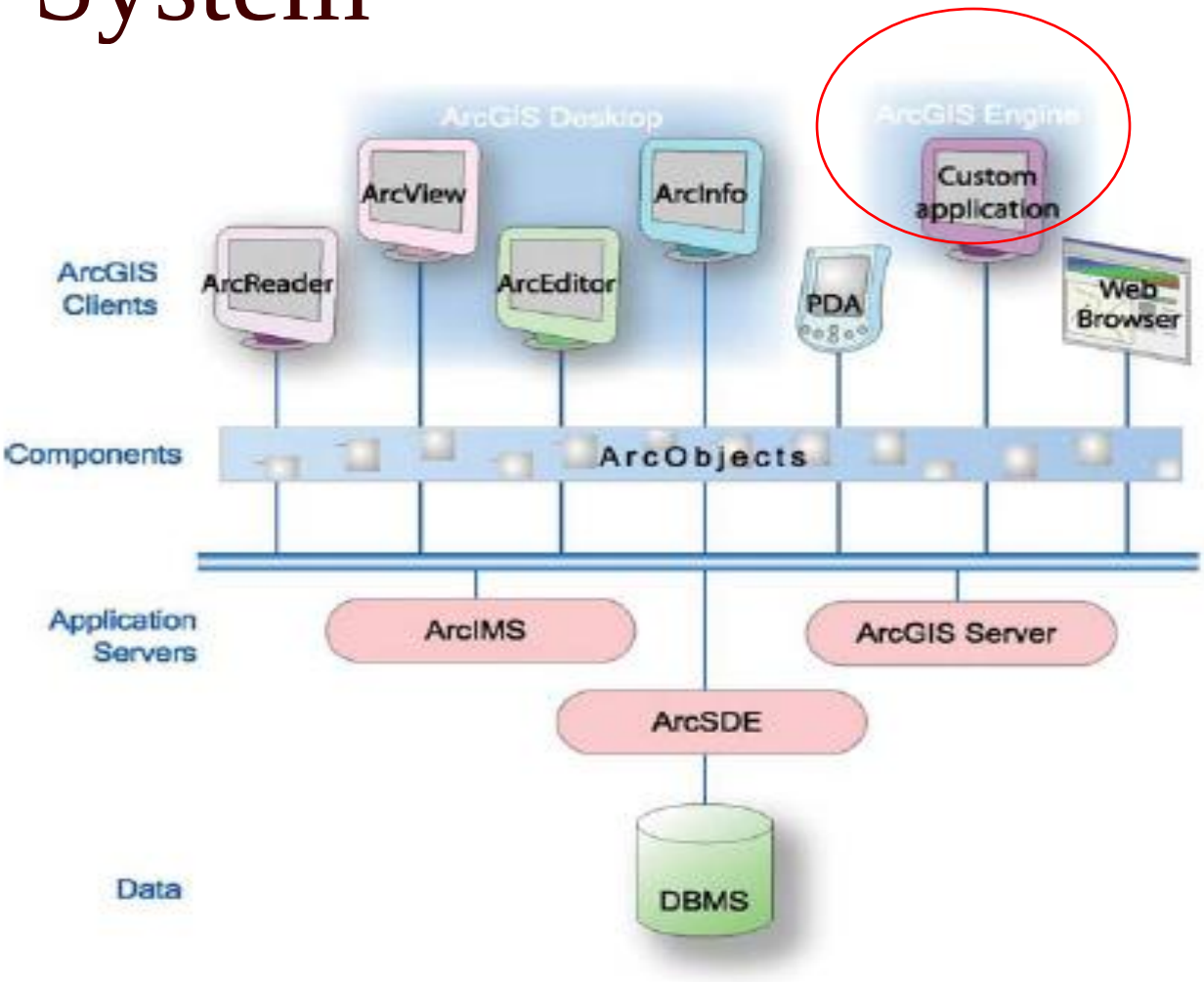
Writing Geoprocessing Scripts With ArcGIS

Lecture 10

GIS System



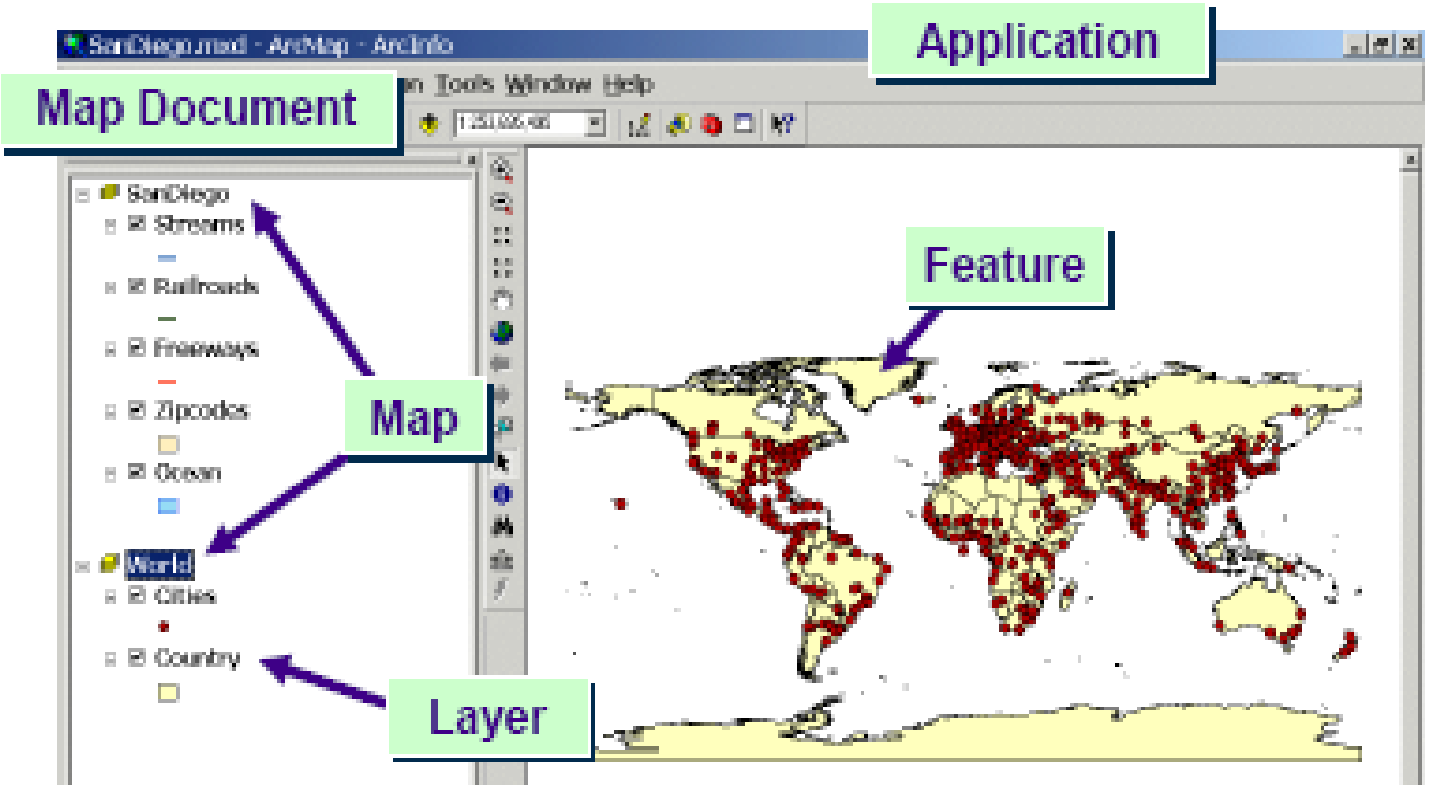
ArcSDE® Gateway is an interface for managing geodatabases in numerous relational database management systems (RDBMSs).

ArcObjects

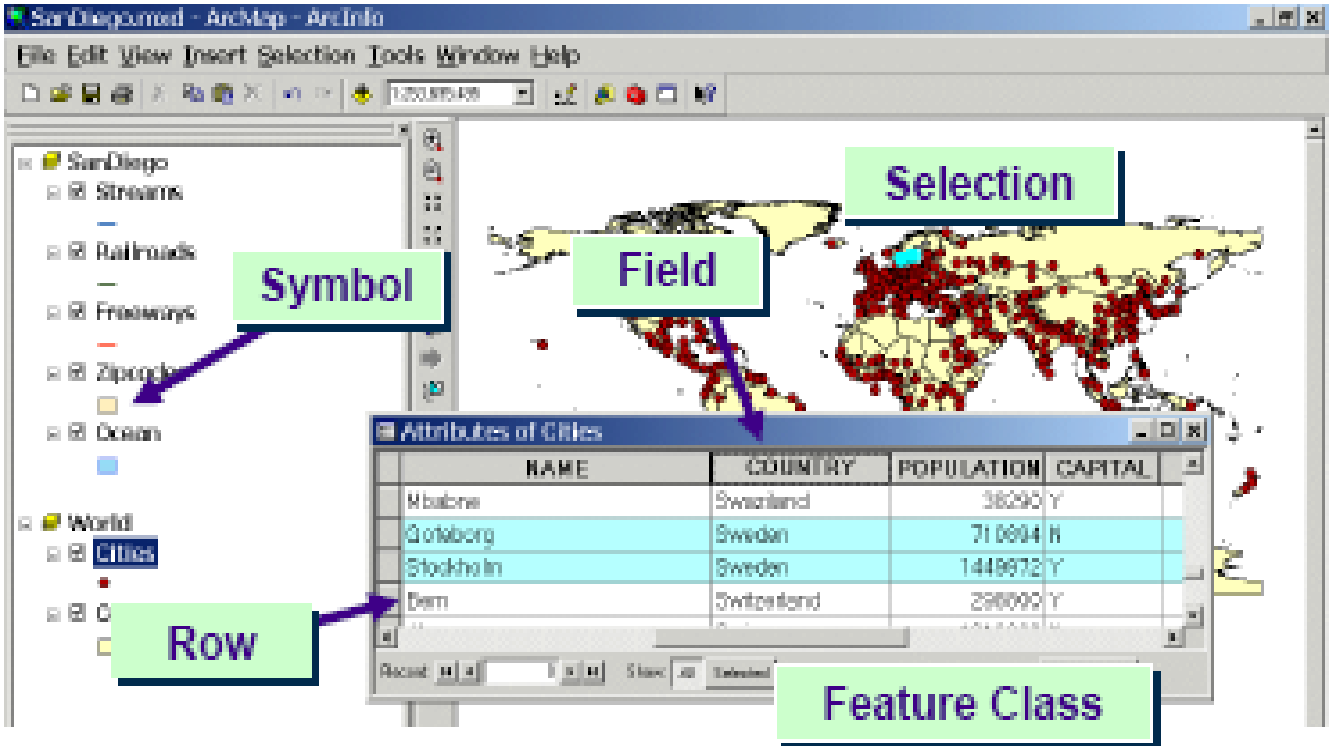
- ArcObjects are the building blocks of ArcGIS. With ArcObjects, you can create your own menus, tools, workflows, applications, and custom feature classes for use with ArcGIS.
- ESRI ArcObjects is the development platform for the ArcGIS family of applications, such as ArcMap, ArcCatalog, ArcScene, ArcGIS 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

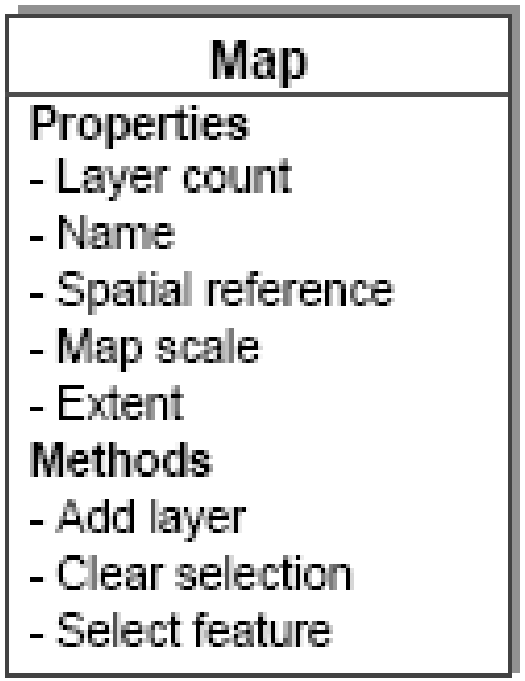


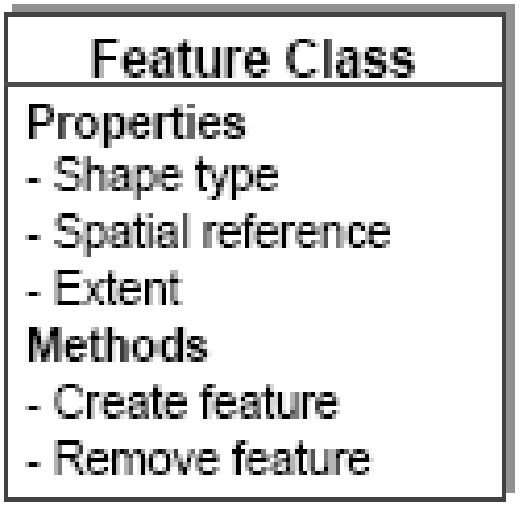
Common ArcObjects



Interacting with ArcObjects

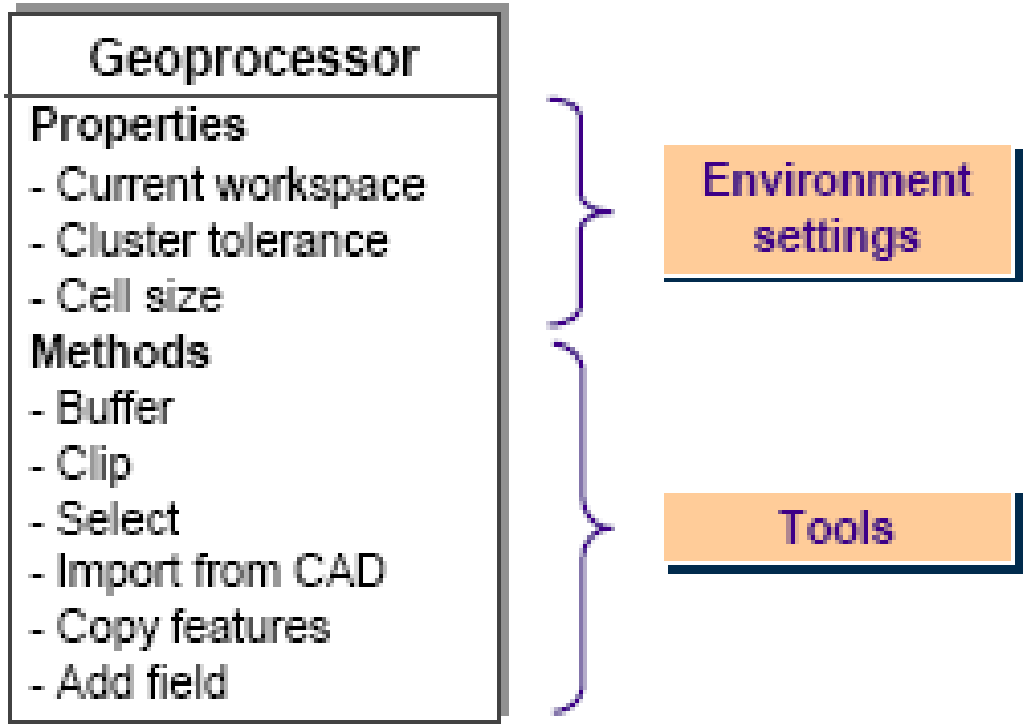
- Each ArcObject has properties and methods
  - Property: Characteristic of an object (a noun)
  - Method: Something the object knows how to do (a verb)
- Interact with objects through properties and methods





