Using Python for Problem Solving

Vincent A. DiNoto, Jr.
Director of GeoTech Center
Vince.dinoto@kctcs.edu

Empowering Colleges:
Expanding the
Geospatial Workforce

Based upon work supported by the National Science Foundation under Grant DUE ATE 1304591. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

Concepts

- Learning an Open Source language
- Learning a scripting language, what are the differences
- Learning scripting for specific applications in a single field of study.

GeoTech Center Model Courses

- Based on the nationally normed competencies
- Geospatial Technology Competency Model
- Meta-DACUM
- Course Content Tool

GeoTECH
GEOTECHCENTER.ORG

Model Course

- Course requirements determined by a group of national educators.
- GST 106 GIS Program was developed.

Concept

- The basic concept was learning Python for use in mapping applications and not learning Python for just general knowledge.
  - There are multiple flavors of Python and multiple mapping softwares. Therefore, a specific application software was selected with the concept that the skills could be transferred to other mapping programs.
  - The learner would have basic mapping knowledge as obtained in an introductory geospatial technology course.
  - No previous knowledge of Python or programming was required.
- Automation of tasks is the reason for learning Python and not to program extensive operations.
- Generated code would generally be short and maybe only a few lines in length.

What was developed?

- A complete course with PowerPoint Slides and audio and video.
  - The **course** is free and accessible from the GeoTech Center website http://geotechcenter.org
  - The **course** is composed of 9 modules with the last module being optional

Software

- Free
- Tailored to Esri ArcGIS Desktop (32 bit, Python 2.x)
- Should be modifiable for QGIS (32 bit, Python 2.x) and Esri ArcGIS Pro (64 bit, Python 3.x)
- Not all Python 2.x commands will run in Python 3.x, it is not forward or backward compatible at 100%.

Modules

1. Introduction
2. Python Basics
3. Using Python in an IDE with ArcPy
4. Loops and decision making operations
5. Coding in a dialog box in ArcMap
6. Create simple tools in a personal toolbox in ArcMap
7. Using Python with the Python Window of ArcMap
8. Additional Python geoprocessing commands in an IDE
9. Using Python in open software and ArcGIS Pro (optional)

Assumptions

- Course Design Assumptions:
  - **Students** will have potential done some small amounts of programming, but have little to no experience with Python.
  - **Students** may never have used a scripting language.
  - **Students** have the mathematical abilities to understand order of operations.
  - **Students** have basic experience with making of maps and the use of Esri ArcGIS Desktop.

Editor Required

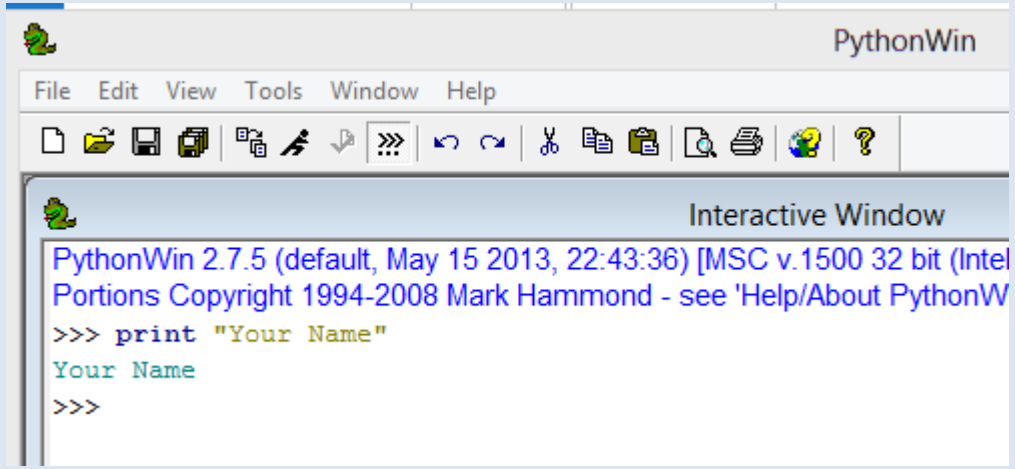
- PythonWin selected to be used for the course.
  - It is a smart Editor: Integrated Development Environment (IDE)
  - Can be run in immediate mode or scripting mode, not all commands will work in immediate mode.
- It is not suggested that Wordpad or Notepad be used.
- In general Python should work on any operating system platform. The Arcpy module must be accessed and since it is a component part of ArcGIS Desktop thus a Windows environment is required.
- QGIS has an intelligent editor as a plugin.

