Writing Geoprocessing Scripts With ArcGIS

Lecture 10
GIS System

ArcSDE® Gateway is an interface for managing geodatabases in numerous relational database management systems (RDBMSs).
ArcObjects

- ArcObjects are the building blocks of ArcGIS. With ArcObjects, you can create your own menus, tools, workflows, applications, and custom feature classes for use with ArcGIS.

- ESRI ArcObjects is the development platform for the ArcGIS family of applications, such as ArcMap, ArcCatalog, ArcScene, ArcGIS Engine, and ArcGIS Server. The ArcObjects software components expose the full range of functionality available in ArcInfo, ArcEditor, and ArcView to software developers.

- Can use VBA, Python, C++, Java to program.
Common ArcObjects

- ArcGIS is built with a set of ArcObjects
Common ArcObjects

- Symbol
- Field
- Row
- Feature Class
- Selection

[Image of ArcGIS interface showing map and data attributes]
Interacting with ArcObjects

- Each ArcObject has properties and methods
  - Property: Characteristic of an object (a noun)
  - Method: Something the object knows how to do (a verb)

- Interact with objects through properties and methods

<table>
<thead>
<tr>
<th>Map</th>
<th>Feature Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>Properties</td>
<td>Properties</td>
</tr>
<tr>
<td>- Layer count</td>
<td>- Shape type</td>
</tr>
<tr>
<td>- Name</td>
<td>- Spatial reference</td>
</tr>
<tr>
<td>- Spatial reference</td>
<td>- Extent</td>
</tr>
<tr>
<td>- Map scale</td>
<td>- Methods</td>
</tr>
<tr>
<td>- Extent</td>
<td>- Create feature</td>
</tr>
<tr>
<td>Methods</td>
<td>- Remove feature</td>
</tr>
<tr>
<td>- Add layer</td>
<td></td>
</tr>
<tr>
<td>- Clear selection</td>
<td></td>
</tr>
<tr>
<td>- Select feature</td>
<td></td>
</tr>
</tbody>
</table>
The Geoprocessor (GpDispatch) ArcObject

- Most geoprocessing functionality on one ArcObject
  - Geoprocessor or GpDispatch
- The Geoprocessor has many properties and methods

<table>
<thead>
<tr>
<th>Geoprocessor</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Properties</td>
<td></td>
</tr>
<tr>
<td>- Current workspace</td>
<td></td>
</tr>
<tr>
<td>- Cluster tolerance</td>
<td></td>
</tr>
<tr>
<td>- Cell size</td>
<td></td>
</tr>
<tr>
<td>Methods</td>
<td></td>
</tr>
<tr>
<td>- Buffer</td>
<td></td>
</tr>
<tr>
<td>- Clip</td>
<td></td>
</tr>
<tr>
<td>- Select</td>
<td></td>
</tr>
<tr>
<td>- Import from CAD</td>
<td></td>
</tr>
<tr>
<td>- Copy features</td>
<td></td>
</tr>
<tr>
<td>- Add field</td>
<td></td>
</tr>
</tbody>
</table>

Environment settings

Tools
Accessing the Geoprocessor from Python

- Geoprocessor can be used in many languages
  - Perl, VBScript, JScript, Python, VBA, VB, C#, and so on
  - Any COM compliant language

For ArcGIS 10 and Python 2.6.5:

```python
import arcpy
```

Tells Python to import basic ArcGIS geoprocessing functionality

```python
from arcpy import env
```

Tells Python to import ability to control the ArcGIS Environment

```python
from arcpy.sa import *
arcpy.CheckOutExtension("Spatial")
```

Example: Tells Python to import functionality from ArcGIS Spatial Analyst and a second command to get/check the Spatial Analyst license
Syntax for properties and methods

♦ To assign a value to a property

    Object.Property = Value

    env.workspace = "C:/Temp"

