Geospatial Programming in Python

Tony Castronova
Utah State University

October 22, 2014
Learning Objectives

1. Understand the benefits of extending ArcGIS through programming
2. Introduce Python programming basics
3. Understand how to collect data using web services
4. Demonstrate how to build a Python script that uses ArcMap tools
Suggested Reading

Using ArcMap Functions

Using Python Classes

Why Programming?

- Automation of repetitive tasks →
- Implementation of functionality that would not otherwise be available ↓

Proportion flowing to neighboring grid cell 4 is $\alpha_1/(\alpha_1+\alpha_2)$
Steepest direction downslope
Proportion flowing to neighboring grid cell 3 is $\alpha_2/(\alpha_1+\alpha_2)$
Flow direction.
Extending ArcGIS

- TauDEM
- ArcSWAT, *Soil and Water Assessment Tool*
- Hec-GeoRAS, *River Analysis System*
- etc ...
Workflow Automation

**Let's Consider Data Collection**

- United State Geological Survey, USGS
- Interactive map selection
- Tiled Data
- Emailed links to download datasets
Downloading Data from the National Map Viewer
ArcGIS Processing

Mosaic illustration
Shortcomings of this Approach

**Shortcomings**
- Manual process (e.g. following documentation/tutorial)
- Difficult to share workflows with others
- Time consuming and tedious
- Repetitive
- Not feasible for large datasets

**Solution**
- Build tools
- Write scripts
- Use web services
Introduction to Programming

- Set of the instructions that direct the computer to perform certain tasks
- User input and output
- Mathematical representations and algorithms
- Logical structure (sequence, repetition)
- Modular design
- EXCEL/VBA, Matlab, Maple, Mathematica
- Fortran, C/C++, .NET, Ruby, Perl, Python
Programming Logic/Control

IF <condition> THEN
  true block
ENDIF

IF <condition> THEN
  block 1
ELSEIF <condition> THEN
  block 2
ELSE
  block 3
ENDIF

DO
  block 1
  IF <condition> THEN
    exit
  ENDIF
ENDDO

DOFOR i = start, finish, step
  block
ENDDO

if i % 2 == 0: print 'i is even!'

if i % 2 == 0: print 'i is even!'
elif i % 2 == 1: print 'i is odd'
else: print 'i is a float'

while 1:
  i += 1
  value *= i
  if i > 10:
    break

for i in range(0, 10, 1):
  print 'Hello'
def quadratic(a, b, c):
    # calculate the discriminant
    d = b ** 2 - 4 * a * c

    if d < 0:
        print("This equation has no real solution")

    # return the answer
    return (None, None)

    elif d == 0:
        x = (-b + math.sqrt(b**2 - 4*a*c)) / 2*a
        print("This equation has one solution: "), x

    # return the answer
    return (x, None)

    else:
        x1 = (-b + math.sqrt((b**2) - (4*a*c))) / (2*a)
        x2 = (-b - math.sqrt((b**2) - (4*a*c))) / (2*a)
        print("This equation has two solutions: ", x1, " or", x2)

    # return the answer
    return (x1, x2)
What is Python

- Interpreted programming language
- Built from C
- Platform Independent (unlike .Net)
- Relatively lightweight
- No need to compile
- Many different versions (latest 3.4.2, most used is 2.7.8)
- ArcGIS already installed it!
- Execute tools inside ArcMap
- Execute tools/code using CMD/Terminal/Console
- Many Interactive Development Environments
Interactive Development Environments

- Code completion
- Error checking
- Debugging
- Interactive Console

https://wiki.python.org/moin/IntegratedDevelopmentEnvironments
(Very Brief) Introduction to Python, pt1

- General purpose programming language
- Indenting matters
- Duck typing (not type specific)
- Core data types

