to flag certain processes (we will do this!)
  - Selectively commenting out code while testing (you might shoose to do thist)
  - Using a Python IDE debugger (we won't cover this...sorry)

Add some `print` statements to the code

- Add a descriptive print statement after each process
- If a print statement does not execute, then that is the process to check for issues
- See the example below:

In [ ]:
```
# Copy CSV table to geodatabase stand-alone table

xy_table = 'JeopardyContestants_Table'

arcpy.CopyRows_management(csv_file, xy_table)

print arcpy.Describe(csv_file).name + ' converted to table.'
```

Handeling Errors Using conditional statements

- Replace the following code for the identified process
- Note the indents after the **`if:`** statement
- Try to test the error by changing a field name

In [ ]:
```
# Processes
# ...

# Convert table to XY Layer

field_list = [field.name for field in arcpy.ListFields(xy_table)]

if lat in field_list and lon in field_list:
    print '{0} and {1} fields exist'.format(lat, lon)

else:
    raise Exception('{0} or {1} fields DO NOT exist. Check field names.'.format(l
at, lon))

xy_layer = 'Jeopardy Contestants'

arcpy.MakeXYEventLayer_management(xy_table, lon, lat, xy_layer)

print 'XY Layer created'
```

Handeling Errors Using `try-except`

- Add the follow **`try-except`** statement before executing the processes
- Note the indents after the **`try:`** statement
- Try to test the error

In [ ]:
```
# Processes

try:

    # Indent all processes...

    # Copy CSV table to geodatabase stand-alone table

    xy_table = 'JeopardyContestants_Table'

    arcpy.CopyRows_management(csv_file, xy_table)

    print arcpy.Describe(csv_file).name + ' converted to table.'

    # Convert table to XY Layer

    field_list = [field.name for field in arcpy.ListFields(xy_table)]

    if lat in field_list and lon in field_list:
        print '{0} and {1} fields exist'.format(lat, lon)
    else:
# Change this line also
        print '{0} or {1} fields DO NOT exist. Check field names.'.format(lat, lo
n)

    #.................

    # Sum the number of contestants from each county and total winnings and outpu
t as a table

    arcpy.Frequency_analysis(fc_output, tbl_output, 'NAME_1', 'totalwinnings')

    print 'Contestants by county summary complete. Script complete.'

except arcpy.ExecuteError:
    print arcpy.GetMessages(2)
    print 'Could not complete script'

except:
    print 'Could not complete script'
```

VI. Adding Python Script Tools to Toolboxes

Senario:

Your supervisor has finally learned to use ArcGIS! After he was so impressed by your script he finally recognized your value and had you teach him GIS the last 6 months (and, no, no raise for you...yet!). Now, he's forgotten Python and he only knows how to use the tools in ArcGIS, so he is looking for something similar to map his contestants. Now you need to figure out how to get that script into an ArcGIS toolbox for him to use (and maybe he will give you a raise!...or he will fire you because you keep coding yourself out of job!!).

Creating and sharing scripts with custom ArcGIS toolboxes

Two ways to run a script:

- Stand-alone (as we have done)
- Script tool in ArcGIS

Advantages to using script tools over stand-alone scripts:

- Allows for integration into ArcGIS
- Can be run from ArcToolbox
- Can be used in ModelBuilder
- Can be used in other scripts
- Relativiely easy to create script tools
- Very useful for traditional ArcGIS users using the familiar tool interface
- Reduces errors (and error checking) by specifying parameters and contoling user input
- Easy to share with other users with the right set up
- Using script tools requires no knowledge of Python

Steps to creating a tool

- Create a stand-alone script (DONE!)
- Modify the script to seamlessly pass inputs and outputs through the script tool (More coding...you got this!...)
- Create a custom toolbox to store the tool (I created one for you already!)
- Add a script tool to the custom toolbox and follow the Add Script wizard (Ok, we haven't done this...)

Note: You will also notice options for Python Toolboxes. These are more advanced and allow additional capabilities. However, they are beyond the scope of this workshop.

Help Documentation: http://desktop.arcgis.com/en/desktop/latest/analyze/creating-tools/a-quick-tour-of-creating-tools-in-python.htm (http://desktop.arcgis.com/en/desktop/latest/analyze/creating-tools/a-quick-tour-of-creating-tools-in-python.htm)

Exercise: Modify Script Inputs and Outputs to Be Used in the Script Tool

- Inputs and outputs use the **`arcpy.GetParameterAsText(<index>)`** funtion to pass variable between the tool and the script.
- All inputs and outputs must be modified to utilize tools

1. In the IDE, save a copy (i.e. save as...) of your stand-alone script and add '_ScriptTool.py' to the name

2. Modify the scripts code to align with the code below

- Part 1: Inputs and outputs converted to user input parameters
- Part 2: Environments are not managed by the tool