♦ To get the value of a property

    Object.Property

    print "The name of the workspace is " + env.workspace

♦ To use a method

    Object.Method (arg, arg, ...)

    arcpy.Buffer_analysis ("C:/input/roads.tif", "C:/output/Output.gdb/buffer_output", 100)

♦ Parentheses around arguments

♦ Arguments separated by commas
Toolbox aliases

Always suffix the tool with the toolbox alias

```python
arcpy.Clip_arc()  
arcpy.Clip_analysis()
```

Always assign names to custom toolboxes
The Select tool

- Syntax

```
Select_analysis (in_features, out_feature_class, where_clause)
```

- Example

```
env.workspace = "c:/basedata/roads.gdb"

arcpy.Select_analysis("nfroads", "paved", '[ROAD_CLASS] = "PAVED"')
```

- Notes

Python uses forward slashes, different than Windows using back slashes

Double quotes pass text strings

Single quotes contain text strings intended to pass variable names
The Where clause Syntax

- SQL expression
  - Dependant on the underlying database
  - Shape file
    - "NAME" = ‘Toronto’
  - Geodatabase
    - [NAME] = ‘Toronto’
  - The easiest way
    - Open the Select tool, or the Select By Attributes dialog, and
    - Copy the syntax that these dialogs use.
Python Basics
Variables in Python

- Variables are dynamically typed
  - No declaration keyword
  - No type assignment
    
    \[ fc = "P:/Qiu/PYTH/Database/World/Cities.shp" \]

- Variables are case sensitive

  \[
  \begin{align*}
  \text{scale} & = 10000 \\
  \text{Scale} & = 20000
  \end{align*}
  \]

- Variables can hold different data types
  - Strings, numbers, lists, tuples, dictionaries, files
Strings

- Variables can hold strings
  ```
  folder = "C:/Student"
  whereClause = "[STREET_NAM] = 'CATALINA'"
  ```

- Strings surrounded in double (" ) or single ( ' ) quotes
  - Can embed one string in another

- Pathnames use two back (\ ) or one forward (/ ) slash
  - \ is a reserved escape character and a line continuation character.

- Strings can be combined together
  ```
  gdbPath = "C:/SouthAfrica.mdb"
  fc = "Roads"
  fullPath = gdbPath + "/" + fc
  ```

- Strings are indexed, 0-based from the left and 1-based from right
  ```
  fc = "Street.shp"
  newFC = fc[:-4]
  fc[0] ---> "S" # S is in the 0 position
  fc[1:3] ---> "tr" # the last position is never included
  fc[:-4] ---> "Street" # get rid of the last 4 characters
  ```
Numbers and lists

- **Variables can hold numbers and expressions**
  
  \[
  \begin{align*}
  num1 &= 1.2 \\
  num2 &= 3 + 5
  \end{align*}
  \]

- **Variables can hold lists**
  
  \[
  \begin{align*}
  numList &= [1, 2, 3] \\
  fcList &= ["Roads", "Streets", "Parcels", "Zipcodes"]
  \end{align*}
  \]

- **Lists are indexed, 0-based from the left and 1-based from right**
  
  \[
  \begin{align*}
  fc1 &= fcList[1] \quad \rightarrow \text{"Streets"} \\
  fc2 &= fcList[0:2] \quad \rightarrow \text{"Roads", "Streets"} \\
  fc3 &= fcList[0:-1] \quad \rightarrow \text{"Roads", "Streets", "Parcels"} \\
  fc4 &= fcList[2:] \quad \rightarrow \text{"Parcels", "Zipcodes"}
  \end{align*}
  \]
  
  # Lists are zero-based, position one is "Streets".
  
  # The last position is not included.
  
  # Take values from zero to minus one. Last position not included.
  
  # Take all values from position two to the end.
Variable index

Word =

- Word[0] = ‘H’
- Word[:3] = ‘Hel’
- Word[-2:-4] = ‘el’
- Word[-3:] = ‘lpA’
Variable naming conventions

