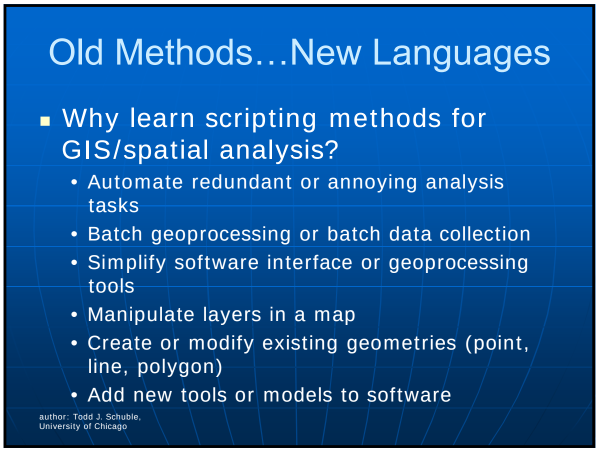
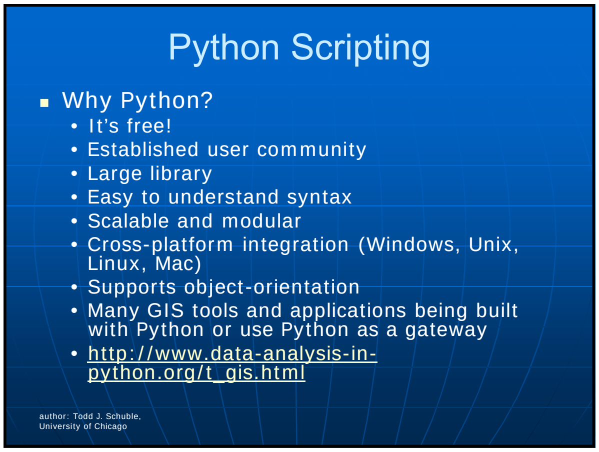


Old Methods…New Languages

- Why learn scripting methods for GIS/spatial analysis?
  - Automate redundant or annoying analysis tasks
  - Batch geoprocessing or batch data collection
  - Simplify software interface or geoprocessing tools
  - Manipulate layers in a map
  - Create or modify existing geometries (point, line, polygon)
  - Add new tools or models to software

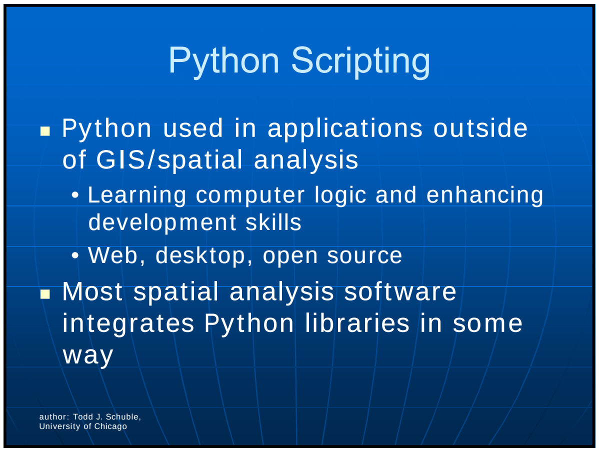
author: Todd J. Schuble,
University of Chicago



Python Scripting

- Why Python?
  - It's free!
  - Established user community
  - Large library
  - Easy to understand syntax
  - Scalable and modular
  - Cross-platform integration (Windows, Unix, Linux, Mac)
  - Supports object-orientation
  - Many GIS tools and applications being built with Python or use Python as a gateway
  - http://www.data-analysis-in-python.org/t_gis.html

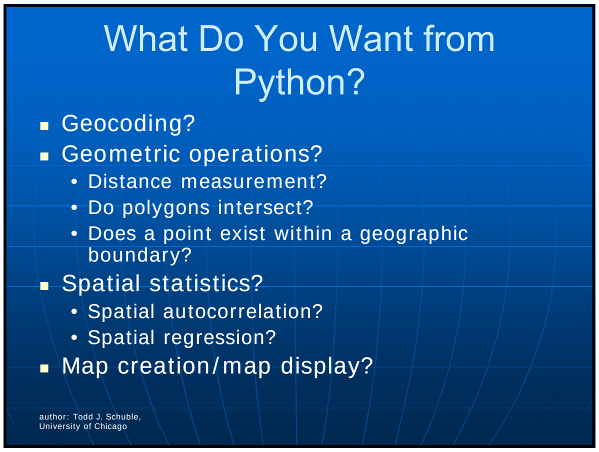
author: Todd J. Schuble,
University of Chicago



Python Scripting

- Python used in applications outside of GIS/spatial analysis
  - Learning computer logic and enhancing development skills
  - Web, desktop, open source
- Most spatial analysis software integrates Python libraries in some way

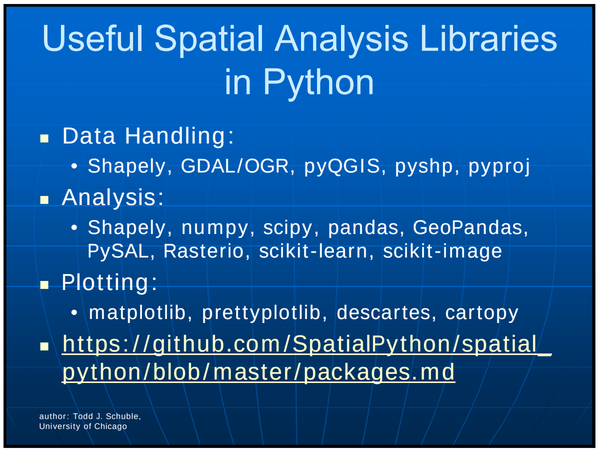
author: Todd J. Schuble,
University of Chicago



What Do You Want from Python?

- Geocoding?
- Geometric operations?
  - Distance measurement?
  - Do polygons intersect?
  - Does a point exist within a geographic boundary?
- Spatial statistics?
  - Spatial autocorrelation?
  - Spatial regression?
- Map creation/map display?

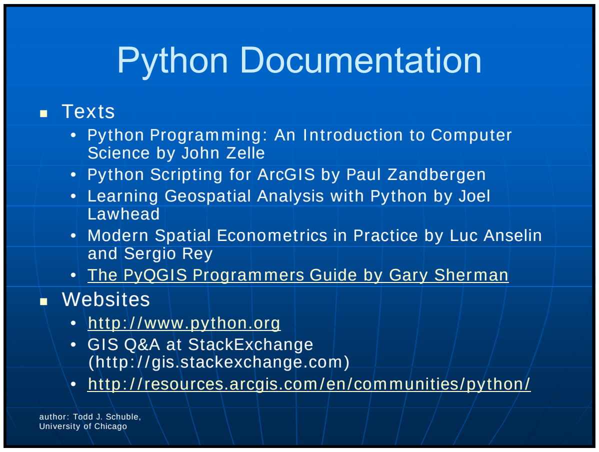
author: Todd J. Schuble,
University of Chicago



Useful Spatial Analysis Libraries in Python

- Data Handling:
  - Shapely, GDAL/OGR, pyQGIS, pyshp, pyproj
- Analysis:
  - Shapely, numpy, scipy, pandas, GeoPandas, PySAL, Rasterio, scikit-learn, scikit-image
- Plotting:
  - matplotlib, prettyplotlib, descartes, cartopy
- https://github.com/SpatialPython/spatial_python/blob/master/packages.md

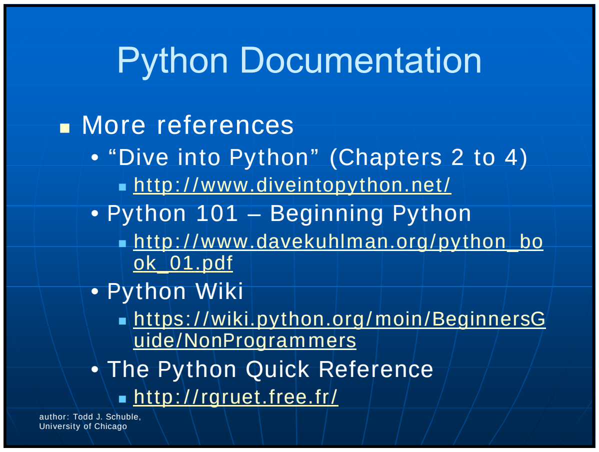
author: Todd J. Schuble,
University of Chicago



Python Documentation

- Texts
  - Python Programming: An Introduction to Computer Science by John Zelle
  - Python Scripting for ArcGIS by Paul Zandbergen
  - Learning Geospatial Analysis with Python by Joel Lawhead
  - Modern Spatial Econometrics in Practice by Luc Anselin and Sergio Rey
  - The PyQGIS Programmers Guide by Gary Sherman
- Websites
  - http://www.python.org
  - GIS Q&A at StackExchange (http://gis.stackexchange.com)
  - http://resources.arcgis.com/en/communities/python/

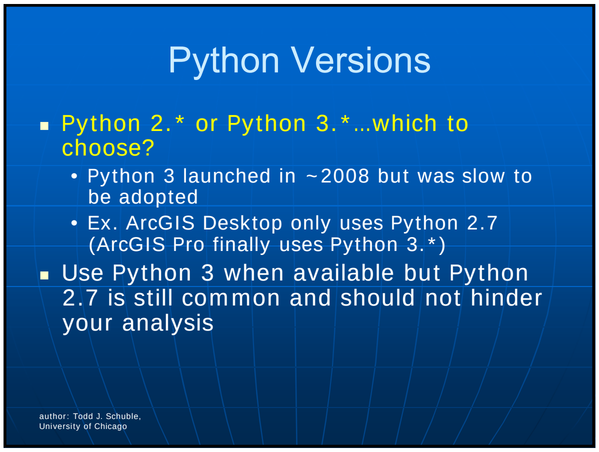
author: Todd J. Schuble,
University of Chicago



Python Documentation

- More references
  - "Dive into Python" (Chapters 2 to 4)
    - http://www.diveintopython.net/
  - Python 101 – Beginning Python
    - http://www.davekuhlman.org/python_book_01.pdf
  - Python Wiki
    - https://wiki.python.org/moin/BeginnersGuide/NonProgrammers
  - The Python Quick Reference
    - http://rgruet.free.fr/