The Geoprocessor (GpDispatch) ArcObject

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



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:

```
import arcpy
```

Tells Python to import basic ArcGIS geoprocessing functionality

*****

```
from arcpy import env
```

Tells Python to import ability to control the ArcGIS Environment

*****

```
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)

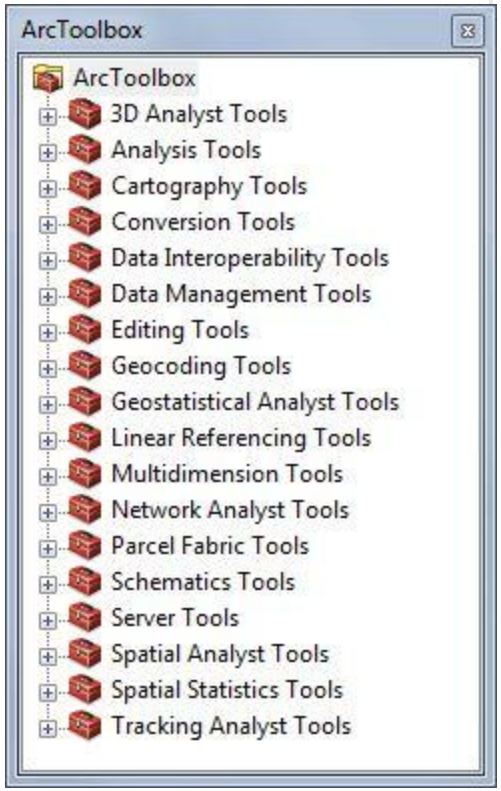
  - Parentheses around arguments
  - Arguments separated by commas

Toolbox aliases

- Always suffix the tool with the toolbox alias

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

```
scale = 10000
Scale = 20000
```

Two different variables

- 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**

```
num1 = 1.2
num2 = 3 + 5
```

- **Variables can hold lists**

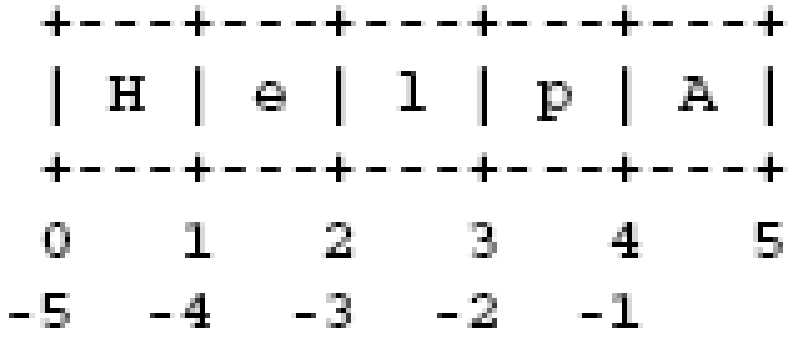
```
numList = [1, 2, 3]
fcList = ["Roads", "Streets", "Parcels", "Zipcodes"]
```

  - Lists are indexed, 0-based from the left and 1-based from right

```
fc1 = fcList[1]          ---> "Streets"

# Lists are zero-based, position one is "Streets".

fc2 = fcList[0:2]        ---> "Roads", "Streets"
# The last position is not included.
fc3 = fcList[0:-1]       ---> "Roads", "Streets", "Parcels"
# Take values from zero to minus one. Last position not included.
fc4 = fcList[2:]         ---> "Parcels", "Zipcodes"
# Take all values from position two to the end.
```

Variable index

Word =



- Word[0]=‘H’
- Word[2:4]=‘lp’
- 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 \
- Indentation is automatic

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

fcList = ["Freeways", "Climate", "Ocean", "MajorAttractions",
          "Railroads", "Mines", "Zipcodes", "Streams"]

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

```
fc = "Railroads.shp"
len(fc) ---> 13
fields = ["OID", "Shape", "Name"]
len(fields) ---> 3
```

  - `max()` – Returns the maximum value

```
xExtent = (6260474.996464, 6338807.996464)
max(xExtent)
```

  - `open()` – Opens a file

```
coord = open( "C:/XY.txt"    , "r").read()
```

  - `round()` – Rounds a number

```
yCoord = 1811884.623964
round(yCoord)
```

Accessing Python modules

- **Most functions need to be imported from modules**
  - The `math` module

```
import math

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

  - The `string` module

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

  - The `os.path` module

```
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

