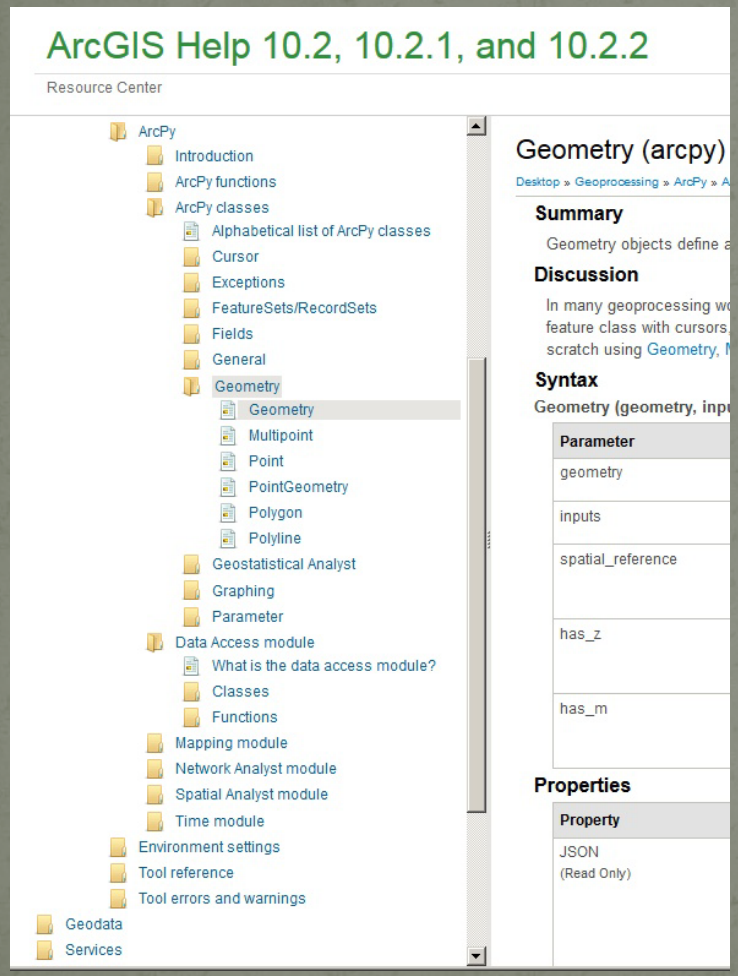
What I did with Python over the Summer

What can you actually do with Python and arcpy?

Why Python?

- VBA going away
- Open source
- Easy to learn?
- Growing popularity

Rich help pages



- Help is directed toward Python
- Examples on nearly every page
- Detailed documentation for properties and methods

ArcPy Esri's core Python API

- Nearly everything in the Data Management toolbox
- Additional classes and methods
- arcpy.da and arcpy.mapping

arcpy.mapping .mxd's & the TOC

- Can reach into a project file, and modify layers
- Can access the currently running project
  - Make layers
  - Modify layers
  - Remove layers

arcpy.da The Data Access Module

- Probably the most important functionality of arcpy
- Cursors
  - SearchCursor - Read only
  - InsertCursor - Write one new record
  - UpdateCursor - Full access to read & write

Cursors Control your data

arcpy.da.SearchCursor(FC, fields, whereClause)

- Restricting your data (that's a good thing)
  - Restrict your columns with the 'fields' parameter
  - Restrict your rows with the 'whereClause' parameter
- Full access to returned rows and cell values
  - Field Calculator on steroids

Geometry Get to the point

- All Feature classes have a shape field
- arcpy has classes and methods to get spatial data
  - Geometry tokens for cursors "SHAPE@XY"
  - Classes
    - Geometry
    - Point
    - Polyline
    - Polygon

Python is Object-Oriented

- Nearly everything in Python is an object
  - Strings have methods
  - Lists and tuples are powerful, out of the box data structures which ArcPy takes advantage of
  - Dictionaries are less used, but highly useful.

