Beginning GIS Programming Using ArcGIS 10.0 and Python

Nick Santos, Josh Viers, and Anna Fryjoff-Hung

Feb 2013

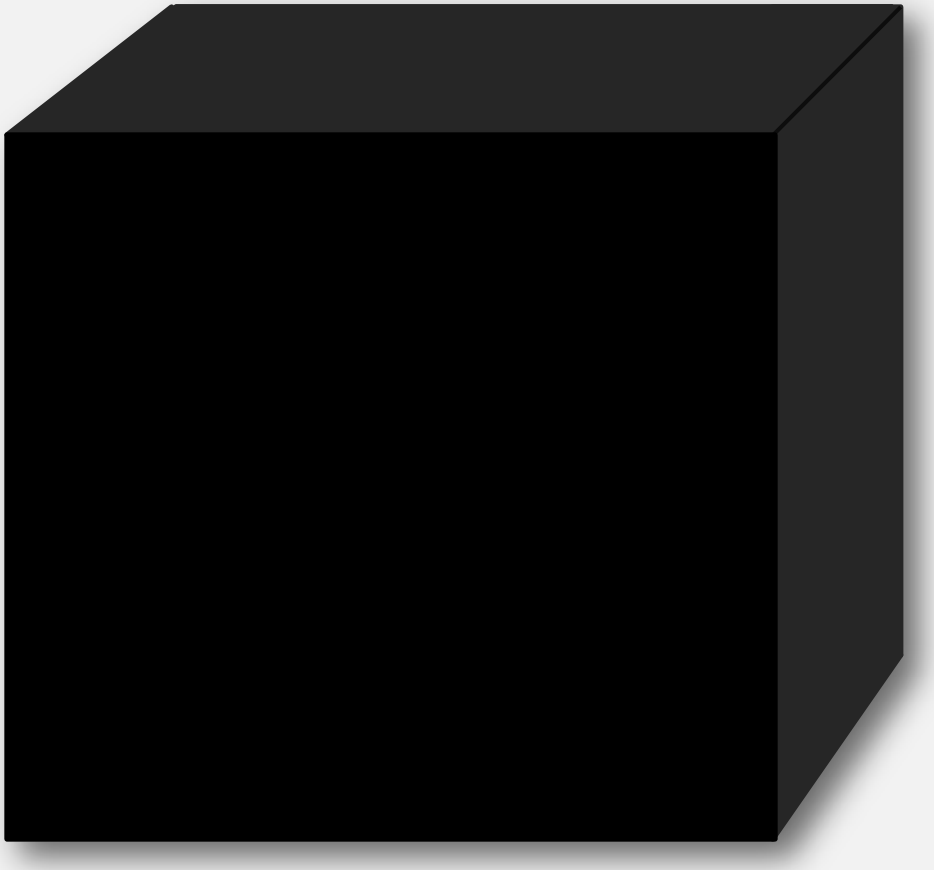
University Extension

Contact: nrsantos@ucdavis.edu

This presentation will available online at
http://watershed.ucdavis.edu/resources/python-for-gis

Most People's Idea of a Program

Or, these days, an "app"