```
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

```
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

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

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

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

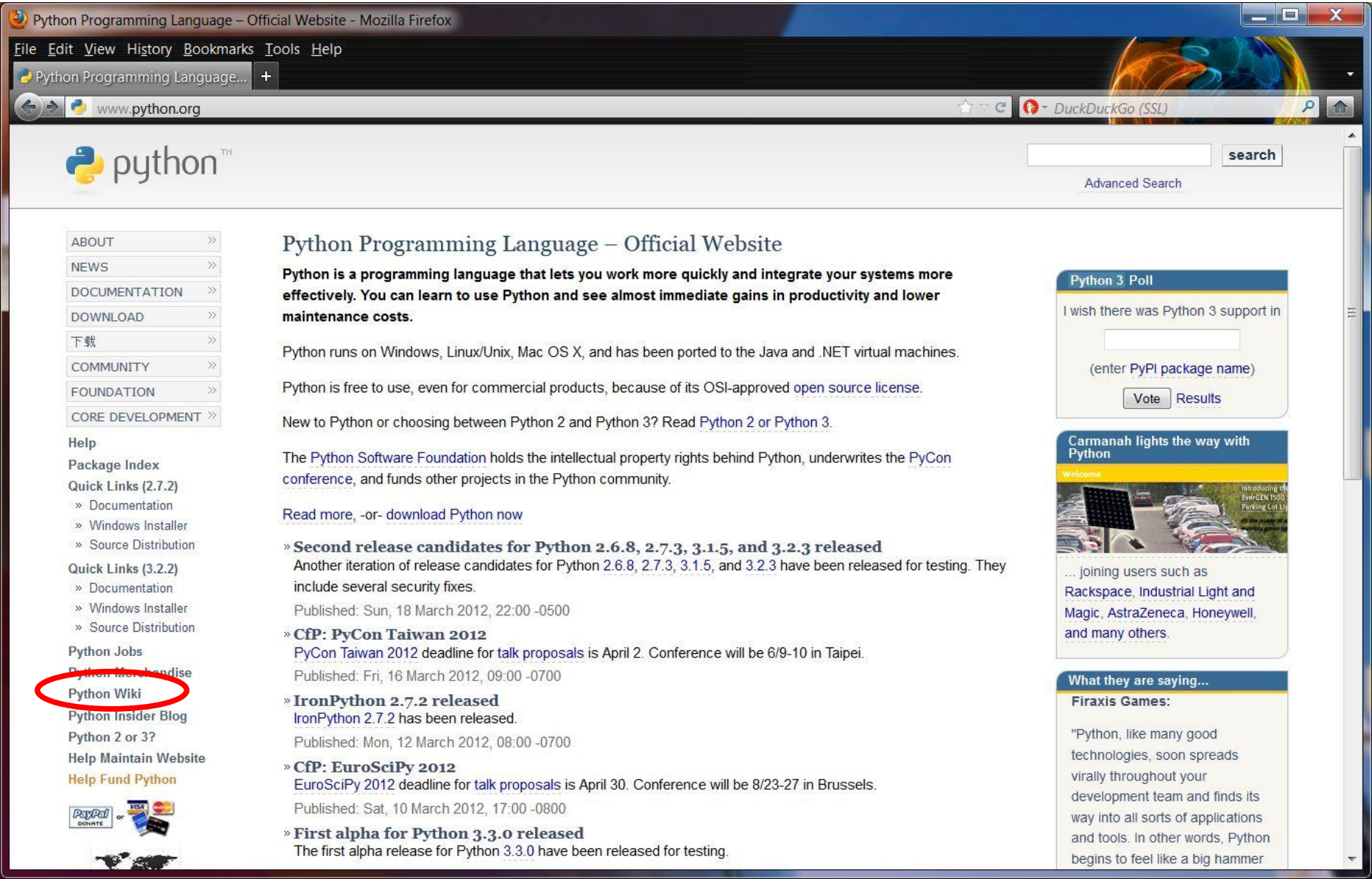
    "C:/STUDENT" = "c:/StUdEnT" — Equal
  - Geoprocessing properties and methods

    arcpy.BUFFER = arcpy.buffer — Equal

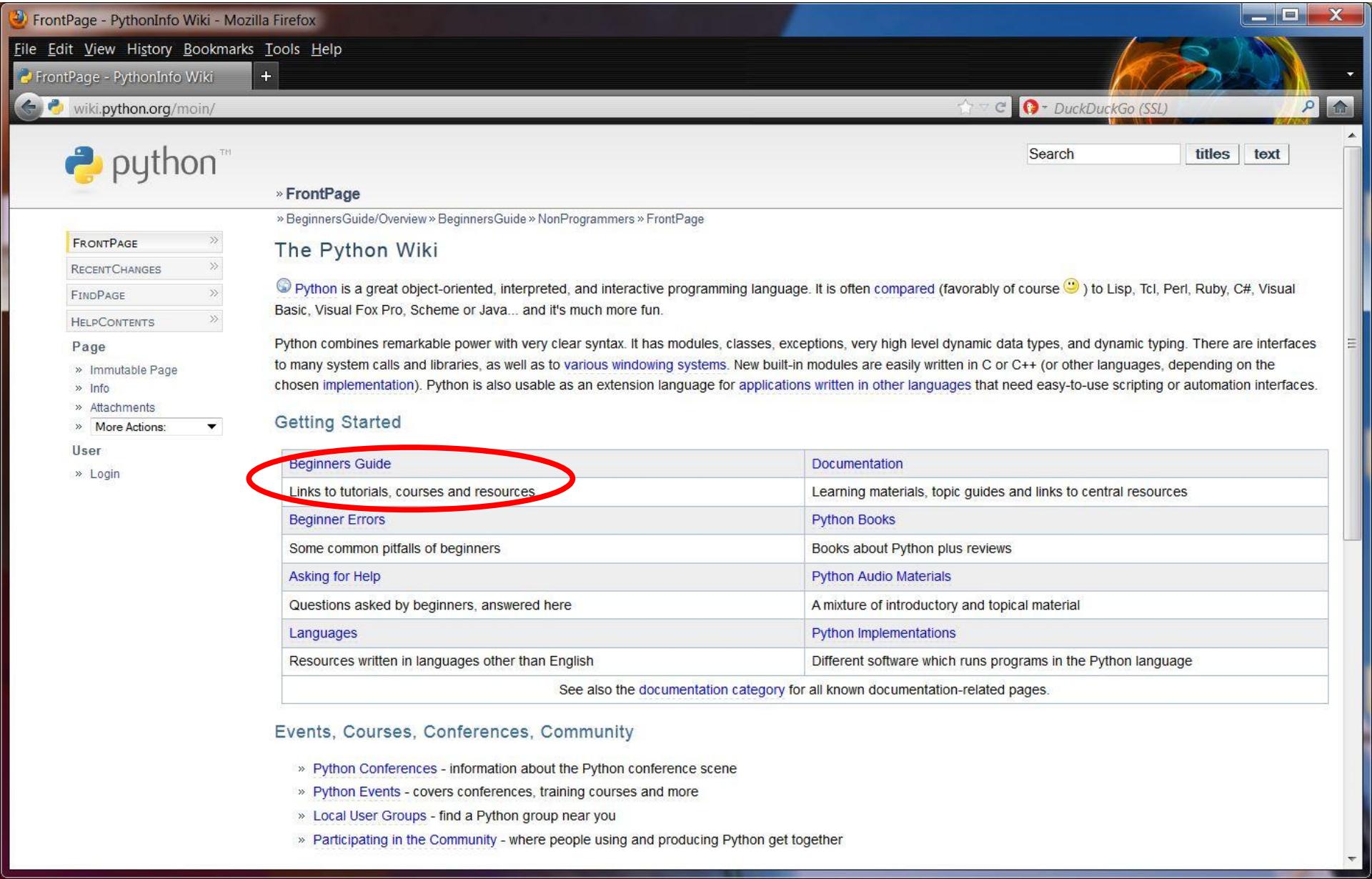
Introduction to Python and ArcGIS for Geoprocessing Scripts

Lecture 10

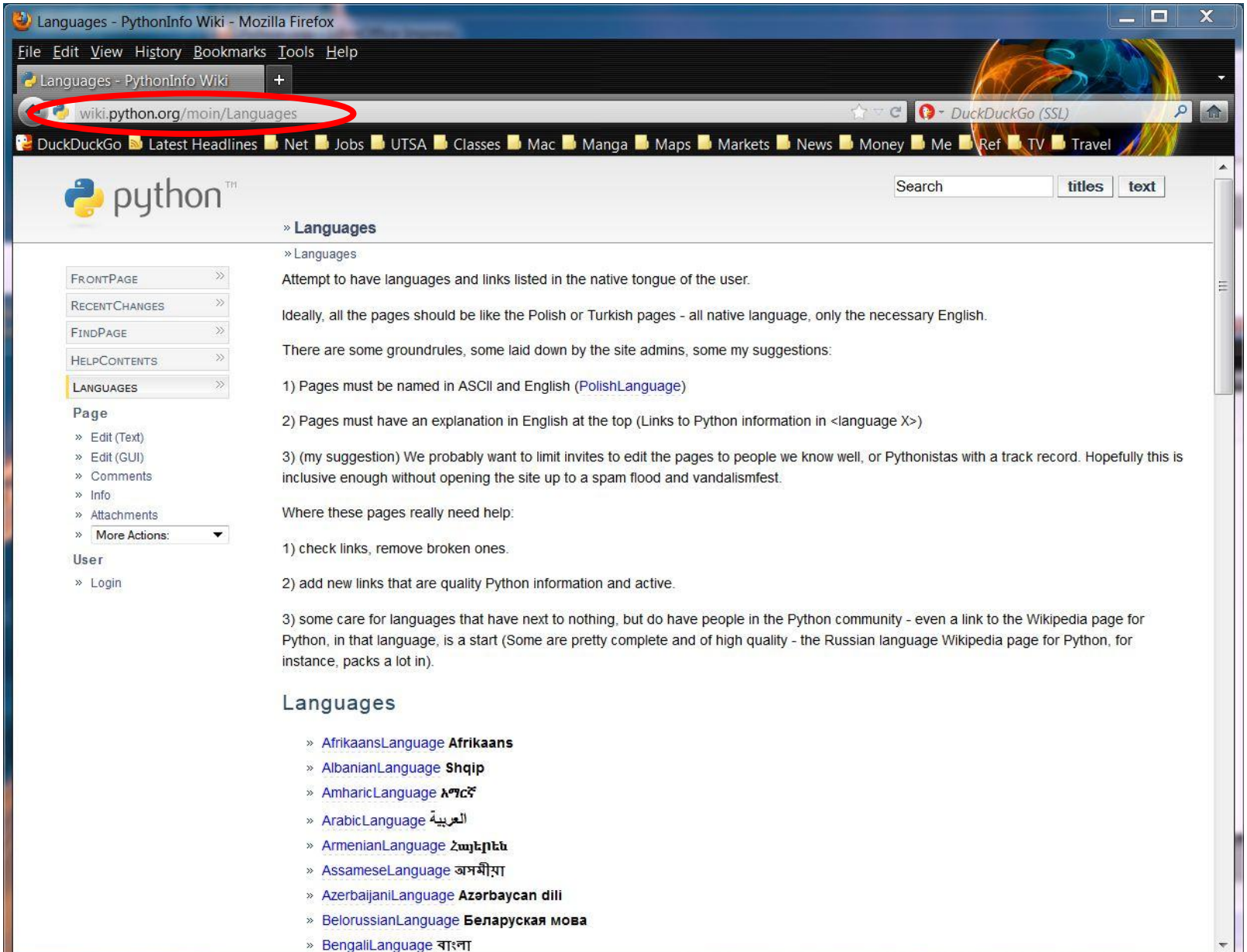
Python: An open-source programming language



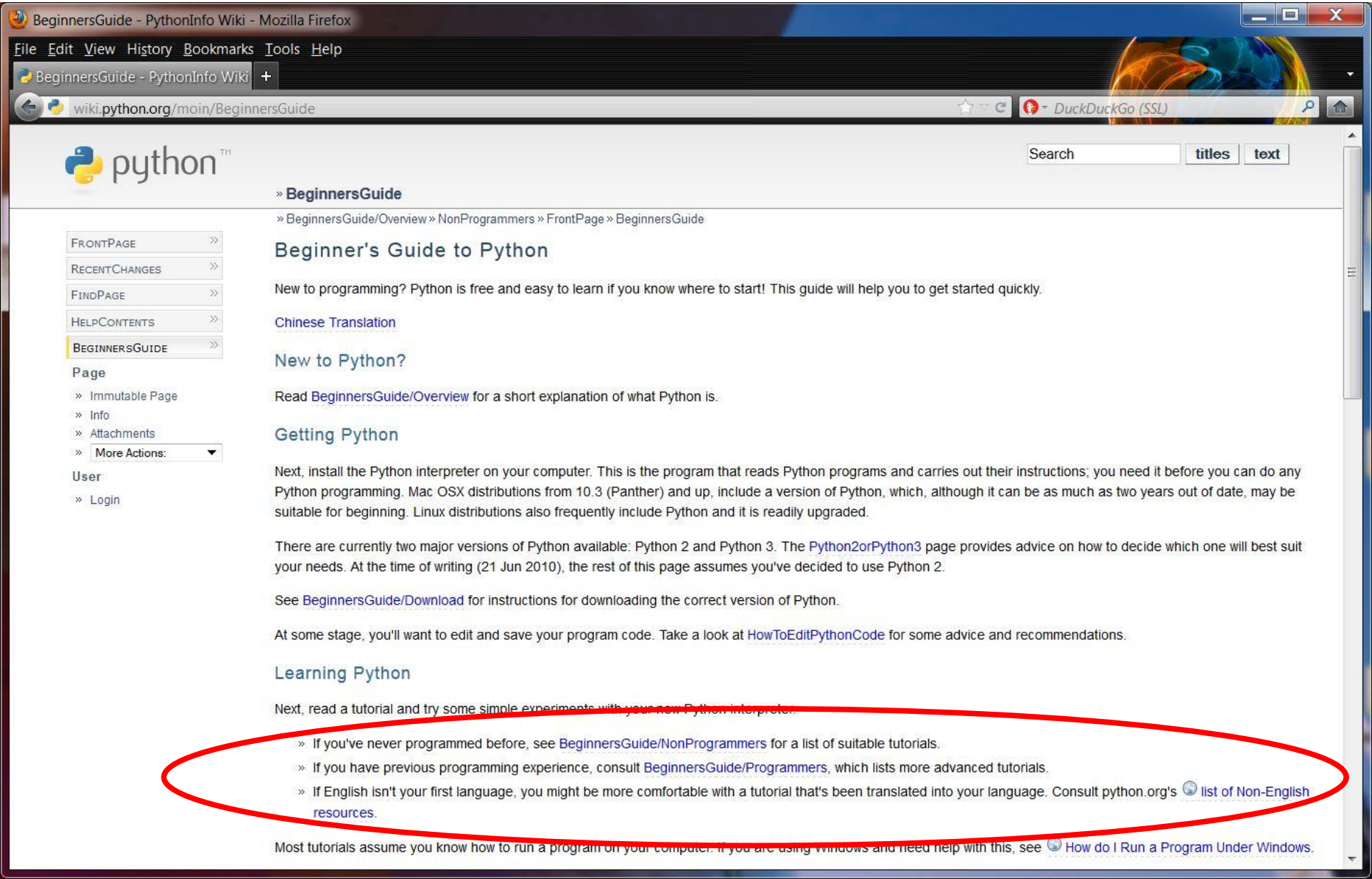
Many places for help, not just Python.org



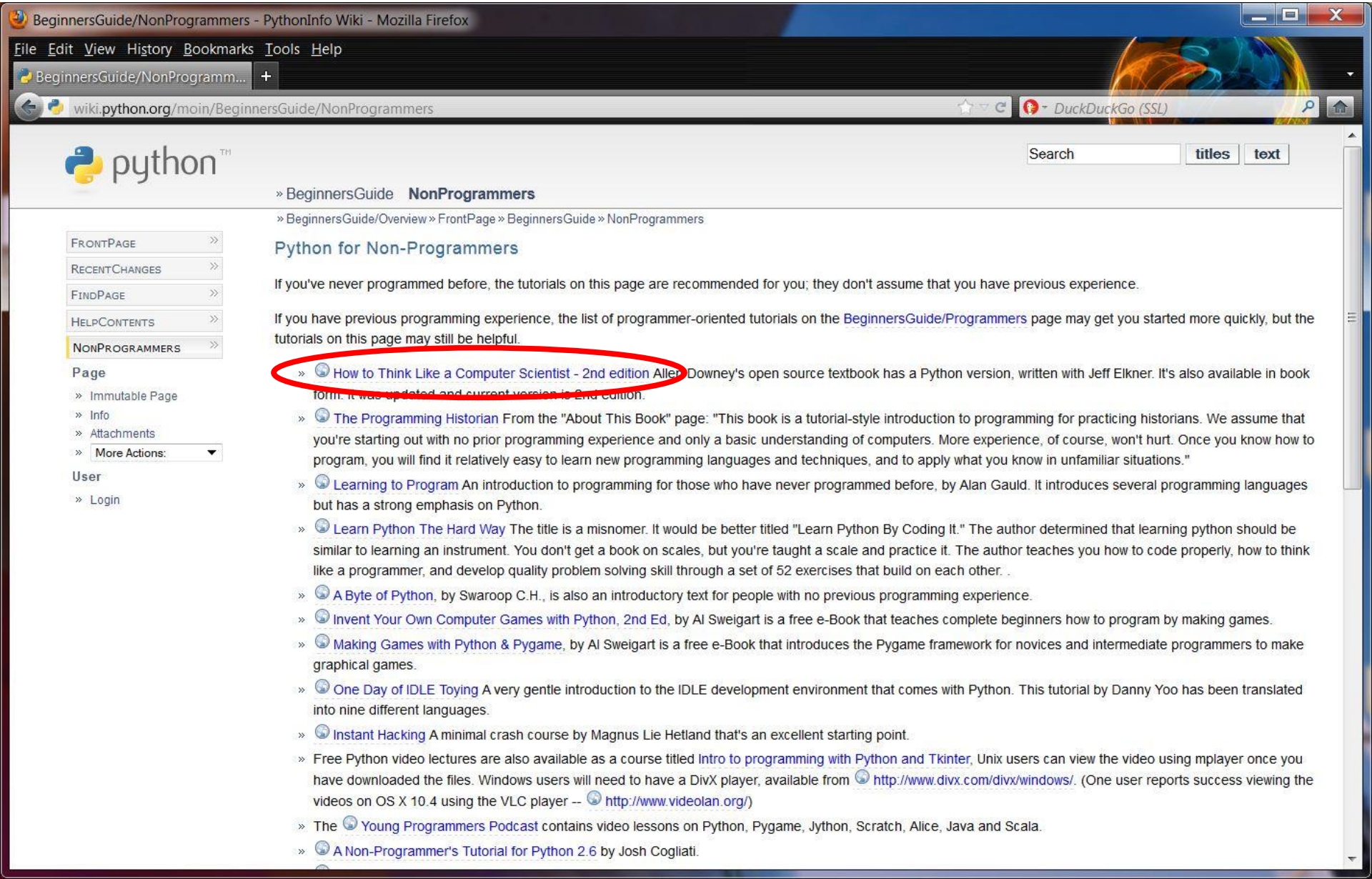
Support comes in many languages



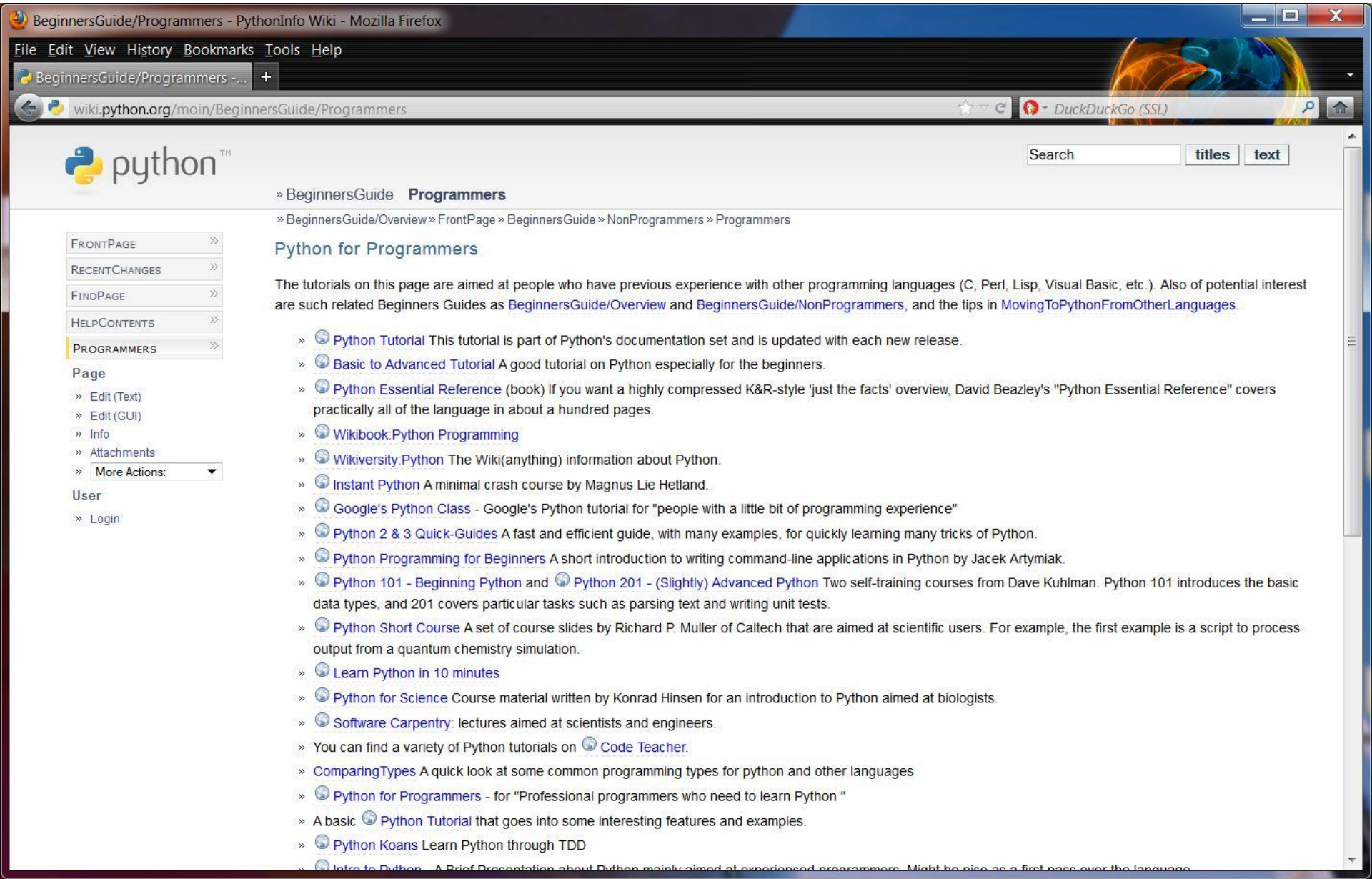
Search, and you will find many guides



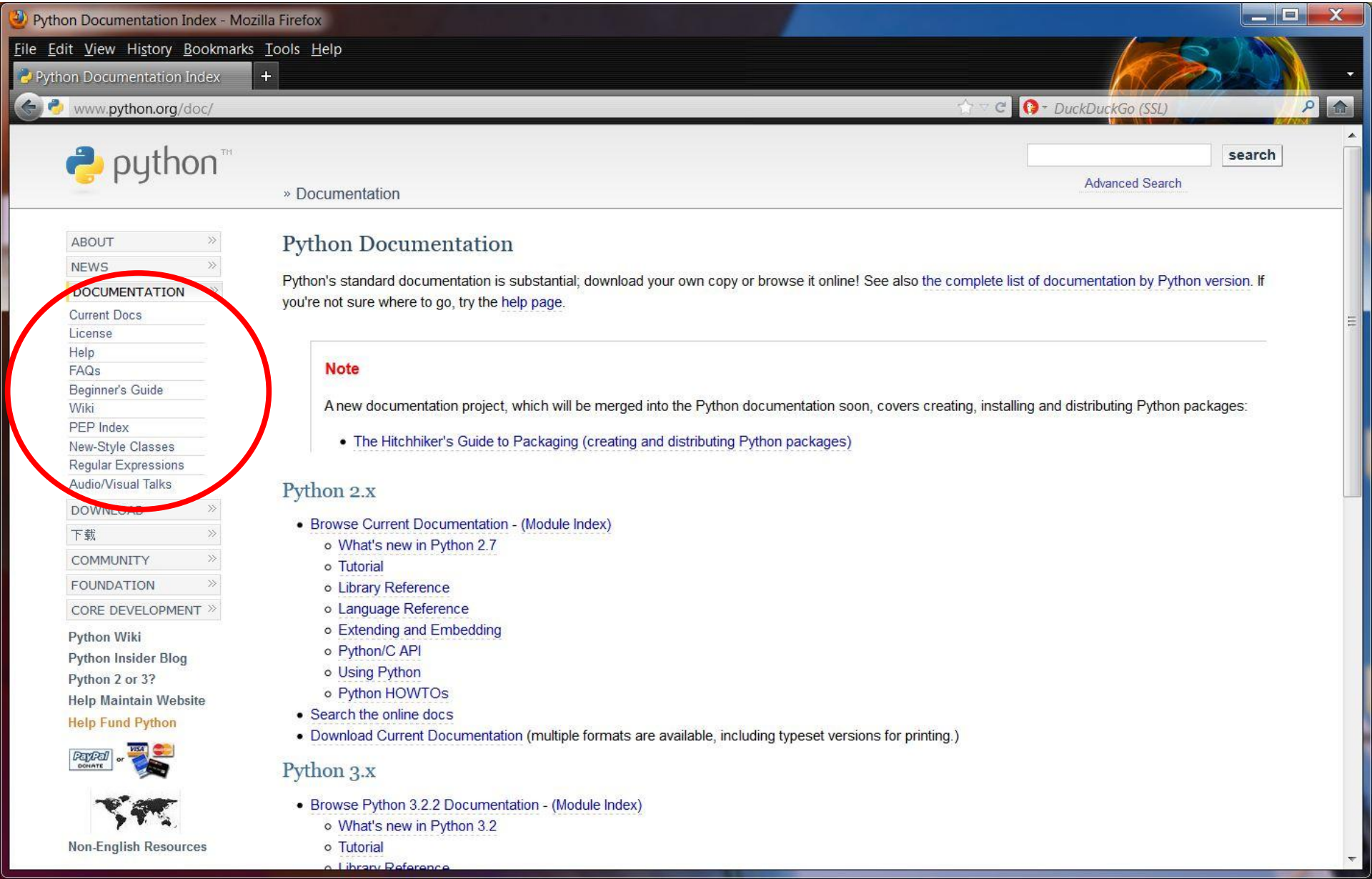
Python as a first language



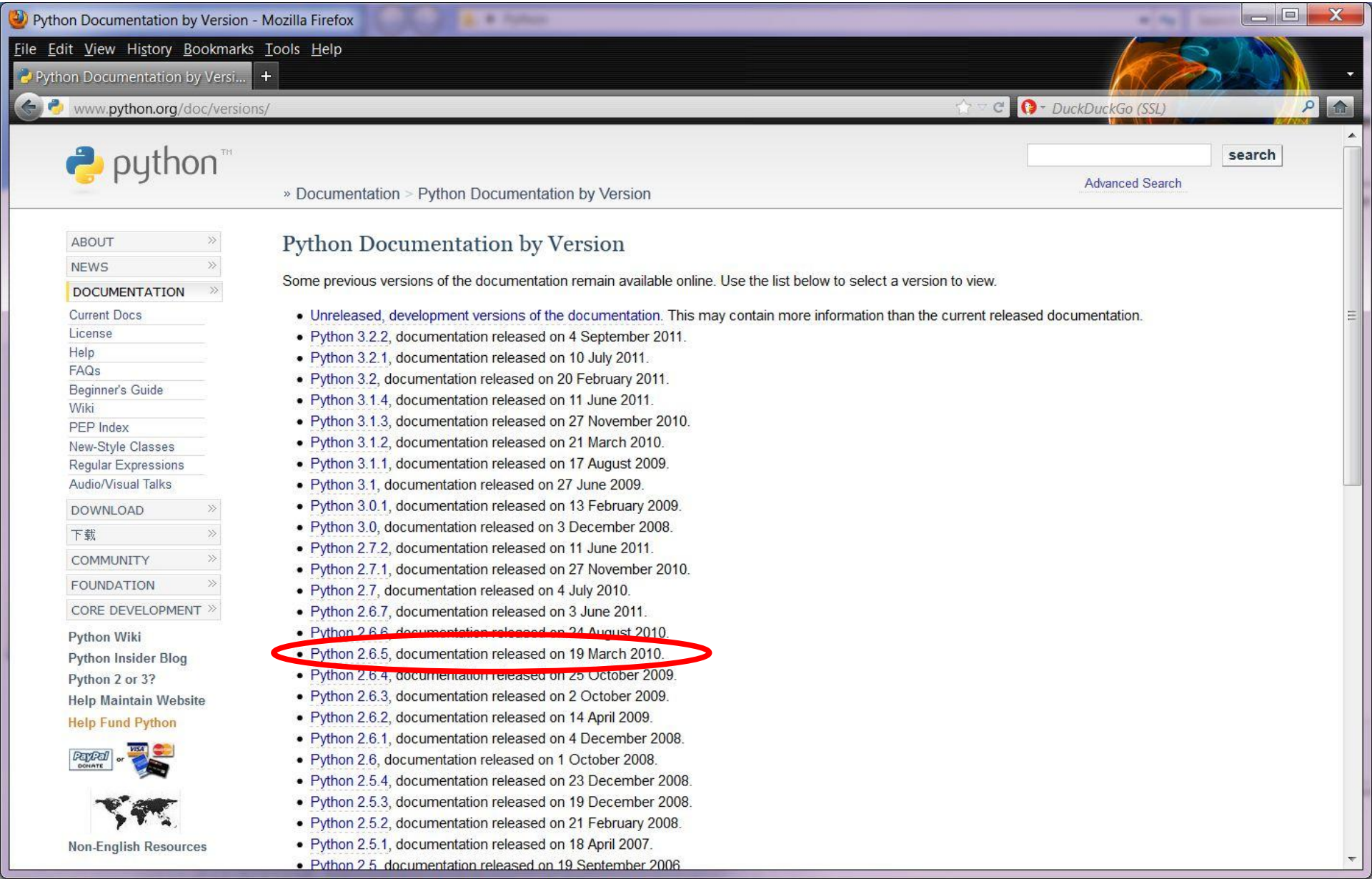
Python for Programmers



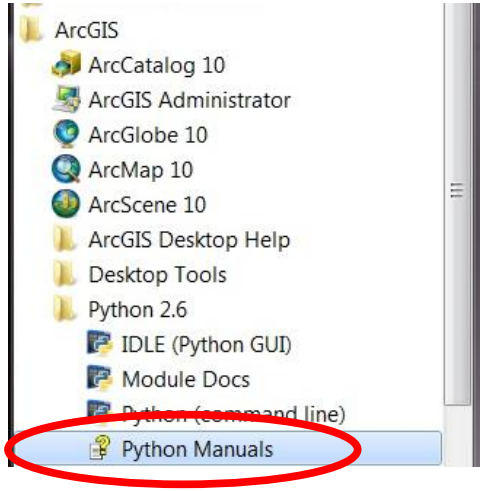
As with any language: “Read the freaking manual”



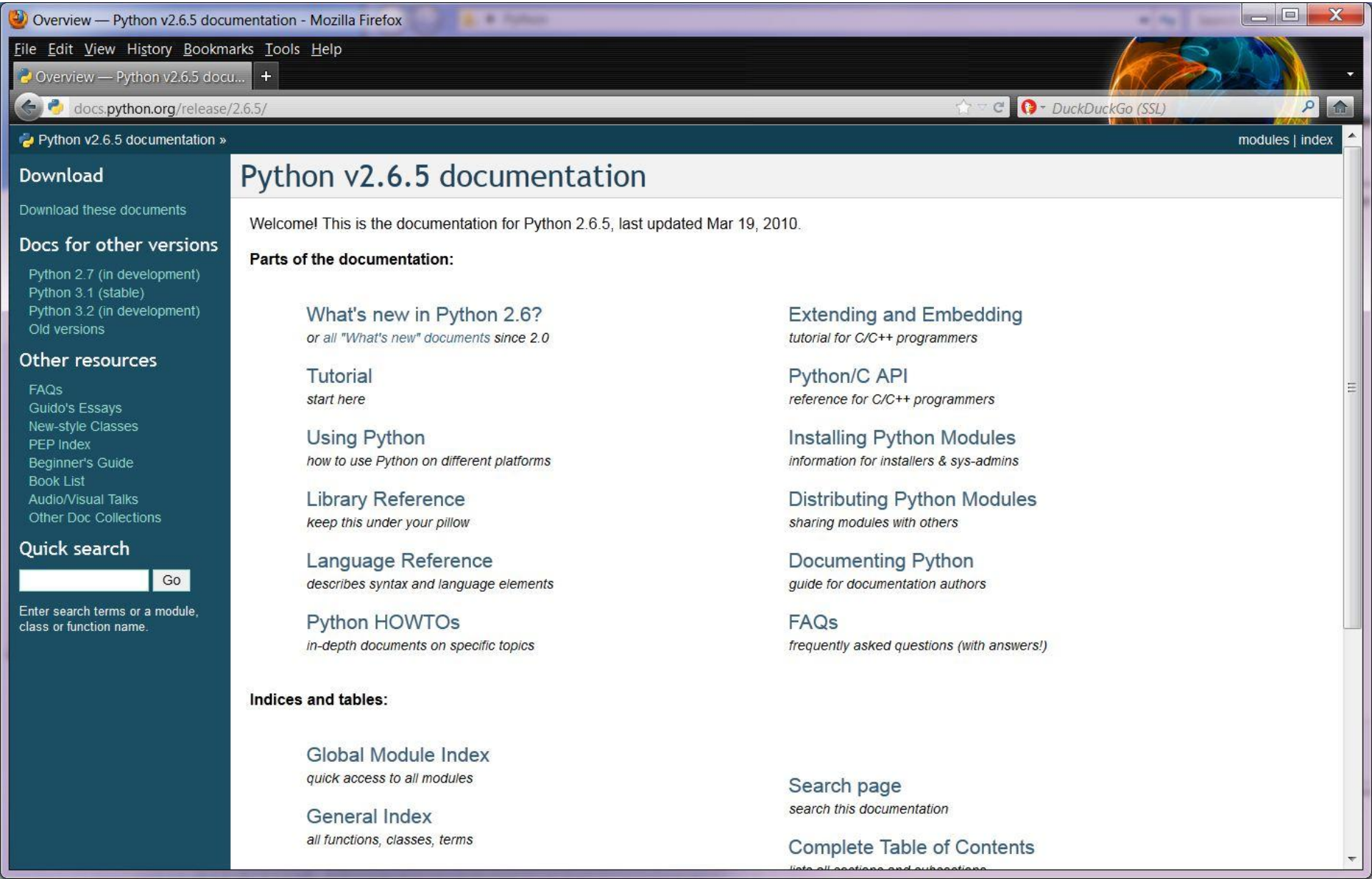
ArcGIS 10 installs Python 2.6.5



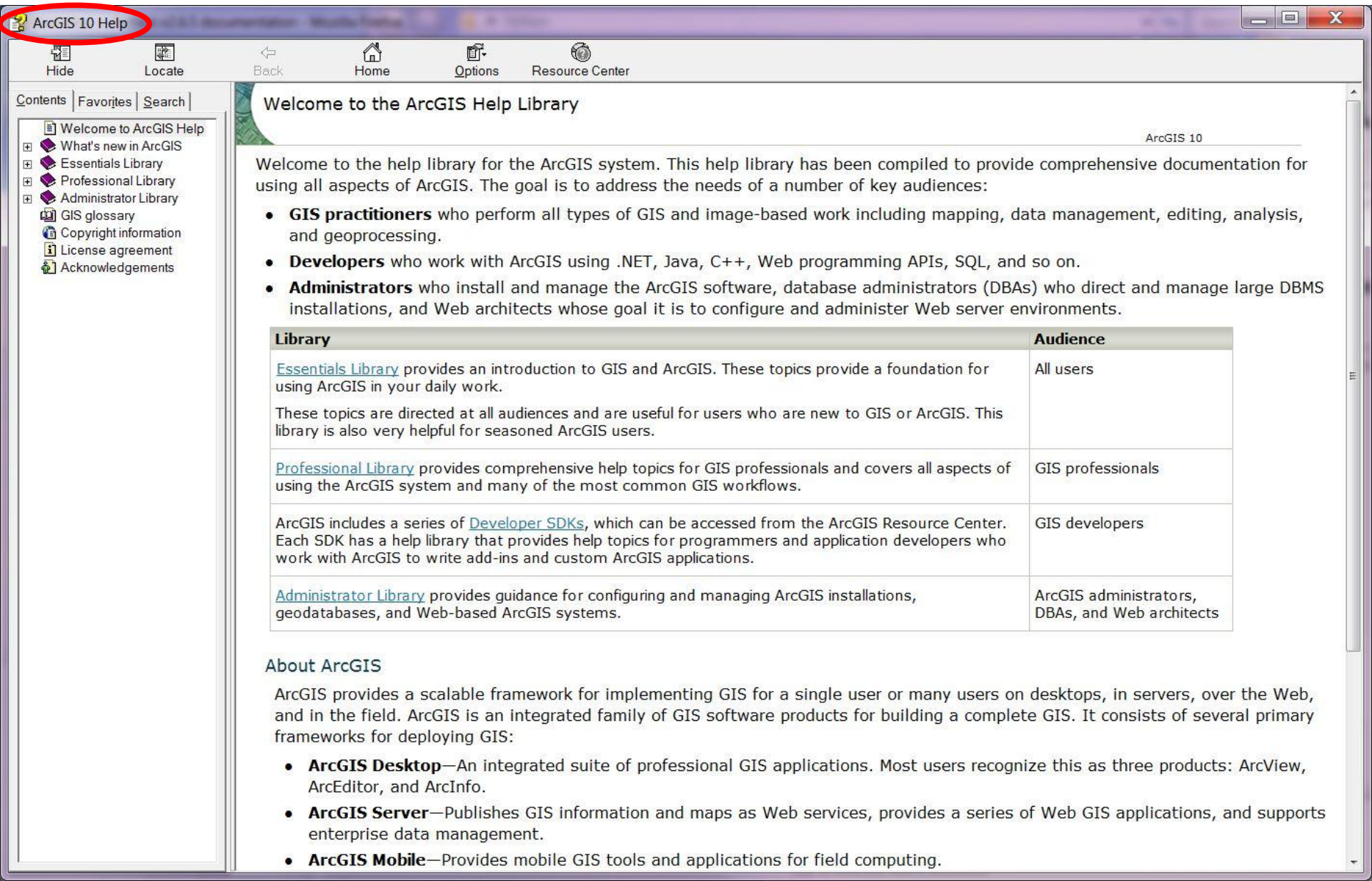
Only use ArcGIS 10 help / tutorials / examples
Version 9.3 is significantly different



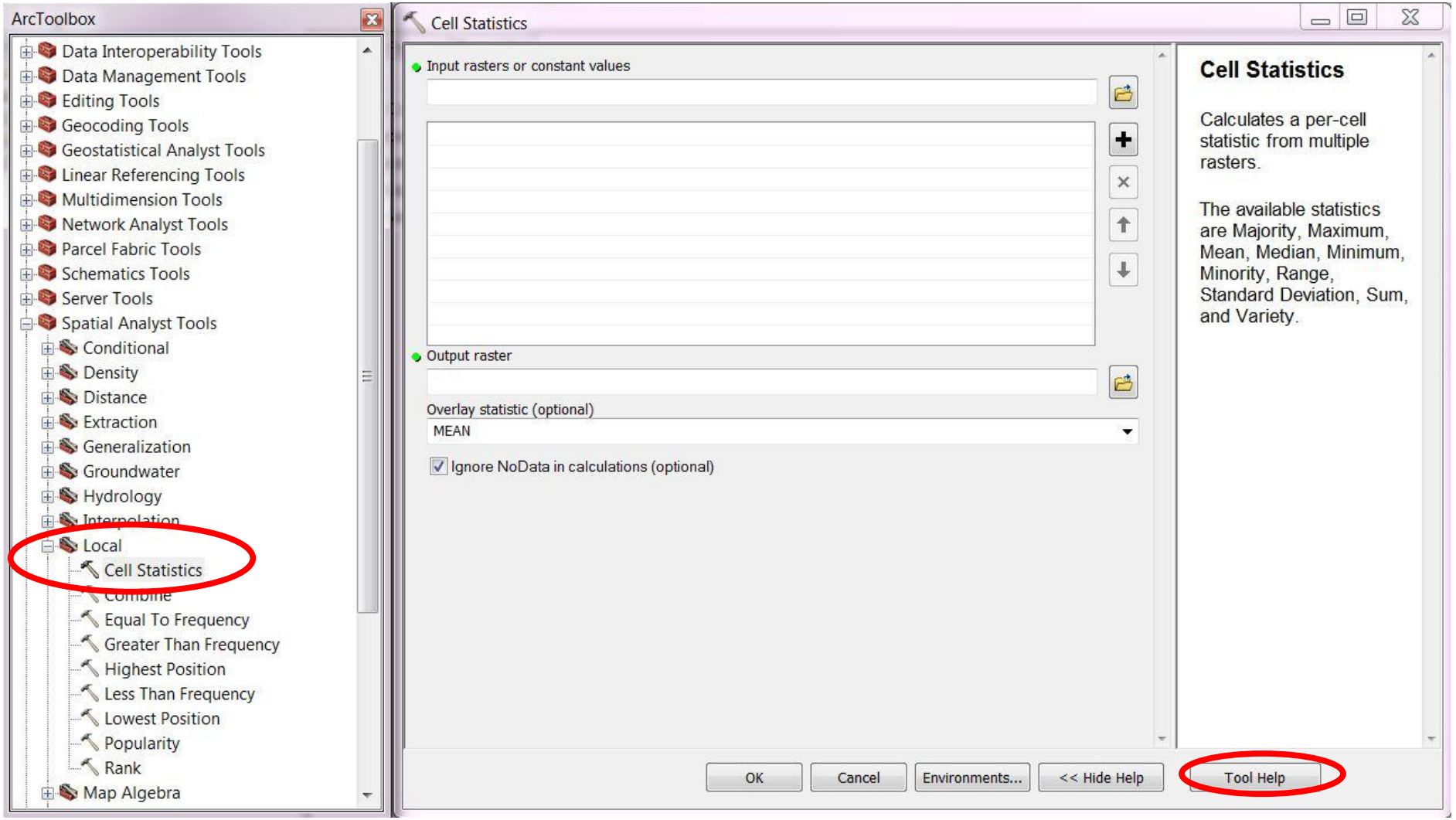
Only use Python 2.6.5 help / tutorials / examples
Upgrading might break link with ArcGIS



ESRI created Python routines to automate GIS



Pick any ArcGIS tool - help is not helpful



Tool Help is much better

Contents | Favorites | Search

- Welcome to ArcGIS Help
- What's new in ArcGIS
- Essentials Library
- Professional Library