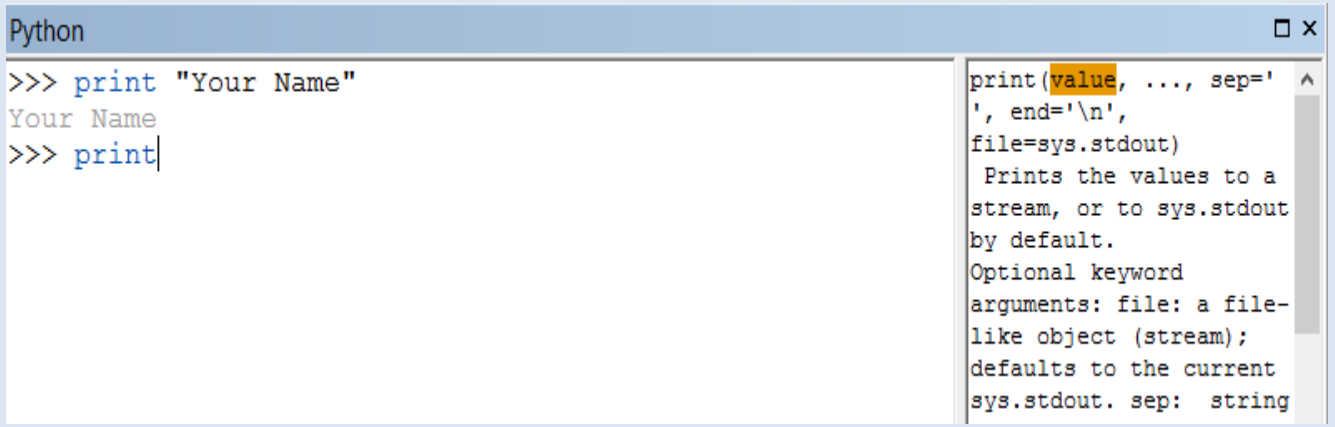
Logic and Formatting

- Basic computer programming logic and Python specific formatting is used throughout the course.
  - Variables are words that should be descriptive that hold information in numerical or text format. Variables can contain a single data item or multiple data items (a list or an array).
  - There are predefined names that cannot be used as a name variable (reserved words that are defined for specific operations).
  - Additional modules can be loaded into python, this keeps the code smaller, loading only what is needed.
  - In general the first letter of all program lines **should be** lower case
  - Anything contained within double quotation marks (“”) or single quotation marks (‘ ‘) is text (string)
  - A pound sign (#) is used for comments, it is **strongly** suggested that comments be used throughout the program to explain what is being done, this is required in the course. In addition when trying to locate errors a # sign van be placed in front of an operational line, thus making it a non-operational line. Comments can be placed at the end of an operational lines, to comment about the specific line but it is recommend that comments be placed on their own line. Comments can also be used to separate parts of a script.

Getting Started

- Use an IDE
  - Work in immediate mode
  - print “Your Name”
  - Result is Your Name
- ArcGIS Python Window
  - The Python Window should be located on the toolbar near to the the ArcCatalog and Toolbox buttons.
  - Note that assistance is shown in the right window, as you start typing a list of commands will be visible.





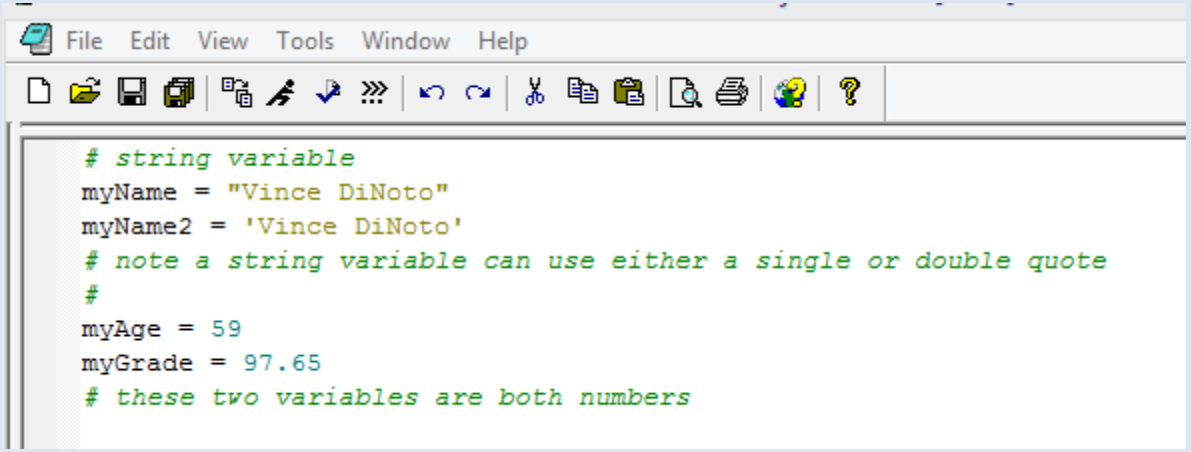
Purpose

- Many times we run the same operation on multiple data sets which causes the user to do:
  - Repetitive operations
  - Manually make decisions based upon the data
  - Create maps that can be less functional because we use a one size fits all approach.
- In the use of a Python script:
  - Learn how to automate multiple functions into a single script,
  - Create solutions based upon the data,
  - Use decision making routines on how a field is displayed and/or calculated,
  - Create specialized tools for your own operational needs.
- In the course Python scripting will be introduced and the use of geoprocessing tools within the code, but is not meant to make the user an expert Python programmer. No single course can do that, but will provide the user the expertise to use Python Scripts to make redundant processes simpler and quicker.