<table>
<thead>
<tr>
<th>Object Type</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Numbers</td>
<td>1234, 3.1415</td>
</tr>
<tr>
<td>Strings</td>
<td>'spam', &quot;eggs&quot;</td>
</tr>
<tr>
<td>Lists</td>
<td>[1, 2, 3], [1, ['spam', 'eggs']]</td>
</tr>
<tr>
<td>Dictionaries</td>
<td>'food': 'spam', 'taste': 'yum'</td>
</tr>
</tbody>
</table>

```
# this is a type double
my_variable = 10.123

# now its a string
my_variable = 'some string'

# now its a list
my_variable = [1, 2, 3, 4, 5]
```

http://www.codecademy.com/courses/introduction-to-python-6WeG3/
https://docs.python.org/2/tutorial/
(Very Brief) Introduction to Python, pt2

- Import statements
- Control statements
- Functions
- Classes

```python
import os
import sys
import numpy
import suds

for i in range(0, 10):
    # do something

while i == True:
    # do something
    if i == False:
        break

while 1:
    # do something
    if condition:
        break

class MyClass():
    def __init__(self):
        d = 1

        def some_calc(self, a, b, c):
            d = d + 1
            return d + (a * b + c**2)

    # create instance of MyClass
    mycalc = MyClass()

    print mycalc.some_calc(1,2,3)  # 13
    print mycalc.some_calc(1,1,2)  # 7

def myFuction(a, b, c):
    # do something
    some_calc = a * b + c**2

    # return the result
    return some_calc
```
Package Management

- Enables you to extend Python’s native functionality
- Using 3rd party libraries, e.g. ArcGIS
- PIP

• ArcPy is the ArcGIS python module
• Provides a productive way to perform geographic data analysis using ArcGIS tools
• Organized into tools, functions, classes, and modules
ArcGIS Tools

- Every tool has its own reference page
- Tools produce results, while functions do not
- Tools produce messages
- Tools are licensed

```python
import arcpy

arcpy.AddField_management("c:/data/Portland.gdb/streets", "LENGTH_MILES", "TEXT")
arcpy.CalculateField_management("c:/data/Portland.gdb/streets", "LENGTH_MILES", "!shape.length@miles!", "PYTHON9.3")
arcpy.FeatureClassToFeatureClass_conversion("c:/data/Portland.gdb/streets", "Database Connections/MySDE.sde/PortlandDataset", "streets")
```

IDW example 2 (stand-alone script)

This example inputs a point shapefile and interpolates the output surface as a Grid raster.

```python
# Name: IDW_3d_Ex_02.py
# Description: Interpolate a series of point features onto a rectangular raster using Inverse Distance Weighting (IDW).
# Requirements: 3D Analyst Extension

# Import system modules
import arcpy
from arcpy import env

# Set environment settings
env.workspace = "C:/data"

# Set local variables
inPointFeatures = "co_ozone_pts.shp"
sField = "ozone"
outRaster = "C:/output/idwout01"
cellsize = 2000.0
power = 2
searchRadius = 150000

# Check out the ArcGIS 3D Analyst extension license
arcpy.CheckOutExtension("3D")

# Execute IDW
arcpy.IDW_3d(inPointFeatures, sField, outRaster, cellsize, power, searchRadius)
```

ArcPy Functions

- Small piece of functionality
- Incorporated into larger programs
- list datasets, retrieve properties, check for existence of data

```python
import arcpy

# prints True
print arcpy.Exists("c:/data/Portland.gdb/streets")

# prints NAD_1983.StatePlane_Oregon_North_FIPS_3601_Feet
sr = arcpy.Describe("c:/data/Portland.gdb/streets").spatialReference
print sr.name

# prints Available
print arcpy.CheckExtension("spatial")

arcpy.CheckOutExtension("spatial")
```
ArcPy Classes

- Architectural blueprint
- Class objects are called instances
- Often used as shortcuts for completing a geoprocessing tool

```python
import arcpy

inputWorkspace = "c:/temp"
outputName = "rivers.shp"

prjFile = "c:/projections/North America Equidistant Conic.prj"
spatialRef = arcpy.SpatialReference(prjFile)

# Run CreateFeatureclass using the spatial reference object
#
# arcpy.CreateFeatureclass_management(inputWorkspace, outputName, "POLYLINE", 
#                                          ",", ",", ",", 
spatialRef)
```
ArcPy Environment Settings