In [ ]:
```
# Parameters

# Inputs:
# CSV File with XY coordinates
csv_file = arcpy.GetParameterAsText(0) #r'C:\...\Data\CSV\JeopardyContestants_Lat
Lon.csv' # Replace the '...' with the location where you placed your downloaded d
ata

# Fields that holds the lat and long values
lon = arcpy.GetParameterAsText(1) # 'lon' # this is the x value
lat = arcpy.GetParameterAsText(2) #'lat' # this is the y value

# Spatial reference CSV XY coordinates
csv_spatialref = arcpy.GetParameterAsText(3) # arcpy.SpatialReference('WGS 1984')

# Counties feature class
counties = arcpy.GetParameterAsText(4) #'Illinois_Counties'

# Outputs:
# Contestants Intersect Feature Class or Shapefile
fc_output = arcpy.GetParameterAsText(5) # 'JeopardyContestants_Income'

# Sum of Contestants by County Table
tbl_output = arcpy.GetParameterAsText(6) # 'JeopardyContestants_ByCounty'


# Environments

#arcpy.env.workspace = r'C:\...\Data\Illinois.gdb' # Replace the '...' with the l
ocation where you placed your downloaded data
#arcpy.env.overwriteOutput = True
```

Exercise: Adding a Script to a Toolbox

- Inputs and outputs are actually passed as text strings
- The script tool allows for those text strings to be pre-validated before using them in the script
- This is essentially a form of error handeling

1 .In the Catalog window in ArcMap, right-click on the custom toolbox and hover over 'Add' and click 'Script...'

This will activate the Add Script wizard

2. Naming the script tool

- Name: IllinoisInJeopardyScript
- Label and Description: Illinois in Jeopardy Script (Final)
- Check 'Store relative path names...'
- Check 'Always run in foreground' (for now...)

3. Adding the script file (.py)

- Add the script ending in '_ScriptTool.py'.
- Check 'Run Python script in process'

4. Adding inputs and outputs

- Reference the script for all inputs and outputs
- Add descriptitive names - spaces are ok
- Choose the correct and expected data types for the inputs and outputs
- Set up the appropriate parrameter properties

Hints:

- For the CSV file, note that there are many data types that could be used.
  - What is the best one?
  - You may need to experiment
  - It is helpful to check the help documentation of the tools you are using to help determine data types
- For the lat and lon fields:
  - Filter the input to number fields only
  - You can read in the fields from the CSV file to create a drop-down list of usable fields
- Yes, there is a spatial reference data type!
- For the county layer with income, there is more than one data type choice. Which is the best?
- Don't forget to change the Direction property for outputs

5. Run and test the script tool

- What messages were returned?
- Learn how to add messages to the tool output: http://desktop.arcgis.com/en/desktop/latest/analyze/creating-tools/writing-messages-in-script-tools.htm

(http://desktop.arcgis.com/en/desktop/latest/analyze/creating-tools/writing-messages-in-script-tools.htm)
- Hint on Messages: Look for the print statements
- Try outputting a shapefile and DBF table in a folder
- Did it work? If it didn't, how can you fix this?

6. Edit and debug the tool as needed

- Shapefiles???
- Are there any outputs you don't want to keep?
- Try adding code to delete those unwanted feature classes
- Or learn about the `in_memory` workspace: http://desktop.arcgis.com/en/desktop/latest/analyze/modelbuilder/the-in-memory-workspace.htm (http://desktop.arcgis.com/en/desktop/latest/analyze/modelbuilder/the-in-memory-workspace.htm)
- If there are no errors now, just wait until people start using it a lot...they'll find them if they exist! This is an iterative process

7. Share the tool

- Make sure the script and/or script folder is easily accessible by the toolbox and tool (i.e. let relative paths work for you)...Zip and ship!
- Try embedding the script

Help Documentation: http://desktop.arcgis.com/en/desktop/latest/analyze/sharing-workflows/a-quick-tour-of-sharing-tools.htm (http://desktop.arcgis.com/en/desktop/latest/analyze/sharing-workflows/a-quick-tour-of-sharing-tools.htm) Oldie, but a goodie: http://help.arcgis.com/en/arcgisdesktop/10.0/help/#/A_structure_for_sharing_tools/005700000004000000/

Moving Forward

Exapnd on today's scripts

- Learn about modifying the script tool documentation and metadata: http://desktop.arcgis.com/en/desktop/latest/analyze/creating-tools/a-quick-tour-of-documenting-tools-and-toolboxes.htm (http://desktop.arcgis.com/en/desktop/latest/analyze/creating-tools/a-quick-tour-of-documenting-tools-and-toolboxes.htm)
- Try to combine the morning and afternoon final scripts to create a 'mega' script
- Create a (FREE!) geocoder using Nominatim and ArcPy

Start using python in your everyday workflows!

- Resist the temptation to fall back on ModelBuilder
- Read Python and ArcPy documentation...and read it again!
- Collaborate with and try to teach colleagues
- Get involved with a local development group...Esri has some

```
In [ ]: print "Goodbye!"
```