Important Concept

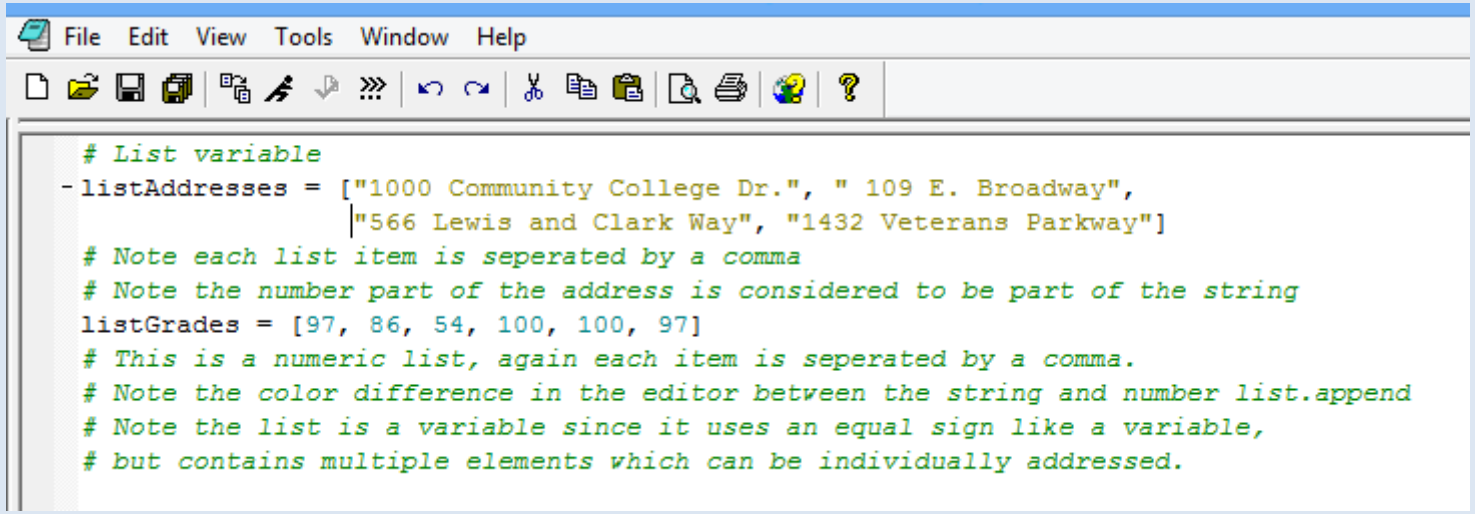
- Command names are case sensitive and must be used appropriate.
- Variable names are also case sensitive. The variable name countyName and countyname are not the same variable.
- Formatting of the script is critical to the correct operation, this includes the indention of lines.
- Immediate mode means that when an enter (return) is pressed on the keyboard, the line executes the command, the exception would be for some specific functions such as a loop.

Variable Example



- Note the comment statement has been used to explain the different variable types.
- A single or double quote can be used to define a string, therefore the variable myName and myName2 do the same operation.
- Note the PythonWin IDE shows numbers and strings in different colors to make it easier to understand and edit (troubleshoot), as well as the comments. The colors are not part of the script and varies with different IDEs.