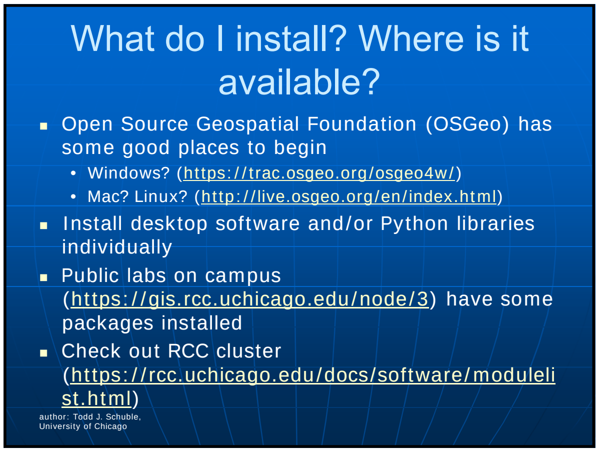
author: Todd J. Schuble,
University of Chicago



Python Versions

- Python 2.* or Python 3.*…which to choose?
  - Python 3 launched in ~2008 but was slow to be adopted
  - Ex. ArcGIS Desktop only uses Python 2.7 (ArcGIS Pro finally uses Python 3.*)
- Use Python 3 when available but Python 2.7 is still common and should not hinder your analysis

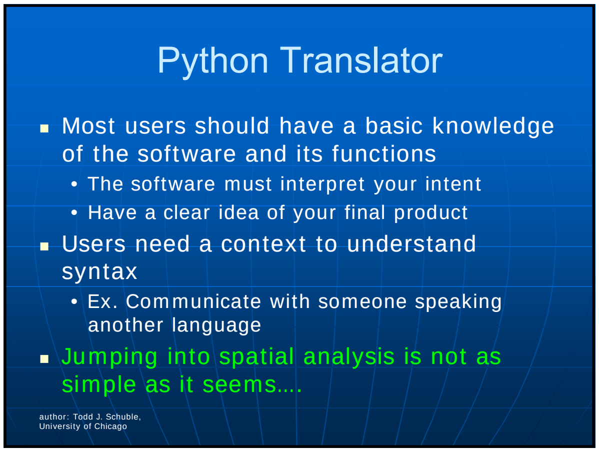
author: Todd J. Schuble,
University of Chicago



What do I install? Where is it available?

- Open Source Geospatial Foundation (OSGeo) has some good places to begin
  - Windows? (https://trac.osgeo.org/osgeo4w/)
  - Mac? Linux? (http://live.osgeo.org/en/index.html)
- Install desktop software and/or Python libraries individually
- Public labs on campus (https://gis.rcc.uchicago.edu/node/3) have some packages installed
- Check out RCC cluster (https://rcc.uchicago.edu/docs/software/modulelist.html)

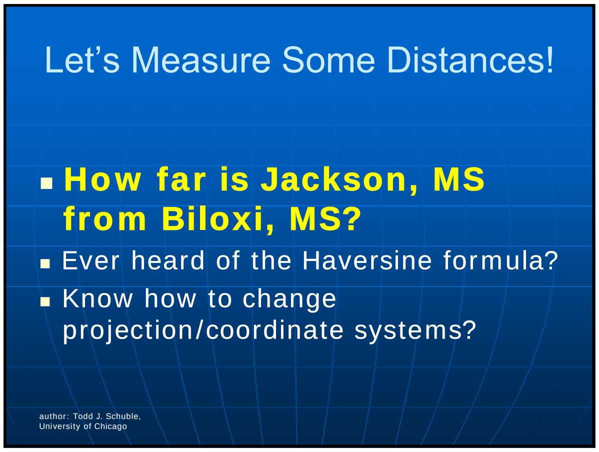
author: Todd J. Schuble,
University of Chicago



Python Translator

- Most users should have a basic knowledge of the software and its functions
  - The software must interpret your intent
  - Have a clear idea of your final product
- Users need a context to understand syntax
  - Ex. Communicate with someone speaking another language
- Jumping into spatial analysis is not as simple as it seems….

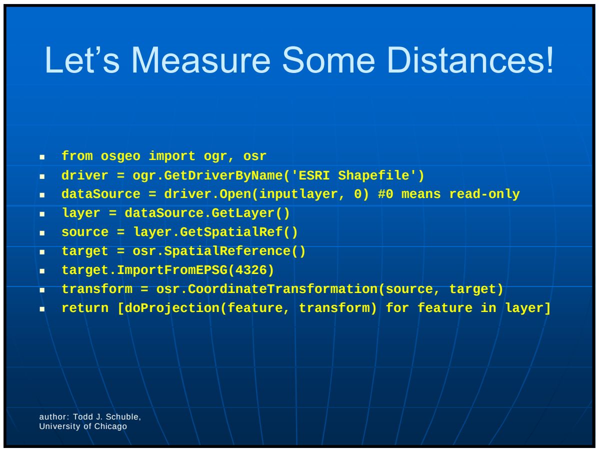
author: Todd J. Schuble,
University of Chicago



Let’s Measure Some Distances!

- **How far is Jackson, MS from Biloxi, MS?**
- Ever heard of the Haversine formula?
- Know how to change projection/coordinate systems?

author: Todd J. Schuble,
University of Chicago



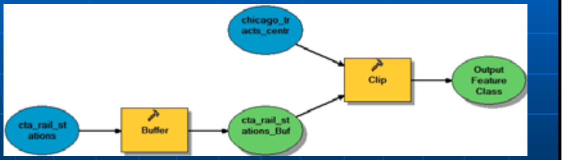
Let’s Measure Some Distances!

```
from osgeo import ogr, osr
driver = ogr.GetDriverByName('ESRI Shapefile')
dataSource = driver.Open(inputlayer, 0) #0 means read-only
layer = dataSource.GetLayer()
source = layer.GetSpatialRef()
target = osr.SpatialReference()
target.ImportFromEPSG(4326)
transform = osr.CoordinateTransformation(source, target)
return [doProjection(feature, transform) for feature in layer]
```

author: Todd J. Schuble,
University of Chicago

ArcGIS Modelbuilder

- ModelBuilder provides a flow diagram interface to start your task

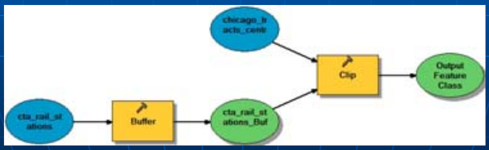


- A majority of parameters can and should be set in ModelBuilder

author: Todd J. Schuble, University of Chicago

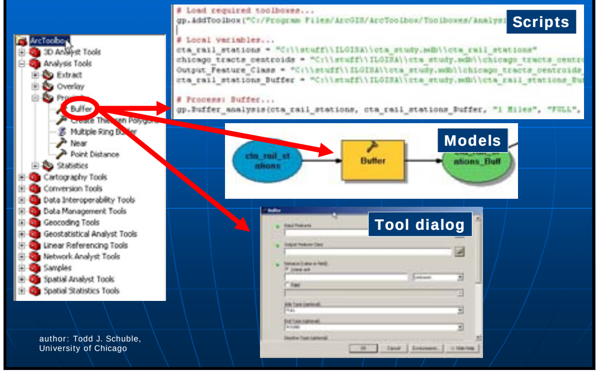
ArcGIS ModelBuilder

- Graphic environment to build a geoprocessing workflow
- Use existing tools to create a new framework
  - New framework will serve as the basis for your script



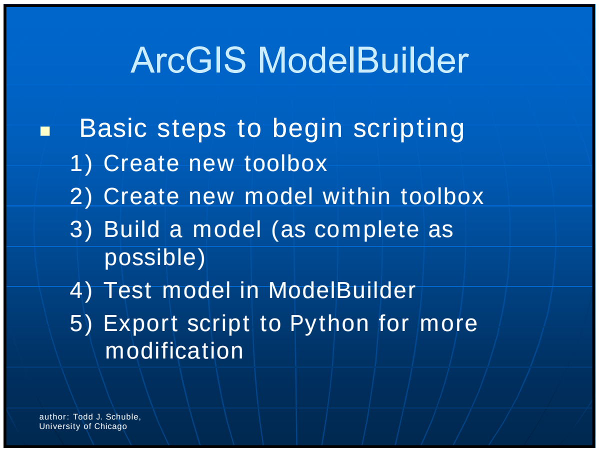
author: Todd J. Schuble, University of Chicago

ArcGIS ModelBuilder



```
# Load required toolboxes...
gp.AddToolbox("C:/Program Files/ArcGIS/ArcToolbox/Toolboxes/Analysi

# Local variables...
cta_rail_stations = "C:\\stuff\\ILOISA\\cta_study.mdb\\cta_rail_stations"
chicago_tracts_centroids = "C:\\stuff\\ILOISA\\cta_study.mdb\\chicago_tracts_centro
Output_Feature_Class = "C:\\stuff\\ILOISA\\cta_study.mdb\\chicago_tracts_centroids_
cta_rail_stations_Buffer = "C:\\stuff\\ILOISA\\cta_study.mdb\\cta_rail_stations_Bu

# Process: Buffer...
gp.Buffer_analysis(cta_rail_stations, cta_rail_stations_Buffer, "1 Miles", "FULL",
```

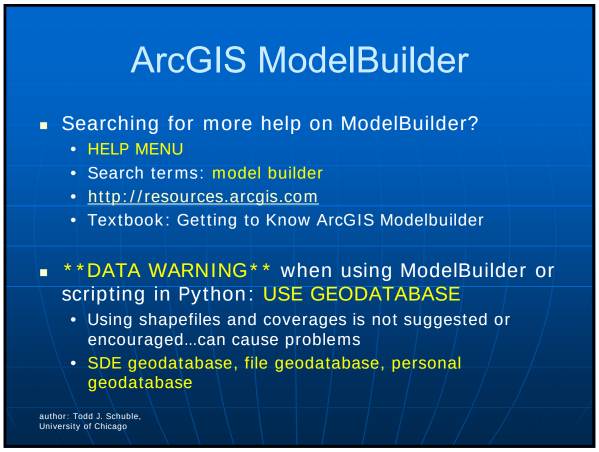
author: Todd J. Schuble, University of Chicago



ArcGIS ModelBuilder

- Basic steps to begin scripting
  1) Create new toolbox
  2) Create new model within toolbox
  3) Build a model (as complete as possible)
  4) Test model in ModelBuilder
  5) Export script to Python for more modification

