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

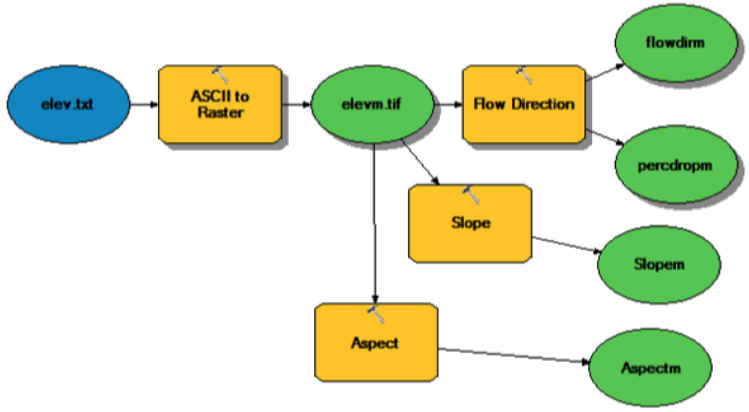
Using ArcMap Functions
http://resources.arcgis.com/en/help/main/10.1/index.html#/Using_functions_in_Python/002z0000000m000000

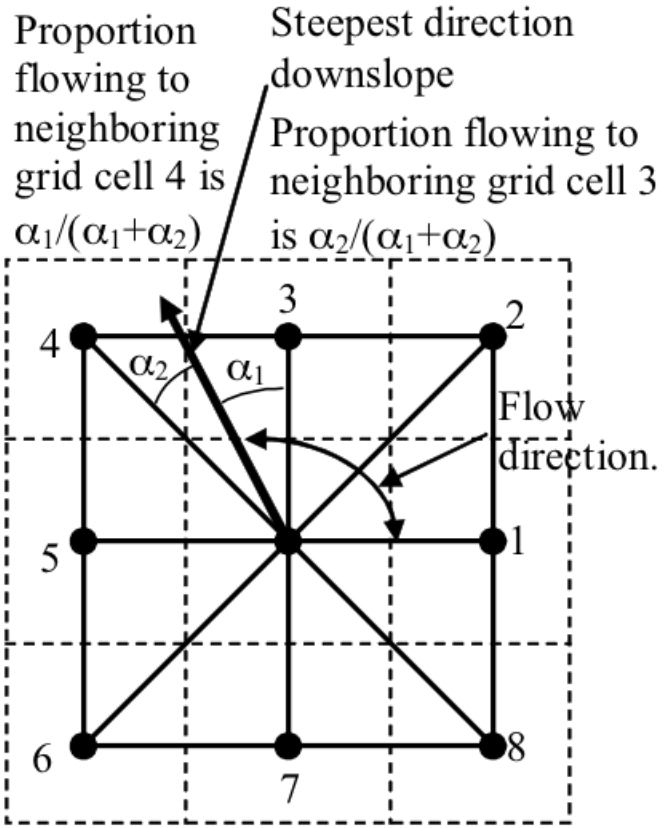
Using Python Classes
http://resources.arcgis.com/en/help/main/10.1/index.html#/Using_classes_in_Python/002z0000000s000000/

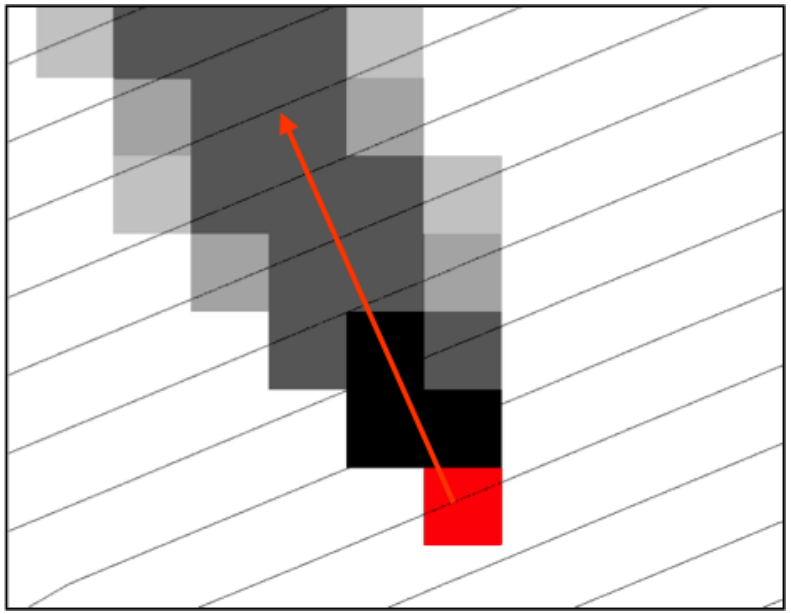
Accessing ArcGIS Licenses http://resources.arcgis.com/en/help/main/10.1/index.html#/Accessing_licenses_and_extensions_in_Python/002z0000000z000000/

Why Programming?

- Automation of repetitive tasks $\rightarrow$
- Implementation of functionality that would not otherwise be available $\downarrow$