- Upper case versus lower case
  - First word lower case, capitalize each successive word
    ouputFieldName = "City"
  - Acronym at the beginning, use lower case letters
    gdbPath = "C:/Stockholm.mdb"
    fc = "Railroads.shp"
  - Acronym in the middle or at the end, use upper case letters
    inputFC = "Streets.shp"
- Avoid special characters (e.g. / \ & * !)
- Use descriptive variable names
Line continuation

◆ Line continuation characters
  ◆ Parentheses ( ), brackets [ ], and braces { }
  ◆ Backslash \n
◆ Indentation is automatic

# Date: May 3, 2004
# Purpose: To buffer the eighth feature class in the list.


distanceValues = 100, 200, 300, 400, 500, 600, 700, 800, \ 900, 1000, 1100, 1200, 1300, 1400, 1500

arcpy.Buffer_analysis (fcList[7], "BuffStreams1000", distanceValues[9])
Built-in functions

- **Python has many built-in functions**
  - `len()` – Returns the length
    ```python
define fc = "Railroads.shp"
define len(fc) → 13
    define fields = ["OID", "Shape", "Name"]
define len(fields) → 3
  - `max()` – Returns the maximum value
    ```python
define xExtent = (6260474.996464, 6338807.996464)
define max(xExtent)
  - `open()` – Opens a file
    ```python
define coord = open( "C:/XY.txt", "r").read()
  - `round()` – Rounds a number
    ```python
define yCoord = 1811884.623964
define round(yCoord)
Accessing Python modules

Most functions need to be imported from modules

The math module

```python
import math

math.sqrt(64)  # Prefix the name of the function with the name of the module
math.pi
```

The string module

```python
import string

string.split("-155.3 -43.5")
string.upper("C:/student")
```

The os.path module

```python
import os.path

os.path.basename("C:/student/Streets.shp")  # returns "Streets.shp"

os.path.dirname("C:/student/Streets.shp")  # returns "C:/Student"
```
Statements

- Python has many statements. Statements do not return values.

  - `print` - Sends output to the Interactive Window
    
    ```
    print "Hello"
    ```

  - `import` - Imports a module
    
    ```
    import math
    import string
    ```

- Other statements (discussed throughout the class)
  
  ```
  if...elif...else
  while
  for...in
  try...except
  ```
Decision making syntax

- Testing conditions
  ```python
  if x == 1:
      print "x is 1"
  elif x == 2:
      print "x is 2"
  else:
      print "x is not 1 or 2"
  ```

- Colons used at end of each condition

- Indentation defines what executes for each condition
  - Python automatically indents when you press Enter
  - Use tabs or spaces, must be consistent

- Two equal signs for conditions, one for assignment
  ```python
  x = 1  # assignment
  y = 8 + 2  # assignment
  if x == 6:  # testing a condition, not assigning a value
  ```
Looping syntax

- While loops, counted loops, and list loops

```python
x = 5
while x < 10:
    print x
    x = x + 1

for x in range(1,5):  #The last value is not executed
    print x
```

```python
x = [1, 2, 3]
for a in x:
    print a
```

- Colons used at end of each statement
- Indentation defines what executes for the loop
Case sensitive rules

- **Case sensitive**
  - **Functions and statements**
    - `max` `len` `open` `print` `import` `if` **Correct**
    - `Max` `LEN` `OpEn` `Print` `importT` `IF` **Incorrect**
  - **Variable names**
    - `Scale `≠ `scale` **Not equal**

- **Not case sensitive**
  - **Path names**
    - "C:/STUDENT" = "c:/StUdEnT" **Equal**
  - **Geoprocessing properties and methods**
    - `arcpy.BUFFER` = `arcpy.buffer` **Equal**
Introduction to Python and ArcGIS for Geoprocessing Scripts

Lecture 10
Python: An open-source programming language

Python Programming Language – Official Website

Python is a programming language that lets you work more quickly and integrate your systems more effectively. You can learn to use Python and see almost immediate gains in productivity and lower maintenance costs.