List Example

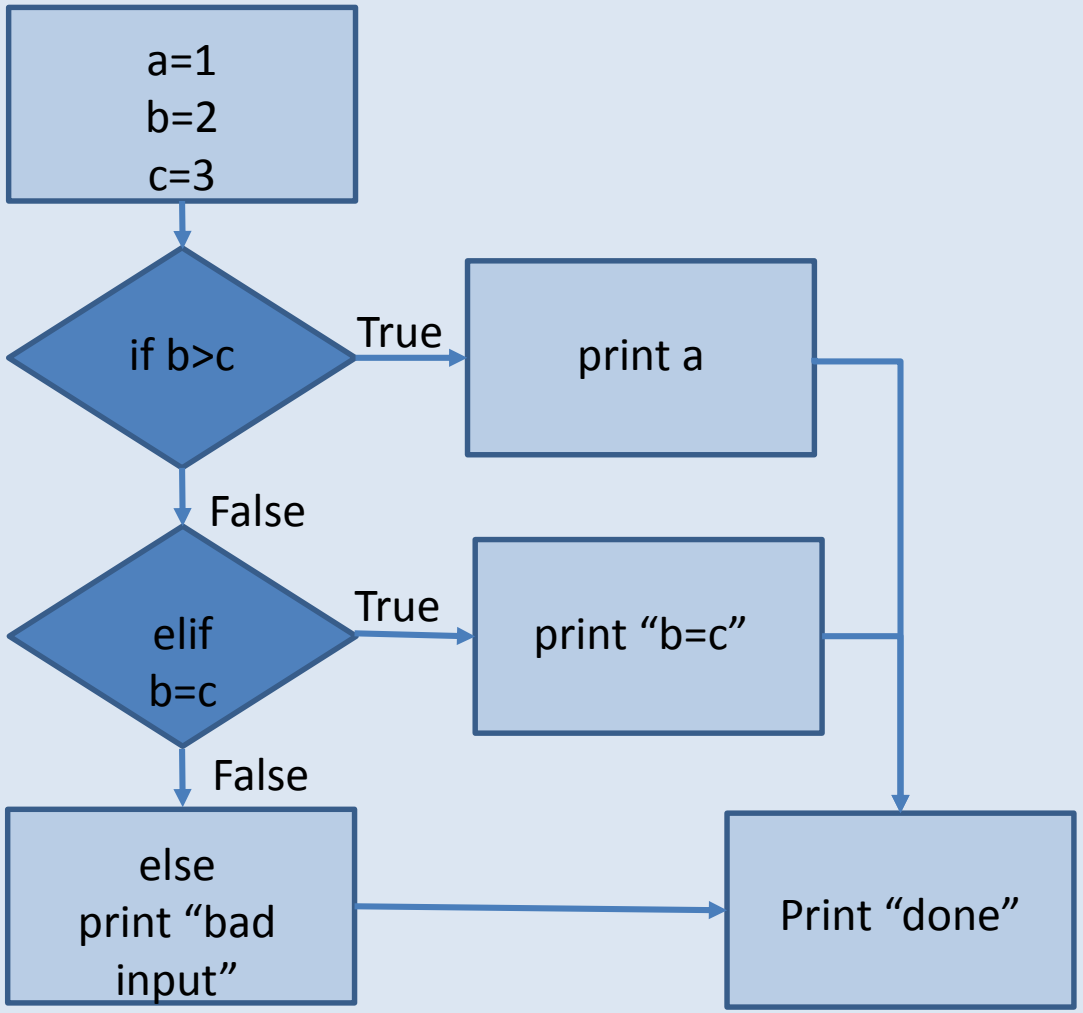


Note if there is an error in coding (it will be in red), the line of code cannot be edited directly since we are working in immediate mode. Go to the line with the error press return, the code will be entered on the next new line, then edit it and press enter. The reason that this method is used is because in immediate mode each line is executed with the return.

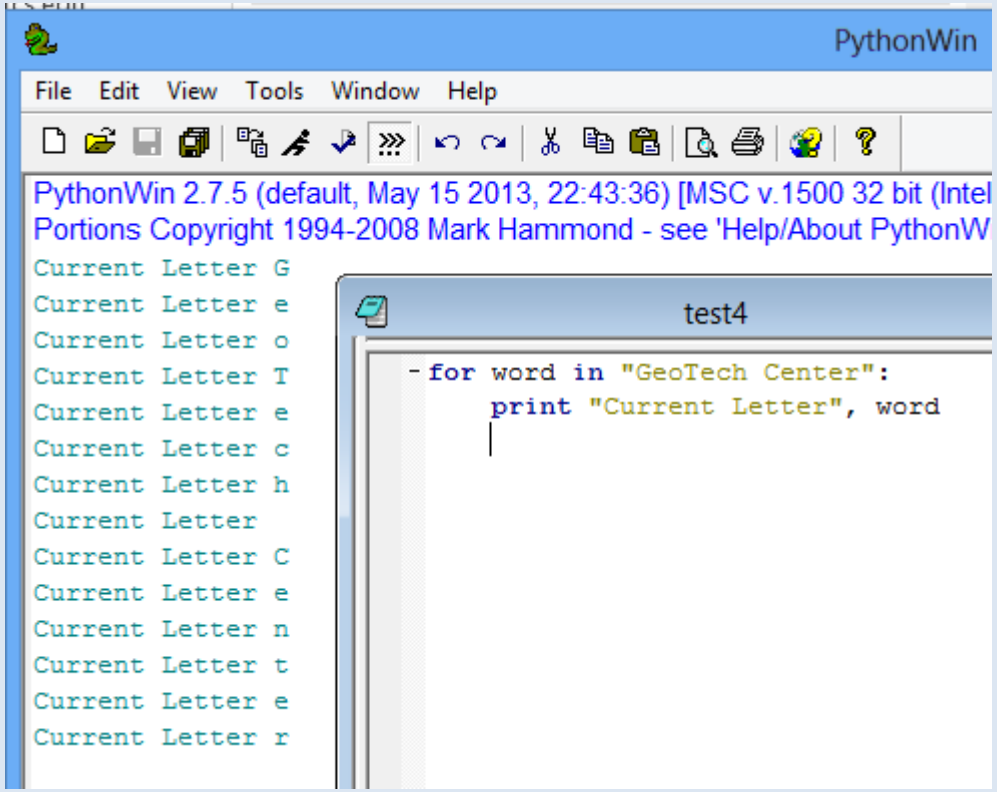
if /elif / else

```
# define variables
a = 1
b = 2
c = 3
#if statement
if b > c:
    print a
#else if statement
elif b == c:
    print "b=c"
#Else statement
else:
    print "bad input"
#end of program
print "Done"
```

- The flow is if b is great than c print a
- if not see if b=c if true print b=c
- if not print bad input.
- No matter which path it takes then done is printed next.
- elif stands for else if
- By varying the values of a, b and c, different paths are followed.



for statement



- The front window shows the code and the back window, shows the result of running the code.
- The for statement, use a variable ‘word’ so the loop is done the number of letters in GeoTech Center including the space, thus the operation runs 14 times.
- The print statement prints what is in the quotes each time and the letter that corresponds to the number of times the loop has been done.
- Each time through the loop, the value in the variable word is incremented one position.

GeoTECH
GEOTECHCENTER.ORG

Programming Mode

- A script is not executed until it told to run.
- The script must be saved
- Errors will show in the immediate mode window.
- Geoprocessing results will not be visible unless ArcMap is opened to review the results.
- To do geocprocessing loading of a module is required.