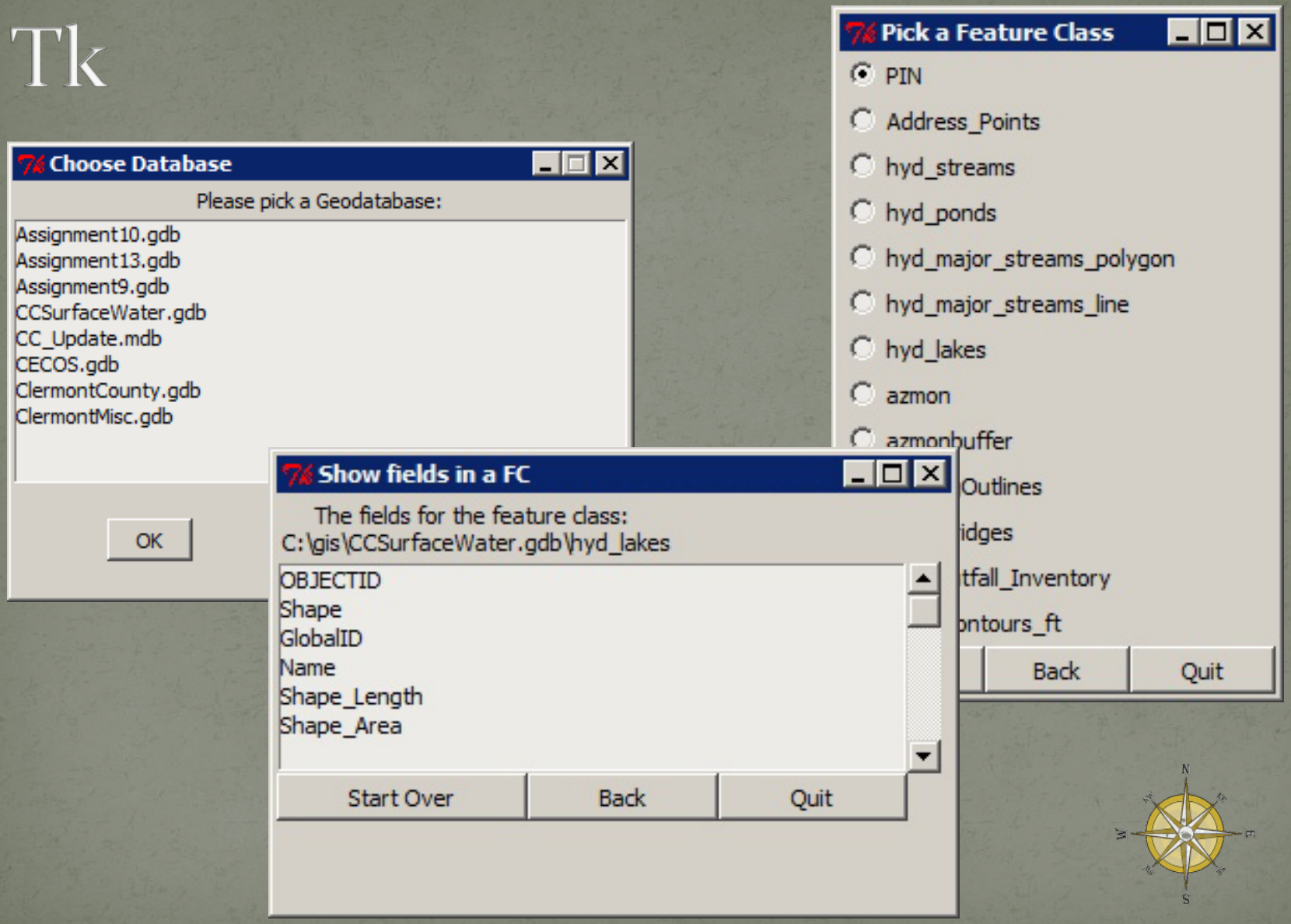
Classes Easily made, very useful

```
class StreamTree:
    StreamCount        = 0
    ProblemStreamCount = 0

    def __init__(self, num):
        self.StreamNum    = num
        self.LeftTrib     = None
        self.RightTrib    = None
        self.ProblemState = "None"
        assert self.ProblemState in (“ThreeWay", "OneChild", "None“, “Leaf”)
        StreamTree.StreamCount += 1

    @staticmethod
    def getEndPts():
        StreamEndPtFields = [ "streamNum", "SHAPE@" ]
       with arcpy.da.SearchCursor(StreamTree.streamFC, StreamEndPtFields) as streamCursor:
           for stream in streamCursor:
               StreamTree.endPointLookup[stream[0]] = ( s[1].lastPoint.X, s[1].lastPoint.Y )
```