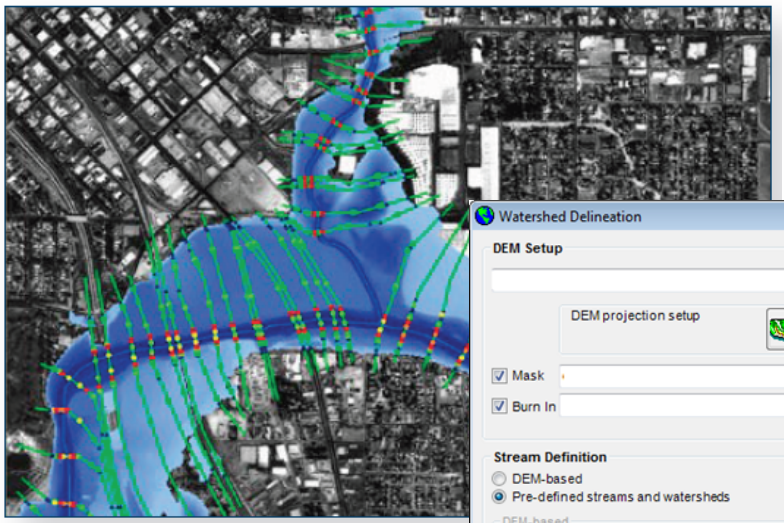


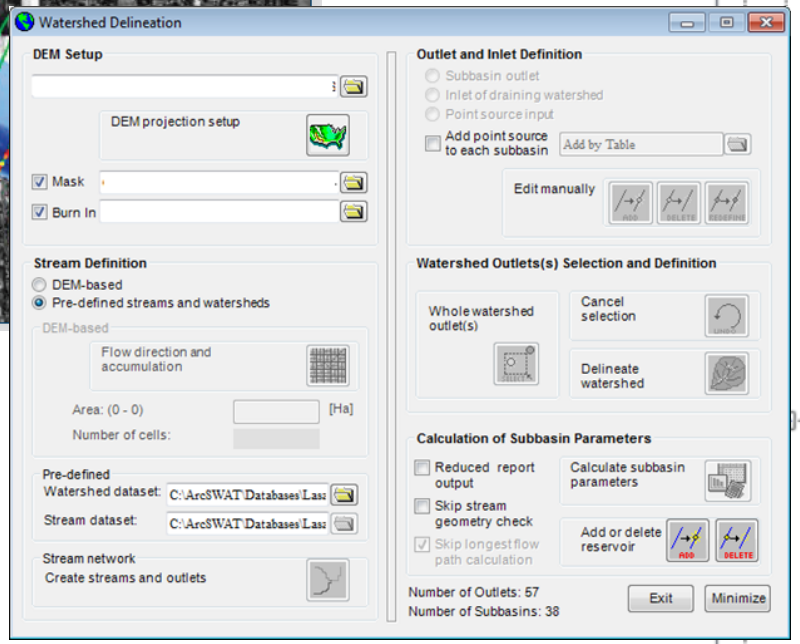


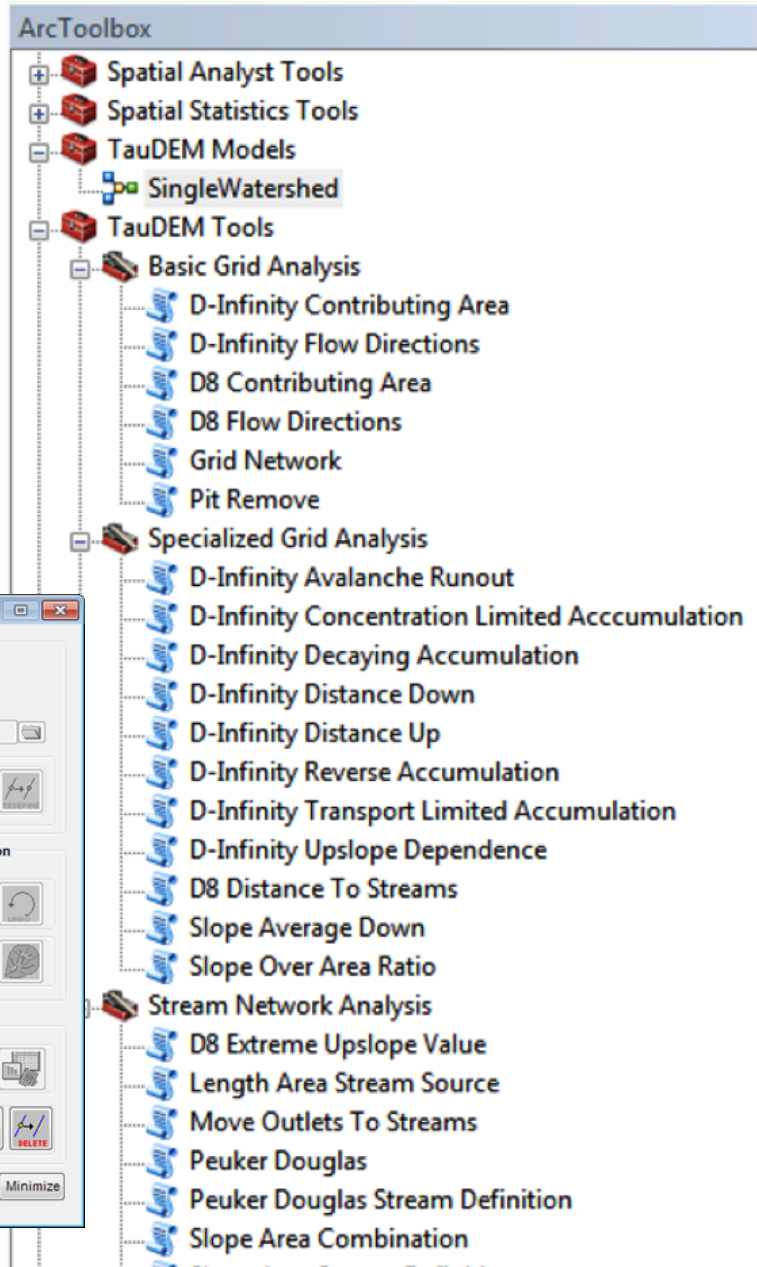


Extending ArcGIS

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



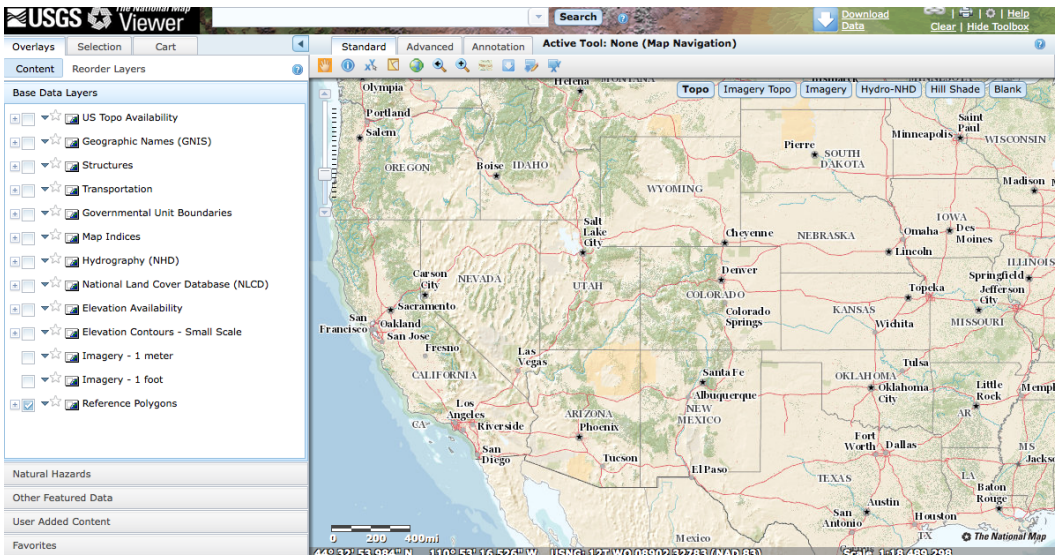


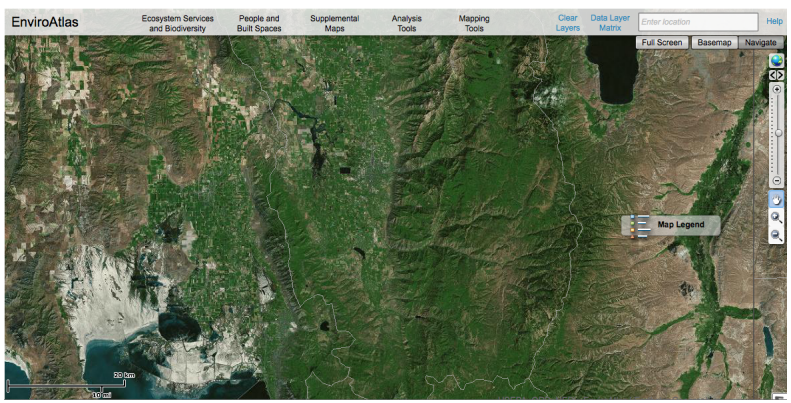


Workflow Automation

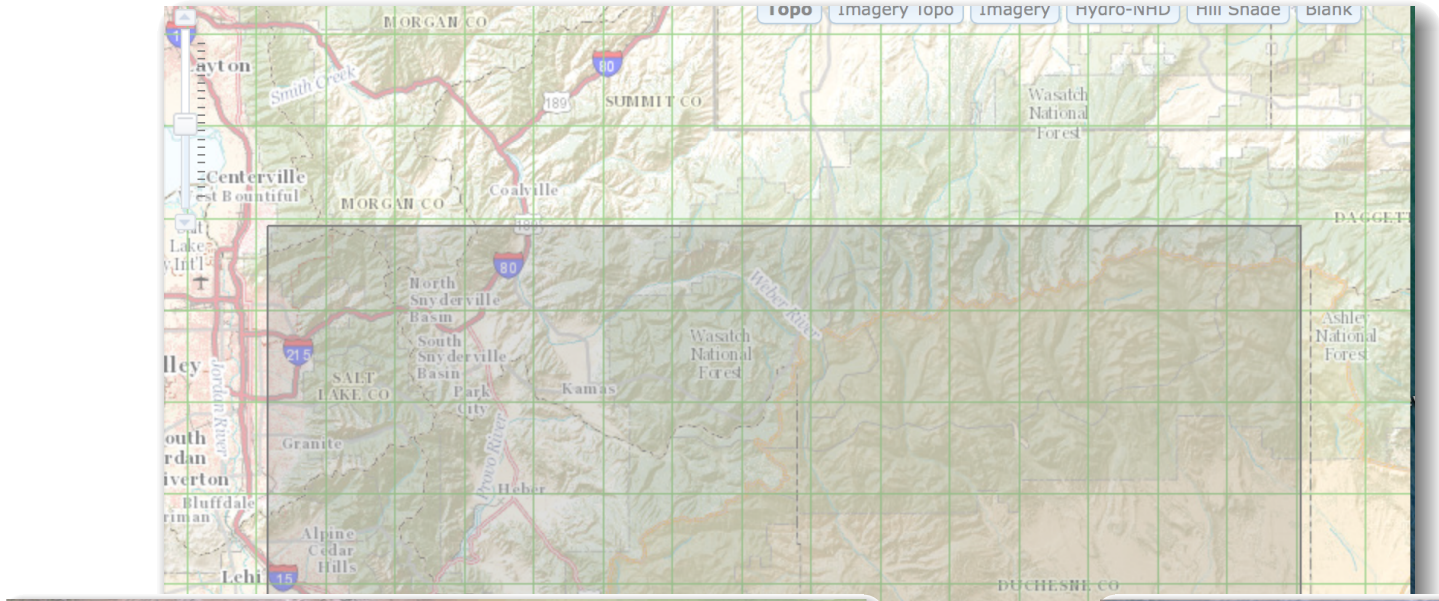
Lets Consider Data Collection

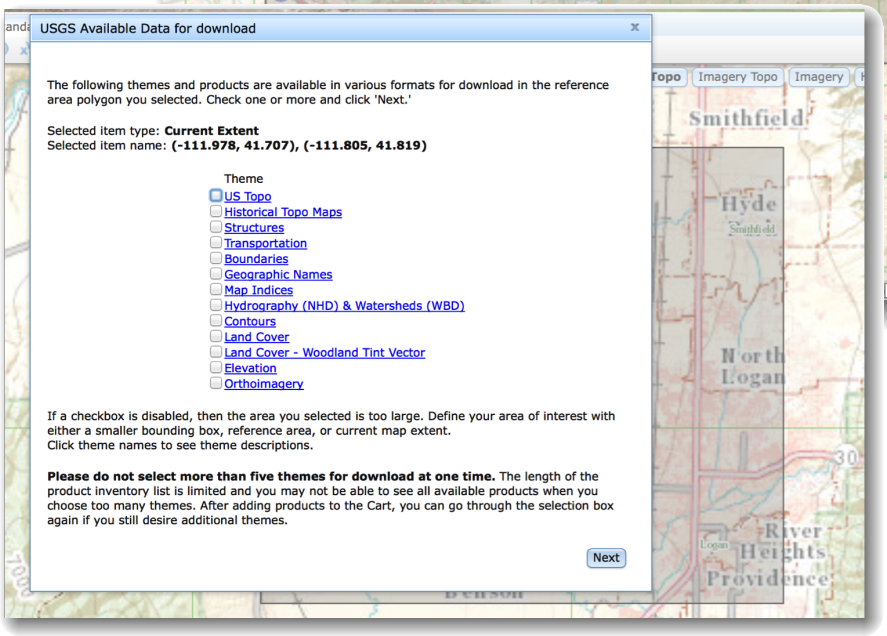
- United State Geological Survey, USGS
- Interactive map selection
- Tiled Data
- Emailed links to download datasets