- Administrator Library
- GIS glossary
- Copyright information
- License agreement
- Acknowledgements

Cell Statistics (Spatial Analyst)

ArcGIS 10

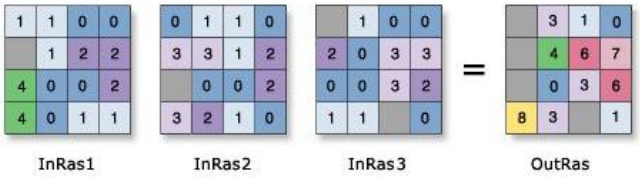
Summary

Calculates a per-cell statistic from multiple rasters.

The available statistics are Majority, Maximum, Mean, Median, Minimum, Minority, Range, Standard Deviation, Sum, and Variety.

Learn more about how Cell Statistics works

Illustration



OutRas = CellStatistics([InRas1, InRas2, InRas3], "SUM", "DATA")

Usage

- The order of the input rasters is irrelevant for this tool.
- For statistic types Maximum, Minimum, Mean, Median, Majority, Minority and Sum, if a single raster is used as the input, the output cell values will be the same as the input cell values. For Range and STD the output cell values will all be 0, and for Variety, all 1.

Syntax

CellStatistics (in_rasters_or_constants, {statistics_type}, {ignore_nodata})

Parameter	Explanation	Data Type
in_rasters_or_constants [in_raster_or_constant,...]	A list of input rasters for which a statistic will be calculated for each cell within the Analysis window. A number can be used as an input; however, the cell size and extent must first be set in the environment.	Raster Layer | Constant
statistics_type (Optional)	Statistic type to be calculated. • MEAN — Calculates the mean (average) of the inputs. • MAJORITY — Determines the majority (value that occurs most often) of the inputs.	String

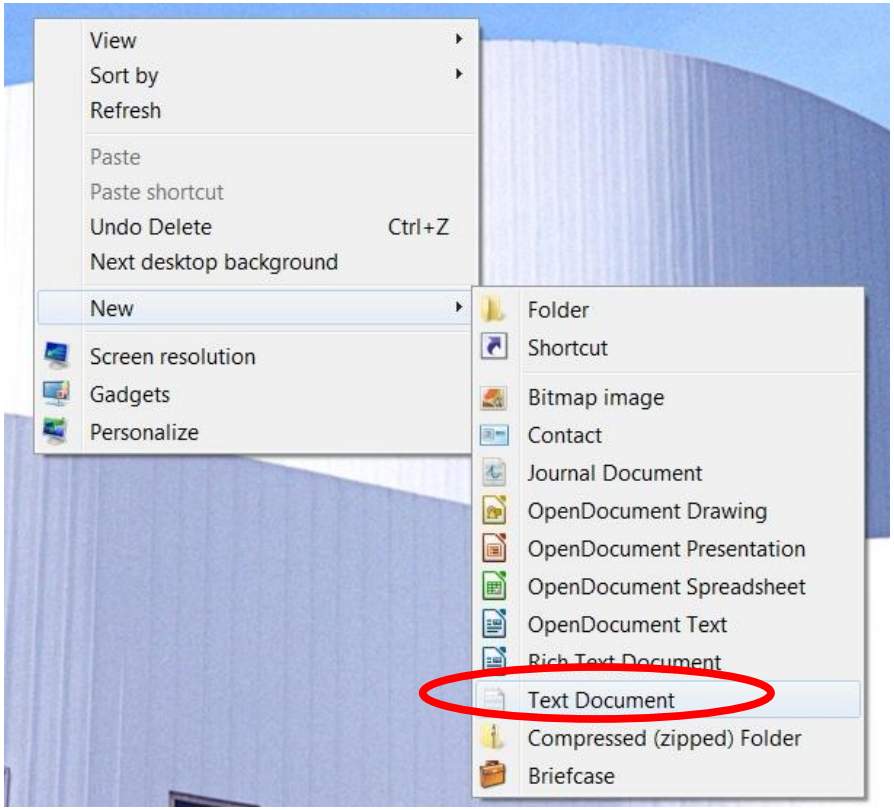
Scroll to bottom for sample Python script

CellStatistics example 2 (stand-alone script)

This example calculates the standard deviation per cell on several input GRID rasters and outputs the result as a GRID raster.