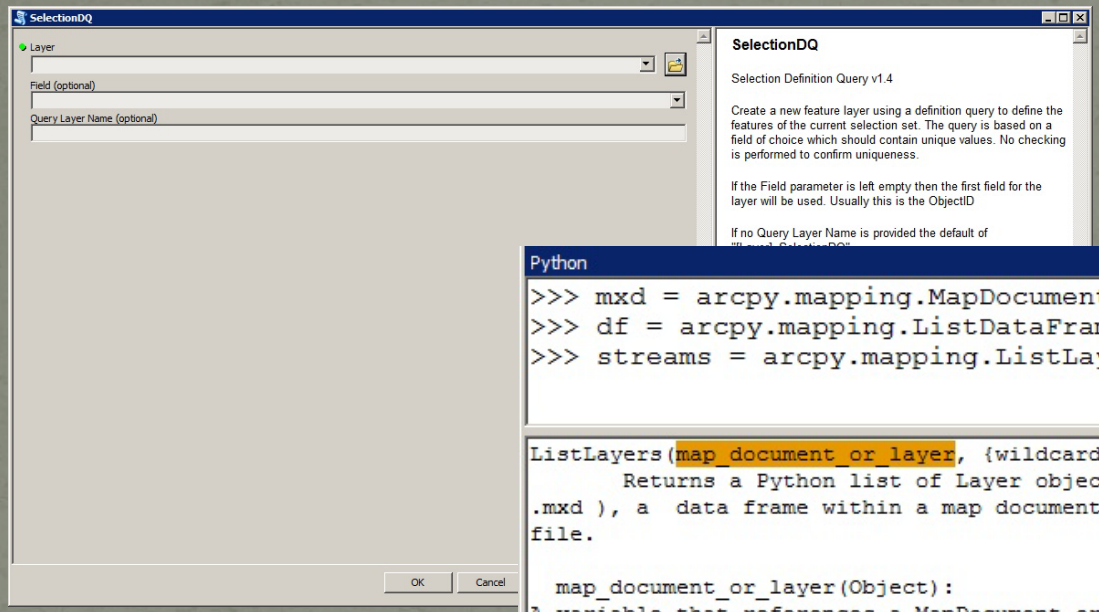
Tk

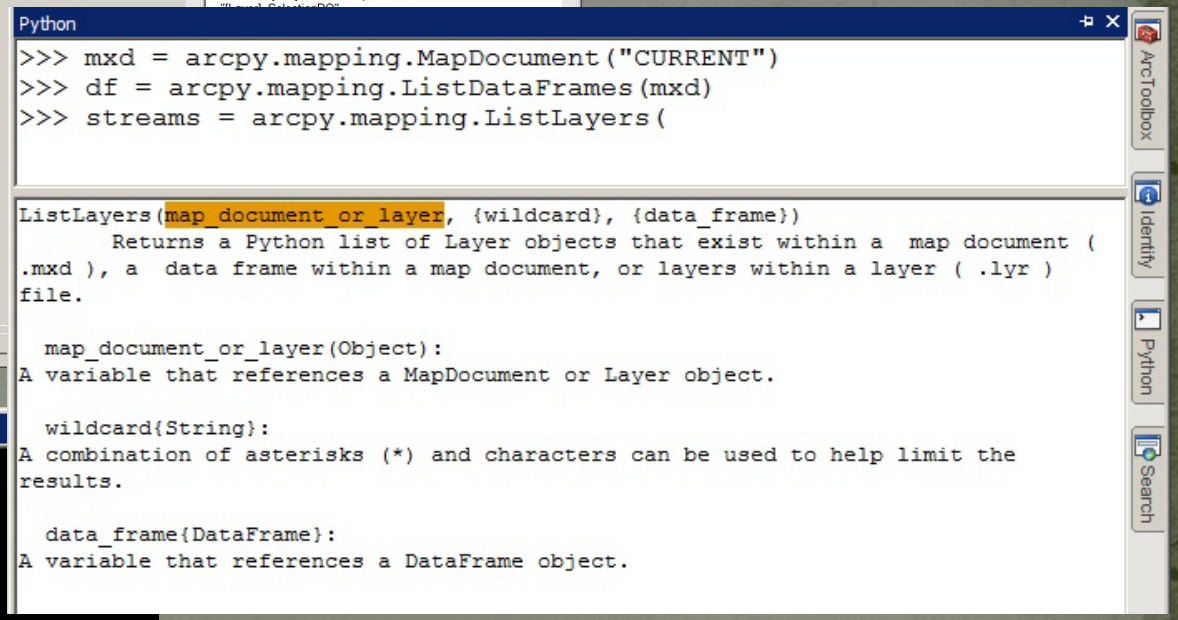

Choose Database
Please pick a Geodatabase:
Assignment10.gdb
Assignment13.gdb
Assignment9.gdb
CCSurfaceWater.gdb
CC_Update.mdb
CECOS.gdb
ClermontCounty.gdb
ClermontMisc.gdb
OK
Pick a Feature Class
PIN
Address_Points
hyd_streams
hyd_ponds
hyd_major_streams_polygon
hyd_major_streams_line
hyd_lakes
azmon
azmonbuffer
Outlines
idges
tfall_Inventory
ontours_ft
Back
Quit
Show fields in a FC
The fields for the feature class:
C:\gis\CCSurfaceWater.gdb\hyd_lakes
OBJECTID
Shape
GlobalID
Name
Shape_Length
Shape_Area
Start Over
Back
Quit