- Global settings
- Default values can be overridden
- nodata value, output spatial reference, cell size

```python
import arcpy
from arcpy import env

# Check the current raster cell size and make sure it is a certain size
# for standard output
#
env.workspace = "c:/avalon/data"

if env.cellSize < 10:
    env.cellSize = 10
elif env.cellSize > 20:
    env.cellSize = 20

arcpy.HillShade_3d("island_dem", "island_shade", 300)
```
Error Checking

- Errors will cause your script to break/fail
- Error catching can prevent this (to and extent)
- Python try-except statements
- ArcPy message retrieval

```python
import arcpy
from arcpy import env
try:
    if arcpy.CheckExtension("3D") == "Available":
        arcpy.CheckOutExtension("3D")
    else:
        raise LicenseError
    env.workspace = "D:/GrosMorne"
    arcpy.HillShade_3d("WesternBrook", "westbrook_hill", 300)
    arcpy.Aspect_3d("WesternBrook", "westbrook_aspect")
except LicenseError:
    print "3D Analyst license is unavailable"
except:
    print arcpy.GetMessages(2)
finally:
    arcpy.CheckInExtension("3D")
```
__author__ = "tonycastronova"

# Tony’s ArcGIS script
# This is a demo for CEE 6440
# 10/22/2014

# Import system modules
import arcpy

# Set workspace
env.workspace = "C:/data"

# Set local variables
in_features = "majorrds.shp"
clip_features = "study_quads.shp"
out_feature_class = "studyarea.shp"
xy_tolerance = ""

try:
    # Execute Clip
    arcpy.Clip_analysis(in_features, clip_features, out_feature_class, xy_tolerance)
except Exception:
    print 'Something went wrong :( ' 
    for i in arcpy.GetMessageCount():
        print i
__author__ = "tonycastronova"

# Tony’s ArcGIS script
# This is a demo for CEE 6440
# 10/22/2014

# Import system modules
import arcpy
import os

# Set workspace
env.workspace = "C:/data"

# Set local variables
#in_features = "majorrds.shp"
clip_features = "study_quads.shp"
#out_feature_class = "studyarea.shp"
xy_tolerance = ""

# find all shapefiles in the current directory
directory_contents = os.listdir('.')
files = []
for item in directory_contents:
    if os.path.isfile(item):
        if item.split('.')\[1\] == 'shp':
            files.append(item)

# continued on next slide ...
ArcPy Scripting, pt3

```python
# loop over files in directory and clip
for file in files:
    in_features = file
    out_features = file.split('.')[0]+'_clip.shp'
    try:
        # Execute Clip
        arcpy.Clip_analysis(in_features, clip_features, out_feature_class, xy_tolerance)
    except Exception:
        print 'Something went wrong :( '
        for i in arcpy.GetMessageCount():
            print i

• This script provides an easy way to automate the Clip tool
• Could be used for clipping dependent datasets
• Could be extended to perform many tasks in sequence
```
Web Services

- Standardized communication between clients and web servers
- Several different standards. SOAP + WSDL, REST, XML-RPC
- XML-based protocols
- Can be used to gather data from national datasets

http://www.opengeospatial.org
What data is available?

- You can get any data that have webservies!
- USGS - streamflow, groundwater, water quality
- EPA STORET - water quality
- DAYMET - meteorological surface temperature, precipitation, humidity, etc.
- MODIS - surface temperature, snow cover, vegetation, etc.

http://his.cuahsi.org/wofws.html
Data Collection using Python

- Suds python package
- Lightweight web services client for python
- Provides an object-like API
- Reads WSDL

```python
from suds.client import Client
url = 'http://river.sdsc.edu/wateroneflow/NWIS/DailyValues.asmx?wsdl'

# create an instance of the suds Client class
client = Client(url)

# see what functions are available
print(client)

# call a function
client.service.GetValues('NWIS:09429000', 'NWIS:00060', '2000-11-01T00:00:00', '2000-12-31T00:00:00', '')
```
Practical Example

- Collect MODIS land surface temperature
- Write a python script to collect this data
- Convert the results into Arc rasters and visualize
- Extend to use other tools (time permitting)
import os  # standard python package
import arcpy  # arcmap package
import modis_base  # package for organizing MODIS data
from suds.client import *  # web service client