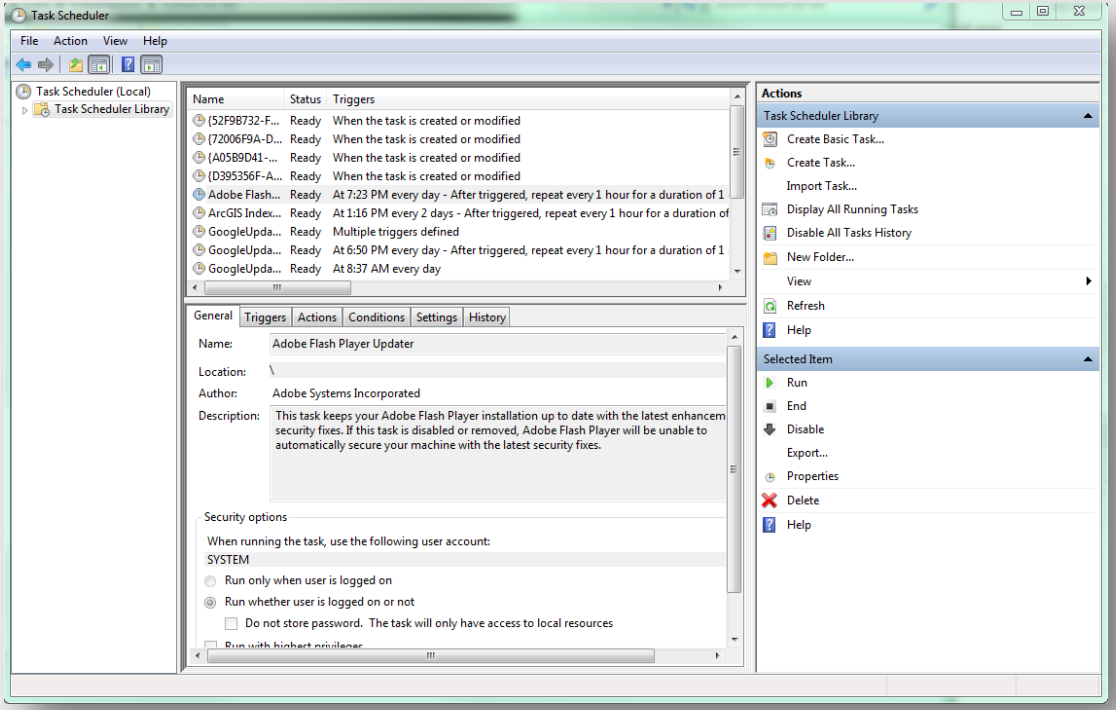


A *maybe* more accurate picture

TinkerToy Source: Wikimedia Commons

So, let's write a program right now

- **Objective**: Every time I log in to my computer, I want ArcMap to be opened (to remind me I should be working)
- **"Language"**: Windows Task Scheduler



Your first program

- Start->Type “Task Scheduler” and click on the result
- Right Click “Task Scheduler Library”, Select **Create Task** and put a name in the box that pops up
- Click triggers, then New
- Select Begin the Task: “**On Workstation Unlock**”
  - Click the radio button for “Specific User”, with the default acceptable, then click OK
- Select Actions, then New
  - Ensure “Start a program” is selected, then click browse and find your ArcMap executable, then Click OK

Program Testing

- Press ⊞ + L to lock your workstation
- Log back in.
- ArcMap should open. If it doesn't let's **debug** it together.

Programming Steps

Problem-Solving Phase

1. Analysis and specification.
   (Define problem and what solution must do.)
2. General solution (algorithm).
   (Develop logical sequence of steps to solve problem.)
3. Verify.
   (follow steps - by hand.)

Implementation Phase

Debug.

1. Specific solution (program).
   (Translate algorithm to code.)
2. Test.
   (Check computed results manually.)

Maintenance Phase

1. Use the program.
2. Maintain.
   (Modify to meet changed requirements or to correct errors.)

(From Dale, Nell and Weems, Chip, *Introduction to Pascal and Structured Design*, 4th edition, D. C. Heath and Company, Lexington, Massachusetts, 1994).

Overview

- A look at GIS Programming in General
- An Introduction to Python
- Learning Programming Terminology
- Python Basics
- Python for GIS
- Resources and Tools
- Hands on Time

What is Programming?

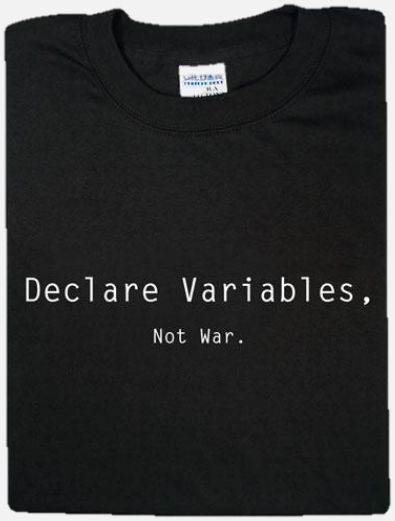
Programming for GIS is principally about ***automation and analysis*** for situations where manual actions are prohibitive or unreproducable.

- Large datasets
- Complex operations
- Subsetting

You're not always writing a large application. Sometimes, you just need it to run your operations without intervention.

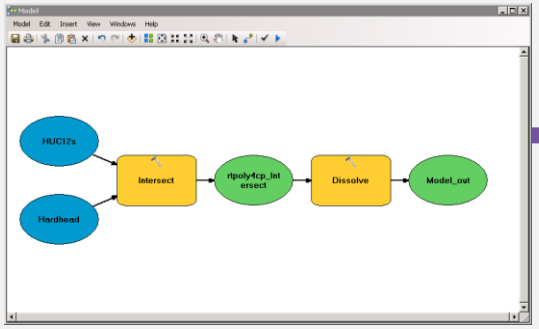
What can a script help with?