```
# 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 = "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")
```

Python scripts are text files with ".py" extension

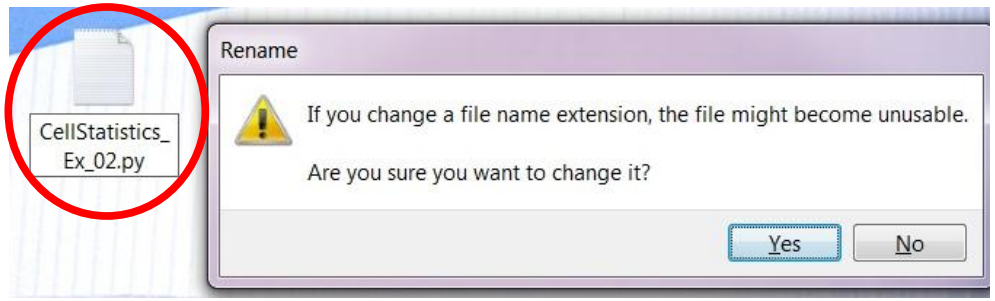


Create new document, copy / paste text

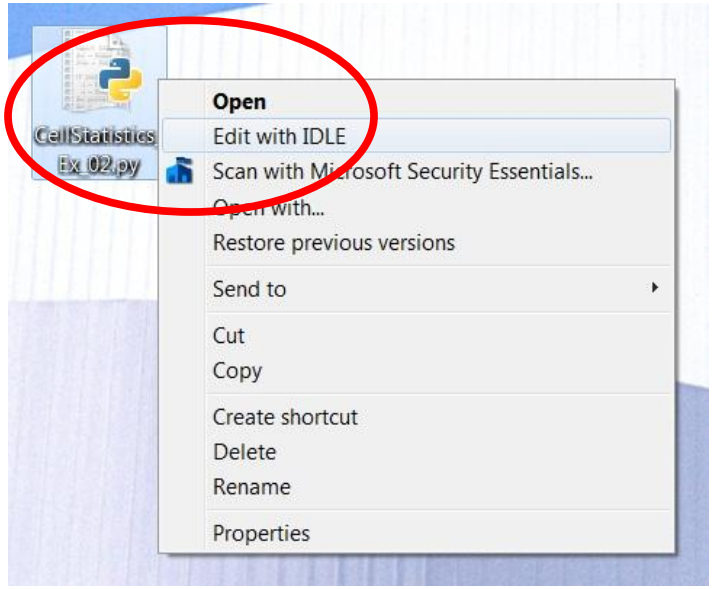
New Text Document.txt - Notepad

File Edit Format View Help

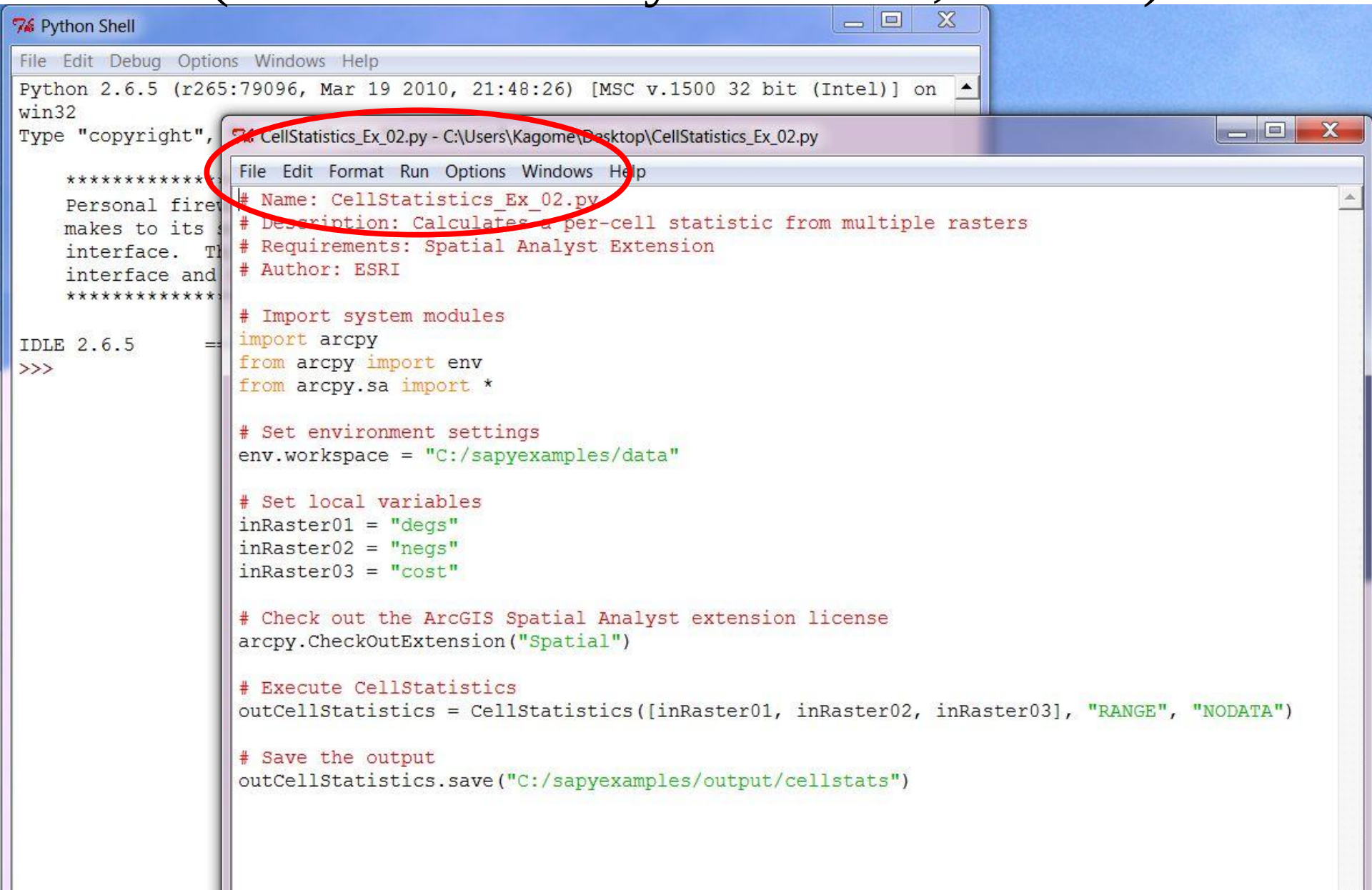
```
# Name: CellStatistics_Ex_02.py
# Description: Calculates a per-cell statistic from multiple rasters
# Requirements: Spatial Analyst Extension
# Author: ESRI