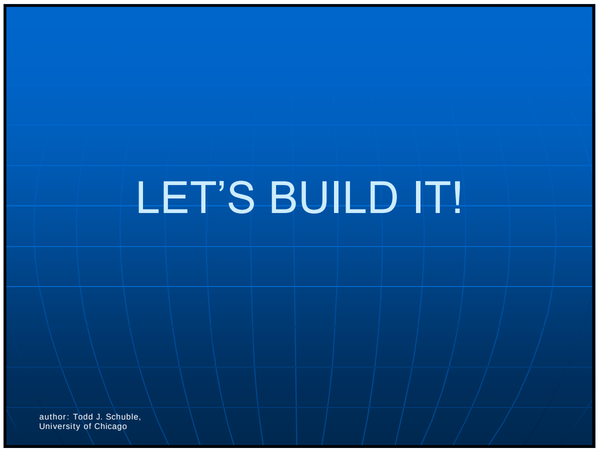
author: Todd J. Schuble,
University of Chicago



ArcGIS ModelBuilder

- Searching for more help on ModelBuilder?
  - HELP MENU
  - Search terms: model builder
  - http://resources.arcgis.com
  - Textbook: Getting to Know ArcGIS Modelbuilder
- **DATA WARNING** when using ModelBuilder or scripting in Python: USE GEODATABASE
  - Using shapefiles and coverages is not suggested or encouraged…can cause problems
  - SDE geodatabase, file geodatabase, personal geodatabase

author: Todd J. Schuble,
University of Chicago



LET’S BUILD IT!

author: Todd J. Schuble,
University of Chicago

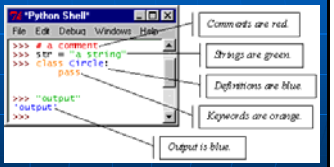
Python Interface

- Multiple ways to edit Python scripts
  - IDLE – a cross-platform Python development environment
  - Other Python editors: PyCharm, Notepad++, etc.
  - PythonWin – a Windows only interface to Python
  - Python Shell – running 'python' from the Command Line opens this interactive shell

author: Todd J. Schuble,
University of Chicago

Python Interface

- Editors helps you program in Python by:
  - color-coding your program code
  - debugging
  - auto-indent
  - interactive shell



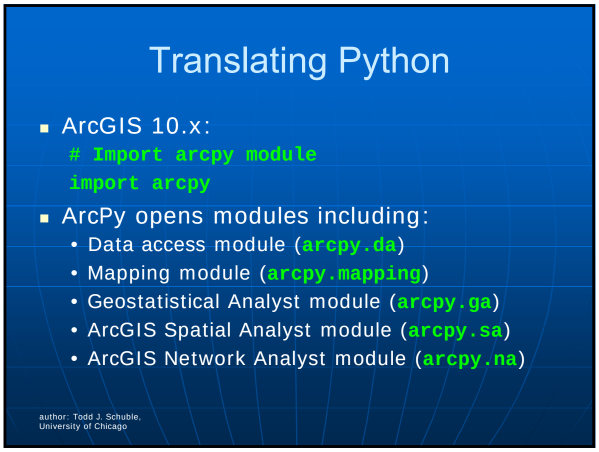
```
>>> # a comment
>>> str = "a string"
>>> class Circle:
        pass


>>> "output"
'output'
>>>
```

author: Todd J. Schuble,
University of Chicago

Translating Python

- ModelBuilder produces Python scripts in a specific format
  - Description, software and library settings, variable settings, functions
- Descriptions
  - Name of script
  - When it was created
  - Generated by ModelBuilder
- Import system modules or libraries
  - Statement references how Python will communicate with its libraries and operating system
  - http://help.arcgis.com/en/arcgisdesktop/10.0/help/index.html#/Importing_ArcPy/002z00000008000000/

author: Todd J. Schuble,
University of Chicago



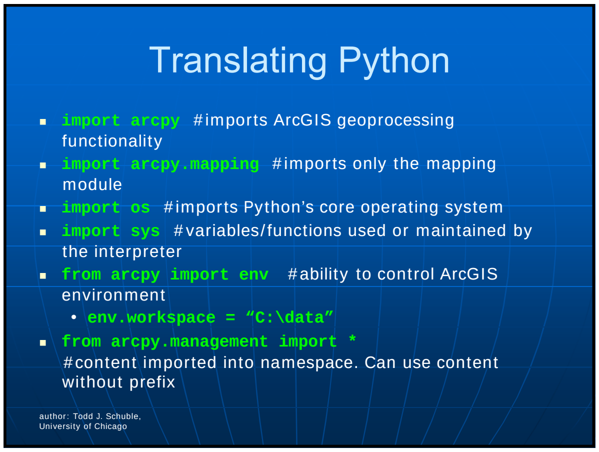
Translating Python

- ArcGIS 10.x:

```
# Import arcpy module
import arcpy
```

- ArcPy opens modules including:
  - Data access module (`arcpy.da`)
  - Mapping module (`arcpy.mapping`)
  - Geostatistical Analyst module (`arcpy.ga`)
  - ArcGIS Spatial Analyst module (`arcpy.sa`)
  - ArcGIS Network Analyst module (`arcpy.na`)

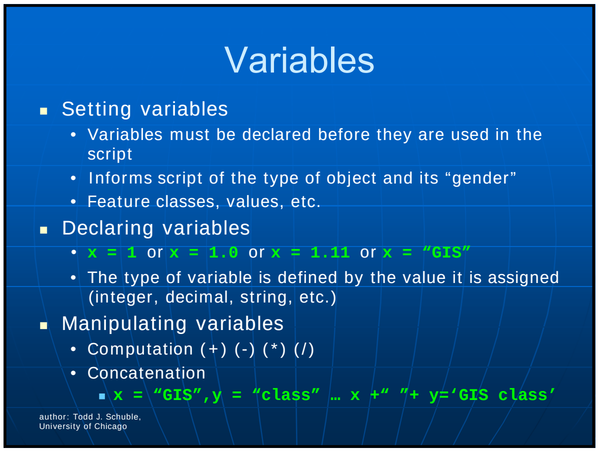
author: Todd J. Schuble,
University of Chicago



Translating Python

- `import arcpy` #imports ArcGIS geoprocessing functionality
- `import arcpy.mapping` #imports only the mapping module
- `import os` #imports Python's core operating system
- `import sys` #variables/functions used or maintained by the interpreter
- `from arcpy import env` #ability to control ArcGIS environment
  - `env.workspace = “C:\data”`
- `from arcpy.management import *`
  #content imported into namespace. Can use content without prefix

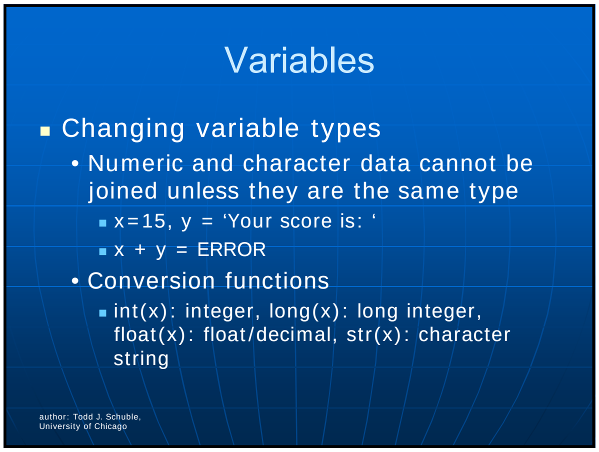
author: Todd J. Schuble,
University of Chicago



Variables

- Setting variables
  - Variables must be declared before they are used in the script
  - Informs script of the type of object and its “gender”
  - Feature classes, values, etc.
- Declaring variables
  - `x = 1` or `x = 1.0` or `x = 1.11` or `x = “GIS”`
  - The type of variable is defined by the value it is assigned (integer, decimal, string, etc.)
- Manipulating variables
  - Computation (+) (-) (*) (/)
  - Concatenation
    - `x = “GIS”,y = “class” … x +“ ”+ y=‘GIS class’`

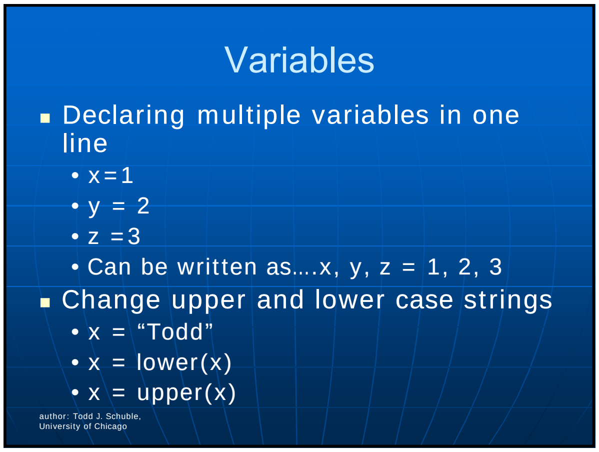
author: Todd J. Schuble,
University of Chicago



Variables

- Changing variable types
  - Numeric and character data cannot be joined unless they are the same type
    - x=15, y = ‘Your score is: ‘
    - x + y = ERROR
  - Conversion functions
    - int(x): integer, long(x): long integer, float(x): float/decimal, str(x): character string

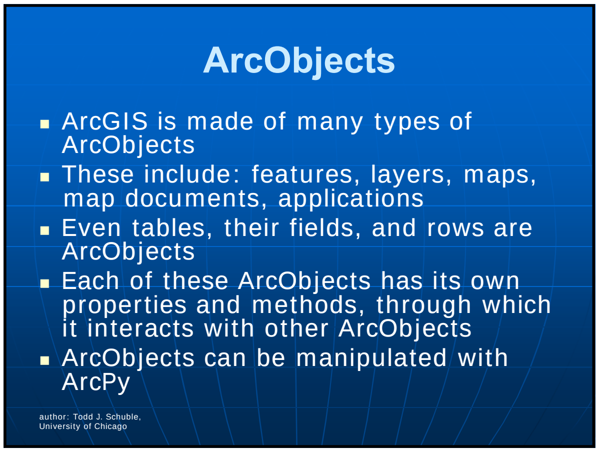
author: Todd J. Schuble,
University of Chicago



Variables

- Declaring multiple variables in one line
  - x=1
  - y = 2
  - z =3
  - Can be written as….x, y, z = 1, 2, 3
- Change upper and lower case strings
  - x = “Todd”
  - x = lower(x)
  - x = upper(x)