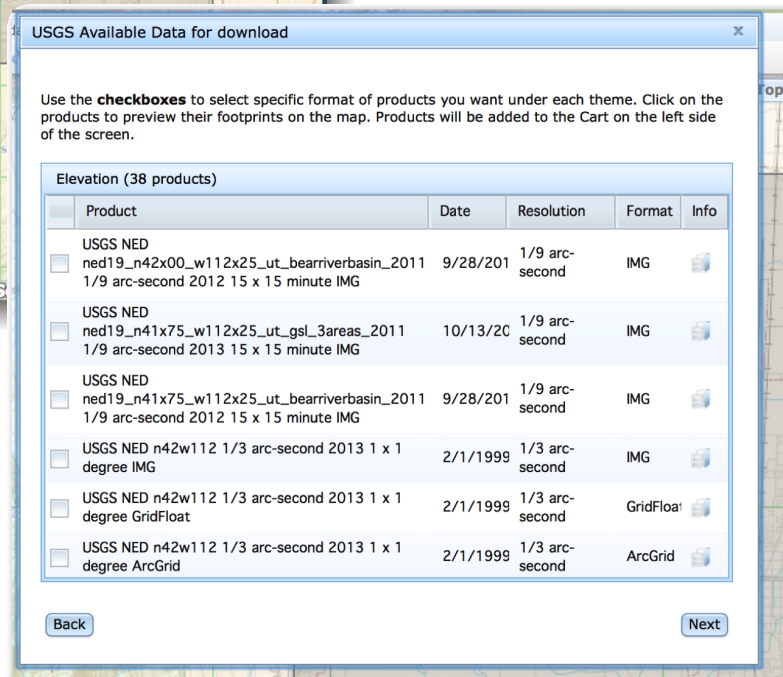




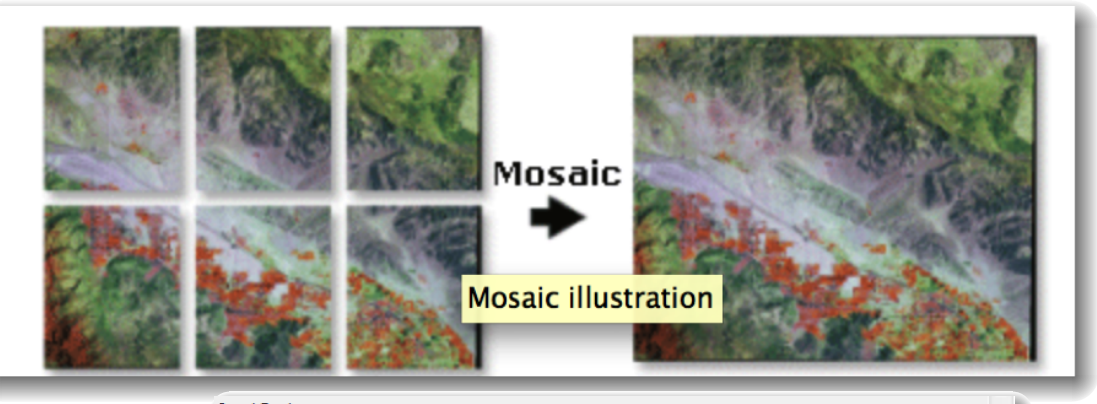
Downloading Data from the National Map Viewer

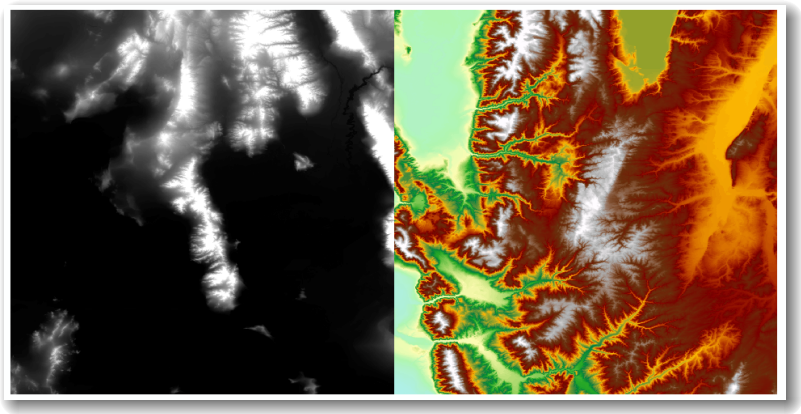


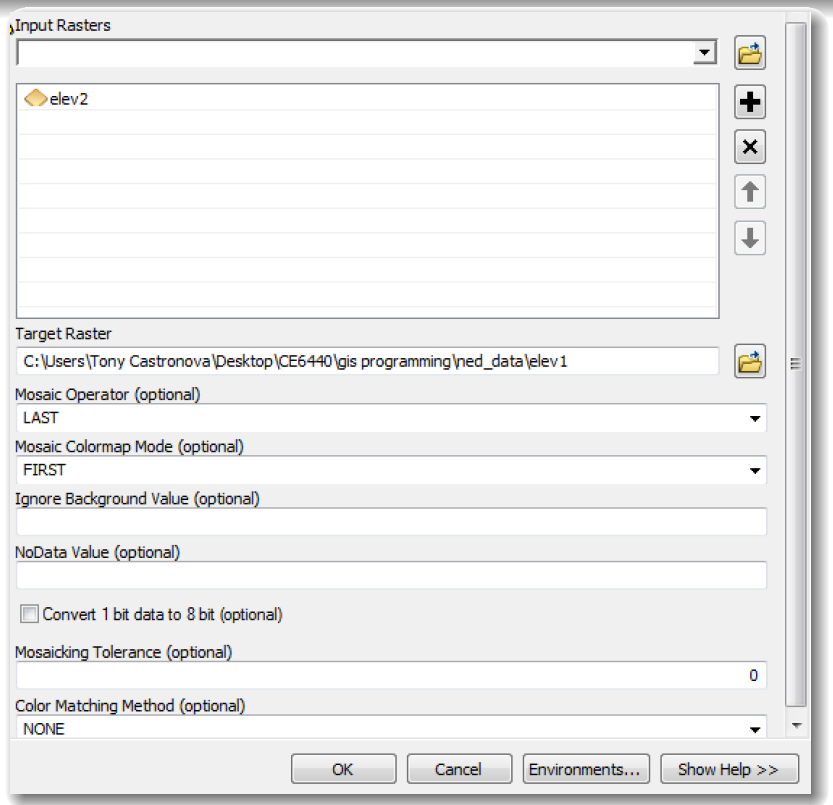


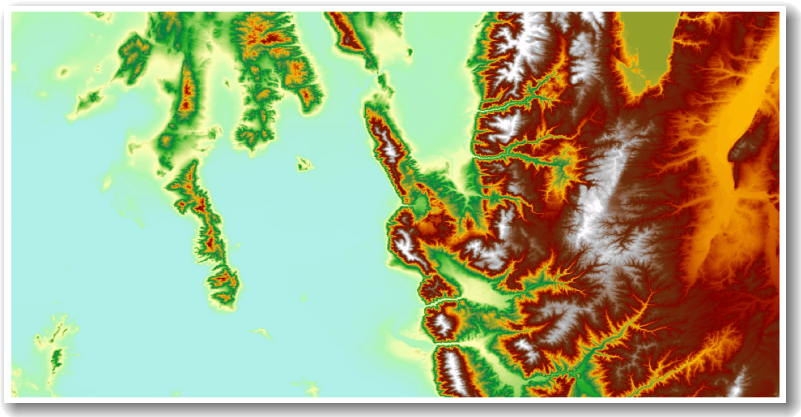


ArcGIS Processing









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

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

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

```
DO
    block 1
IF <condition>
    exit
ENDIF
```