- Anything with repetition
  - Really, it's designed for this.
  - Mapping, intersect operations, getting data, etc
- Large, complex geoprocessing operations
  - Anything a model can do
  - Can help (or harm) debugging and logical flow
  - Database-backed operations
- Plugging in external data to your geoprocessing
  - Python has LOTS of modules for interfacing
- Quick tasks in ArcGIS itself – either on multiple layers, or multiple rows in a layer
- Running geoprocessing tasks outside of Arc



Why not a Model?

- Models have some excellent use cases
- Large, complicated models are often good candidates for scripts instead
  - The logic is often cleaner
  - If/Else statements and recurring parts (functions/loops) are complicated in models.




```
import arcpy

# Local variables:
Hardhead = "Hardhead"
HUC12s = "HUC12s"
rtpoly4cp_Intersect =
        "E:\\FSFISH\\Scripts\\Calculations.mdb\\rtpo
        ly4cp_Intersect"
Model_out =
        "E:\\FSFISH\\MappingResults.mdb\\Model_o
        ut"

arcpy.Intersect_analysis("Hardhead #;HUC12s #",
        rtpoly4cp_Intersect, "ALL", "", "INPUT")

arcpy.Dissolve_management(rtpoly4cp_Intersect,
        Model_out, "", "", "MULTI_PART",
        "DISSOLVE_LINES")
```

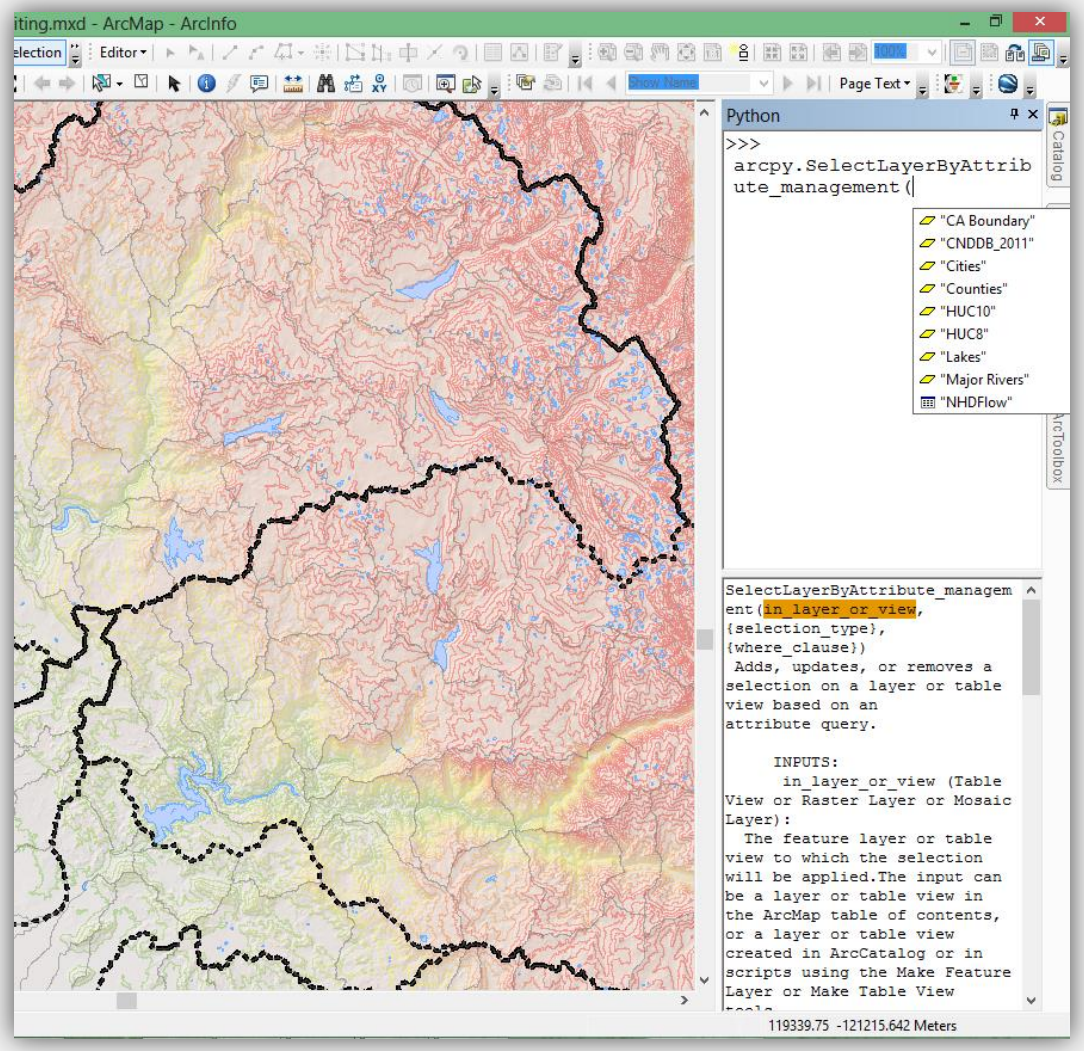
An Overview of GIS Code

- Basic->Advanced GIS programming is principally done using a language called **Python**.
  - Other languages can be used, but have higher learning curves. Python is most important for geoprocessing