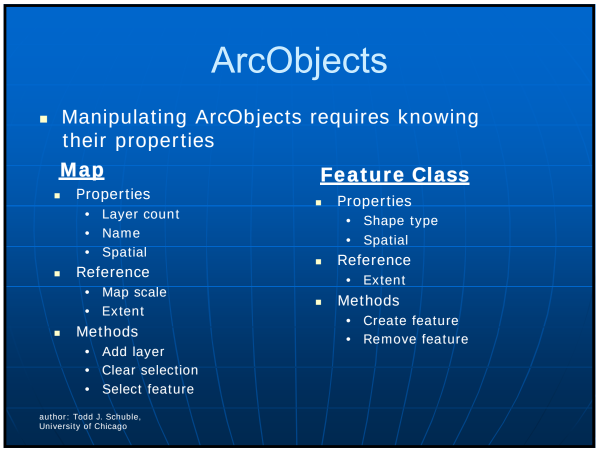
author: Todd J. Schuble,
University of Chicago



ArcObjects

- ArcGIS is made of many types of ArcObjects
- These include: features, layers, maps, map documents, applications
- Even tables, their fields, and rows are ArcObjects
- Each of these ArcObjects has its own properties and methods, through which it interacts with other ArcObjects
- ArcObjects can be manipulated with ArcPy

author: Todd J. Schuble,
University of Chicago



ArcObjects

- Manipulating ArcObjects requires knowing their properties

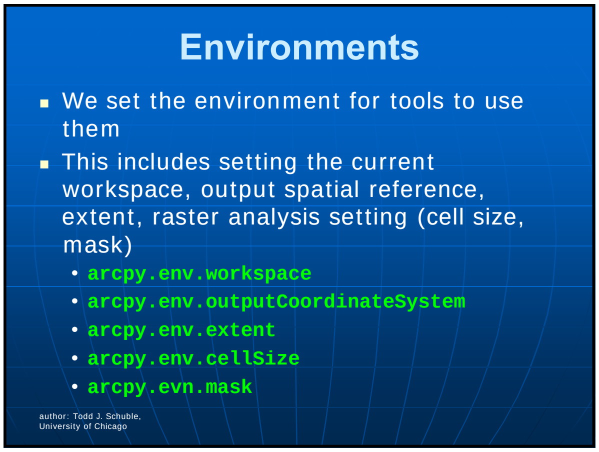
Map

- Properties
  - Layer count
  - Name
  - Spatial
- Reference
  - Map scale
  - Extent
- Methods
  - Add layer
  - Clear selection
  - Select feature

Feature Class

- Properties
  - Shape type
  - Spatial
- Reference
  - Extent
- Methods
  - Create feature
  - Remove feature

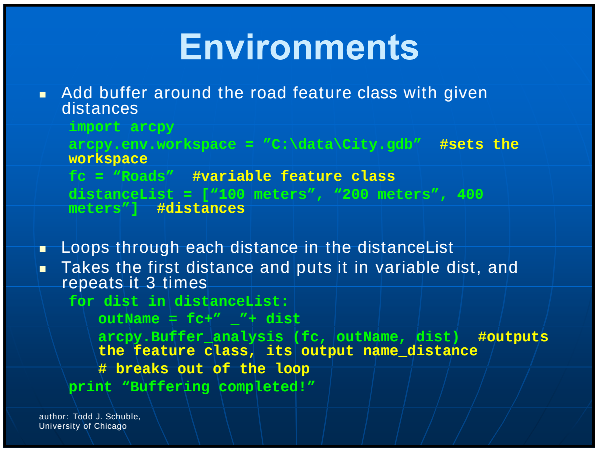
author: Todd J. Schuble,
University of Chicago



Environments

- We set the environment for tools to use them
- This includes setting the current workspace, output spatial reference, extent, raster analysis setting (cell size, mask)
  - arcpy.env.workspace
  - arcpy.env.outputCoordinateSystem
  - arcpy.env.extent
  - arcpy.env.cellSize
  - arcpy.evn.mask

author: Todd J. Schuble,
University of Chicago



Environments

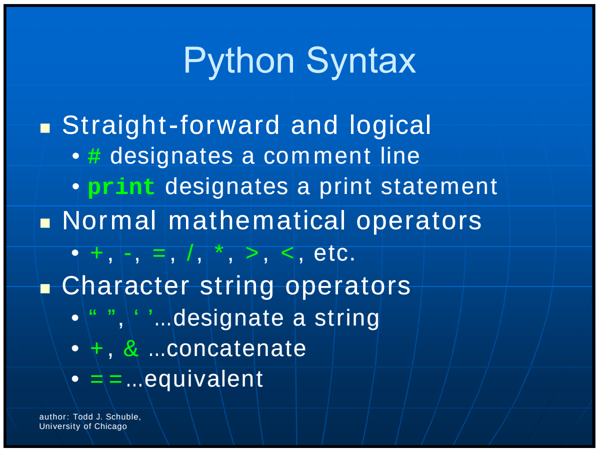
- Add buffer around the road feature class with given distances

```
import arcpy
arcpy.env.workspace = ”C:\data\City.gdb”  #sets the
workspace
fc = “Roads”  #variable feature class
distanceList = [“100 meters”, “200 meters”, 400
meters”]  #distances
```

- Loops through each distance in the distanceList
- Takes the first distance and puts it in variable dist, and repeats it 3 times

```
for dist in distanceList:
    outName = fc+” _”+ dist
    arcpy.Buffer_analysis (fc, outName, dist)  #outputs
    the feature class, its output name_distance
    # breaks out of the loop
print “Buffering completed!”
```

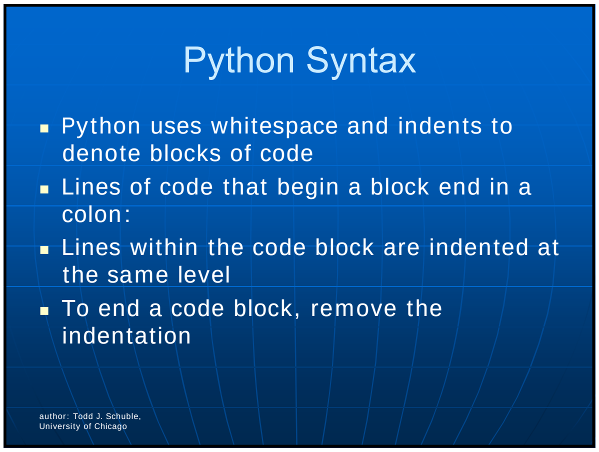
author: Todd J. Schuble,
University of Chicago



Python Syntax

- Straight-forward and logical
  - # designates a comment line
  - print designates a print statement
- Normal mathematical operators
  - +, -, =, /, *, >, <, etc.
- Character string operators
  - “ ”, ‘ ’...designate a string
  - +, & ...concatenate
  - = =...equivalent

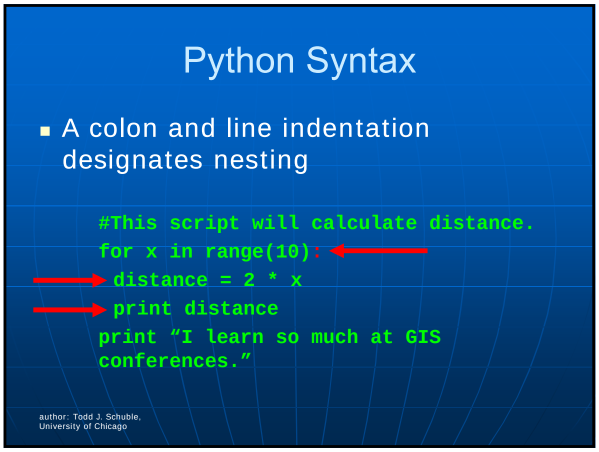
author: Todd J. Schuble,
University of Chicago



Python Syntax

- Python uses whitespace and indents to denote blocks of code
- Lines of code that begin a block end in a colon:
- Lines within the code block are indented at the same level
- To end a code block, remove the indentation

author: Todd J. Schuble,
University of Chicago

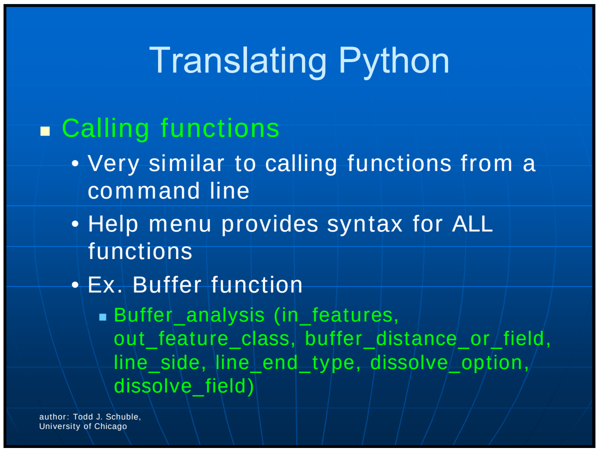


Python Syntax

- A colon and line indentation designates nesting

```
#This script will calculate distance.
for x in range(10):
  distance = 2 * x
  print distance
print “I learn so much at GIS
conferences.”
```

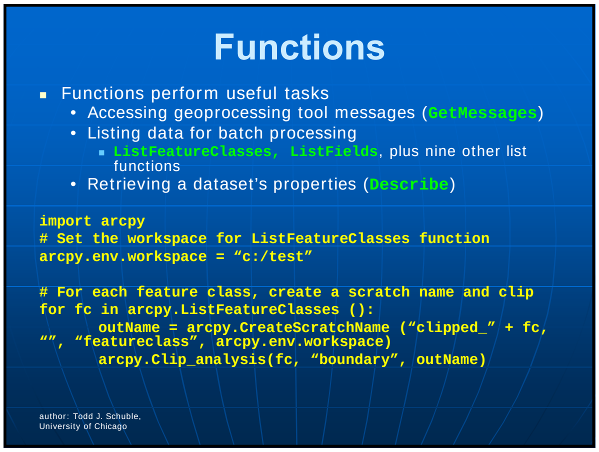
author: Todd J. Schuble,
University of Chicago