Python runs on Windows, Linux/Unix, Mac OS X, and has been ported to the Java and .NET virtual machines.

Python is free to use, even for commercial products, because of its OSI-approved open source license.

New to Python or choosing between Python 2 and Python 3? Read Python 2 or Python 3.

The Python Software Foundation holds the intellectual property rights behind Python, underwrites the PyCon conference, and funds other projects in the Python community.

Read more, or download Python now

- Second release candidates for Python 2.6.8, 2.7.3, 3.1.5, and 3.2.3 released
  Another iteration of release candidates for Python 2.6.8, 2.7.3, 3.1.5, and 3.2.3 have been released for testing. They include several security fixes.
  Published: Sun, 18 March 2012, 22:00 -0600

- CFP: PyCon Taiwan 2012
  PyCon Taiwan 2012 deadline for talk proposals is April 2. Conference will be 6-9-10 in Taipei.
  Published: Fri, 16 March 2012, 09:00 -0700

- IronPython 2.7.2 released
  IronPython 2.7.2 has been released.
  Published: Mon, 12 March 2012, 08:00 -0700

- CFP: EuroSciPy 2012
  Published: Sat, 10 March 2012, 17:00 -0800

- First alpha for Python 3.3.0 released
  The first alpha release for Python 3.3.0 have been released for testing.
Many places for help, not just Python.org
Support comes in many languages

---

Languages

» Languages

Attempt to have languages and links listed in the native tongue of the user.

Ideally, all the pages should be like the Polish or Turkish pages - all native language, only the necessary English.

There are some groundrules, some laid down by the site admins, some my suggestions:

1) Pages must be named in ASCII and English (PolishLanguage)

2) Pages must have an explanation in English at the top (Links to Python information in <language X>)

3) (my suggestion) We probably want to limit invites to edit the pages to people we know well, or Pythonistas with a track record. Hopefully this is inclusive enough without opening the site up to a spam flood and vandalism fest.

Where these pages really need help:

1) check links, remove broken ones.

2) add new links that are quality Python information and active.

3) some care for languages that have next to nothing, but do have people in the Python community - even a link to the Wikipedia page for Python, in that language, is a start (Some are pretty complete and of high quality - the Russian language Wikipedia page for Python, for instance, packs a lot in).

Languages

» Afrikaans

» Albanian

» Amharic

» Arabic

» Armenian

» Assamese

» Azerbaijani

» Belarusian

» Bengali

» Bulgarian

» Catalan

» Chinese

» Corsican

» Czech

» Danish

» Dutch

» English

» Eswatini

» Esperanto

» Estonian

» Filipino

» Finnish

» French

» Friulian

» Galician

» German

» Greek

» Gujarati

» Hausa

» Hebrew

» Hindi

» Hungarian

» Icelandic

» Indonesian

» Irish

» Italian

» Japanese

» Javanese

» Kannada

» Kazakh

» Korean

» Kyrgyz

» Latvian

» Lithuanian

» Malay

» Malayalam

» Maltese

» Maori

» Marathi

» Moldovan

» Mongolian

» Norwegian

» NorwegianBokmal

» NorwegianNynorsk

» Pashto

» Persian

» Polish

» Portuguese

» Punjabi

» Quechua

» Romani

» Rombo

» Russian

» Samoan

» Scots

» Scottish

» Serbian

» SimpleEnglish

» Slovenian

» Somali

» Sundanese

» Swedish

» Sundan

» Tagalog

» Thai

» Tibetan

» Turkish

» Ukrainian

» Urdu

» Vietnamese

» Welsh

» Wolof

» Xhosa

» Yoruba

» Zulu
Search, and you will find many guides
Python as a first language

Python for Non-Programmers

If you've never programmed before, the tutorials on this page are recommended for you; they don't assume that you have previous experience.