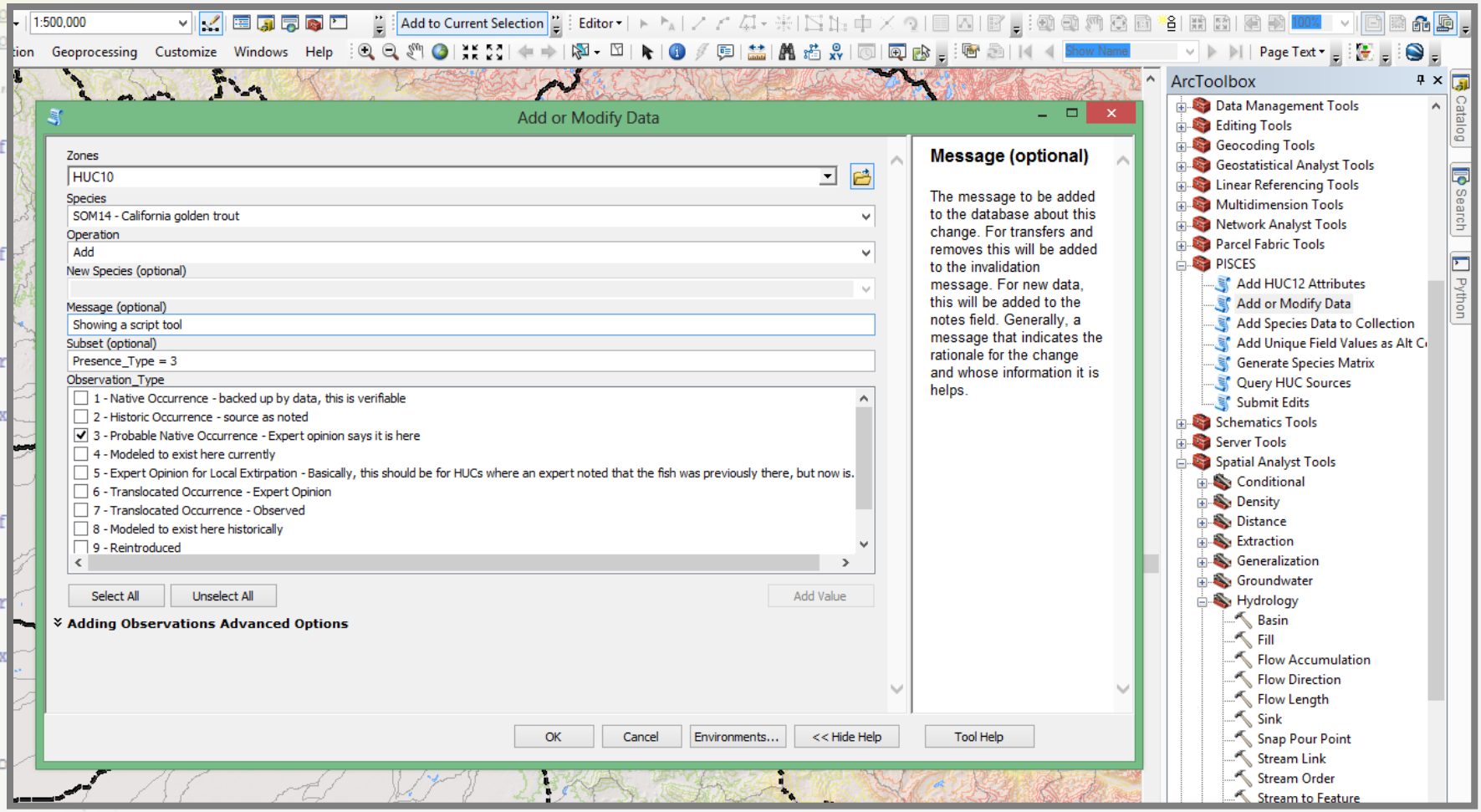
De-Jargon-er

Geoprocessing:
GIS operations that manipulate spatial datasets and return results. Examples include buffering, clipping, and summarization of areas.