Translating Python

- Calling functions
  - Very similar to calling functions from a command line
  - Help menu provides syntax for ALL functions
  - Ex. Buffer function
    - Buffer_analysis (in_features, out_feature_class, buffer_distance_or_field, line_side, line_end_type, dissolve_option, dissolve_field)

author: Todd J. Schuble,
University of Chicago

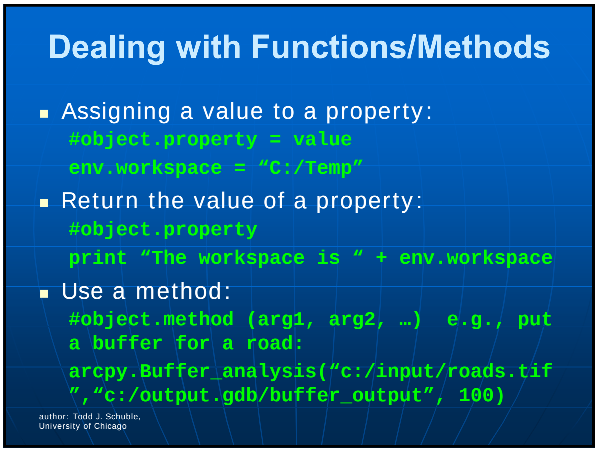


Functions

- Functions perform useful tasks
  - Accessing geoprocessing tool messages (`GetMessages`)
  - Listing data for batch processing
    - `ListFeatureClasses, ListFields`, plus nine other list functions
  - Retrieving a dataset's properties (`Describe`)

```
import arcpy
# Set the workspace for ListFeatureClasses function
arcpy.env.workspace = “c:/test”

# For each feature class, create a scratch name and clip
for fc in arcpy.ListFeatureClasses ():
        outName = arcpy.CreateScratchName (“clipped_” + fc,
“”, “featureclass”, arcpy.env.workspace)
        arcpy.Clip_analysis(fc, “boundary”, outName)
```

author: Todd J. Schuble,
University of Chicago



Dealing with Functions/Methods

- Assigning a value to a property:

```
#object.property = value
env.workspace = “C:/Temp”
```

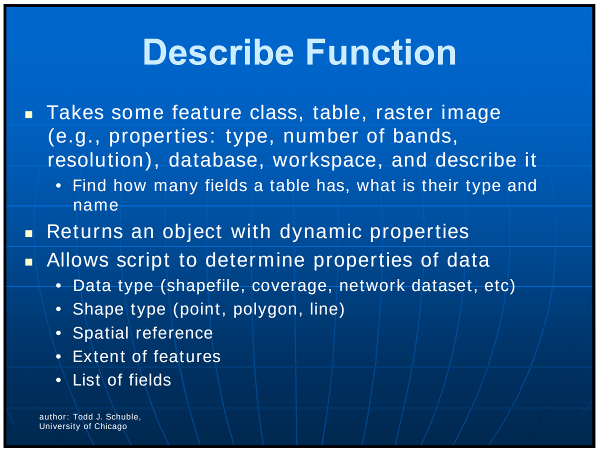
- Return the value of a property:

```
#object.property
print “The workspace is “ + env.workspace
```

- Use a method:

```
#object.method (arg1, arg2, …)  e.g., put
a buffer for a road:
arcpy.Buffer_analysis(“c:/input/roads.tif
”,“c:/output.gdb/buffer_output”, 100)
```

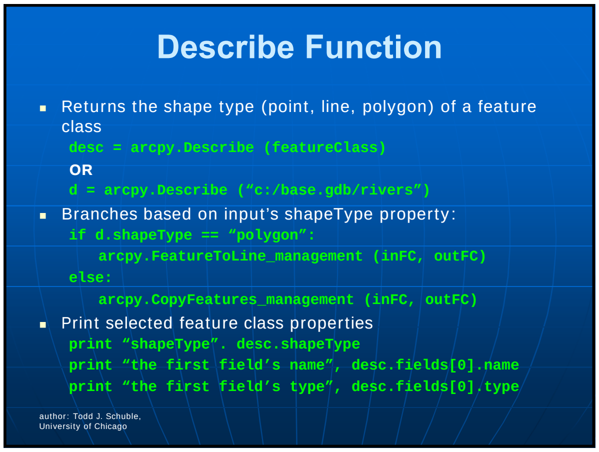
author: Todd J. Schuble,
University of Chicago



Describe Function

- Takes some feature class, table, raster image (e.g., properties: type, number of bands, resolution), database, workspace, and describe it
  - Find how many fields a table has, what is their type and name
- Returns an object with dynamic properties
- Allows script to determine properties of data
  - Data type (shapefile, coverage, network dataset, etc)
  - Shape type (point, polygon, line)
  - Spatial reference
  - Extent of features
  - List of fields

author: Todd J. Schuble,
University of Chicago



Describe Function

- Returns the shape type (point, line, polygon) of a feature class

```
desc = arcpy.Describe (featureClass)
```

OR

```
d = arcpy.Describe (“c:/base.gdb/rivers”)
```

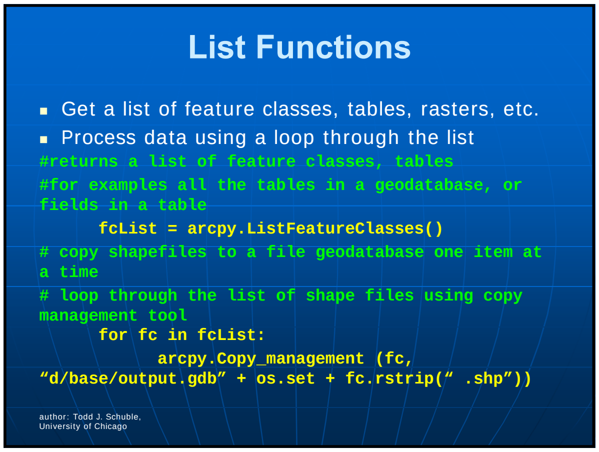
- Branches based on input’s shapeType property:

```
if d.shapeType == “polygon”:
    arcpy.FeatureToLine_management (inFC, outFC)
else:
    arcpy.CopyFeatures_management (inFC, outFC)
```

- Print selected feature class properties

```
print “shapeType”. desc.shapeType
print “the first field’s name”, desc.fields[0].name
print “the first field’s type”, desc.fields[0].type
```

author: Todd J. Schuble,
University of Chicago



List Functions

- Get a list of feature classes, tables, rasters, etc.
- Process data using a loop through the list

```
#returns a list of feature classes, tables
#for examples all the tables in a geodatabase, or
fields in a table
      fcList = arcpy.ListFeatureClasses()
# copy shapefiles to a file geodatabase one item at
a time
# loop through the list of shape files using copy
management tool
      for fc in fcList:
            arcpy.Copy_management (fc,
“d/base/output.gdb” + os.set + fc.rstrip(“ .shp”))
```

author: Todd J. Schuble,
University of Chicago

List Functions

- Adding to the List
  - var[*n*] = *object*
    - replaces *n* with *object*
  - var.append(*object*)
    - adds *object* to the end of the list
- Removing from the List
  - var[*n*] = []
    - empties contents of card, but preserves order
  - var.remove(*n*)
    - removes card at *n*
  - var.pop(*n*)
    - removes *n* and returns its value

author: Todd J. Schuble,
University of Chicago

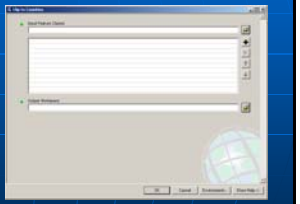
Lists in ArcToolbox

You will create lists:

- Layers as inputs
- Attributes to match
- Arrays of objects

You will work with lists:

- List of field names
- List of selected features

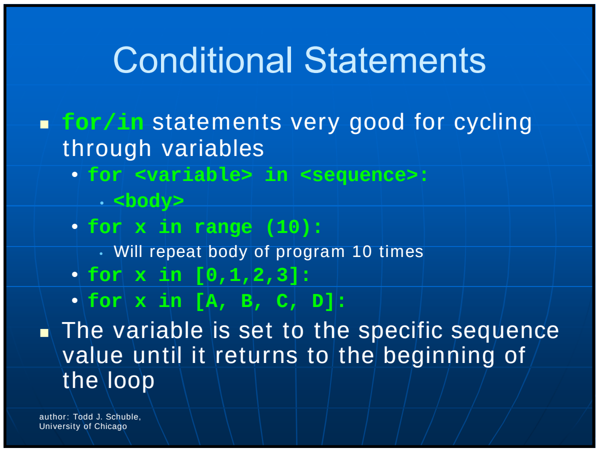


author: Todd J. Schuble,
University of Chicago

Conditional Statements

- Conditional and nested conditional statements
  - **for/in** statements
  - **if/else** statements
  - **try/except** statements
  - **while** statements

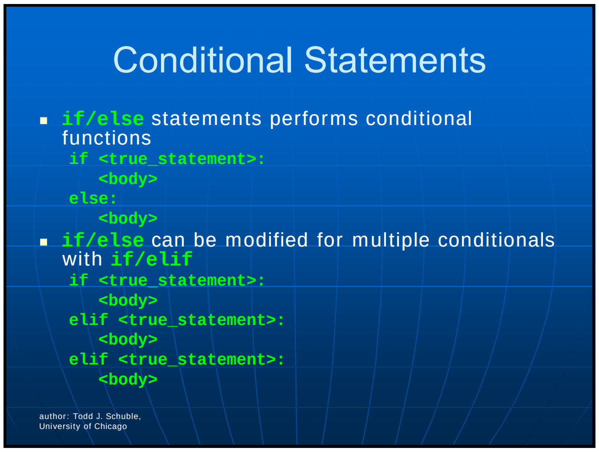
author: Todd J. Schuble,
University of Chicago



Conditional Statements

- for/in statements very good for cycling through variables
  - for <variable> in <sequence>:
    - <body>
  - for x in range (10):
    - Will repeat body of program 10 times
  - for x in [0,1,2,3]:
  - for x in [A, B, C, D]:
- The variable is set to the specific sequence value until it returns to the beginning of the loop

author: Todd J. Schuble,
University of Chicago



Conditional Statements

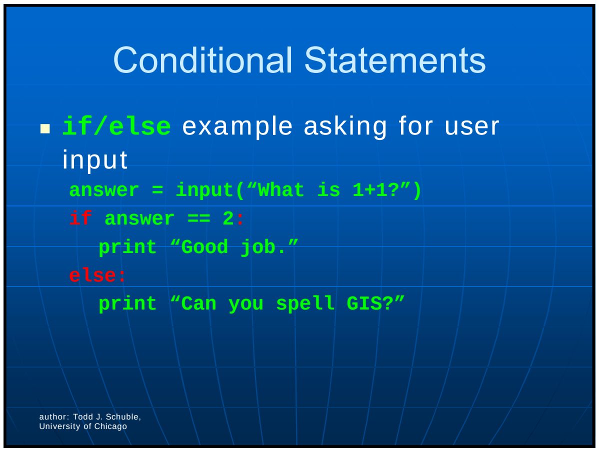
- if/else statements performs conditional functions

```
if <true_statement>:
   <body>
else:
   <body>
```