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

```
DOFOR i = start, finish, step
    block
ENDDO
```

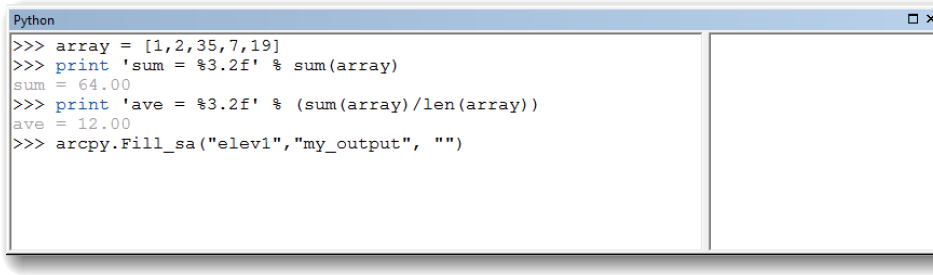
```
for i in range(0, 10, 1):
    print 'Hello'
```

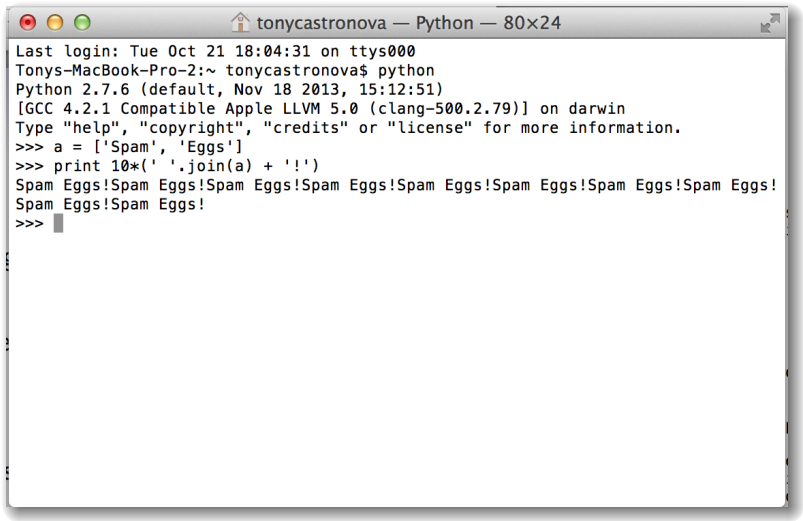
Example

```
1   def quadratic(a, b, c):
2
3       # calculate the discriminant
4       d = b ** 2 - 4 * a * c
5
6       if d < 0:
7
8           print ("This equation has no real solution")
9
10          # return the answer
11          return (None, None)
12
13      elif d == 0:
14
15          x = (-b+math.sqrt(b**2-4*a*c))/2*a
16          print ("This equation has one solutions: "), x
17
18          # return the answer
19          return (x,None)
20
21      else:
22
23          x1 = (-b+math.sqrt((b**2)-(4*(a*c))))/(2*a)
24          x2 = (-b-math.sqrt((b**2)-(4*(a*c))))/(2*a)
25          print ("This equation has two solutions: ", x1, " or", x2)
26
27          # return the answer
28          return (x1, x2)
```

What is Python

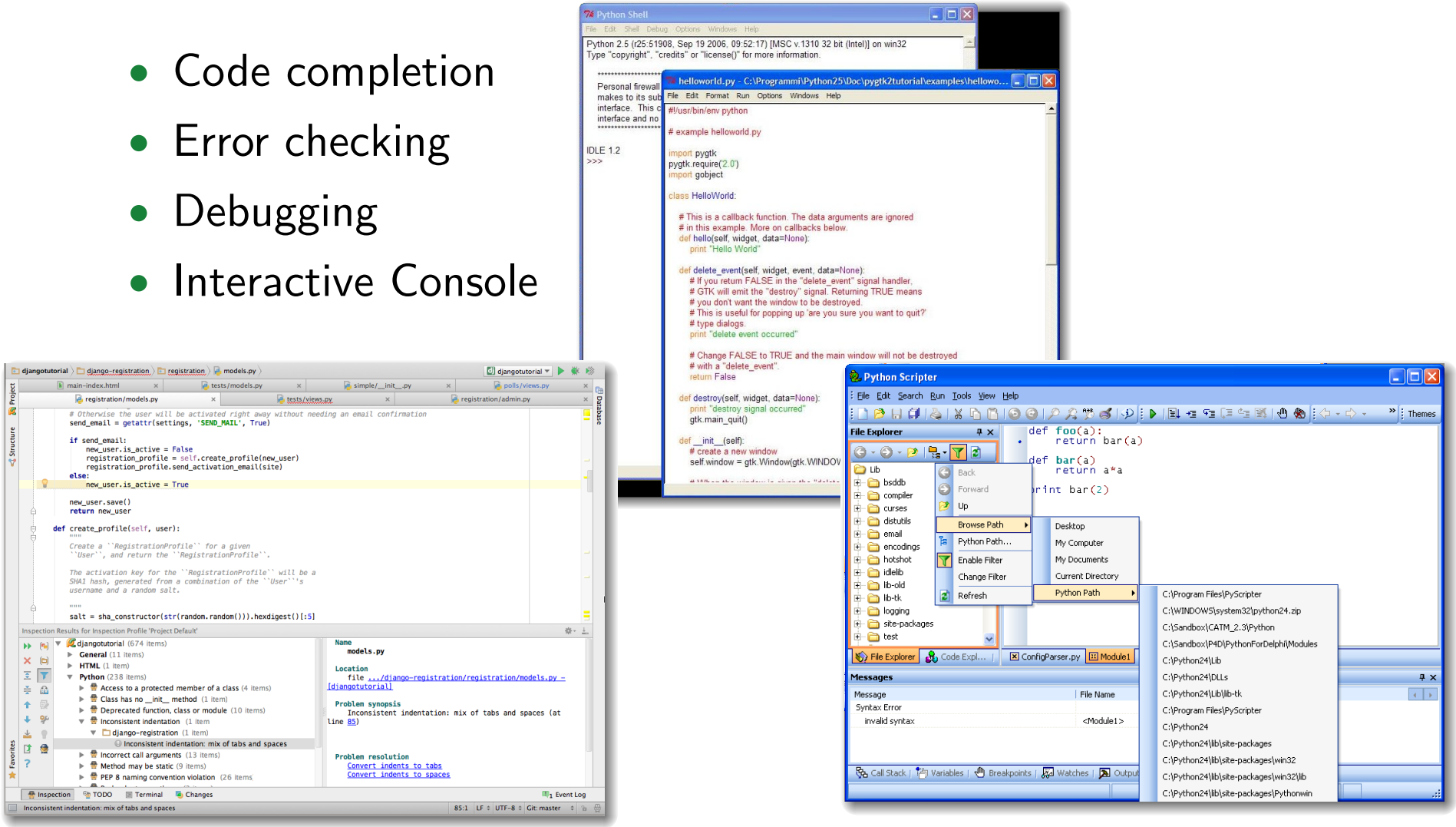
- Interpreted programming language
- Built from C
- Platform Independent (unlike .Net)
- Relatively lightweight
- No need to compile
- Many different versions (latest 3.4.2, most used is 2.7.8)
- ArcGIS already installed it!
- Execute tools inside ArcMap
- Execute tools/code using CMD/Terminal/Console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

Object Type	**Example**
Numbers	1234, 3.1415
Strings	'spam', "eggs"
Lists	[1, 2, 3], [1, ['spam', 'eggs']]
Dictionaries	'food':'spam','taste':'yum'

```
# 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 os
import sys
import numpy
import suds
```