If you have previous programming experience, the list of programmer-oriented tutorials on the BeginnersGuide/Programmers page may get you started more quickly, but the tutorials on this page may still be helpful.

- How to Think Like a Computer Scientist - 2nd edition

Downey's open source textbook has a Python version, written by Jeff Elkner. It's also available in book form. It was updated and current version is 2nd edition.

- The Programming Historian

From the "About This Book" page: "This book is a tutorial-style introduction to programming for practicing historians. We assume that you're starting out with no prior programming experience and only a basic understanding of computers. More experience, of course, won't hurt. Once you know how to program, you will find it relatively easy to learn new programming languages and techniques, and to apply what you know in unfamiliar situations."

- Learning to Program

An introduction to programming for those who have never programmed before, by Alan Gauld. It introduces several programming languages but has a strong emphasis on Python.

- Learn Python The Hard Way

The title is a misnomer. It would be better titled "Learn Python By Coding It." The author determined that learning python should be similar to learning an instrument. You don't get a book on scales, but you're taught a scale and practice it. The author teaches you how to code properly, how to think like a programmer, and develop quality problem solving skills through a set of 52 exercises that build on each other.

- A Byte of Python

by Swaroop C.H., is also an introductory text for people with no previous programming experience.

- Invent Your Own Computer Games with Python, 2nd Ed.

by Al Sweigart is a free e-book that teaches complete beginners how to program by making games.

- Making Games with Python & Pygame

by Al Sweigart is a free e-book that introduces the Pygame framework for novices and intermediate programmers to make graphical games.

- One Day of IDLE

A very gentle introduction to the IDLE development environment that comes with Python. This tutorial by Danny Yoo has been translated into nine different languages.

- Instant Hacking

A minimal crash course by Magnus Lie Hetland that's an excellent starting point.

- Free Python video lectures are also available as a course titled Intro to programming with Python and Tkinter. Unix users can view the video using mplayer once you have downloaded the files. Windows users will need to have a DMX player, available from http://www.dmx.com/dmx/windows/

- The Young Programmers Podcast

contains video lessons on Python, Pygame, Python, Scratch, Alice, Java and Scala.

- A Non-Programmer's Tutorial for Python 2.6 by Josh Cogliati.
Python for Programmers

The tutorials on this page are aimed at people who have previous experience with other programming languages (C, Perl, Lisp, Visual Basic, etc.). Also of potential interest are such related Beginner Guides as BeginnersGuide/Overview and BeginnersGuide/NonProgrammers, and the tips in MovingToPythonFromOtherLanguages.