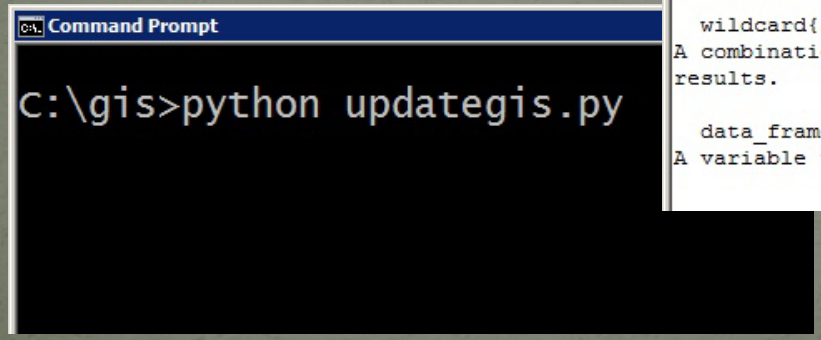
Myriad other modules

- All of the amenities of a fully developed language
  - Full OS access
  - csv, dbfPy
  - smtplib
  - urllib
  - math
  - NumPy, SciPy
  - Regular expressions

ArcMap, ArcCatalog, or stand-alone







Debugging

Well, it depends….

PyScripter - C:\Data\CS380\Work\13_CreateAStreamNetworkClass\StreamTree.py

```
#-------------------------------------------------------------------------------
# Name:        module1
# Purpose:
#
# Author:      William F. Mellman
#
# Created:     2014-12-04
# Copyright:   (c) William F. Mellman (bmellman) 2014
# Licence:     GNU General Public License
#-------------------------------------------------------------------------------

import arcpy
from MyModules import c360ify

MeanLines = "c:/gis/Assignment13.gdb/MeanLines"


#    @staticmethod
def getCompassA(num):
    MeanLineFields = ["CompassA"]
    whereClause = "StreamNum = " + str(num)
    with arcpy.da.SearchCursor(MeanLines, MeanLineFields, whereClause) as MeanLineCursor:
        return MeanLineCursor.next()[0]


class StreamTree:
    StreamCount        = 0      # Total number of streams, incremented in init
    ProblemStreamCount = 0      # Total number of streams with non-None problem state
    endPointLookup     = {}     # end points of all lines in streamsFC
    streamFC           = "c:/gis/Assignment13.gdb/streams"


    def __init__(self, num):
        self.StreamNum    = num
        self.LeftTrib     = None
        self.RightTrib    = None
        self.ExtraTrib    = []
        self.ProblemState = "None"
        assert self.ProblemState in ("FourWay", "OneChild", "None")
        StreamTree.StreamCount += 1


    @staticmethod
    def getEndPts():
        StreamEndPtFields = [ "streamNum", "SHAPE@" ]
        with arcpy.da.SearchCursor(StreamTree.streamFC, StreamEndPtFields) as streamCursor:
            for stream in streamCursor:
                StreamTree.endPointLookup[stream[0]] = ( stream[1].lastPoint.X, stream[1].last


    @staticmethod
    def getRightLeft(trunk, A, B):
        AngleTrunk = getCompassA(trunk)
        Angle_A    = getCompassA(A)
```

Python Interpreter

```
*** Python 2.7.5 (default, May 15 2013, 22:43:36) [MSC v.1500 32 bit (Intel)
] on win32. ***
>>>
[Dbg]>>>
245
217      Four Way
215
213
```