# set overwrite to true
arcpy.env.overwriteOutput = True
import os  # standard python package
import arcpy  # arcmap package
import modis_base  # package for organizing MODIS data
from suds.client import *  # web service client

# setup environment
arcpy.env.overwriteOutput = True
arcpy.env.workspace = os.getcwd()

pts = [(40.115,-110.025),
       (40.144341, -109.807629),
       (39.927865, -109.867392),
       (40.193480, -110.353466),
       (40.068641, -110.150605),
       (40.124420, -109.910932)]

output_files = {}  # dictionary to store the results
i = 0  # counter
for lat, lon in pts:  # loop over each lat,lon in pts list
    print 'Querying MODIS data...'
    wsdl = 'http://daac.ornl.gov/cgi-bin/MODIS/GLBVIZ_1_Glb_subset/MODIS_webservice.wsdl'
    client = Client(wsdl)  # Create instance of suds client

    # call the web service to retrieve data
    lat, lon = pt
    result = client.service.getsubset(lat, lon, "MOD11A2","LST_DAY_1km",
                                       "A2001001","A2001025",10,10)
# ... continued from previous slide

# organize the results
print 'Building Modis Object...'
modis_data = modis_base.ModisData(result)

# get info about the dataset
data = modis_data.get_data()
modisObj = modis_data.mosisObj()
rows = modisObj.nrows
cols = modisObj.ncols
xmin, ymin, xmax, ymax = modis_data.get_bbox()
cellsize = modis_data.get_cellsize()
```python
# create ASCII Raster for all dates retrieved
for date in data.keys():

    filename = 'modis_temp_' + date + '.txt'
    ascii_path = os.getcwd() + '/temp/' + filename
    with open(ascii_path, 'w') as f:

        # write header data
        f.write('NCOLS %s
' % cols)
        f.write('NROWS %s
' % rows)
        f.write('XLLCORNER %s
' % xmin)
        f.write('YLLCORNER %s
' % ymin)
        f.write('CELLSIZE %s
' % cellsize)
        f.write('NODATA_VALUE %s
' % 0)

    d = data[date]
    for row in d:
        for value in row:
            if value != 0.0:
                value = (value * .02 - 273.15) * 1.8 + 32.

            f.write('%3.2f ' % value)
        f.write('
')

    print 'Process: ASCII to Raster'
    raster_name = filename.split('_')[-1][:-4] + '_' + str(i)
    raster = os.path.join(os.getcwd() + '/raster', raster_name)
    arcpy.ASCIIToRaster_conversion(ascii_path, raster, 'FLOAT')

    # continued on next slide ...
```
# ... still inside loop

print 'Process: Define Projection'

# Define a custom projection as WKT (MODIS projection)
prj = "PROJCS['Sinusoidal', " + \n"GEOGCS['GCS_Undefined',DATUM['D_Undefined',SPHEROID['User_DEFINED_Spheroid',"+\n"6371007.181,0.0]],PRIMEM['Greenwich',0.0],UNIT['Degree',0.0174532925199433]],"+\n"PROJECTION['Sinusoidal'],PARAMETER['False_Easting',0.0],"+\n"PARAMETER['False_Northing',0.0],PARAMETER['Central_Meridian',0.0],"+\n"UNIT['Meter',1.0]]"

arcpy.DefineProjection_management(raster, prj)

# store all output file names in a dictionary based on date
if date in output_files:
    output_files[date].append(raster)
else:
    output_files[date] = [raster]
Practical Example - Build Raster Results