- **Python Tutorial** This tutorial is part of Python's documentation set and is updated with each new release.
- **Basic to Advanced Tutorial** A good tutorial on Python especially for the beginners.
- **Python Essential Reference** (book) If you want a highly compressed K&R-style 'just the facts' overview, David Beazley's "Python Essential Reference" covers practically all of the language in about a hundred pages.
- **WikiBook Python Programming**
- **Wikiiversity Python** The Wiki(anything) information about Python.
- **Instant Python** A minimal crash course by Magnus Lie Hetland.
- **Google's Python Class** - Google's Python tutorial for "people with a little bit of programming experience".
- **Python 2 & 3 Quick-Guides** A fast and efficient guide, with many examples, for quickly learning many tricks of Python.
- **Python Programming for Beginners** A short introduction to writing command-line applications in Python by Jacek Artymiak.
- **Python 101 - Beginning Python** and **Python 201 - (Slightly) Advanced Python** Two self-training courses from Dave Kuhiman. Python 101 introduces the basic data types, and 201 covers particular tasks such as parsing text and writing unit tests.
- **Python Short Course** A set of course slides by Richard P. Muller of Caltech that are aimed at scientific users. For example, the first example is a script to process output from a quantum chemistry simulation.
- **Learn Python in 10 minutes**
- **Python for Science** Course material written by Konrad Hinsen for an introduction to Python aimed at biologists.
- **Software Carpentry**: lectures aimed at scientists and engineers.
- **You can find a variety of Python tutorials on** CodeTeacher.
- **ComparingTypes** A quick look at some common programming types for python and other languages.
- **Python for Programmers** - for "Professional programmers who need to learn Python".
- **A basic Python Tutorial** that goes into some interesting features and examples.
- **Python Koans Learn Python through TDD**
- **Intro to Python** - A brief presentation about Python mainly aimed at experienced programmers. Might be use as a first pass overview the language.
As with any language: “Read the freaking manual”
ArcGIS 10 installs Python 2.6.5
Only use ArcGIS 10 help / tutorials / examples
Version 9.3 is significantly different
Only use Python 2.6.5 help / tutorials / examples
Upgrading might break link with ArcGIS
ESRI created Python routines to automate GIS
Pick any ArcGIS tool - help is not helpful
Tool Help is much better
This example calculates the standard deviation per cell on several input GRID rasters and outputs the result as a GRID raster.

```python
# Name: CellStatistics_Ex_02.py
# Description: Calculates a per-cell statistic from multiple rasters
# Requirements: Spatial Analyst Extension
# Author: ESRI

# Import system modules
import arcpy
from arcpy import env
from arcpy.sa import *

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

# Set local variables
inRaster01 = "degs"
inRaster02 = "negr"
inRaster03 = "cost"

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

# Execute CellStatistics
outCellStatistics = CellStatistics([inRaster01, inRaster02, inRaster03], "RANGE", "NODATA")

# Save the output
outCellStatistics.save("C:/sapyexamples/output/cellstats")
```
Python scripts are text files with “.py” extension
Create new document, copy / paste text

# Name: Cellstatistics_Ex_02.py
# Description: Calculates a per cell statistic from multiple rasters
# Requirements: Spatial Analyst Extension
# Author: ESRI

import arcpy
from arcpy import env
from arcpy.sa import *

env.workspace = "C:/sapyexamples/data"

inRaster01 = "degs"
inRaster02 = "nogs"
inRaster03 = "cost"

arcpy.CheckOutExtension("Spatial")

outCellStatistics = CellStatistics([inRaster01, inRaster02, inRaster03], "RANGE", "NODATA")

outCellStatistics.save("C:/sapyexamples/output/cellstats")
For existing scripts, right click and edit with IDLE
IDLE displays color-coded script (Don't need the Python Shell, close it)

 mortar_cube = 222.1
 # Description: Calculate a per-cell statistic from multiple rasters
 # Requirements: Spatial Analyst Extension
 # Author: ESRI

 # Import system modules
 import arcpy
 from arcpy import env
 from arcpy.sa import *

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

 # Set local variables
 inRaster01 = "degs"
inRaster02 = "negs"
inRaster03 = "cost"

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

 # Execute CellStatistics
 outCellStatistics = CellStatistics([inRaster01, inRaster02, inRaster03], "RANGE", "NODATA")

 # Save the output
 outCellStatistics.save("C:\sapyexamples\output\cellstats")
### Data Distribution Site Description

In addition to browsing the directory structure below, you may create a Customizable File Search.

<table>
<thead>
<tr>
<th>Filename</th>
<th>Last Modified</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ancillary</td>
<td></td>
<td>-</td>
</tr>
<tr>
<td>CZCS</td>
<td></td>
<td>-</td>
</tr>
<tr>
<td>MERIS</td>
<td></td>
<td>-</td>
</tr>
<tr>
<td>MODISA</td>
<td></td>
<td>-</td>
</tr>
<tr>
<td>MODIST</td>
<td></td>
<td>-</td>
</tr>
<tr>
<td>OCTS</td>
<td></td>
<td>-</td>
</tr>
<tr>
<td>SeaWiFS</td>
<td></td>
<td>-</td>
</tr>
<tr>
<td>VIIRS</td>
<td></td>
<td>-</td>
</tr>
<tr>
<td>Aquarius</td>
<td></td>
<td>-</td>
</tr>
</tbody>
</table>