Arcpy Interfaces – Python Window



Arcpy Interfaces – Script Tools



Arcpy Interfaces – Command Line

C:\WINDOWS\System32\cmd.exe

```
C:\Users\Nick\Desktop>cd C:\Users\Nick\Documents\CWS\PISCES\scripts\PISCES

C:\Users\Nick\Documents\CWS\PISCES\scripts\PISCES>cmd
Microsoft Windows [Version 6.2.9200]
(c) 2012 Microsoft Corporation. All rights reserved.

C:\Users\Nick\Documents\CWS\PISCES\scripts\PISCES>main.py stats

Calculating data stats
Total Number of Valid Obervations in the Quality Controlled Set: 50925
Total Number of *Native* Fish Species with Data: 126
Total Number of Fish Species with Data: 192
Total Number of NonNative Species: 83
Total Number of Datasets Included: 154
Total Number of Observations: 274555
Total Number of Species tracked (no data bins): 212
Total Number of Species with Historic QC Data: 42
Total Number of Species with Present QC Data: 67
Total Number of Species tracked (including data bins): 224


C:\Users\Nick\Documents\CWS\PISCES\scripts\PISCES>
```

Basic Python Terminology

- Statement
  - A line of code that does some work
- Variable
  - Just like in algebra, these are names for values that can change
- String
  - Think of it as text – letters strung along one after another
- Function
  - A named block of code that can be reused
- Block
  - A set of code that executes together
  - This will make sense when we start looking at code

Additional Terminology

- Class
  - An abstracted collection of variables and methods that represent some larger concept
    - Eg: a car – *generic concept*
  - Instance Object or Instance
    - The class, when in use, and with data – like a variable with information, where the variable has a structure predefined by the class
      - Eg: Your 1996 Ford Taurus – *specific incarnation*
  - Method
    - Like a function, but operates on class data
      - Eg: Drive! – *do something*
- Module or Package
  - Reusable code that you can bring into your own code. Arcpy is an example of a package

Talking like a programmer

- Argument/Parameter
  - Variable data passed into a function or script to provide the context and information for the code
- Exception
  - An unexpected condition in the program - difficult to recover from without additional coding to handle them. For our purposes, a crash
- Comment
  - Embedded, non-code English (or other human language) explanations of what is contained in the code.

Failing Gracefully

- Rule #1 of programs is – they *break*, and never work on the first try.
  - So, we go back to **debugging**
- Google is your friend, but you may need to adjust your searches a bit.
  - Language and version (Python 2.6)
  - Major package (arcpy)
  - Error codes or descriptions ("'NoneType' object has no attribute")
- Comment your code – it really helps. Seriously.

Important Items in Python

- **print**

Statement

String

Comment

```
print "hello world!" # prints text to console
```

- **=**

Direction of Assignment

Variable

```
a = b # sets the value of a to the value of b
```

- **==**

"If Statement"

Block

Statements

```
if a == b: # tests for equality of variables
    print "a and b are the same!"
```

- **is**

Predefined Constant

```
if a is True: # tests if they are the same object - special case
    print "You've found the truth!"
```

More parts!

- Import

```
import time
import tempfile
```

- If/Else

```
if upstream_layer: #if this exists
    arcpy.SetParameter(7,upstream_layer)
else: #otherwise, do this
    log.error("No Upstream Layer to Return")
```

- For Loops

```
for fid in fish_subset: # do something with each fish id in the set
    l_query = "select Common_Name from Species where FID = ?“
    l_result = db_cursor.execute(l_query, fid)
    map_fish[fid] = l_result[0].Common_Name # Index by FID
```

Cursors

- Special way for looping
  - If a feature in a feature class is just a single record, a cursor can help you iterate through each one and read, modify, or add new records.

```
rows = arcpy.SearchCursor("C:/Data/Counties.shp", "", "",
"NAME; STATE_NAME; POP2000", "STATE_NAME A; POP2000 D")

currentState = ""

# Iterate through the rows in the cursor
for row in rows:
    if currentState != row.STATE_NAME:
        currentState = row.STATE_NAME
        # Print out the state name, county, and population
        print "State: %s, County: %s, population: %i" % \
             (row.STATE_NAME, row.NAME, row.POP2000)
```

Code source: ESRI Documentation

Functions

Reusable code – importable to other scripts, in order to make commonly needed code available

python

```
Microsoft Windows [Version 6.2.9200]
(c) 2012 Microsoft Corporation. All rights reserved.

C:\Users\Nick>python
Python 2.7.3 (default, Apr 10 2012, 23:31:26) [MSC v.1500 32 bit (Intel)] on win
32
Type "help", "copyright", "credits" or "license" for more information.
>>> print "I am not a function"
I am not a function
>>> def reuse_me(please = False):
...     if please:
...             print "you can use this"
...     else:
...             print "you didn't say please"
...
>>> reuse_me()
you didn't say please
>>> reuse_me(please = "please")
you can use this
>>>
```