# 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 = "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")
```



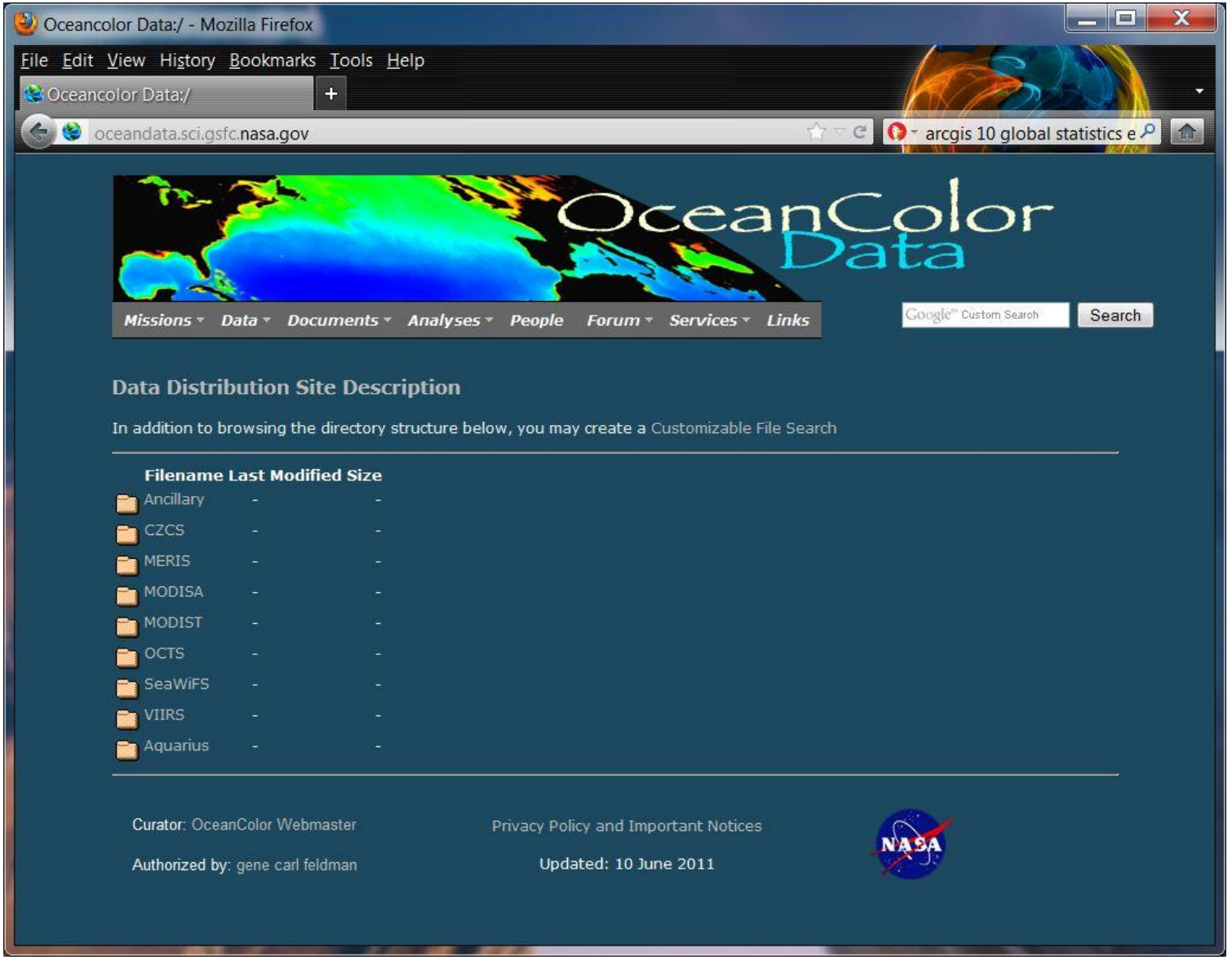
For existing scripts, right click and edit with IDLE



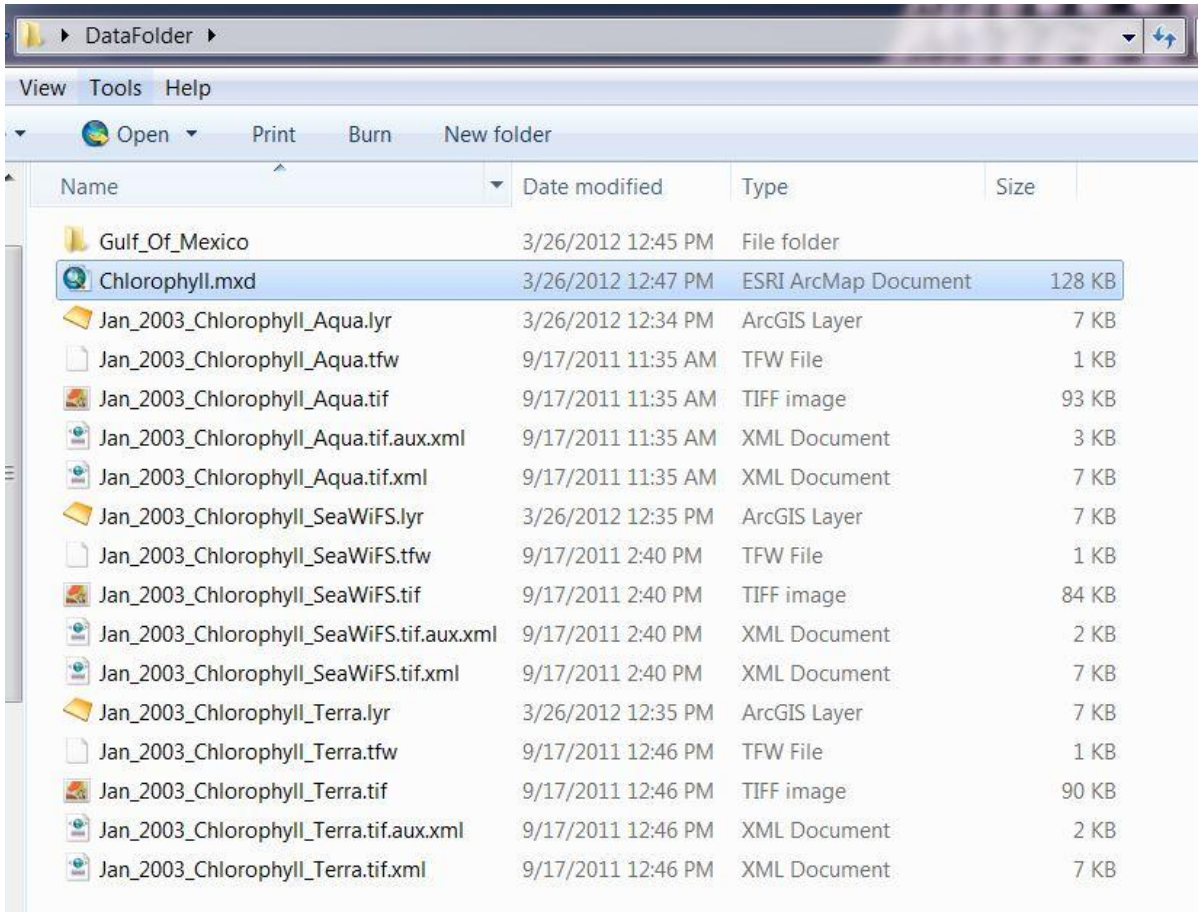
IDLE displays color-coded script (Don't need the Python Shell, close it)



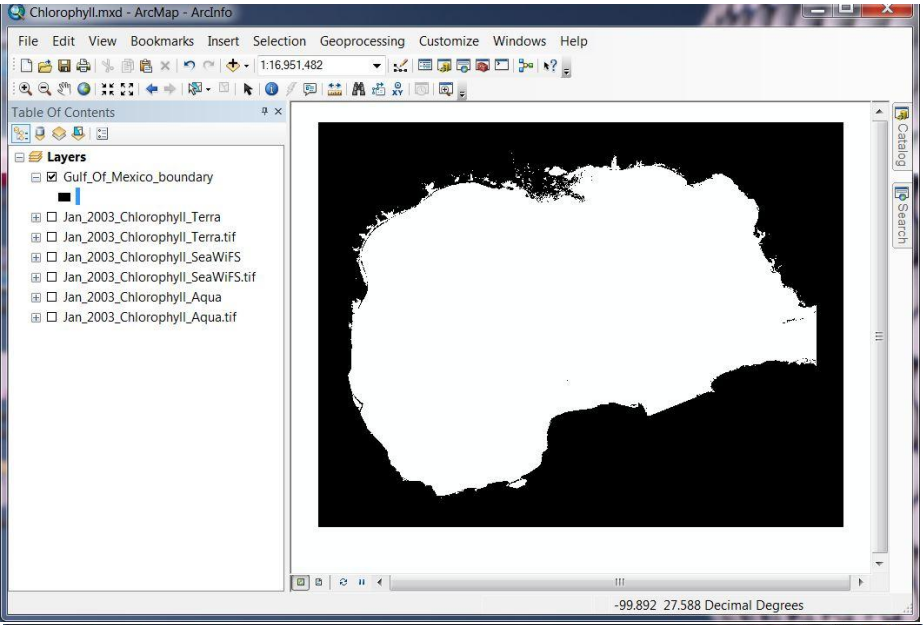
Example
Get GIS data

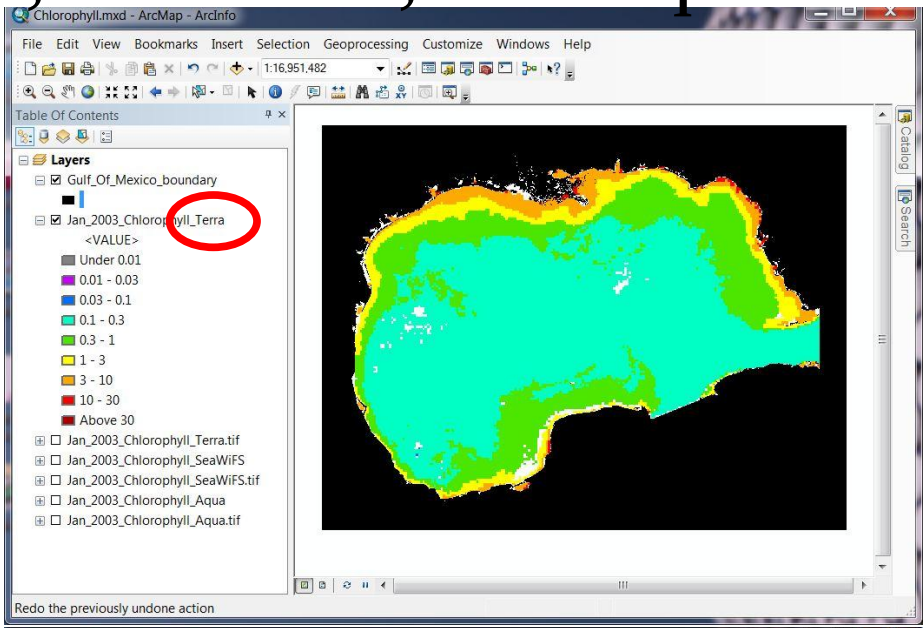


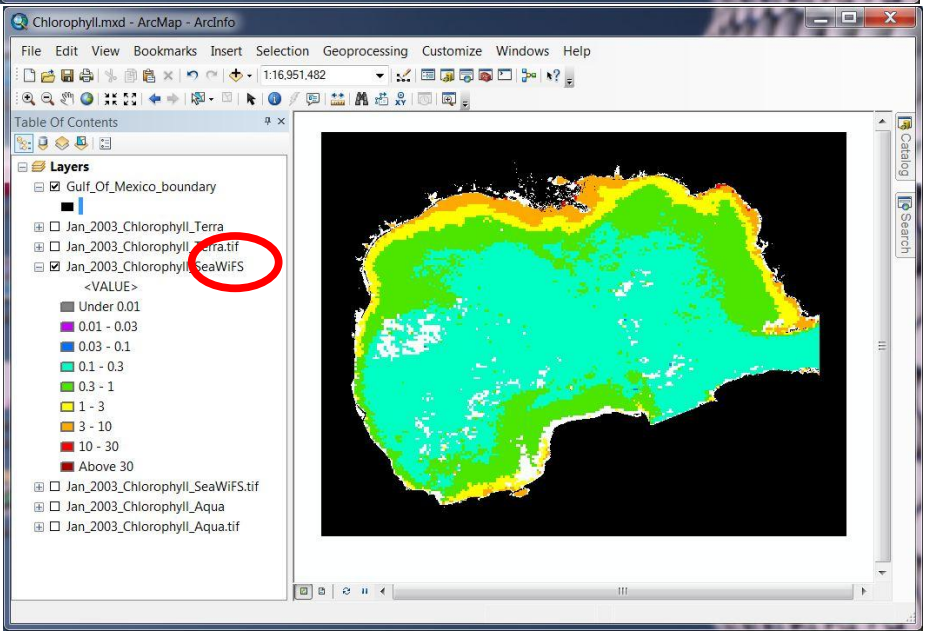
Example
Downloaded, processed, and symbolized layers

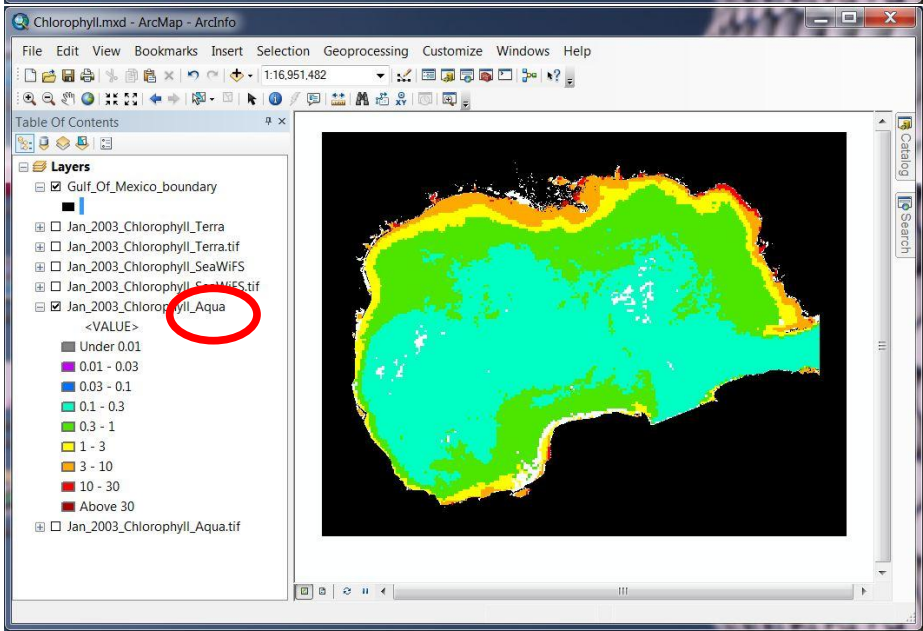


Gulf Of Mexico outline, plus monthly Chlorophyll concentration from Terra, SeaWiFS, and Aqua









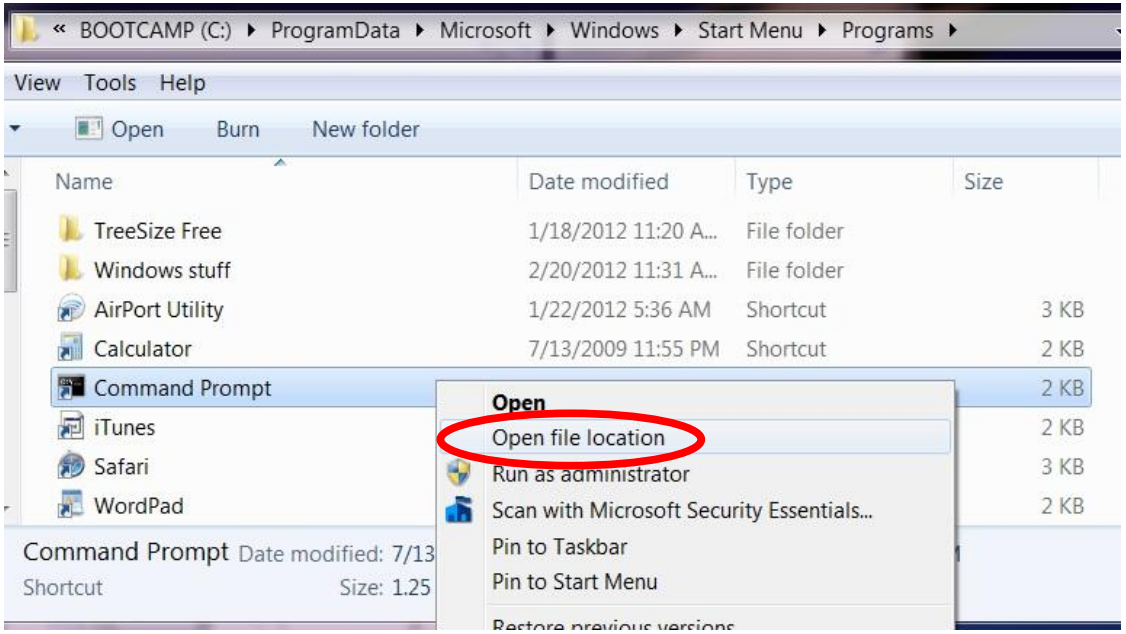
Edit script to match data location and names

Mean.py - C:\Users\Kagome\Desktop\Mean.py

File Edit Format Run Options Windows Help

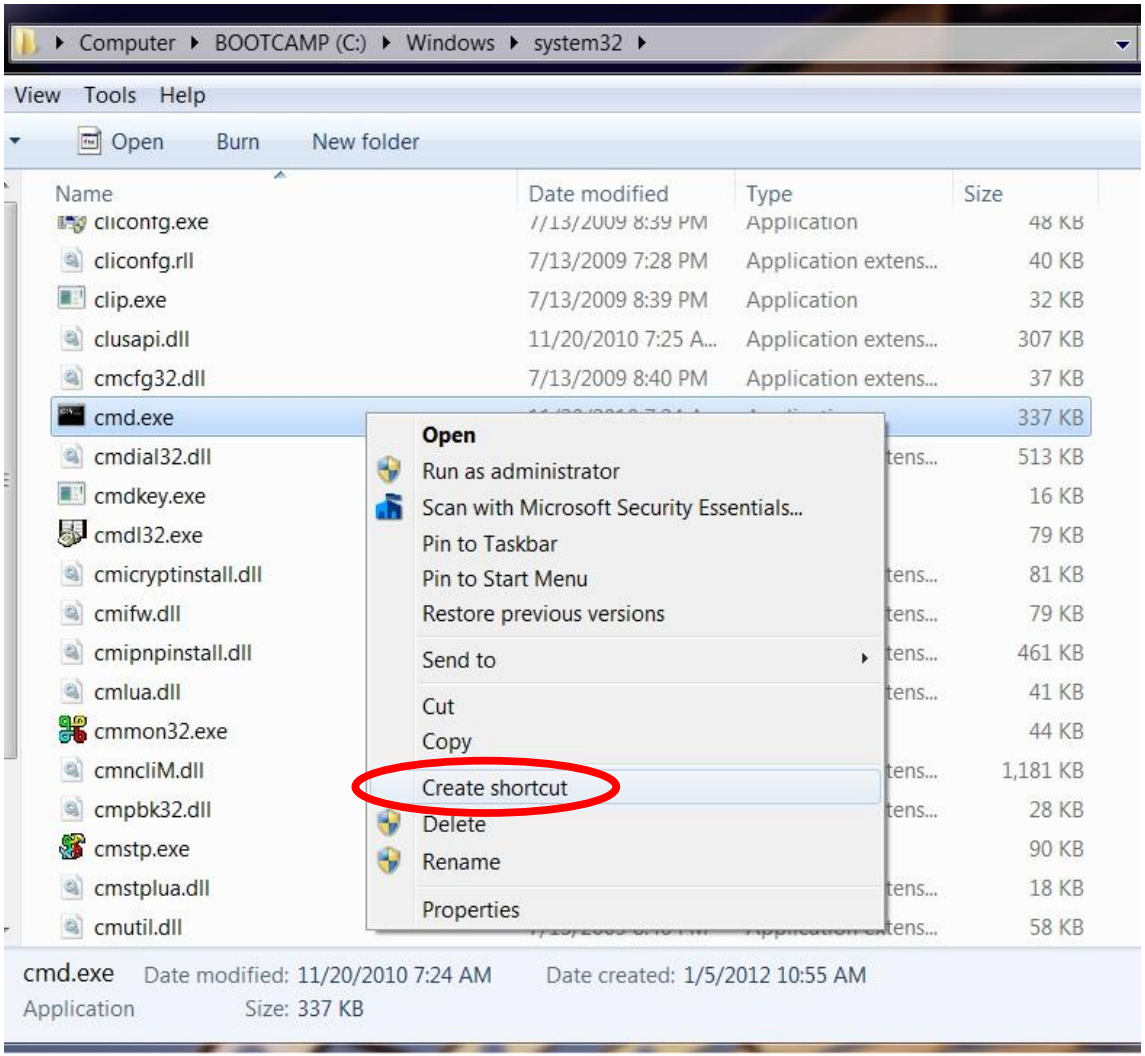
```
# Name: Mean.py
# Description: Calculates mean chlorophyll concentration from 3 input rasters
# Requirements: Spatial Analyst Extension
# Author: Name