ArcPy module

- These examples will assume that an IDE is being used in the programming mode and not in the immediate mode.
- The Arcpy module contains all the geoprocessing commands that can be used in Python script creation.
- When working with the Python Window inside of ArcMap, the Arcpy module is automatically loaded and there is no need to load the module. Normally, it is suggested to load the module even within ArcMap so that your code will run both within ArcMap and from an IDE, this will not cause any issues when doing Python in ArcMap.
- The IDE will be used, the Arcpy module must be loaded before it can be used.
- The loading of the Arcpy module should be at the **beginning** of the script using: **import arcpy**.
  - Note it must be all lower case letters (Python is case sensitive), and when discussing Arcpy many times we do not use this convention.
- It is suggested when writing a script, first go to immediate mode window and import arcpy, by doing this prior to entering the program mode window, the IDE will suggest code as you type, the same way it suggests basic Python code.
- NOTE: since the IDE was installed after the installation of Esri ArcMap Desktop, the location of the Arcpy module was established, if the installation was done in the reverse direction, mapping to the module would be required.

Using a Tool from ArcMap Toolbox

- Tools in the ArcMap Toolbox
  - Formatting of a tool is:
    - arcpy.tool_location(properties)
      - tool is the name of the tool
      - location is the Toolbox location (don't forget the _ between the tool and location)
      - properties is the different parameters that the tool uses each separated by a comma
  - Example:
    - arcpy.Union_analysis(input features, output features, join attributes, tolerance, gaps)
    - The last three parameters are optional.
  - The Union tool from the toolbox, will run within the Python script assuming that Arcpy is loaded (imported).

Simple Example Script, Buffering

```
#Created by Vince DiNoto
#
#Example of creating a 25 mile buffer around a set of data points
#
#load the ArcPy module
import arcpy
#defining where original data is located
featureInput=r"C:\Users\vdinotojr0001\gis data\KY.gdb\college"
#defining where to save the buffer file
featureOutput=r"C:\Users\vdinotojr0001\gis data\KY.gdb\buffer"
#defining the radius of the buffer
radius="25 miles"
#creating buffer
arcpy.Buffer_analysis(featureInput,featureOutput,radius)
```

GeoTECH
GEOTECHCENTER.ORG

Clip Script


```
#Created by Vince DiNoto
#
#Example of using the Clipping of multiple files with a single surface
#
#load the ArcPy module
import arcpy
#load the environment
from arcpy import env
env.workspace=r"c:\Users\vdinotojr0001\gis data\KY1.gdb"
county="Trimble"
#clip
arcpy.Clip_analysis("kyrivers","KY_Trimble",county+"_rivers")
arcpy.Clip_analysis("kyrds","KY_Trimble",county+"_roads")
arcpy.Clip_analysis("Landmarks","KY_Trimble",county+"_Landmarks")
```

- In this example we will clip three shapefiles based on a single county boundary, the initial data is statewide.
- The environment is set and the arcpy module is loaded
- A variable is created called county
- The Clip tool which is in the analysis toolbox is used. Each line clips a different feature. The parameters for this tool is the input file, the clipping boundary file and the output file.
- The result of this script is the creation of three new files one for each feature.
- The naming of the new file uses the county variable along with a text component. Thus the first name would be Trimble_rivers.
- All files are stored in the KY1.gdb.