- Import statements
- Control statements
- Functions
- Classes

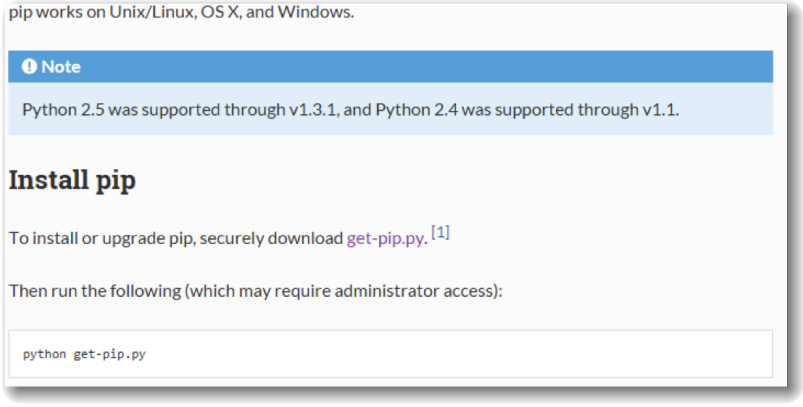
```
for i in range(0,10):
    # do something

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


pip works on Unix/Linux, OS X, and Windows.

Note

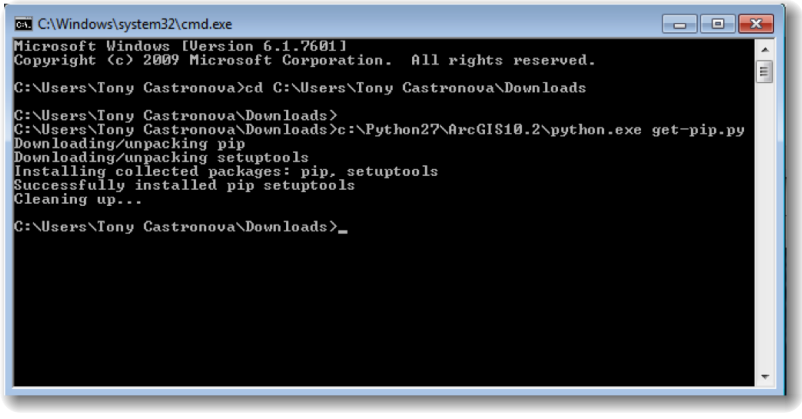
Python 2.5 was supported through v1.3.1, and Python 2.4 was supported through v1.1.

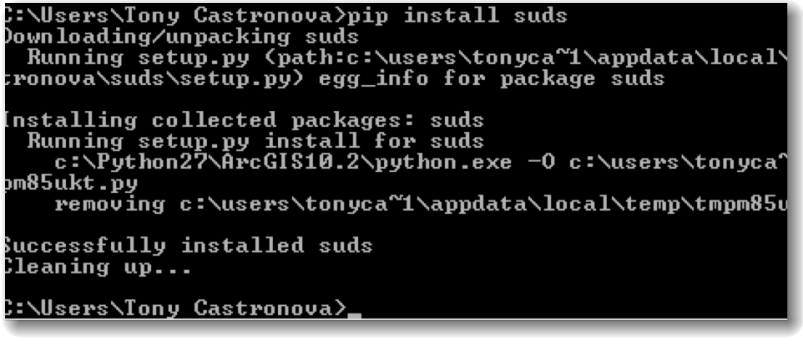
Install pip

To install or upgrade pip, securely download get-pip.py. [1]

Then run the following (which may require administrator access):

```
python get-pip.py
```

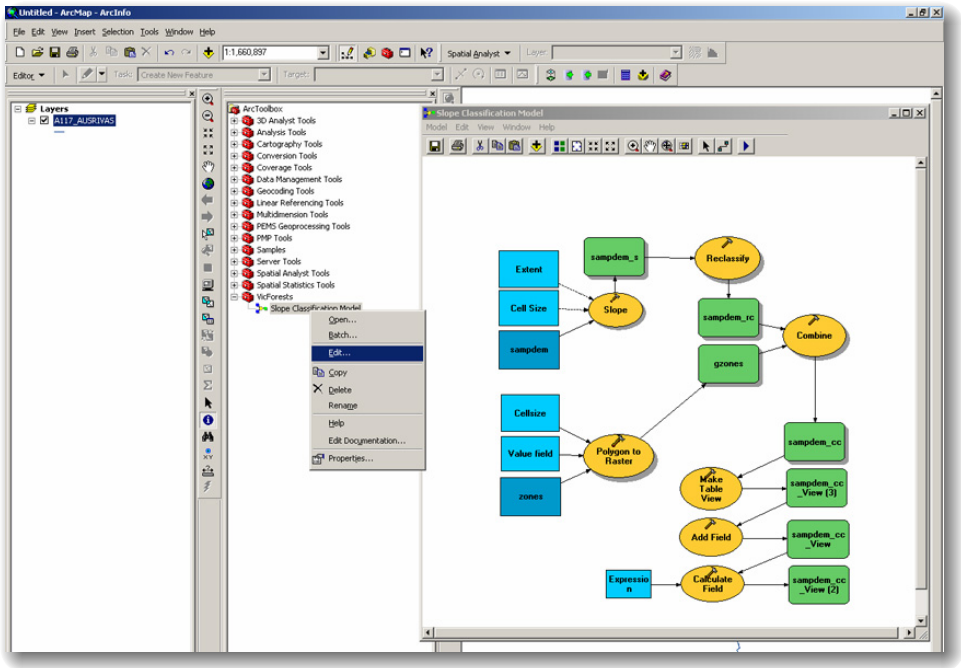




https://pip.pypa.io/en/latest/installing.html

ArcGIS Programming in Python

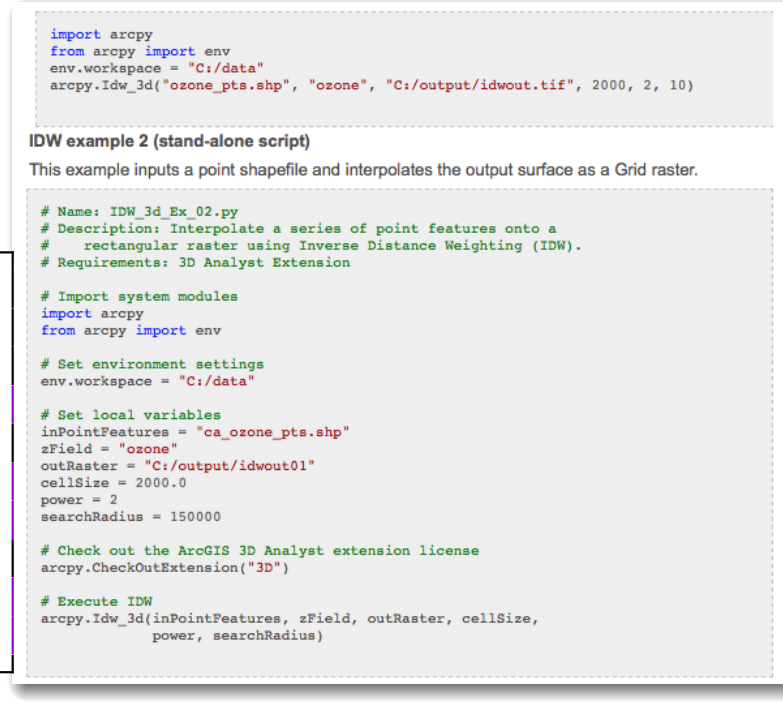
- ArcPy is the ArcGIS python module
- Provides a productive way to perform geographic data analysis using ArcGIS tools
- Organized into tools, functions, classes, and modules



ArcGIS Tools

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

```
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!", "
    PYTHON_9.3")
arcpy.FeatureClassToFeatureClass_conversion("c:/data/
    Portland.gdb/streets", "Database Connections/MySDE.
    sde/PortlandDataset", "streets")
```



```
import arcpy
from arcpy import env
env.workspace = "C:/data"
arcpy.Idw_3d("ozone_pts.shp", "ozone", "C:/output/idwout.tif", 2000, 2, 10)
```

IDW example 2 (stand-alone script)

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

```
# 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 = "ca_ozone_pts.shp"
zField = "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, zField, outRaster, cellSize,
             power, searchRadius)
```

http://resources.arcgis.com/en/help/main/10.1/index.html#//002t0000000z000000

ArcPy Functions

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