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

# 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")

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

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

Setup
Run Python from Windows Command Prompt

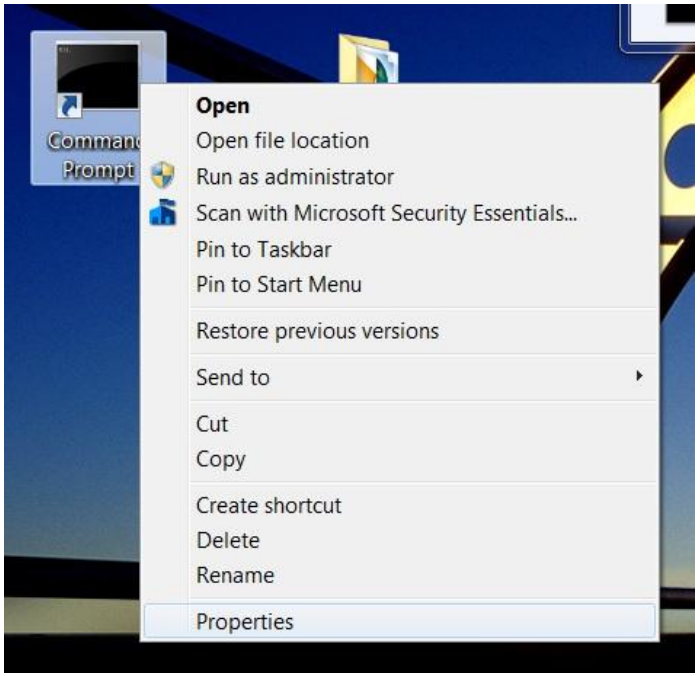


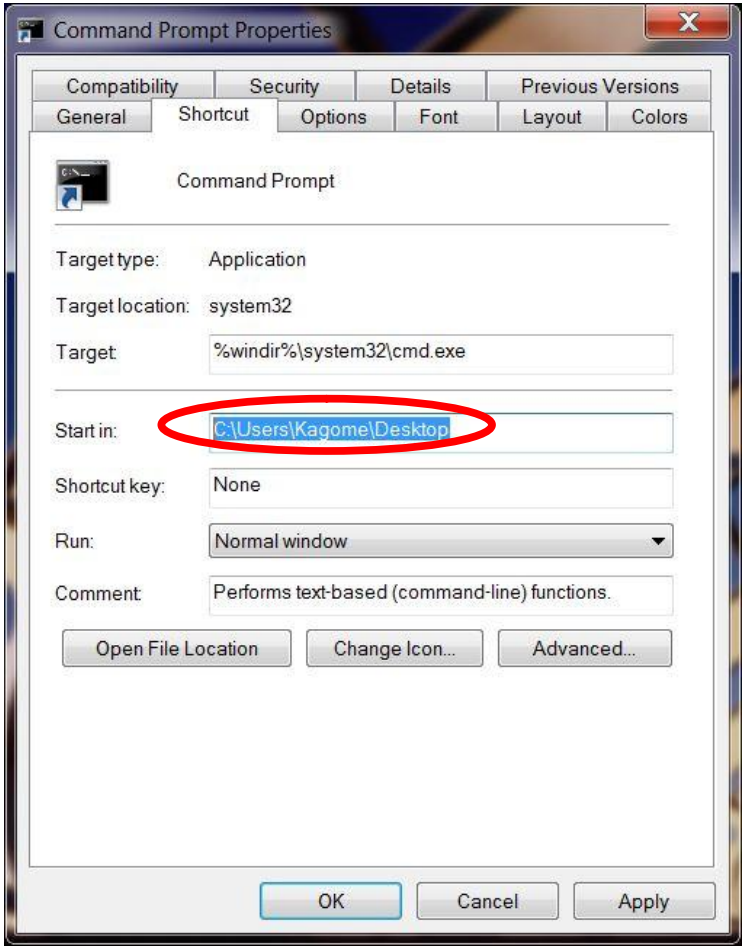
Setup

Easier to create a customized shortcut

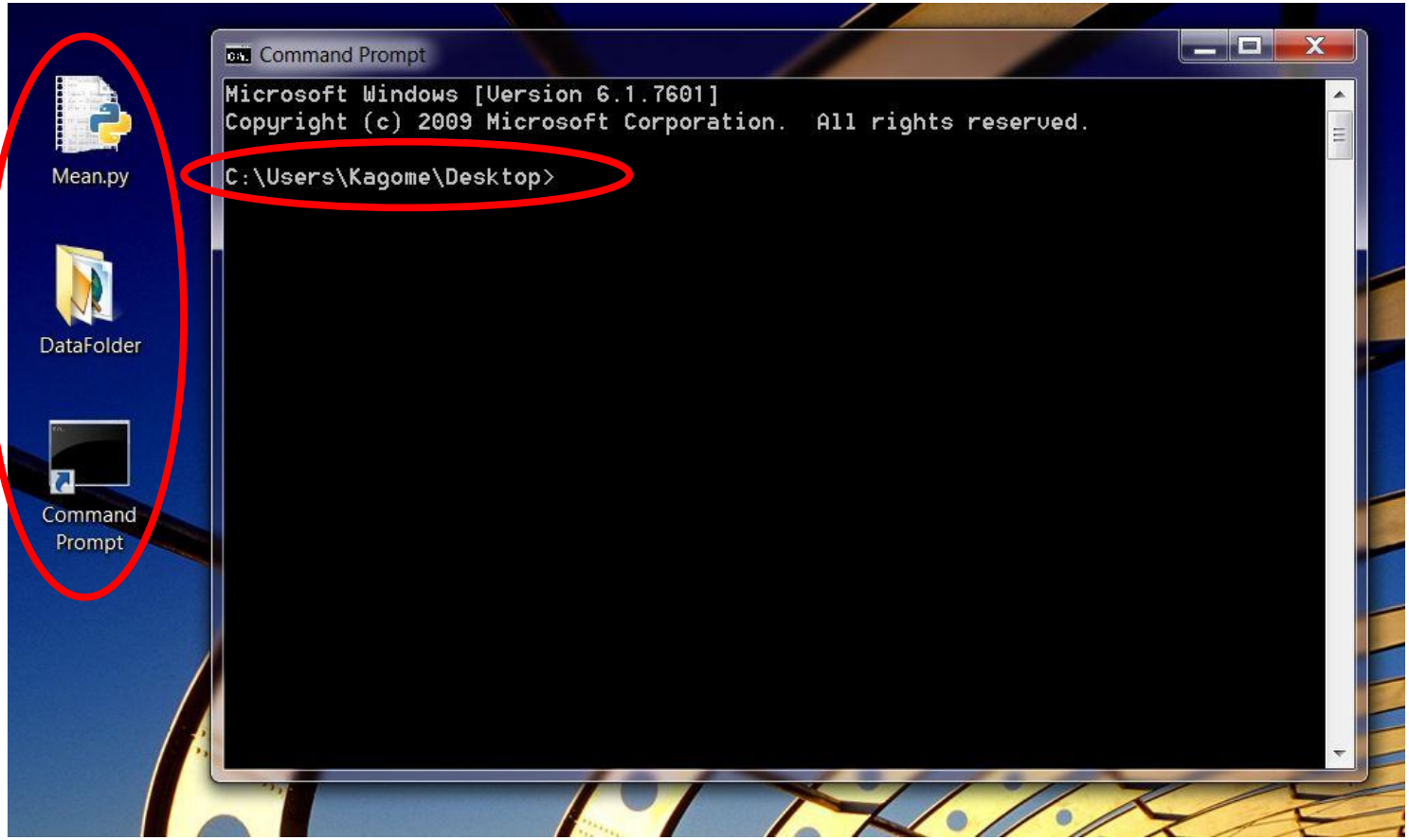


Setup
Start Command Prompt in same folder as script

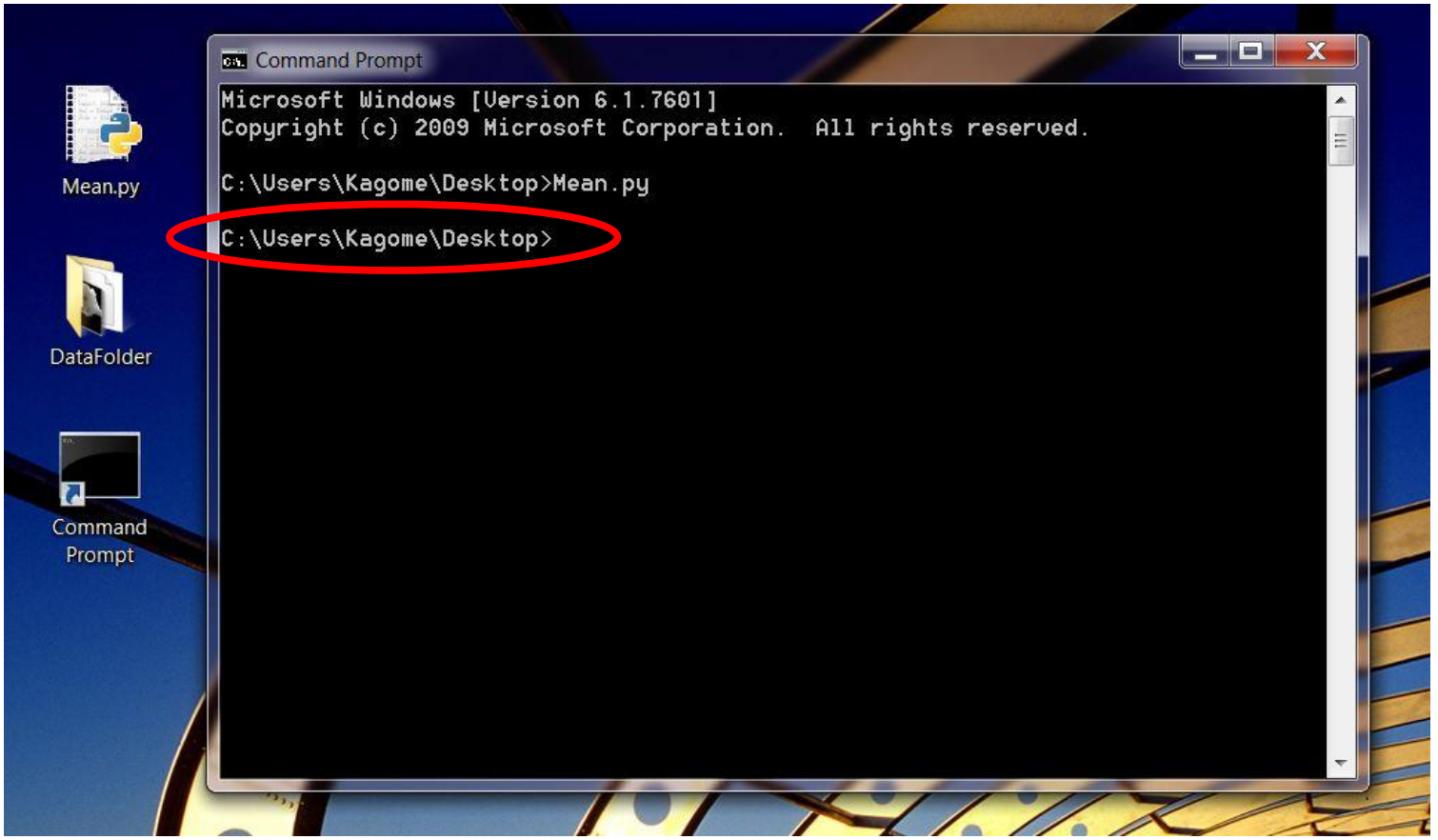




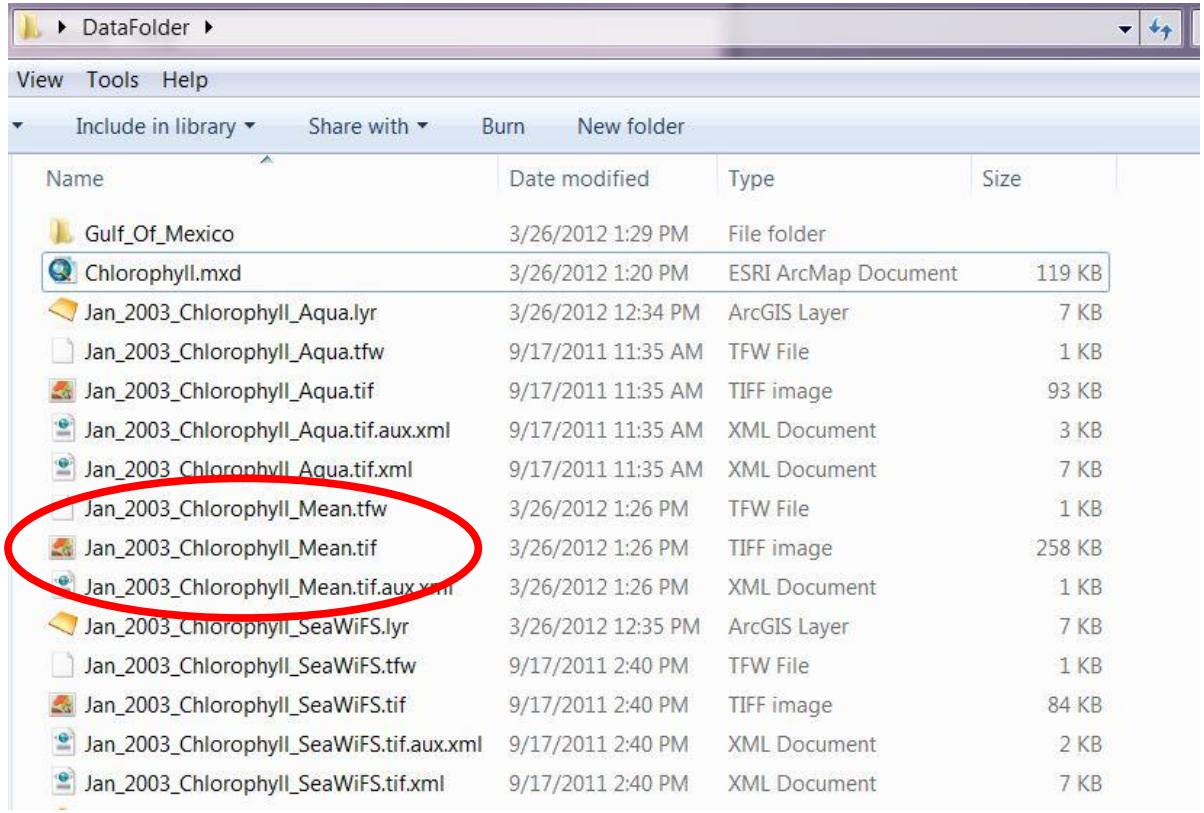
Desktop and Command Prompt ready



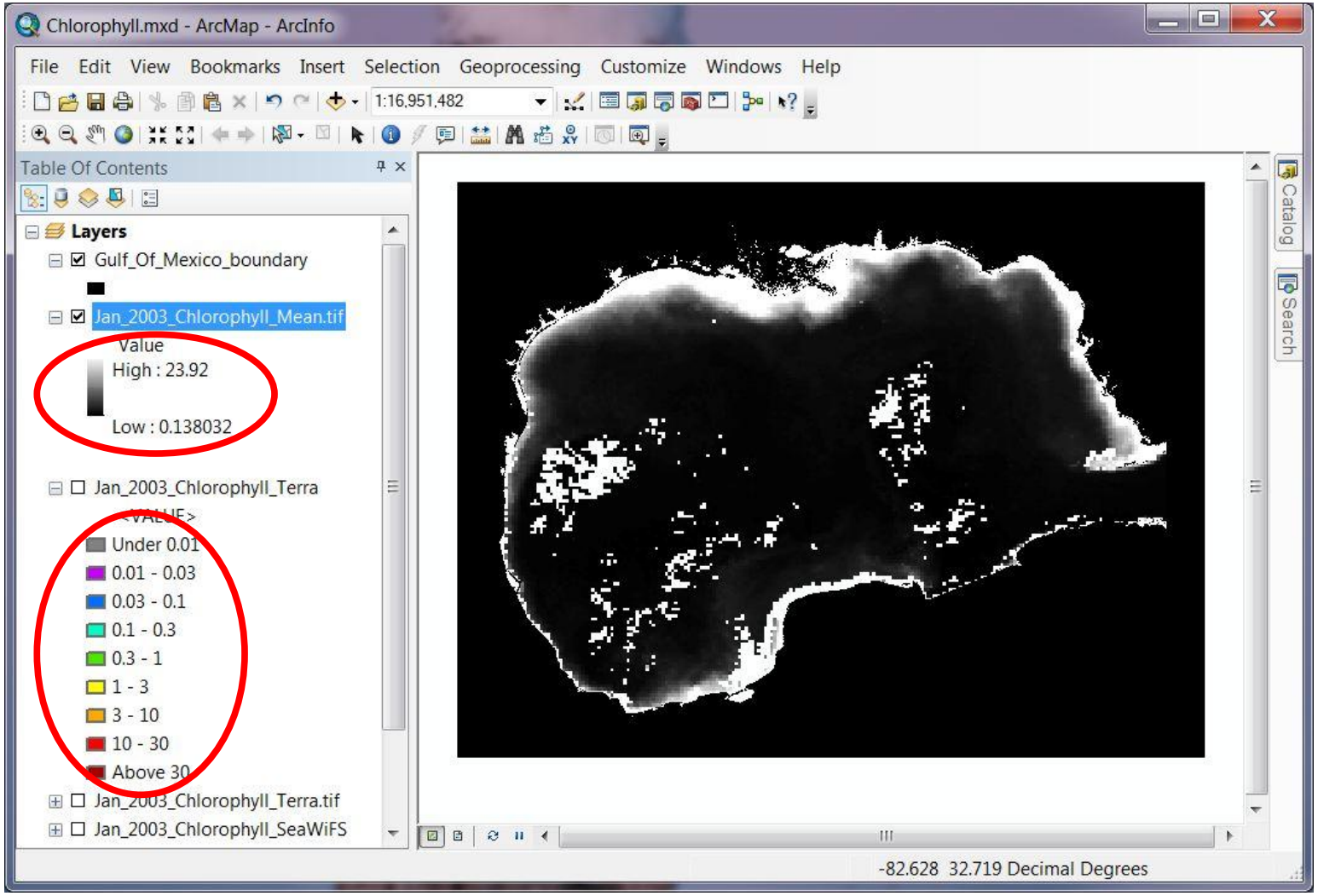
Run... no feedback (maybe you like that)