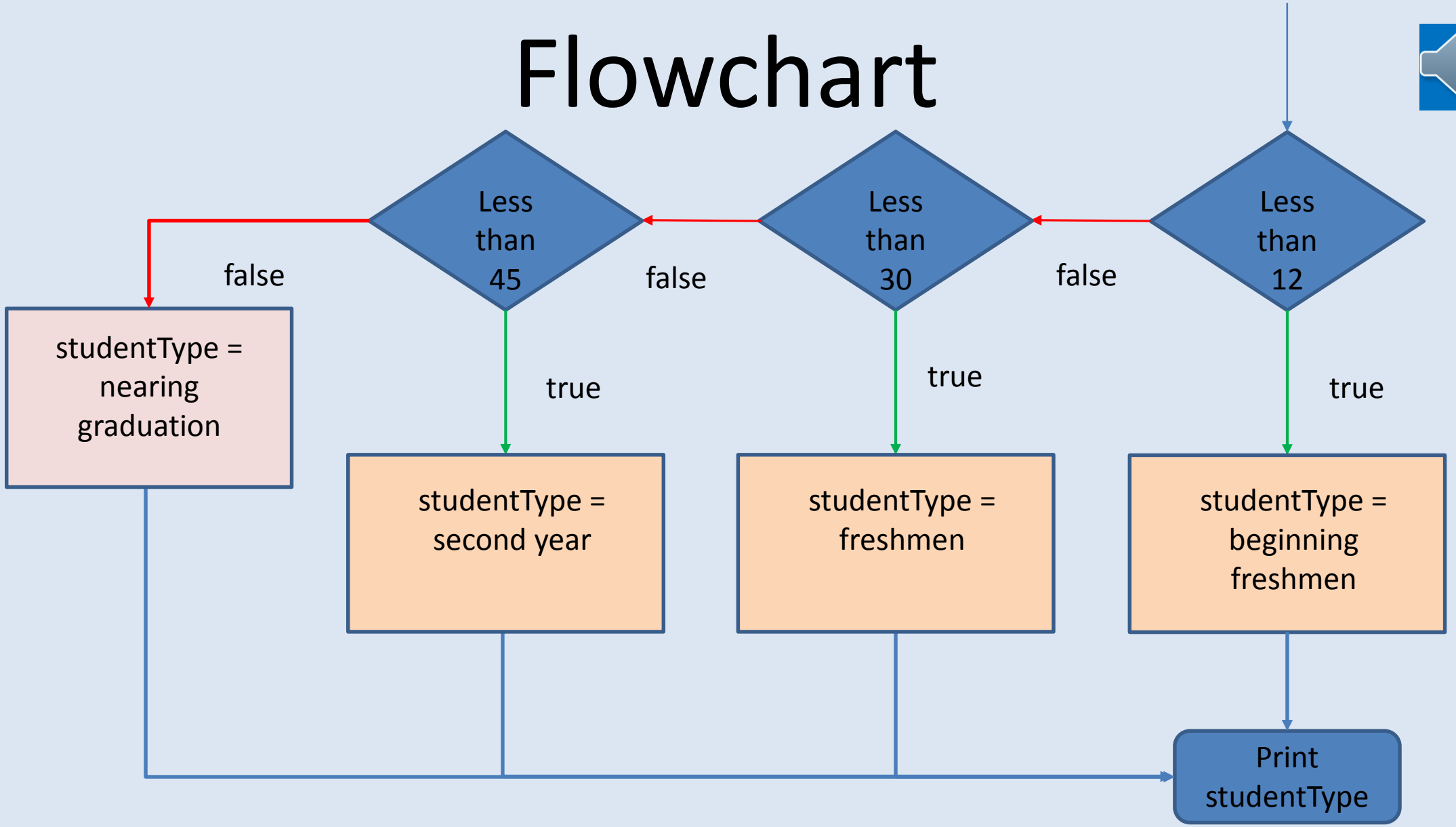
Assignment

- Your inputs:
  - a point shapefile for the United States
  - state county shapefile (polygon)
- Your outputs:
  - Create a 30 mile buffer about each data point for your state.
  - Create a single Python script file to accomplish the operation.
  - Provide the results (geodatabase), a map and the script to your instructor.
- Learning objectives:
  - Utilize the concepts of clipping and buffering within a script.

Example If/Elif/Else

- A college wants to automatically determine the status level of students that have been geocoded.
  - If the students have less than 12 credit hours, they will be listed as beginning freshmen
  - If between 12 and 29 credit hours as freshmen
  - If between 30 and 44 credit hours as second year
  - And 45 or more credit hours as nearing graduation
- Print status level

Flowchart



Code View

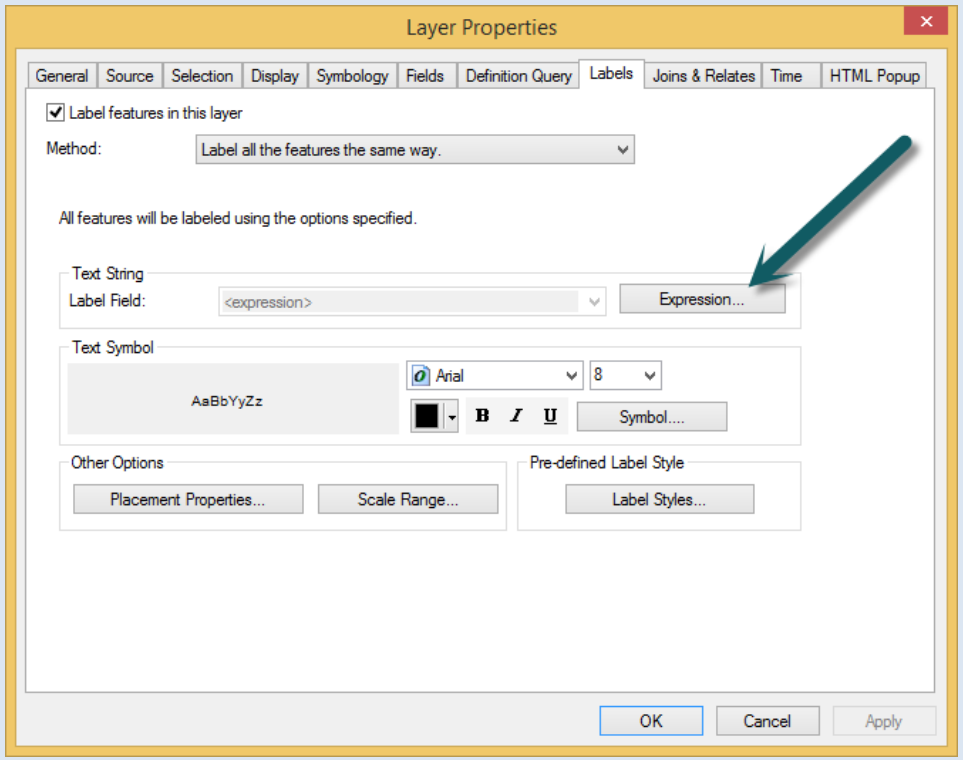
```
#Created by Vince DiNoto
#
#Example of using the if, else if and else commands from python with ar
#
#load the ArcPy module
import arcpy
#load the environment
from arcpy import env
#if not imported could by used like arcpy.env.workspace
#since imported it is
env.workspace = r"C:\Users\vdinotojr0001\gis data\KY.gdb"
#if command
if hours < 12:
    studentType="beginning freshman"
elif hours <30:
    studentType="freshman"
elif hours < 45:
    studentType="second year"
else:
    studentType="nearing graduation"
print studentType
```

- This code does no geoprocessing and an input for hours is required for successful operation (could be done in immediate mode or added into the script).
- Import arcpy module and set the work environment (not required since no geoprocessing is done).
- Variable must be defined (hours)
- The if statement checks to see if the variable hours is less than 12 hours so this would be 11 hours and lower. If true it will print “beginning freshman” and end. If it is not true it will drop to the first elif line.
- Elif (else if), we know the variable hours must be 12 or greater and if it is between 29 and 12 it will be true and print “freshman” if it is not true it will drop to the next elif line.
- Elif (2), we know the variable hours must be 30 or greater. If it is between 30 and 44 then it will print “second year” if it is false it will drop to the else line.
- Else completes the statement setting the variable studentType to “nearing graduation” and then exits the loop and the variable studentType is printed.