```
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

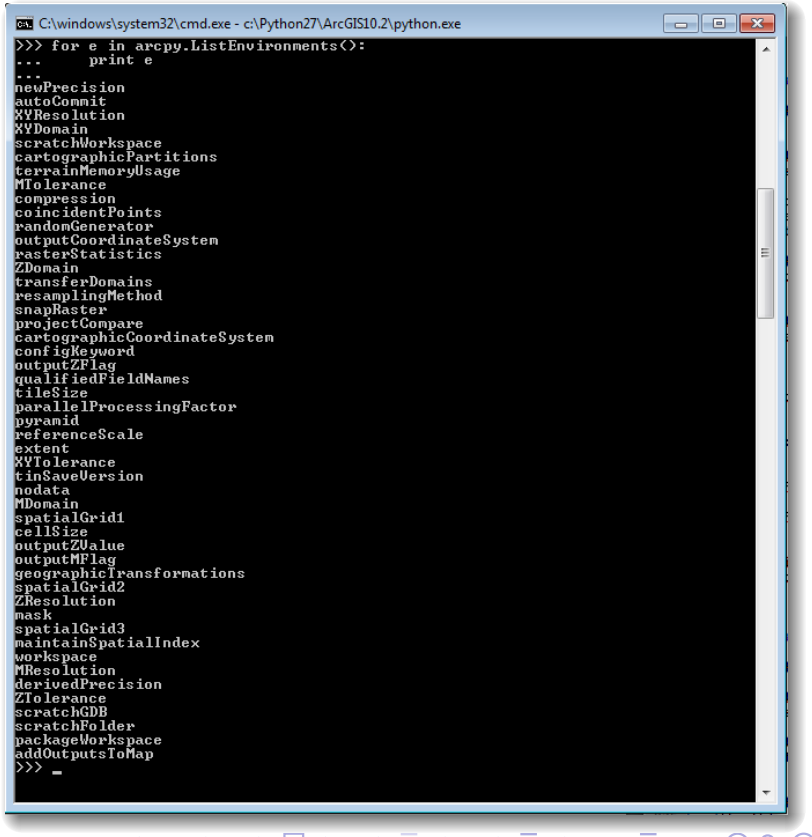
- Architectural blueprint
- Class objects are called instances
- Often used as shortcuts for completing a geoprocessing tool

```
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

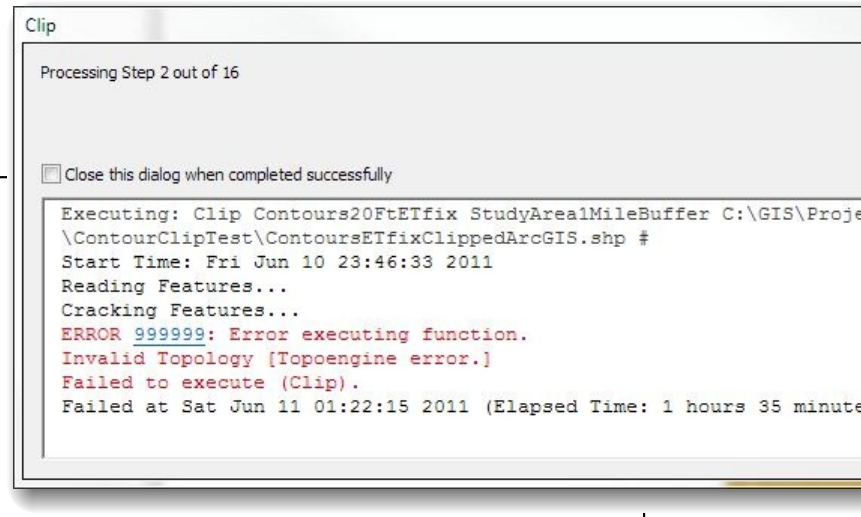
- Global settings
- Default values can be overridden
- nodata value, output spatial reference, cell size

```
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




```
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")
```

ArcPy Scripting, pt1

```
1   __author__ = "tonycastronova"
2
3   # Tony's ArcGIS script
4   # This is a demo for CEE 6440
5   # 10/22/2014
6
7   # Import system modules
8   import arcpy
9
10  # Set workspace
11  env.workspace = "C:/data"
12
13  # Set local variables
14  in_features = "majorrds.shp"
15  clip_features = "study_quads.shp"
16  out_feature_class = "studyarea.shp"
17  xy_tolerance = ""
18
19  try:
20      # Execute Clip
21      arcpy.Clip_analysis(in_features, clip_features, out_feature_class, xy_tolerance)
22  except Exception:
23      print 'Something went wrong :( '
24      for i in arcpy.GetMessageCount():
25          print i
```

- Establish the basic structure of the script
- Look on ArcGIS website for tool definitions
- Make sure this runs before we move on

ArcPy Scripting, pt2

```
__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 ...
```

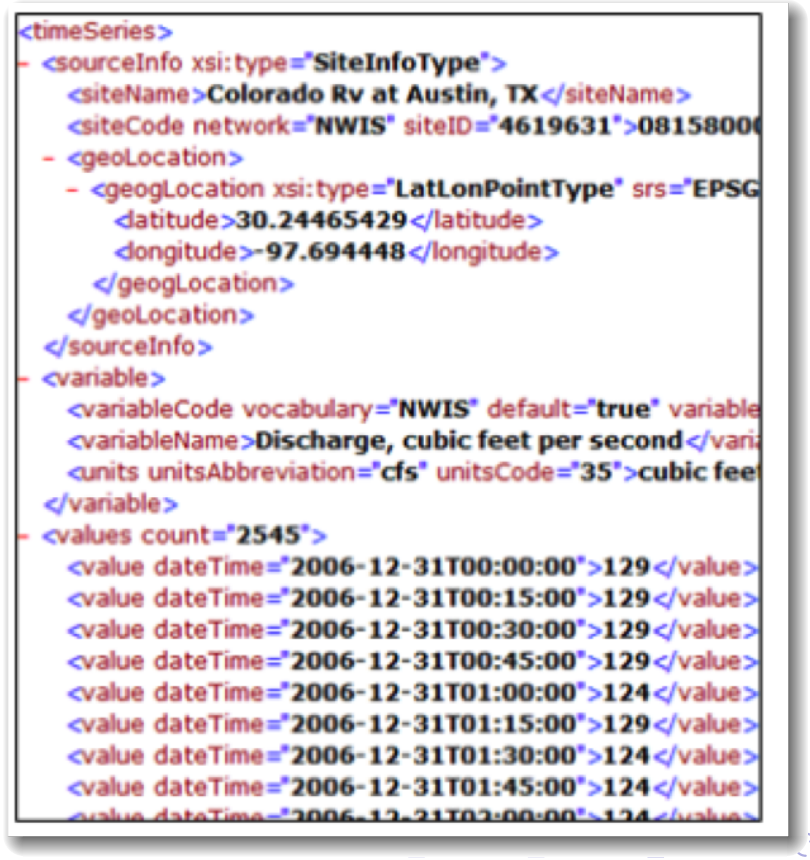
ArcPy Scripting, pt3

```
30  # loop over files in directory and clip
31  for file in files:
32      in_features = file
33      out_features = file.split('.')+'_clip.shp'
34      try:
35          # Execute Clip
36          arcpy.Clip_analysis(in_features, clip_features, out_feature_class, xy_tolerance)
37      except Exception:
38          print 'Something went wrong :( '
39          for i in arcpy.GetMessageCount():
40              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 webservices!
- USGS - streamflow, groundwater, water quality
- EPA STORET - water quality
- DAYMET - meteorological surface temperature, precipitation, humidity, etc.
- MODIS - surface temperature, snow cover, vegetation, etc.

`http://daymet.ornl.gov` `http://www.epa.gov/storet/dbtop.html`

`http://modis.gsfc.nasa.gov` `http://waterdata.usgs.gov/nwis`

`http://his.cuahsi.org/wofws.html`

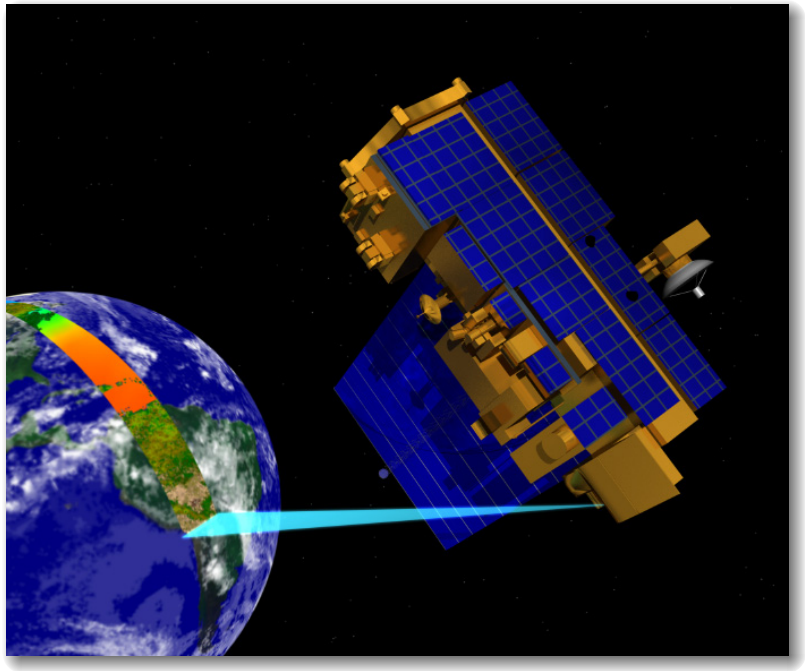
Data Collection using Python

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

```
1  from suds.client import Client
2  url = 'http://river.sdsc.edu/wateroneflow/NWIS/DailyValues.asmx?wsdl'
3
4  # create an instance of the suds Client class
5  client = Client(url)
6
7  # see what functions are available
8  print client
9
10 # call a function
11 client.service.GetValues('NWIS:09429000', 'NWIS:00060',
12 '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)



Practical Example - Build the base script

```
1  import os                    # standard python package
2  import arcpy                 # arcmap package
3  import modis_base            # package for organizing MODIS data
4  from suds.client import *    # web service client
5
6  # set overwrite to true
7  arcpy.env.overwriteOutput = True
```

```
import numpy

class ModisData():
    def __init__(self, modisdata):
        self.__modisdata = modisdata

        self.__data = {}

        self._set_data()

    def mosisObj(self):
        return self.__modisdata

    def get_cellsize(self):
        return self.__modisdata.cellsize

    def get_bbox(self):
        #xmin ymin xmax ymax]
        return [self.__modisdata.xllcorner,
                self.__modisdata.yllcorner,
                self.__modisdata.xllcorner+self.__modisdata.ncols*self.__modisda
                self.__modisdata.yllcorner+self.__modisdata.nrows*self.__modisda


    def _set_data(self):

        subsets = self.__modisdata.subset
        data = []
        cols = int(self.__modisdata.ncols)

        for row in subsets:
            data = []
            values = row.split(',')
            date = values[2]
            for idx in range(5,len(values),cols):
                start = idx
                end = idx+cols
                data.append([float(val) for val in values[start:end]])

            self.__data[date] = numpy.array(data)

    def get_data(self):
        return self.__data
```