**Curator:** OceanColor Webmaster

**Authorized by:** gene can fieldman

**Updated:** 10 June 2011
Example
Downloaded, processed, and symbolized layers
Gulf Of Mexico outline, plus monthly Chlorophyll concentration from Terra, SeaWiFS, and Aqua
Edit script to match data location and names

```python
# Name: Mean.py
# Description: Calculates mean chlorophyll concentration from 3 input rasters
# Requirements: Spatial Analyst Extension
# Author: Your Name

# Import system modules
import arcpy
from arcpy import env
from arcpy.sa import *

# Set environment settings
env.workspace = "C:/Users/Kagome/Desktop/DataFolder"

# Set local variables
inRaster01 = "Jan_2003_Clorophyll_Terra.tif"
inRaster02 = "Jan_2003_Clorophyll_SeaWiFS.tif"
inRaster03 = "Jan_2003_Clorophyll_Aqua.tif"
outRaster = "Jan_2003_Clorophyll_Mean.tif"

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

# Execute CellStatistics
outCellStatistics = CellStatistics([inRaster01, inRaster02, inRaster03], "MEAN", "NODATA")

# Save the output
outCellStatistics.save(outRaster)
```
Setup

Run Python from Windows Command Prompt
Setup
Easier to create a customized shortcut
Setup

Start Command Prompt in same folder as script
Desktop and Command Prompt ready
Run... no feedback (maybe you like that)
Script created “Mean” datafiles
Open in ArcMap... unsymbolized
ArcPython does not like to overwrite files
Add status messages, create symbolize layer

```python
# Name: Mean.py
# Description: Calculates mean chlorophyll concentration from 3 input rasters
# Requirements: Spatial Analyst Extension
# Author: Name

# Import system modules
import arcpy
from arcpy import env
from arcpy.sa import *

print "Initializing script"

# Set environment settings
env.workspace = "C:/Users/Kagome/Desktop/DataFolder"

# Set local variables
inRaster01 = "Jan_2003_Chlorophyll_Terra.tif"
inRaster02 = "Jan_2003_Chlorophyll_SeaWiFS.tif"
inRaster03 = "Jan_2003_Chlorophyll_Aqua.tif"
outRaster = "Jan_2003_Chlorophyll_Mean.tif"

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

print "Calculating mean"

# Execute CellStatistics
outCellStatistics = cellStatistics([inRaster01, inRaster02, inRaster03], "MEAN", "NODATA")

print "Saving results"

# Save the output
outCellStatistics.save(outRaster)

# Make output raster into an ArcMap layer, apply color scheme from first input raster
inRasterLayer = "Jan_2003_Chlorophyll_Terra.lyr"
outLayerName = "Jan_2003_Chlorophyll_Mean"
outLayerFile = "Jan_2003_Chlorophyll_Mean.lyr"
arcpy.MakeRasterLayer_management(outRaster, outLayerName)
arcpy.ApplySymbologyFromLayer_management(outLayerName, inRasterLayer)
arcpy.SaveToLayerFile_management(outLayerName, outLayerFile, "RELATIVE")

print "Script complete"
```
Run again... feedback shows

Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. All rights reserved.

C:\Users\Kagome\Desktop>Mean.py
Initializing script
Calculating mean
Saving results
Script complete

C:\Users\Kagome\Desktop>
Script created “Mean” datafiles and layer
Open in ArcMap... symbolized
Python and ArcGIS are powerful tools

Do not be fooled by simple problems described and solved by scripts in beginner textbooks

Try manually processing multiple GB of data with dozens of processing steps, EVERY DAILY