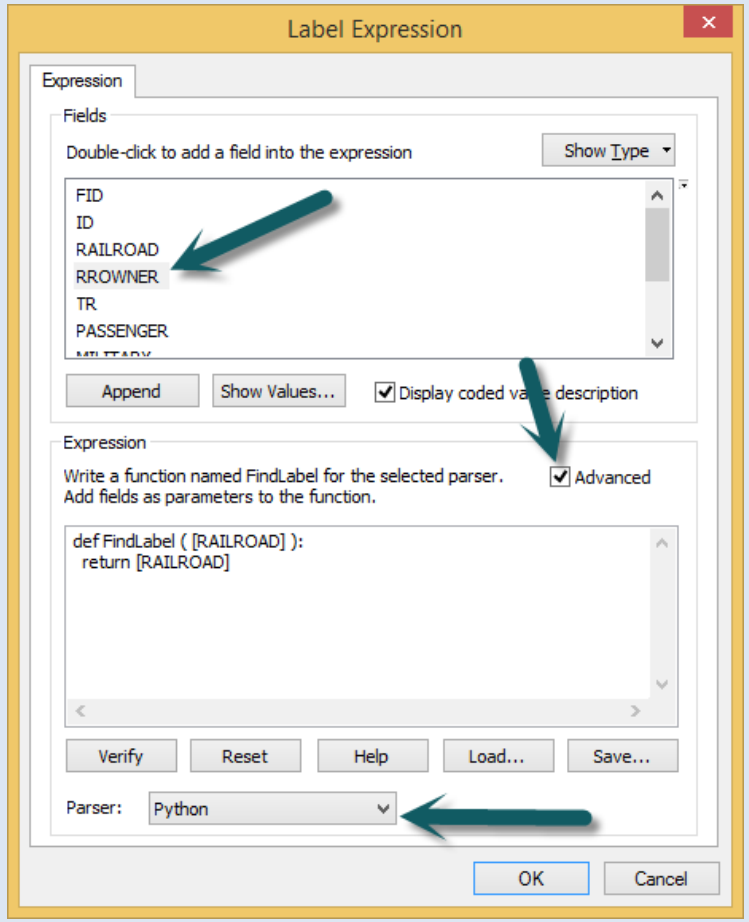
From within ArcMap

- Scripts can be used in selected functions inside of ArcMap.
- There is also a dumb editor inside of ArcMap that can be used in an immediate type mode.
- Code can also be used directly with some tools.

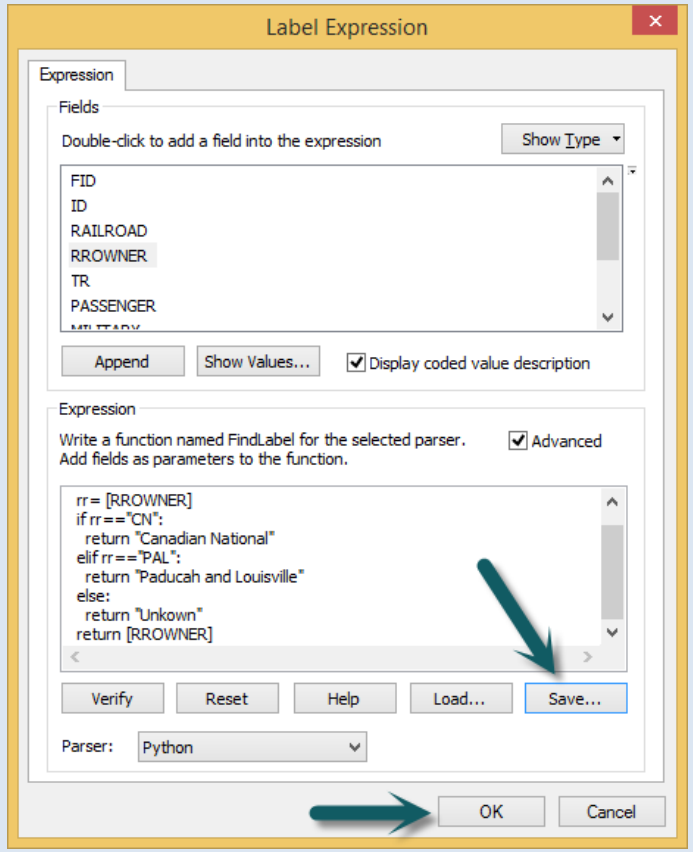
Label Properties



- Select the Expression button



- Select the field, Parser and Advanced



- You can save your code (note not all code is visible)