NCOLS 21.0
NROWS 21.0
XLLCORNER -9355210.41
YLLCORNER 4451508.54
CELLSIZE 926.625433056
NODATA_VALUE 0
48.33 47.82 46.99 46.31 46.13 44.65 44.47 44.47 44.04
47.28 46.71 46.35 45.81 45.77 45.70 45.19 44.58 43.29
45.12 44.73 45.91 45.52 45.59 44.94 44.91 44.44 44.26
43.72 43.61 44.33 46.09 46.35 45.19 44.87 44.83 45.19
44.22 46.24 47.64 47.43 46.09 44.04 44.26 44.26 44.29
48.87 49.05 48.00 46.63 44.47 43.72 43.54 44.73 44.80
0.00 47.17 47.03 46.85 46.67 45.91 0.00 0.00 44.08 45.48
48.15 47.14 46.85 46.45 45.34 44.94 0.00 44.08 44.94 44.74
47.25 46.99 45.34 45.30 46.45 46.56 46.67 45.59
46.81 47.17 45.73 46.71 46.56 46.20 46.78 46.35 45.88
47.50 47.64 47.79 0.00 0.00 47.57 47.35 43.72 43.86 43.00
0.00 0.00 0.00 48.54 45.45 43.57 43.90 43.79 43.72 43.00
0.00 0.00 0.00 44.37 43.75 43.90 43.79 43.68 43.36 43.00
46.13 43.97 44.33 44.62 44.19 44.08 43.83 43.50 43.43
44.91 44.76 44.73 44.47 44.26 43.72 43.25 42.96 43.11
45.05 44.98 44.69 44.22 43.86 43.29 42.82 42.85 43.03
44.76 44.65 44.44 43.86 43.61 43.36 43.21 43.03 42.93
43.86 43.36 43.29 43.29 43.65 43.72 43.68 43.54 43.39
44.29 44.04 44.11 45.19 45.37 45.30 44.37 44.08 43.57
44.62 44.76 44.73 44.91 45.30 45.34 45.01 44.51 44.11
44.58 44.65 45.01 45.19 45.52 45.59 45.41 45.19 44.87
Practical Example - Add Mosaic Tool

```python
print 'Process: Mosaic'
for key in output_files.keys():
    # build a new mosaic dataset
    file_list = output_files[key]
    files = ';'.join(file_list)
arcpy.MosaicToNewRaster_management( files, '/mosaic', key, prj,
                                        '8_BIT_UNSIGNED', '40', '1',
                                        'LAST','FIRST')
```

Mosaic
Usage: Mosaic_management inputs;inputs... target
   {LAST | FIRST | BLEND | MEAN | MINIMUM | MAXIMUM} {FIRST | REJECT | LAST | MATCH}
   {background_value} {nodata_value} {NONE | OneBitTo8Bit} {mosaicking_tolerance}
   {NONE | STATISTIC_MATCHING | HISTOGRAM_MATCHING | LINEARCORRELATION_MATCHING}
How about some Geoprocessing

- Determine locations where surface temperature increased
- Determine locations where surface temperature decreased

```python
# Check out the ArcGIS extension license
arcpy.CheckOutExtension("Spatial")
from arcpy.sa import *

# calculate locations where surface temperature increased
r1 = '/mosaic/' + output_files.keys()[0]
r2 = '/mosaic/' + output_files.keys()[-1]
```
Calculating Temperature Increase

# ... continued from previous slide

# Execute Minus
print 'Executing Minus...'
diff = os.getcwd()+'/temperature/diff'
outMinus = Minus(r2, r1)
outMinus.save(diff)

# Execute GreaterThan: find all cells greater than 0
print 'Execute GreaterThan...'

# Execute Times isolate all cells where temperature increased
print 'Execute Times...'
icc = os.getcwd()+'/temperature/increased'
outTimes = Times(diff, gt)
outTimes.save(inc)
Calculating Temperature Decrease

```python
# ... continued from previous slide

# Execute LessThan: find all cells less than 0
print 'Execute LessThan'
lt = os.getcwd()+'/temperature/lt'
outLessThan = LessThan(diff, 0)
outLessThan.save(lt)

# Execute Times isolate all cells where temperature increased
print 'Execute Times...'
dec = os.getcwd()+'/temperature/decreased'
outTimes = Times(diff, lt)
outTimes.save(dec)
```
Interactive Debugging

DEBUGGING DEMO (time permitting)
Summary of Concepts

• ArcGIS can be extended to implement new functionality using Python
• Python can also be used to automate repetitive tasks
• Tools can leverage web services to download data
• Python (and .Net) bindings have been developed for most of the ArcToolBox
• Documentation for every tool is provided on the ArcGIS website