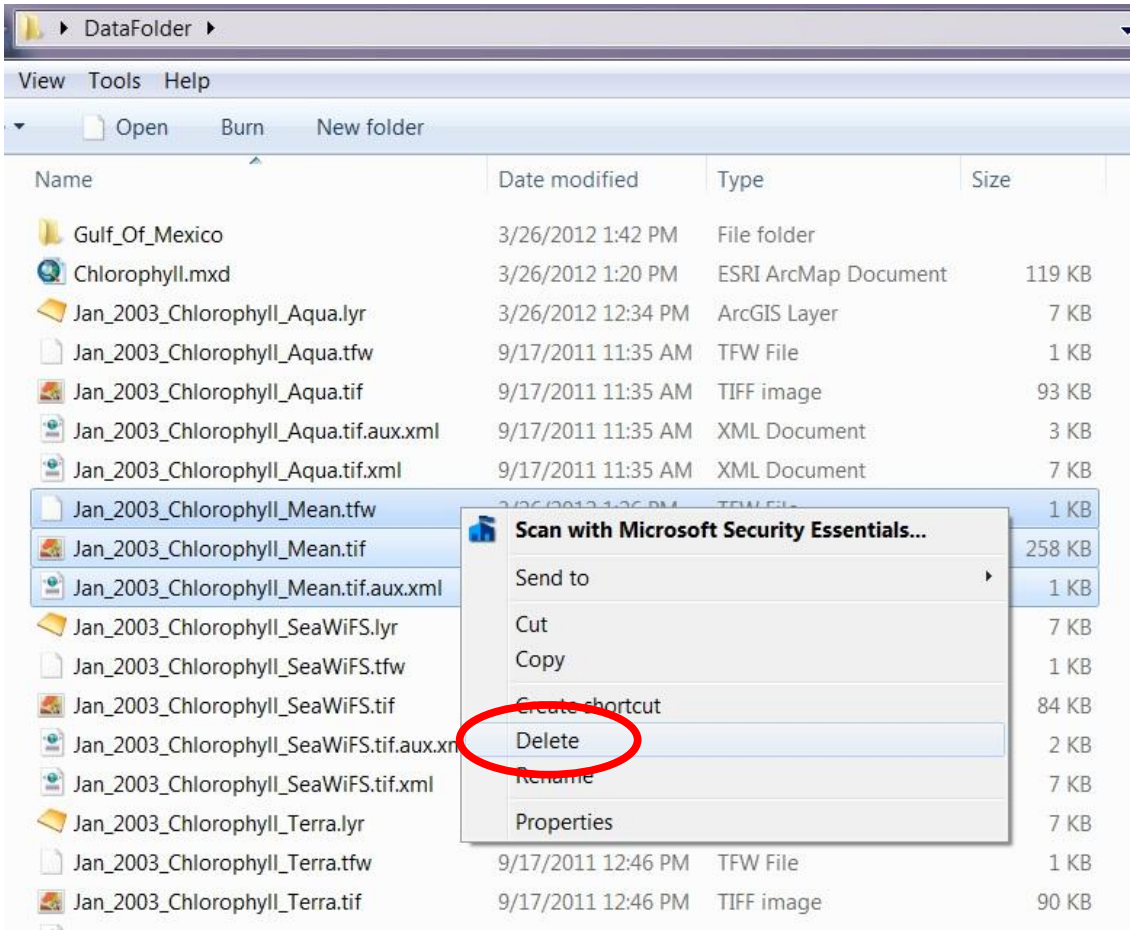
Script created “Mean” datafiles



Open in ArcMap... unsymbolized



ArcPython does not like to overwrite files



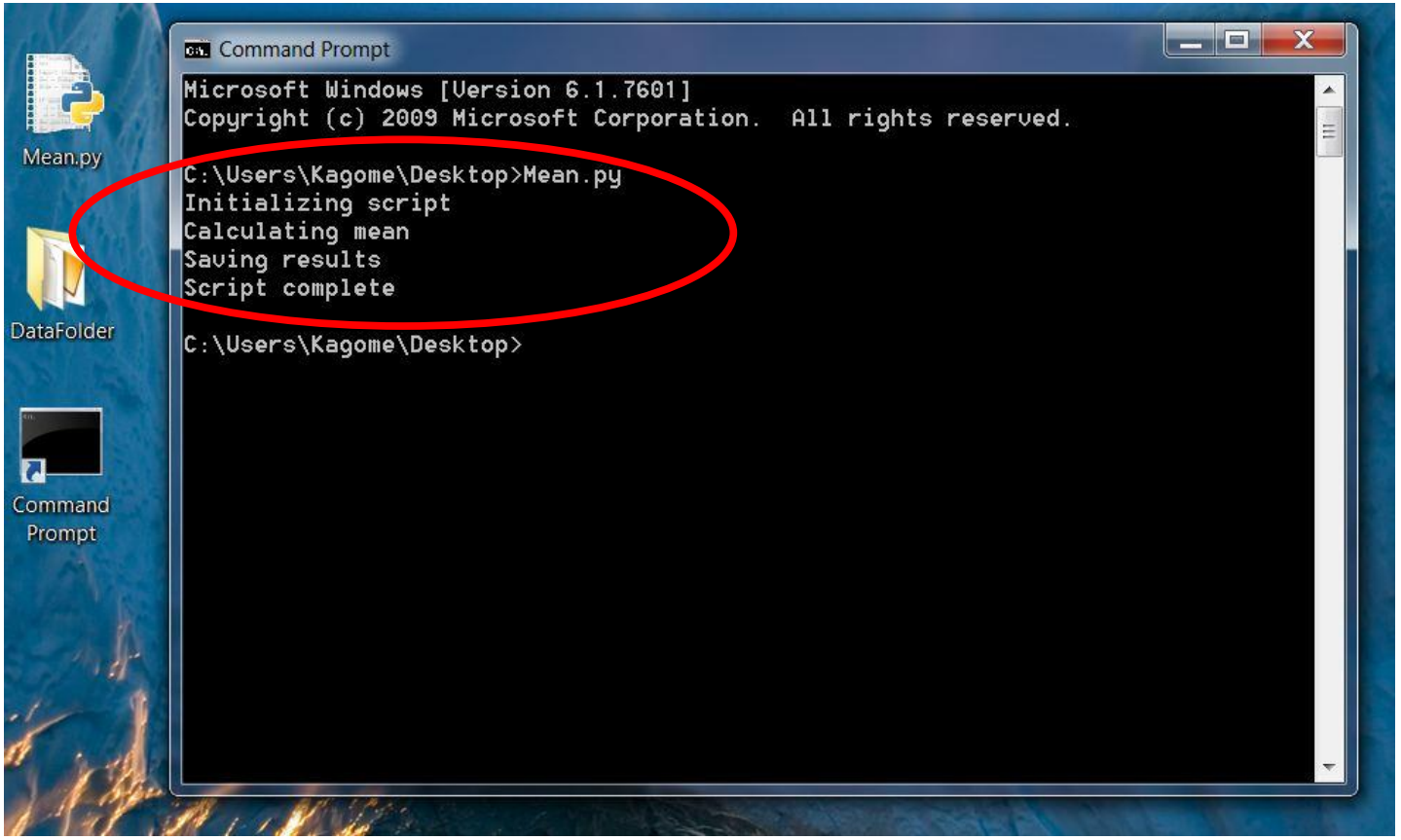
Add status messages, create symbolize layer

Mean.py - C:\Users\Kagome\Desktop\Mean.py

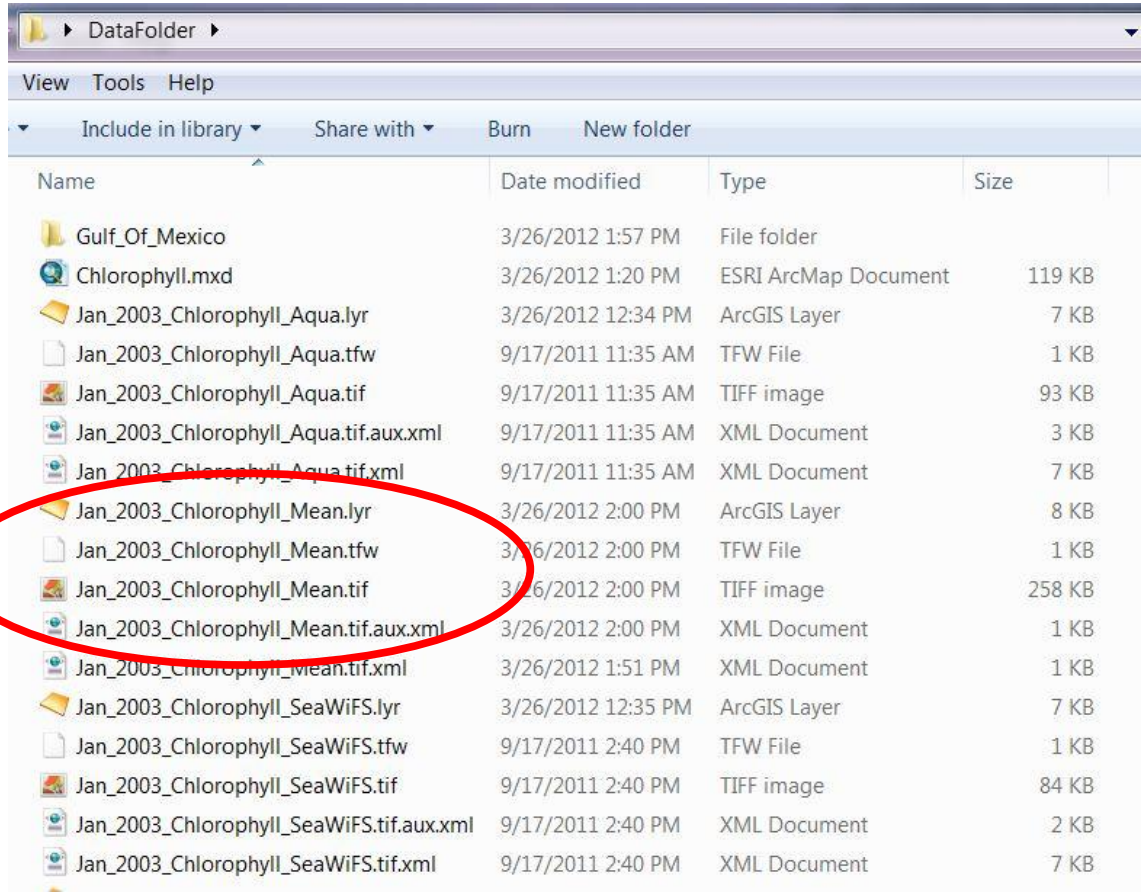
File Edit Format Run Options Windows Help

```
# 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"
```

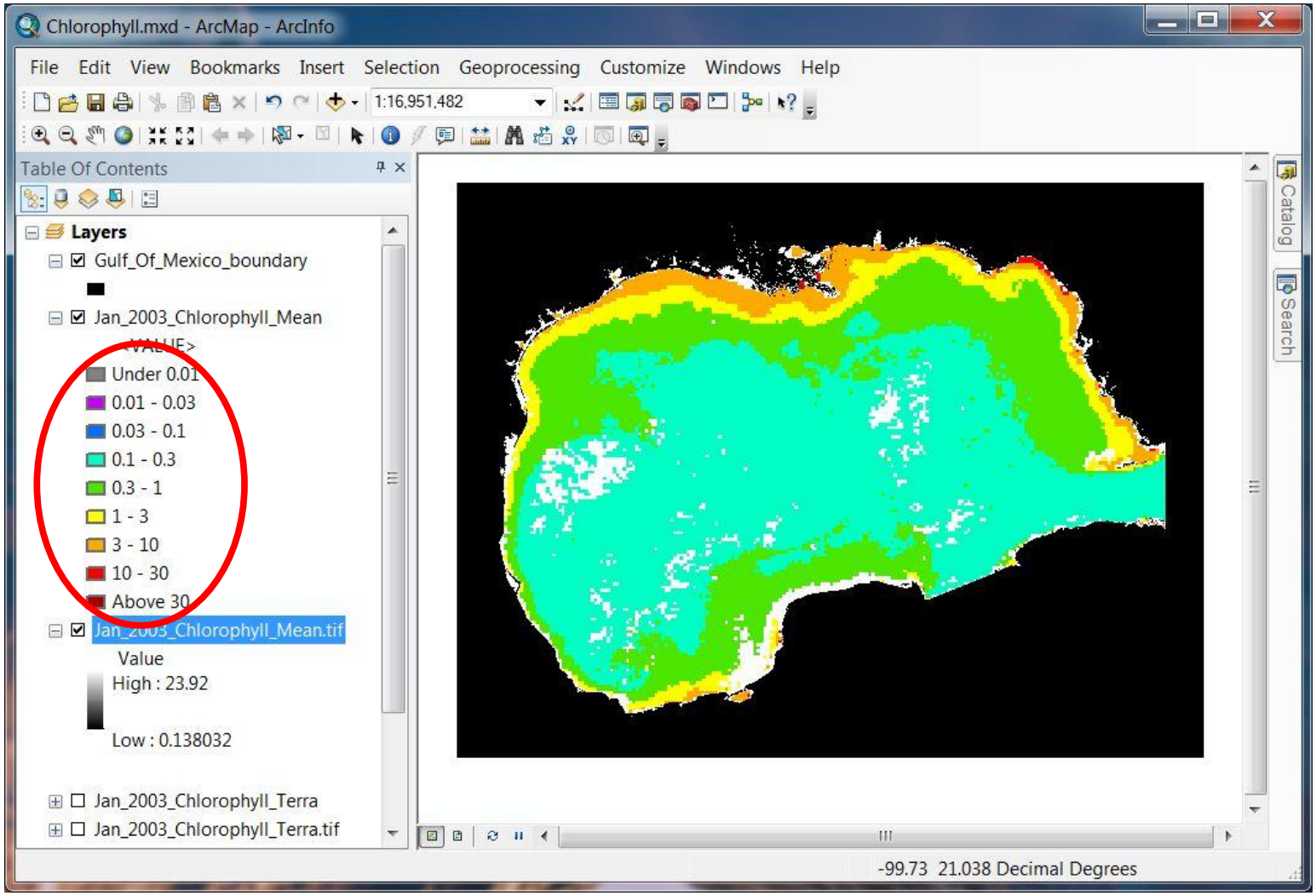
Run again... feedback shows



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