GeoTECH
GEOTECHCENTER.ORG

Label Script

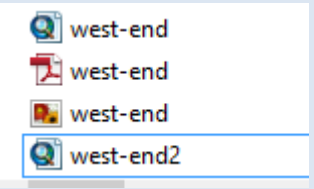
```
def FindLabel ( [RROWNER] ):
  rr= [RROWNER]
  if rr=="CN":
    return  "<CLR red='255'><FNT size = '14'>" + "Canadian National" + "</FNT></CLR>"
  elif rr=="PAL":
    return "<BOL><CLR Green='200'>"+"Paducah and Louisville"+"</CLR></BOL>"
  else:
    return "Unknown

"
  return [RROWNER]
```

- Note the changes to the two return lines
- Note the formatting
- The first return line states the Color (CLR) is red and at full intensity (255 is the maximum) we could also have used two other colors green and blue. Next we change the font (FNT) size to 14 after putting the text in we must close the FNT and CLR statements. These must be done in the reverse order from the creation.
- The next return line uses the Bold (BOL) command which has no attributes and then the CLR to set the color to Green, it is not at full intensity (less than 255), the text to be displayed is next and finally we close the statements.

Export an image

- In this example a map will be exported as a PNG image file with a resolution of 600 dpi.

```
import arcpy
project=arcpy.mapping.MapDocument(r"X:\Database\west-end.mxd")
arcpy.mapping.ExportToPNG(project,r"X:\Database\west-end.png, resolution=600")
del project
```



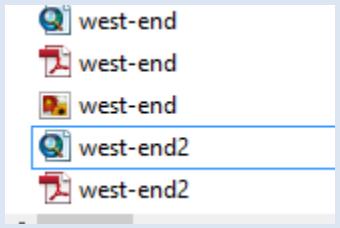
- The results of the script is displayed to the left as a file stored in the specified saving location.

- This operation was done in an IDE thus we imported the arcpy module.
- The variable project is defined to be the mxd file.
- The next command will export the project file as a png file type. This is a typical export format for web applications. Note the specified resolution of the export file is 600 dpi.
- Multiple files could be exported or the same file at various resolutions using a looping function and a list.

Exporting an Adobe Acrobat Format: Part 1

- The command structure is the same as that of the image export.

```
import arcpy
project=arcpy.mapping.MapDocument(r"X:\Database\west-end.mxd")
arcpy.mapping.ExportToPDF(project,r"X:\Database\west-end2.pdf, resolution=600")
del project
```



- Note the results to the left which shows the creation of a new pdf file, refer to the previous slide to see the directory prior to running this script.

- Note the code is the exact the same except the function routine is for PDF and not the PNG as before.
- Note we called the PDF file west-end2 because there already existed a PDF file called west-end.

- There are numerous export formats, it suggested that you look at the help menu inside of ArcMap to see some of the different image formats, as well as the different optional construction elements, such image size, etc.

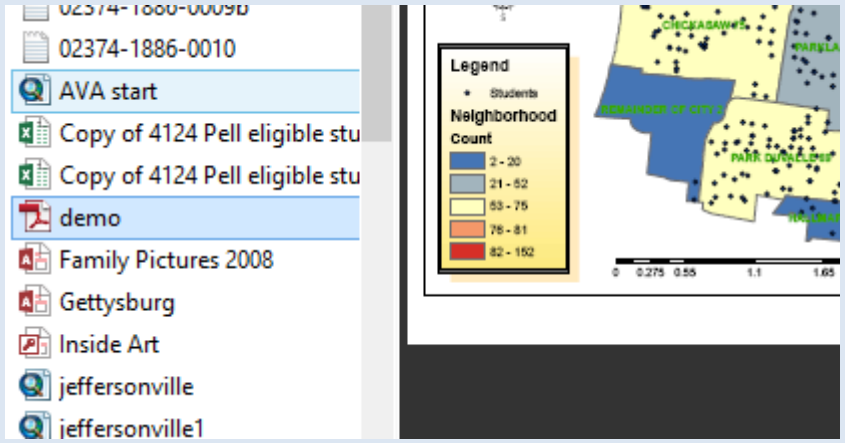
Exporting a Adobe Acrobat Format: Part 2

- One of the functions that many times we do is create multiple maps of the same region with different items displayed on each map.
  - For example: if a demographic study is being done for a regional geography Adobe Acrobat files might be wanted for the following parameters:
    - Population
    - Income
    - Education attainment
    - Race
    - Gender
    - Ethnicity
    - Housing type
    - Geography (rivers and roads)
    - Townships/towns/place names
  - Creating a package of these different maps in automated process would save time.

Creating a Map Book

- In this example a map book of a single regional geography from existing pdf files will be constructed.

```
import arcpy
path=r"X:\database\demo.pdf"
document=arcpy.mapping.PDFDocumentCreate(path)
document.appendPages (r"X:\database\west-end.pdf")
document.appendPages (r"X:\database\west-end2.pdf")
document.saveAndClose()
del document
```



- This example will be done in an IDE so arcpy is imported.
- The path of the final document is set, this could have been done using the environment statement.
- A mapping command to create an empty pdf document is done.
- Multiple pages are appended to the new document, this could and should be done in a loop using a list variable or a common naming convention and a number to locate all the documents.
- Once the appending is completed, the document is saved and closed.
- Make sure all locks are eliminated.
- The results are displayed to the left, which shows the first page in the example document, demo.

GeoTECH
GEOTECHCENTER.ORG

Vince.DiNoto@kctcs.edu

http://geotechcenter.org

Under presentations at the bottom of the page.