Practical Example - Add Web Services

```
1   import os                    # standard python package
2   import arcpy                 # arcmap package
3   import modis_base            # package for organizing MODIS data
4   from suds.client import *    # web service client
5
6   # setup environment
7   arcpy.env.overwriteOutput = True
8   arcpy.env.workspace = os.getcwd()
9
10  pts = [ (40.115,-110.025),       # add some points to collect data at
11      (40.144341, -109.807629),
12      (39.927865, -109.867392),
13      (40.193480, -110.353466),
14      (40.068641, -110.150605),
15      (40.124420, -109.910932)
16      ]
17
18  output_files = {}                # dictionary to store the results
19  i = 0                            # counter
20  for lat, lon in pts:             # loop over each lat,lon in pts list
21
22      print 'Querying MODIS data...'      # print statements can be helpful
23      wsdl = 'http://daac.ornl.gov/cgi-bin/MODIS/GLBVIZ_1_Glb_subset/MODIS_webservice.wsdl'
24
25      client = Client(wsdl)        # Create instance of suds client
26
27      # call the web service to retrieve data
28      lat, lon = pt
29      result = client.service.getsubset(lat, lon, "MOD11A2","LST_DAY_1km",
30                                        "A2001001","A2001025",10,10)
```

http://daac.ornl.gov/MODIS/MODIS-menu/modis_webservice.html

Practical Example - Add Web Services, pt2

```
33      # ... continued from previous slide
34
35      # organize the results
36      print 'Building Modis Object...'
37      modis_data = modis_base.ModisData(result)
38
39      # get info about the dataset
40      data = modis_data.get_data()
41      modisObj = modis_data.mosisObj()
42      rows = modisObj.nrows
43      cols = modisObj.ncols
44      xmin,ymin,xmax,ymax = modis_data.get_bbox()
45      cellsize = modis_data.get_cellsize()
```

DEMO

Practical Example - Build Rasters, pt1

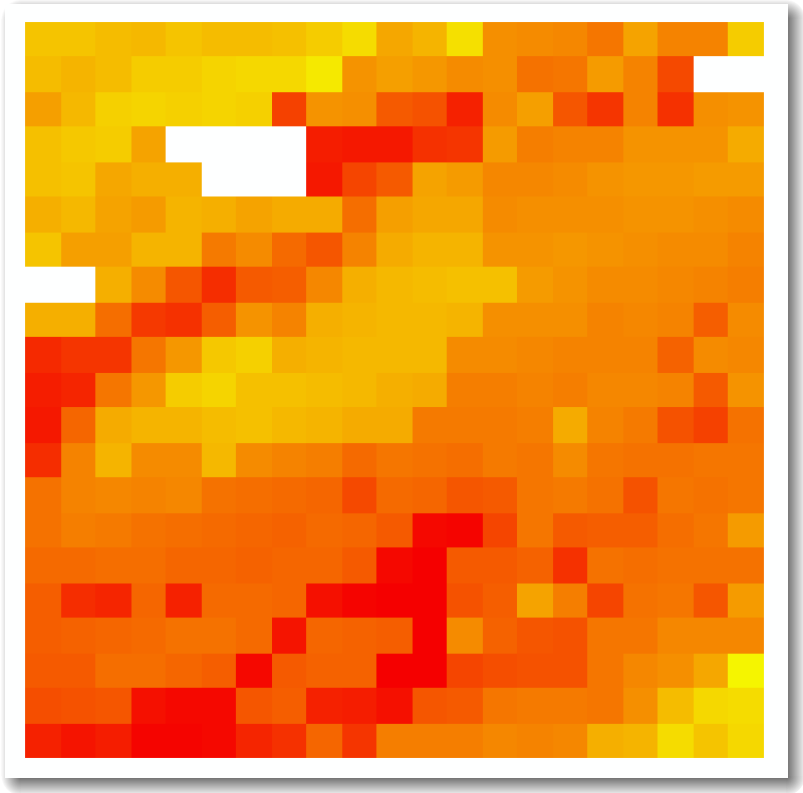
```
33      # create ASCII Raster for all dates retrieved
34      for date in data.keys():
35
36          filename = 'modis_temp_'+date+'.txt'
37          ascii_path = os.getcwd()+'/temp/'+filename
38          with open(ascii_path,'w') as f:
39
40              # write header data
41              f.write('NCOLS %s\n'% cols)
42              f.write('NROWS %s\n'% rows)
43              f.write('XLLCORNER %s\n'% xmin)
44              f.write('YLLCORNER %s\n'% ymin)
45              f.write('CELLSIZE %s\n'% cellsize)
46              f.write('NODATA_VALUE %s\n' % 0)
47
48              d = data[date]
49              for row in d:
50                  for value in row:
51                      if value != 0.0:
52                          value = (value*.02 - 273.15)*1.8 + 32.
53
54                      f.write('%3.2f ' % value )
55                  f.write('\n')
56
57              print 'Process: ASCII to Raster'
58              raster_name = filename.split('_')[-1][:-4]+'_'+str(i)
59              raster = os.path.join(os.getcwd()+'/raster',raster_name)
60              arcpy.ASCIIToRaster_conversion(ascii_path, raster, "FLOAT")
61
62              # continued on next slide ...
```

Practical Example - Build Rasters, pt2

```
63          # ... still inside loop
64
65          print 'Process: Define Projection'
66
67          # Define a custom projection as WKT (MODIS projection)
68          prj = "PROJCS['Sinusoidal', " + \
69                "GEOGCS['GCS_Undefined',DATUM['D_Undefined',SPHEROID['User_Defined_Spheroid',"+\
70                "6371007.181,0.0]],PRIMEM['Greenwich',0.0],UNIT['Degree',0.0174532925199433]],"+\
71                "PROJECTION['Sinusoidal'],PARAMETER['False_Easting',0.0],"+\
72                "PARAMETER['False_Northing',0.0],PARAMETER['Central_Meridian',0.0],"+\
73                "UNIT['Meter',1.0]]"
74
75          arcpy.DefineProjection_management(raster, prj)
76
77          # store all output file names in a dictionary based on date
78          if date in output_files:
79              output_files[date].append(raster)
80          else:
81              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.
48.15 47.14 46.85 46.45 45.34 44.94 0.00 44.08 44.94 4
46.74 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
0.00 0.00 0.00 48.54 45.45 43.57 43.90 43.79 43.72 43.
0.00 0.00 0.00 44.37 43.75 43.90 43.79 43.68 43.36 43.
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
```