- if/else can be modified for multiple conditionals with if/elif

```
if <true_statement>:
   <body>
elif <true_statement>:
   <body>
elif <true_statement>:
   <body>
```

author: Todd J. Schuble,
University of Chicago

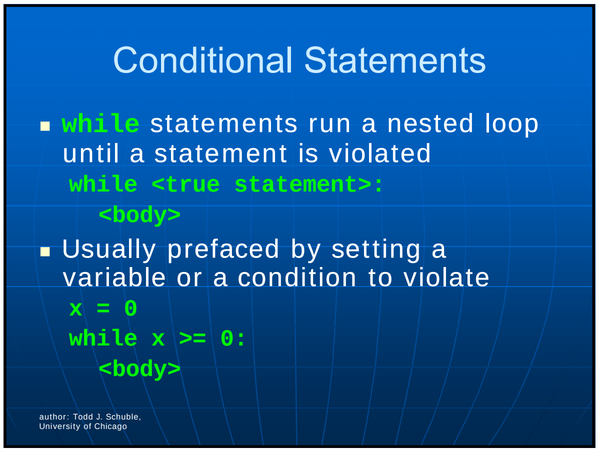


Conditional Statements

- if/else example asking for user input

```
answer = input(“What is 1+1?”)
if answer == 2:
   print “Good job.”
else:
   print “Can you spell GIS?”
```

author: Todd J. Schuble,
University of Chicago



Conditional Statements

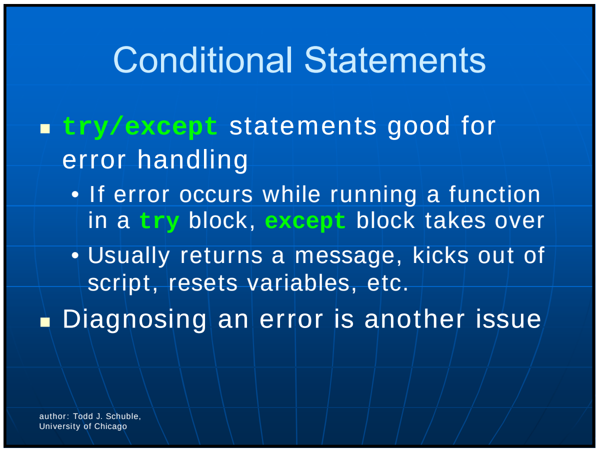
- while statements run a nested loop until a statement is violated

```
while <true statement>:
    <body>
```

- Usually prefaced by setting a variable or a condition to violate

```
x = 0
while x >= 0:
    <body>
```

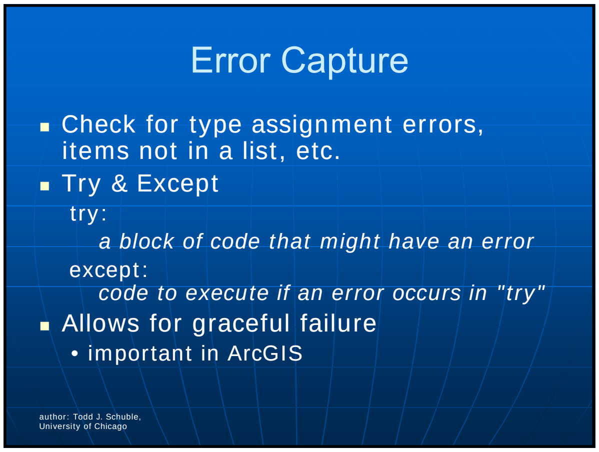
author: Todd J. Schuble, University of Chicago



Conditional Statements

- **try/except** statements good for error handling
  - If error occurs while running a function in a **try** block, **except** block takes over
  - Usually returns a message, kicks out of script, resets variables, etc.
- Diagnosing an error is another issue

author: Todd J. Schuble, University of Chicago



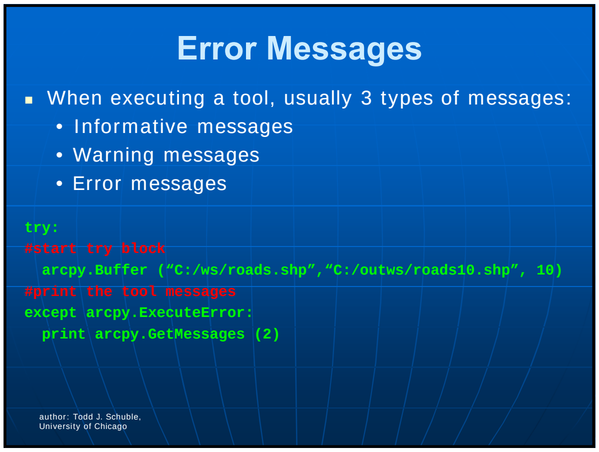
Error Capture

- Check for type assignment errors, items not in a list, etc.
- Try & Except

```
try:
    a block of code that might have an error
except:
    code to execute if an error occurs in "try"
```

- Allows for graceful failure
  - important in ArcGIS

author: Todd J. Schuble, University of Chicago

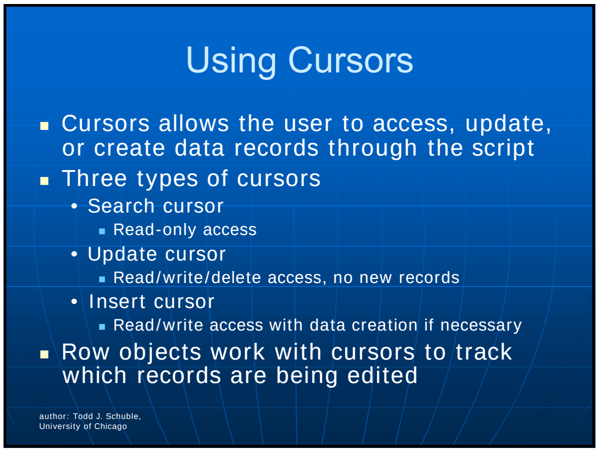


Error Messages

- When executing a tool, usually 3 types of messages:
  - Informative messages
  - Warning messages
  - Error messages

```
try:
#start try block
  arcpy.Buffer (“C:/ws/roads.shp”,“C:/outws/roads10.shp”, 10)
#print the tool messages
except arcpy.ExecuteError:
  print arcpy.GetMessages (2)
```

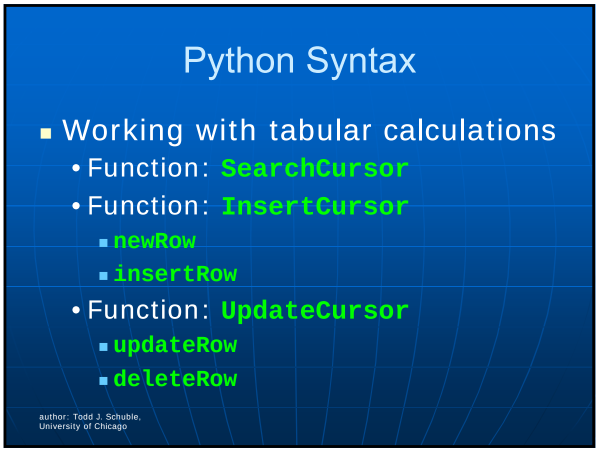
author: Todd J. Schuble,
University of Chicago



Using Cursors

- Cursors allows the user to access, update, or create data records through the script
- Three types of cursors
  - Search cursor
    - Read-only access
  - Update cursor
    - Read/write/delete access, no new records
  - Insert cursor
    - Read/write access with data creation if necessary
- Row objects work with cursors to track which records are being edited

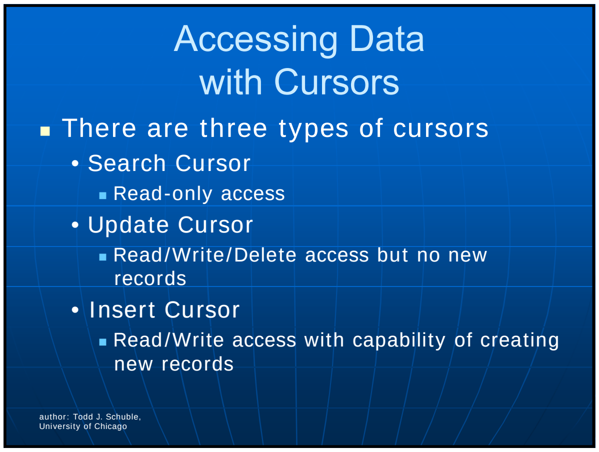
author: Todd J. Schuble,
University of Chicago



Python Syntax

- Working with tabular calculations
  - Function: `SearchCursor`
  - Function: `InsertCursor`
    - `newRow`
    - `insertRow`
  - Function: `UpdateCursor`
    - `updateRow`
    - `deleteRow`

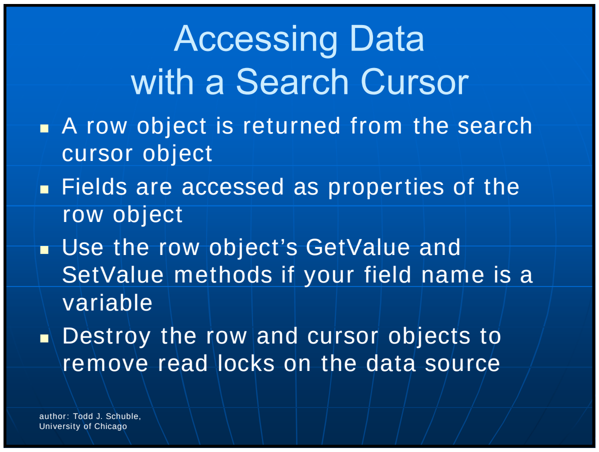
author: Todd J. Schuble,
University of Chicago



Accessing Data with Cursors

- There are three types of cursors
  - Search Cursor
    - Read-only access
  - Update Cursor
    - Read/Write/Delete access but no new records
  - Insert Cursor
    - Read/Write access with capability of creating new records