Conventions

- Python code blocks are defined by indentation
  - Statements that start a new block end with a colon
- import statements usually occur at the top
- Dot Notation and nesting
  - **os.getcwd()** refers to function **getcwd()** in package **os**
  - **os.path.join()** refers to function **join()** in module **path** in package **os**

Reading Code

```
import huc_network
import log
import arcpy

ds_field = arcpy.GetParameterAsText(1) # Get the parameter from ArcGIS
if not ds_field: # if ds_field is still undefined
    log.write("Setting DS field to %s" %
              huc_network.ds_field)
              # write a log message about what we're doing
    ds_field = huc_network.ds_field
       # And use our backup definition as our default
```

This snippet of a script

1. Imports additional packages
2. Obtains a command line argument
3. Checks if the argument is defined (it could have been empty)
4. Prints a message to the log
5. Sets the value of ds_field to a default if it wasn't already set

Diving into Arcpy!

```
def multifeature_to_HUCs(feature = None, relationship = "INTERSECT"):

    zones_layer = "zones_feature_layer"
    arcpy.MakeFeatureLayer_management(vars.HUCS,zones_layer)

    join_shape = os.path.join(arcpy.env.workspace,"temp_sjoin")

    arcpy.SpatialJoin_analysis(zones_layer,feature_layer,join_shape,
    "JOIN_ONE_TO_MANY", "KEEP_COMMON", match_option = relationship)

    l_fields = arcpy.ListFields(join_shape)
    l_cursor = arcpy.SearchCursor(join_shape)

    zones = []
    for row in l_cursor: # for each row in the result
        l_row = empty_row()
        for field in l_fields: # and for every field in that row
            l_row.__dict__[field.name] = row.getValue(field.name)
         zones.append(l_row)

    return zones
```

Programming Resources

- Code Academy *(codecademy)*:
  - http://www.codecademy.com/
- Coursera
  - Has a number of free classes available
  - http://coursera.com
- Getting help via StackOverflow, a programming Q&A site
  - http://stackoverflow.com

Python Resources

- Python is VERY well documented
  - Python 2.6 documentation (ArcGIS 10.0) http://docs.python.org/2.6/
  - Python 2.7 documentation (ArcGIS 10.1) http://docs.python.org/2/
- Learning/Programming Python
  - Learning: http://oreil.ly/pD1rM5
  - Programming: http://oreil.ly/zqD7JK
- GDAL/OGR – open source programming libraries
  - http://www.gdal.org/

Arcpy and GIS Programming Resources

- ESRI standard documentation
  - Geoprocessing Tool Reference has code samples
    - http://bit.ly/9MBUwH
  - Arcpy Site Package Reference
    - 10.0 - http://bit.ly/da5yDT
    - 10.1 - http://bit.ly/T28hlM
- StackExchange Q&A site for GIS
  - http://gis.stackexchange.com/
- A UNEX course? Note it on your evals please if you'd like a more full course.

Tools

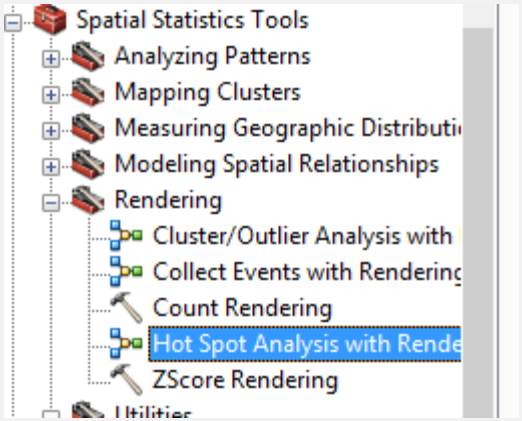
- IDLE
  - Simple Python editor that ships with Python
- Notepad++
  - Better Python Editor. Free.
- PyCharm
  - Commercial Python IDE. $29 and up.
- Command Line
  - Useful for exploration and testing activities
- Aptana Studio (PyDev)
  - Free Python IDE
- Mercurial or Git
  - For versioning your data and code. Helps you revert errors, track changes, and collaborate.