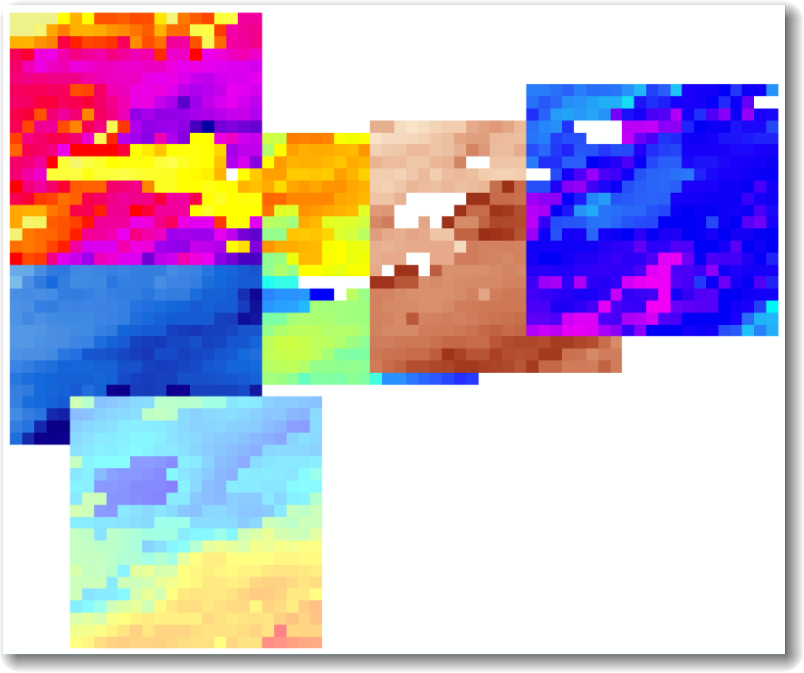


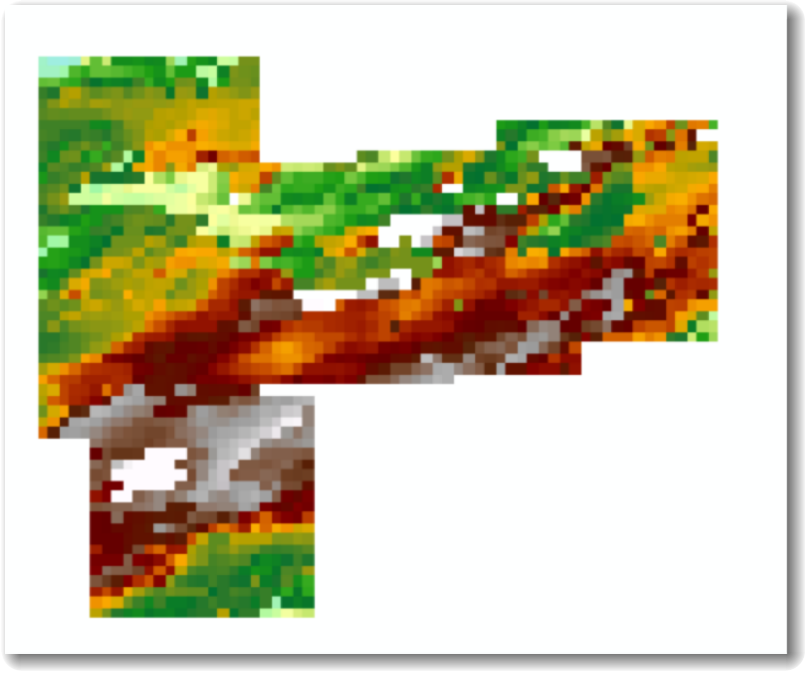
Practical Example - Add Mosaic Tool

```
83    print 'Process: Mosaic'
84    for key in output_files.keys():
85
86        # build a new mosaic dataset
87        file_list = output_files[key]
88        files = ';'.join(file_list)
89        arcpy.MosaicToNewRaster_management( files, '/mosaic', key, prj,
90                                            "8_BIT_UNSIGNED", "40", "1",
91                                            "LAST","FIRST")
```

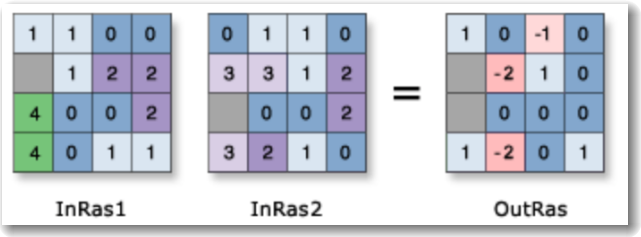
```
==================================
Mosaic
Usage: Mosaic_management inputs;inputs... target
       {LAST | FIRST | BLEND | MEAN | MINIMUM | MAXIMUM} {FIRST | REJECT | LAST | MATCH}
       {background_value} {nodata_value} {NONE | OneBitTo8Bit} {mosaicking_tolerance}
                          {NONE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

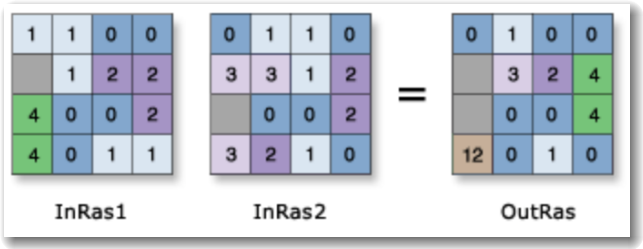
Mosaic Results





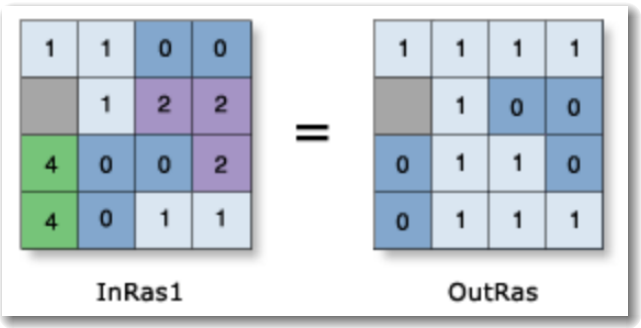
How about some Geoprocessing

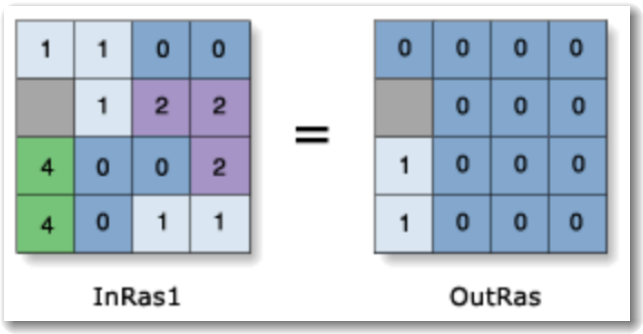




- Determine locations where surface temperature increased
- Determine locations where surface temperature decreased

```
1  # Check out the ArcGIS extension license
2  arcpy.CheckOutExtension("Spatial")
3  from arcpy.sa import *
4
5  # calculate locations where surface temperature increased
6  r1 = '/mosaic/'+output_files.keys()[0]
7  r2 = '/mosaic/'+output_files.keys()[-1]
```



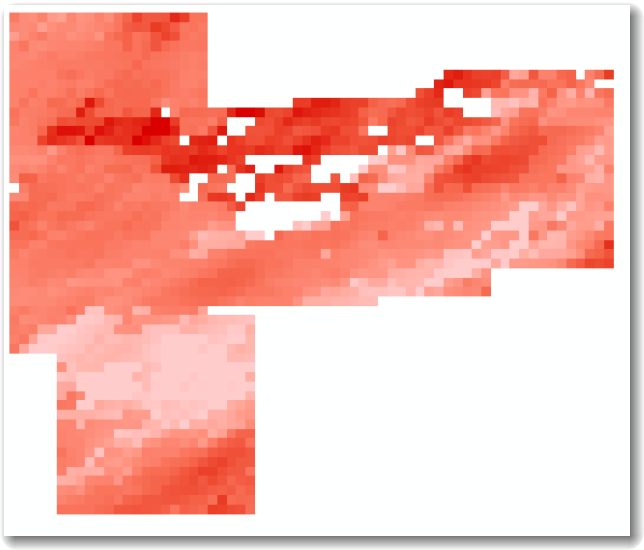


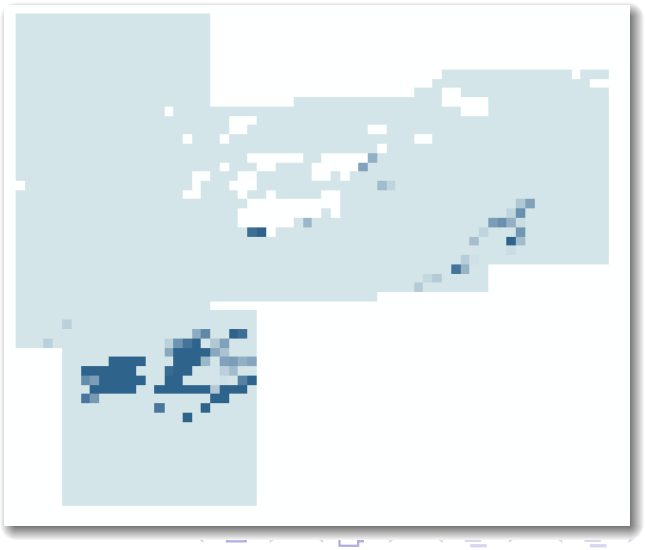
Calculating Temperature Increase

```
 9    # ... continued from previous slide
10
11    # Execute Minus
12    print 'Executing Minus...'
13    diff = os.getcwd()+'/temperature/diff'
14    outMinus = Minus(r2, r1)
15    outMinus.save(diff)
16
17    # Execute GreaterThan: find all cells greater than 0
18    print 'Execute GreaterThan...'
19    gt = os.getcwd()+'/temperature/gt'
20    outGreaterThan = GreaterThan(diff, 0)
21    outGreaterThan.save(gt)
22
23    # Execute Times isolate all cells where temperature increased
24    print 'Execute Times...'
25    inc = os.getcwd()+'/temperature/increased'
26    outTimes = Times(diff, gt)
27    outTimes.save(inc)
```

Calculating Temperature Decrease

```
27  # ... continued from previous slide
28
29  # Execute LessThan: find all cells less than 0
30  print 'Execute LessThan'
31  lt = os.getcwd()+'/temperature/lt'
32  outLessThan = LessThan(diff, 0)
33  outLessThan.save(lt)
34
35  # Execute Times isolate all cells where temperature increased
36  print 'Execute Times...'
37  dec = os.getcwd()+'/temperature/decreased'
38  outTimes = Times(diff, lt)
39  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