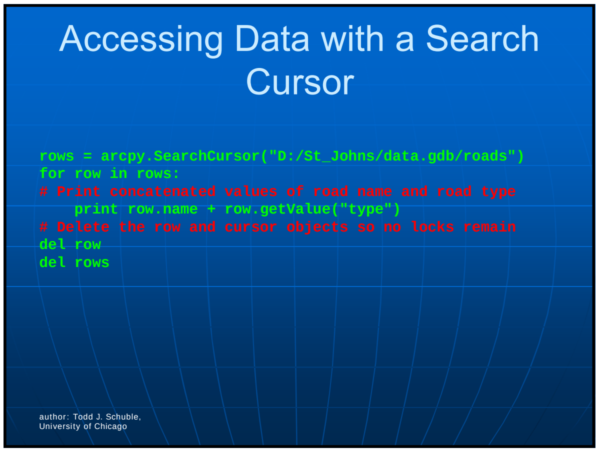
author: Todd J. Schuble,
University of Chicago



Accessing Data with a Search Cursor

- A row object is returned from the search cursor object
- Fields are accessed as properties of the row object
- Use the row object’s GetValue and SetValue methods if your field name is a variable
- Destroy the row and cursor objects to remove read locks on the data source

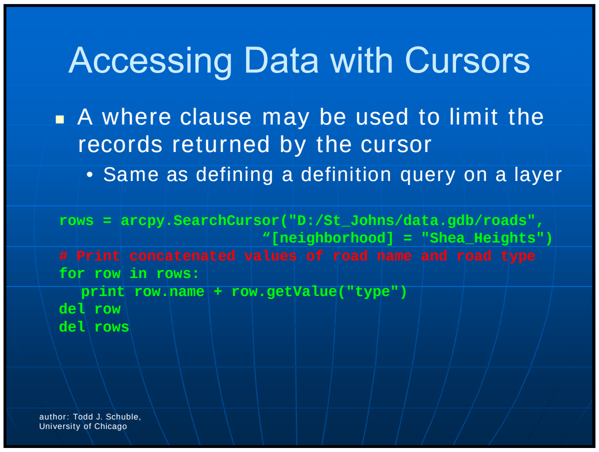
author: Todd J. Schuble,
University of Chicago



Accessing Data with a Search Cursor

```
rows = arcpy.SearchCursor("D:/St_Johns/data.gdb/roads")
for row in rows:
# Print concatenated values of road name and road type
    print row.name + row.getValue("type")
# Delete the row and cursor objects so no locks remain
del row
del rows
```

author: Todd J. Schuble,
University of Chicago

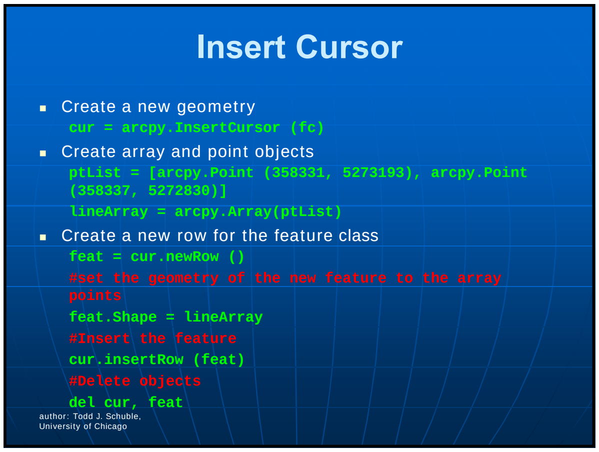


Accessing Data with Cursors

- A where clause may be used to limit the records returned by the cursor
  - Same as defining a definition query on a layer

```
rows = arcpy.SearchCursor("D:/St_Johns/data.gdb/roads",
                       “[neighborhood] = "Shea_Heights")
# Print concatenated values of road name and road type
for row in rows:
  print row.name + row.getValue("type")
del row
del rows
```

author: Todd J. Schuble,
University of Chicago



Insert Cursor

- Create a new geometry

```
cur = arcpy.InsertCursor (fc)
```

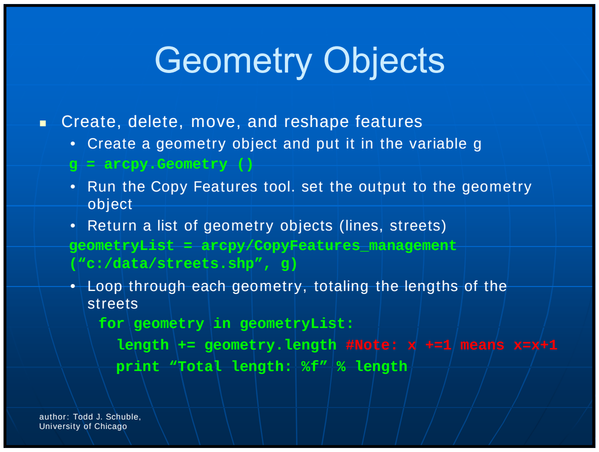
- Create array and point objects

```
ptList = [arcpy.Point (358331, 5273193), arcpy.Point
(358337, 5272830)]
lineArray = arcpy.Array(ptList)
```

- Create a new row for the feature class

```
feat = cur.newRow ()
#set the geometry of the new feature to the array
points
feat.Shape = lineArray
#Insert the feature
cur.insertRow (feat)
#Delete objects
del cur, feat
```

author: Todd J. Schuble,
University of Chicago



Geometry Objects

- Create, delete, move, and reshape features
  - Create a geometry object and put it in the variable g

```
g = arcpy.Geometry ()
```

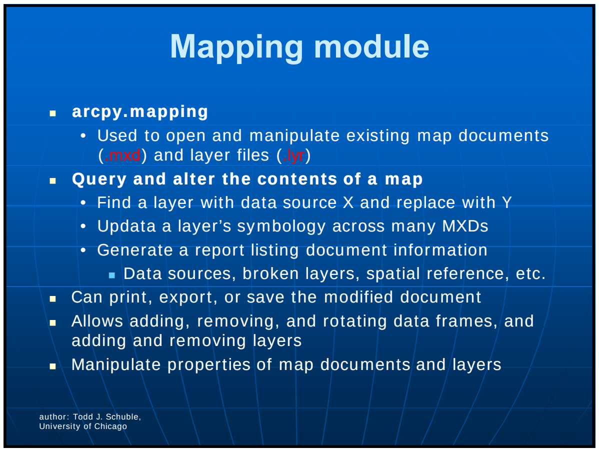
  - Run the Copy Features tool. set the output to the geometry object
  - Return a list of geometry objects (lines, streets)

```
geometryList = arcpy/CopyFeatures_management
(“c:/data/streets.shp”, g)
```

  - Loop through each geometry, totaling the lengths of the streets

```
   for geometry in geometryList:
     length += geometry.length #Note: x +=1 means x=x+1
     print “Total length: %f” % length
```

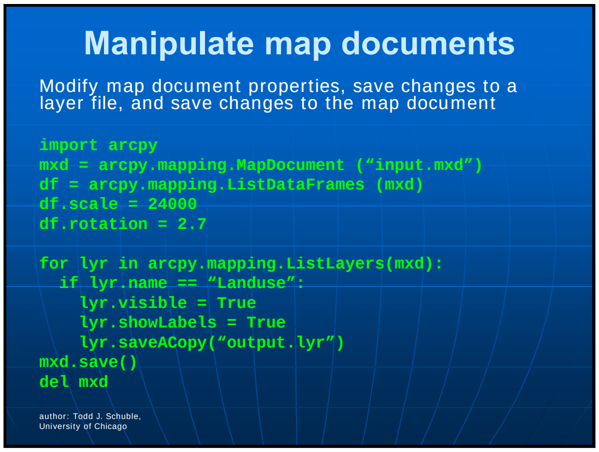
author: Todd J. Schuble,
University of Chicago



Mapping module

- **arcpy.mapping**
  - Used to open and manipulate existing map documents (.mxd) and layer files (.lyr)
- **Query and alter the contents of a map**
  - Find a layer with data source X and replace with Y
  - Updata a layer's symbology across many MXDs
  - Generate a report listing document information
    - Data sources, broken layers, spatial reference, etc.
- Can print, export, or save the modified document
- Allows adding, removing, and rotating data frames, and adding and removing layers
- Manipulate properties of map documents and layers

author: Todd J. Schuble,
University of Chicago

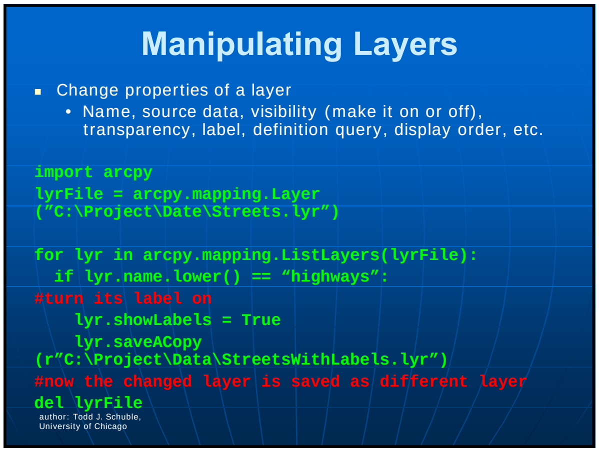


Manipulate map documents

Modify map document properties, save changes to a layer file, and save changes to the map document

```
import arcpy
mxd = arcpy.mapping.MapDocument (“input.mxd”)
df = arcpy.mapping.ListDataFrames (mxd)
df.scale = 24000
df.rotation = 2.7

for lyr in arcpy.mapping.ListLayers(mxd):
  if lyr.name == “Landuse”:
    lyr.visible = True
    lyr.showLabels = True
    lyr.saveACopy(“output.lyr”)
mxd.save()
del mxd
```

author: Todd J. Schuble,
University of Chicago



Manipulating Layers

- Change properties of a layer
  - Name, source data, visibility (make it on or off), transparency, label, definition query, display order, etc.

```
import arcpy
lyrFile = arcpy.mapping.Layer
(”C:\Project\Date\Streets.lyr”)

for lyr in arcpy.mapping.ListLayers(lyrFile):
  if lyr.name.lower() == “highways”:
#turn its label on
    lyr.showLabels = True
    lyr.saveACopy
(r”C:\Project\Data\StreetsWithLabels.lyr”)
#now the changed layer is saved as different layer
del lyrFile
```