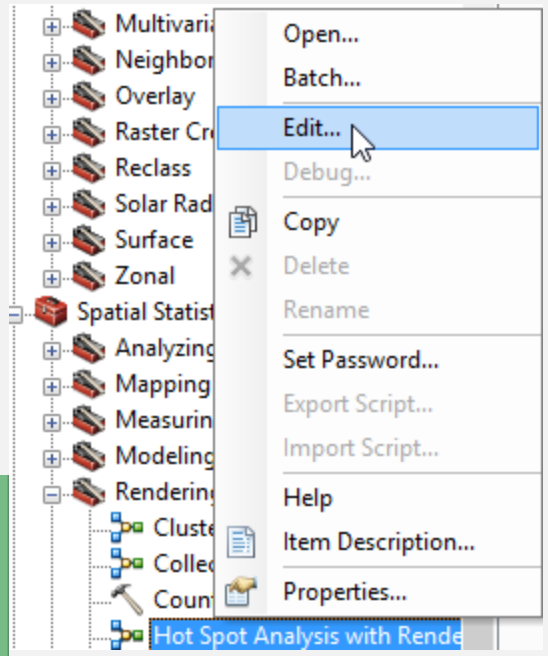
Toolbox -> Script

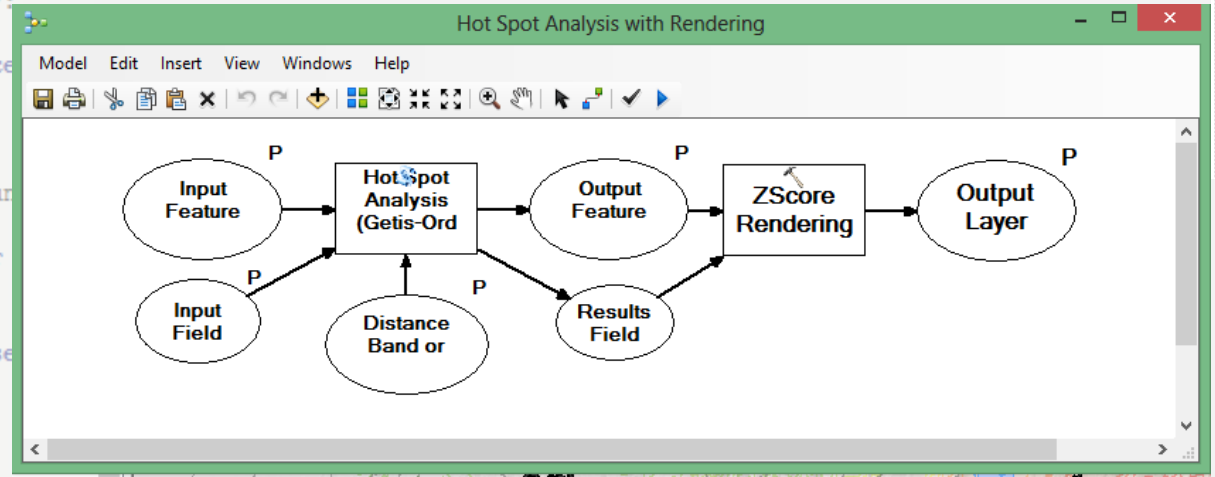
- Open the Hot Spot Anlysis Model in the Spatial Statistics Toolbox

Right Click on the Tool, Select Edit.