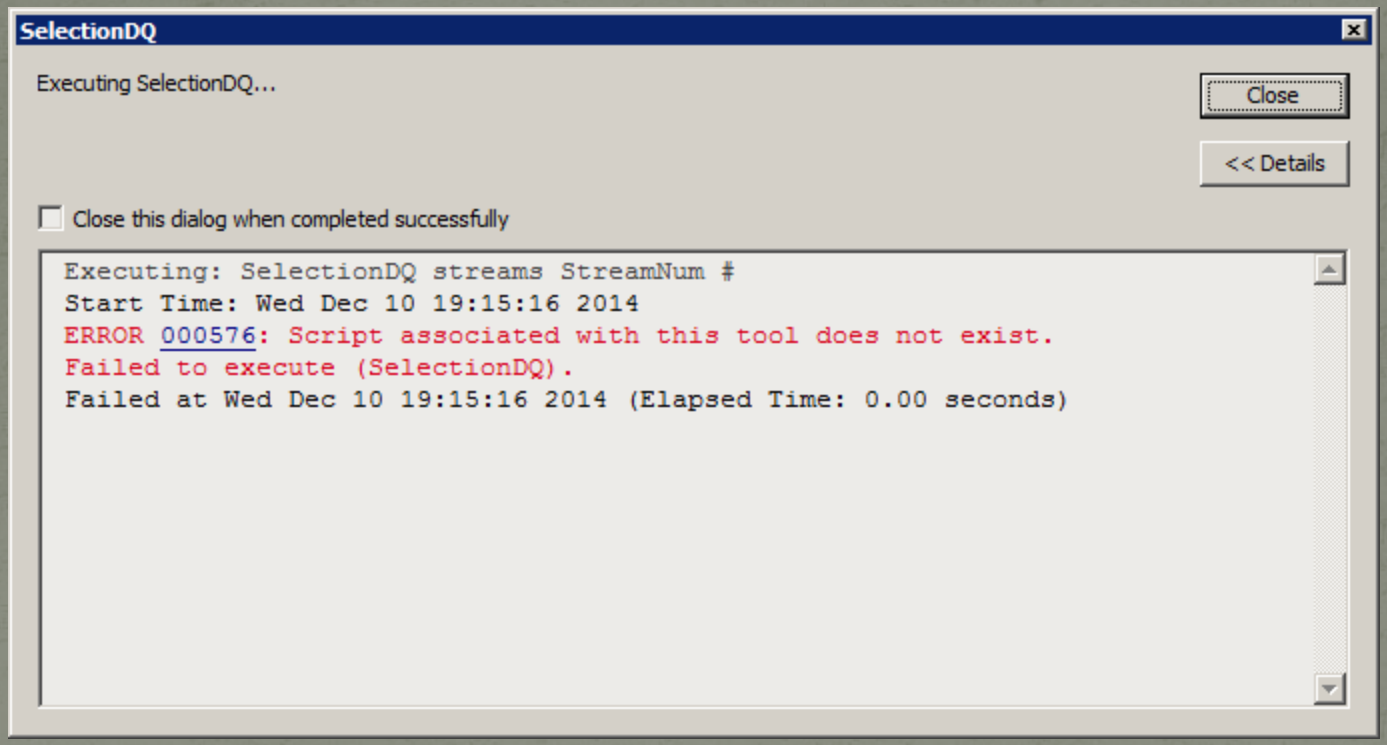
Variables

Name	Type	Value
globals	dict	{'MeanLines': 'c:/gis/Assignment1...
locals	dict	{'num': 210, 'self': <__main__.Strea...
num	int	210
self	StreamTree	<__main__.StreamTree instance at ...
buildBranch	function	<function buildBranch at 0x11815...
buildTree	function	<function buildTree at 0x11815B80>
endPointLookup	dict	{46: (1476756.5279999971, 424034.6...
getChildList	function	<function getChildList at 0x11815B...
getEndPts	function	<function getEndPts at 0x11815AB...
getRightLeft	function	<function getRightLeft at 0x11815...
LeftTrib	NoneType	None
ProblemStreamC...	int	0
RightTrib	NoneType	None
StreamCount	int	7
streamFC	str	'c:/gis/Assignment13.gdb/streams'
StreamNum	int	210
__doc__	NoneType	None
__init__	instancemethod	<bound method StreamTree.__init...
__module__	str	'__main__'

Namespace: Frame(Function: "__init__" of module: "StreamTree" at line 38)

Life is Good with Pyscripter!

But, in ArcMap…. Ouch!



No viable integrated debugging environment,

Can you say “print”?

Try & Except

```
1 try:
2     x = 1/0
3
4 except ZeroDivisionError:
5     print "Don't divide by zero!"
```

Python Interpreter

```
*** Python 2.7.5 (defau
(Intel)] on win32. ***
>>>
[Dbg]>>>
Don't divide by zero!

>>> |
```

- Can check for several different kinds of exceptions
- Compound statement
- Nested exceptions

Exceptions are a Class

- And Classes Inherit, so …
- you can make your own exceptions:

```
Class BillsException(Exception):
   #Do some stuff in here

#Main
raise BillsException()
```

Questions are Free

Answers will cost you, but

there's a special going today.

Bill Mellman
Clermont County Office of Environmental Quality
Clermont County Water Resources
bmellman@clermontcountyohio.gov