author: Todd J. Schuble,
University of Chicago

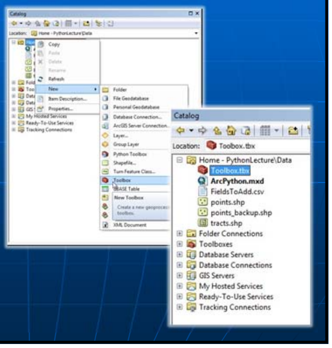
Adding a Python Script as a Tool

- Add a script as a tool to a toolbox
- They become a new tool with all the properties of a tool, e.g.,
  - It will return messages, access to all environment settings, and automatically add the output to our map (to the table of contents in ArcMap)
- Can easily be shared
- Tools automatically create dialog boxes (created by ArcGIS)
- Add the tool into the toolbar and menus

author: Todd J. Schuble,
University of Chicago

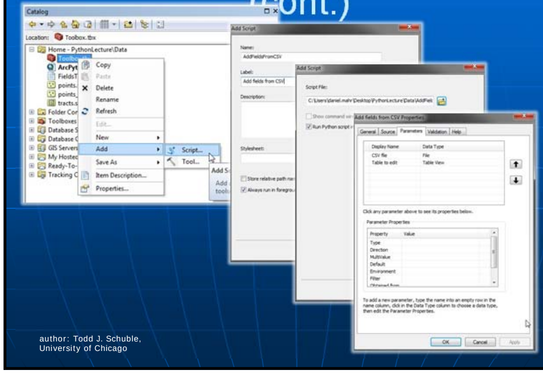
Creating a Custom Toolbox

- Empty custom toolboxes can be made in the *Catalog* pane.
- Each toolbox can contain multiple tools
- Each tool references a Python script file with the *.py* extension.



author: Todd J. Schuble,
University of Chicago

Creating a Custom Toolbox (cont.)



Name: AddFieldsFromCSV

Label: Add fields from CSV

Description:

Stylesheet:

□ Store relative path na

☑ Always run in foregrou

Script File: C:\Users\daniel.mahr\Desktop\PythonLecture\Data\AddFiel

□ Show command wi

☑ Run Python script i

Display Name	Data Type
CSV file	File
Table to edit	Table View

Click any parameter above to see its properties below.

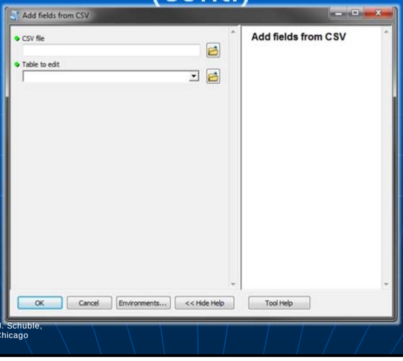
Parameter Properties

Property	Value
Type	
Direction	
MultiValue	
Default	
Environment	
Filter	
Obtained from	

To add a new parameter, type the name into an empty row in the name column, click in the Data Type column to choose a data type, then edit the Parameter Properties.

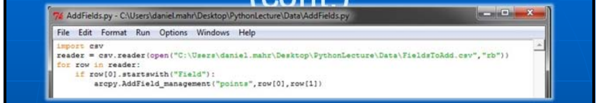
author: Todd J. Schuble,
University of Chicago

Creating a Custom Toolbox (cont.)

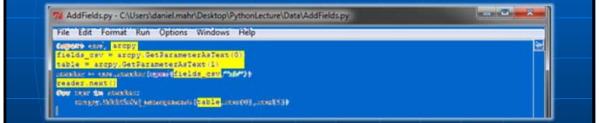


author: Todd J. Schuble, University of Chicago

Creating a Custom Toolbox (cont.)


AddFields.py - C:\Users\daniel.mahr\Desktop\PythonLecture\Data\AddFields.py
File Edit Format Run Options Windows Help

```
import csv
reader = csv.reader(open("C:\Users\daniel.mahr\Desktop\PythonLecture\Data\FieldsToAdd.csv","rb"))
for row in reader:
    if row[0].startswith("Field"):
        arcpy.AddField_management("points",row[0],row[1])
```

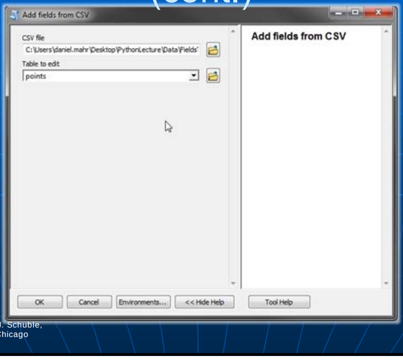

AddFields.py - C:\Users\daniel.mahr\Desktop\PythonLecture\Data\AddFields.py
File Edit Format Run Options Windows Help

```
import csv, arcpy
fields_csv = arcpy.GetParameterAsText(0)
table = arcpy.GetParameterAsText(1)
reader = csv.reader(open(fields_csv,"rb"))
reader.next()
for row in reader:
    arcpy.AddField_management(table,row[0],row[1])
```

1. Import arcpy module
2. Handle header fields more elegantly using next()
3. Use parameter values from our toolbox tool instead of hardcoded .csv file and table layer.

author: Todd J. Schuble, University of Chicago

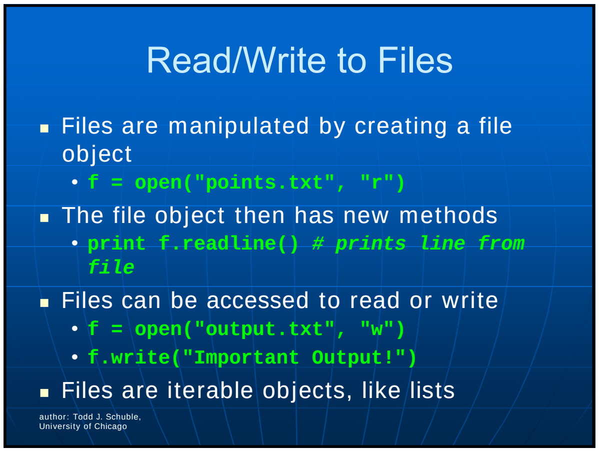
Creating a Custom Toolbox (cont.)



CSV file: C:\Users\daniel.mahr\Desktop\PythonLecture\Data\Fields'

Table to edit: points

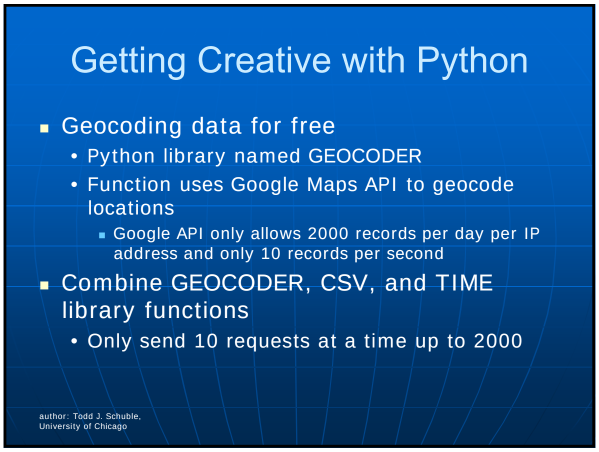
author: Todd J. Schuble, University of Chicago



Read/Write to Files

- Files are manipulated by creating a file object
  - `f = open("points.txt", "r")`
- The file object then has new methods
  - `print f.readline() # prints line from file`
- Files can be accessed to read or write
  - `f = open("output.txt", "w")`
  - `f.write("Important Output!")`
- Files are iterable objects, like lists

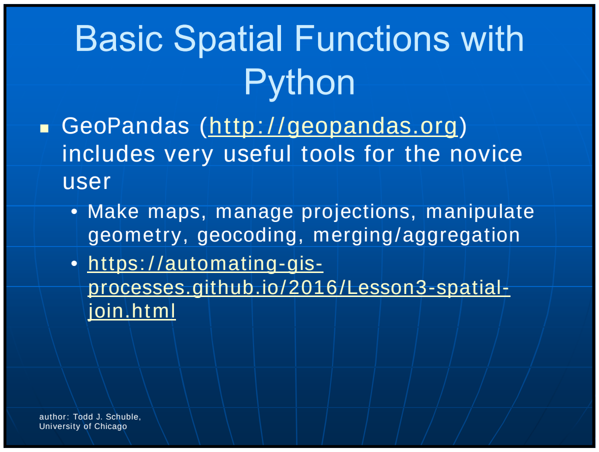
author: Todd J. Schuble,
University of Chicago



Getting Creative with Python

- Geocoding data for free
  - Python library named GEOCODER
  - Function uses Google Maps API to geocode locations
    - Google API only allows 2000 records per day per IP address and only 10 records per second
- Combine GEOCODER, CSV, and TIME library functions
  - Only send 10 requests at a time up to 2000

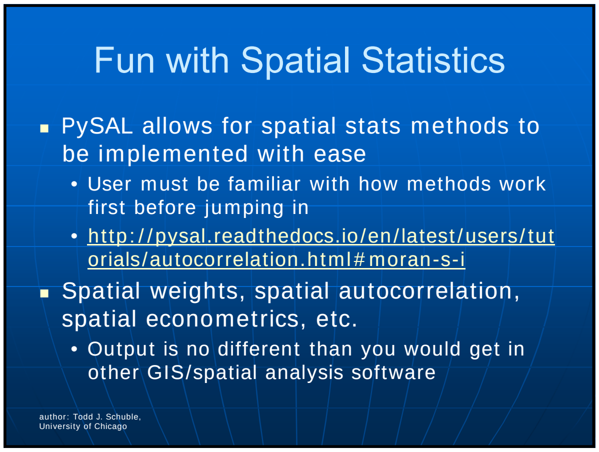
author: Todd J. Schuble,
University of Chicago



Basic Spatial Functions with Python

- GeoPandas (http://geopandas.org) includes very useful tools for the novice user
  - Make maps, manage projections, manipulate geometry, geocoding, merging/aggregation
  - https://automating-gis-processes.github.io/2016/Lesson3-spatial-join.html

author: Todd J. Schuble,
University of Chicago



Fun with Spatial Statistics

- PySAL allows for spatial stats methods to be implemented with ease
  - User must be familiar with how methods work first before jumping in
  - http://pysal.readthedocs.io/en/latest/users/tutorials/autocorrelation.html#moran-s-i
- Spatial weights, spatial autocorrelation, spatial econometrics, etc.
  - Output is no different than you would get in other GIS/spatial analysis software

author: Todd J. Schuble,
University of Chicago