Take a brief look at the model

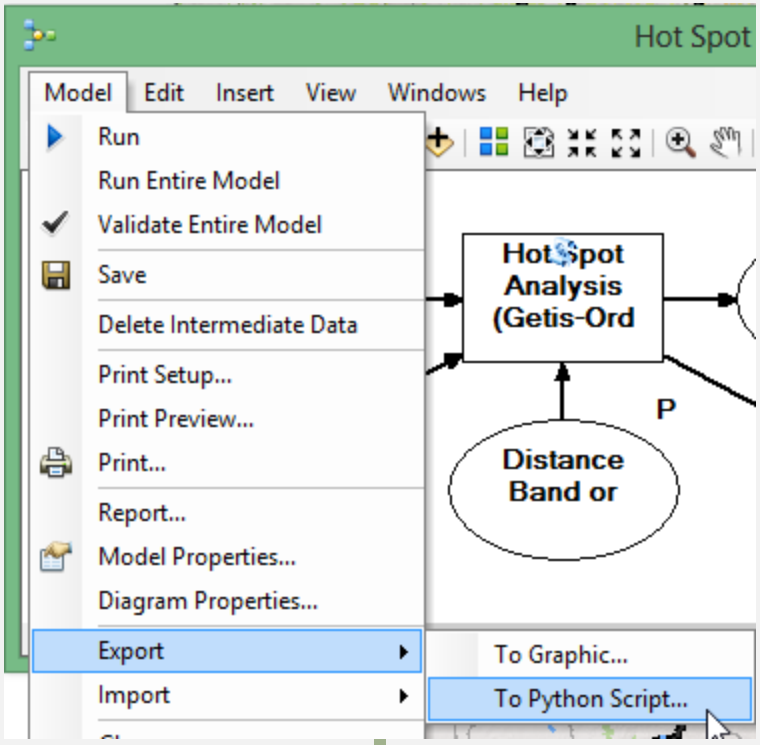


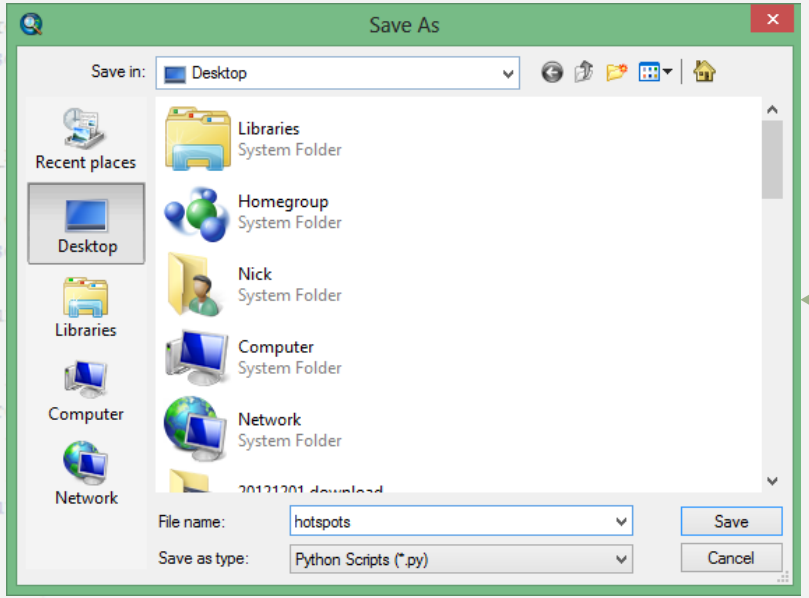




Export the Model to Python

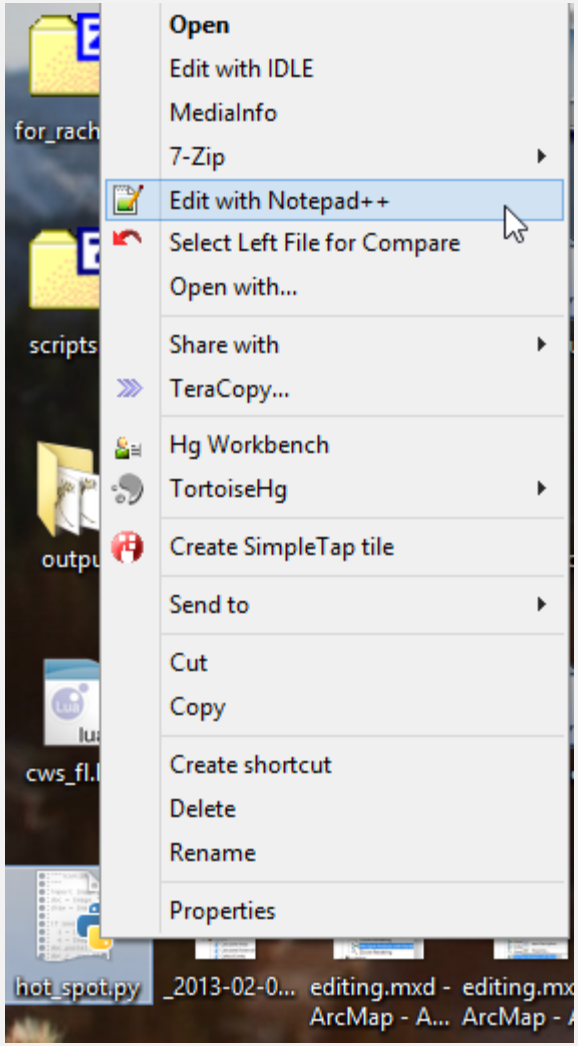
- Model->Export->To Python Script
- Save to hot_spot.py on your desktop





Open the Code and Observe

- Right Click on the file and select "Edit" with your preferred editor



- Major questions:
  - How is data getting into this program?
  - Where do results get saved out?

Version Control

- Let's make a new repository with that code using **Git**

Hands on

- In your web browser, navigate to https://bitbucket.org/UltraAyla/sierra-code-library
  - Go to the downloads tab and download CWS Toolbox-1.3.5.zip