Python for ArcGIS

Part 1

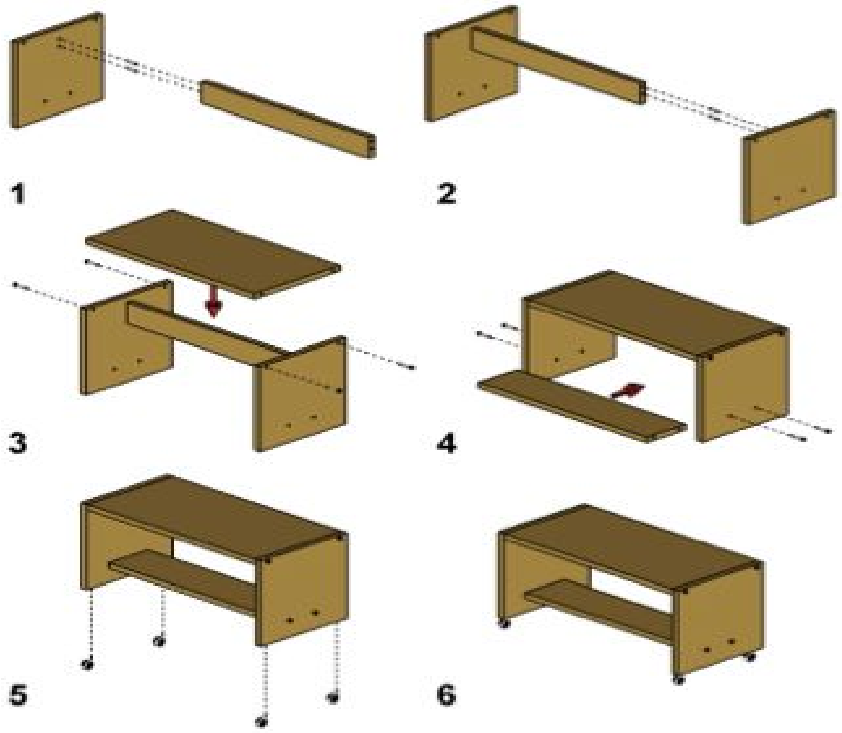
Ahmad Aburizaiza
Data Services Group - GMU Libraries

Spring 2016

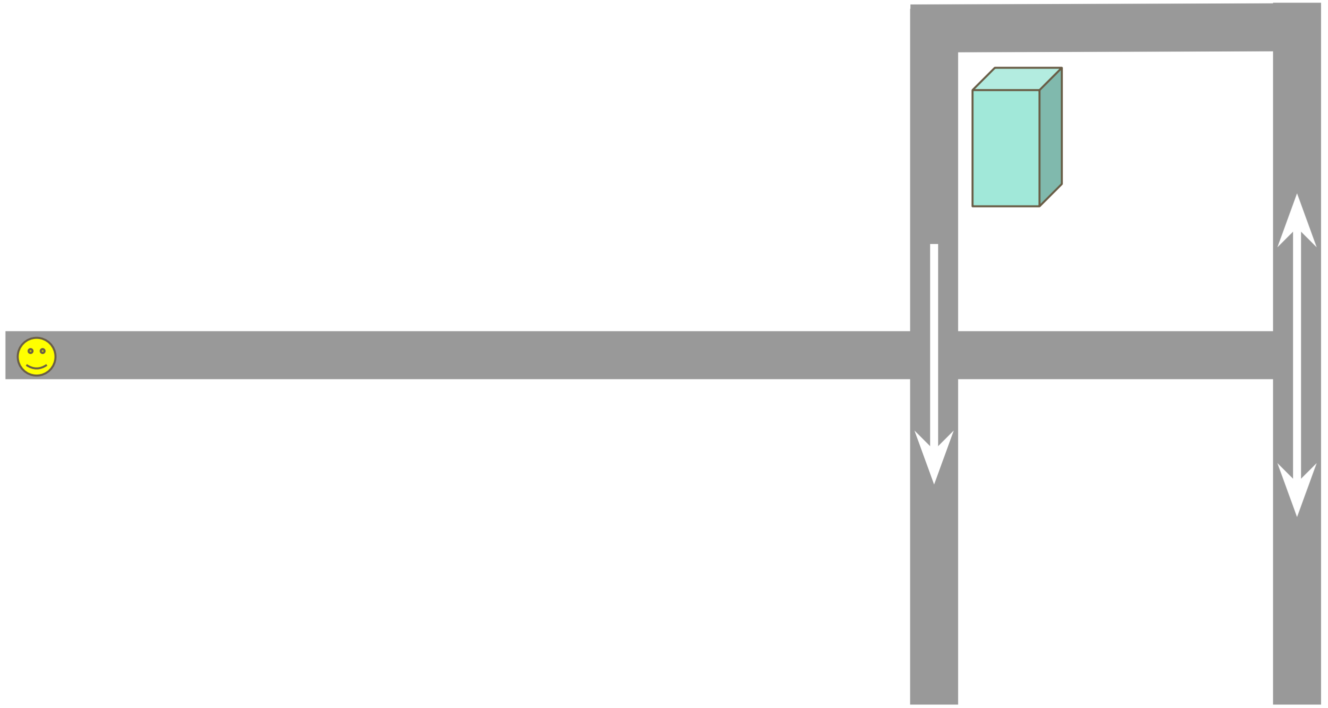
What is Programming?

Programming can be explained as set of ***ordered*** instructions for the computer to do something(s).

What is Programming?



What is Programming?



What is Programming?

Examples of simple programs running ordered instructions:

1.
    a. Add the two numbers 2 and 8
    b. Multiply the addition result by 5
2.
    a. Create a circle
    b. Assign it the red color
    c. Draw it on the computer screen
3.
    a. Open a CSV file
    b. Read x and y coordinates
    c. Create points from the read coordinates
    d. Draw the points on the map

Programming Languages

A programming language is a set of instructions and commands in a specific syntax different from other programming languages.

JavaScript

python™

Java

Swift 2.0

Ruby

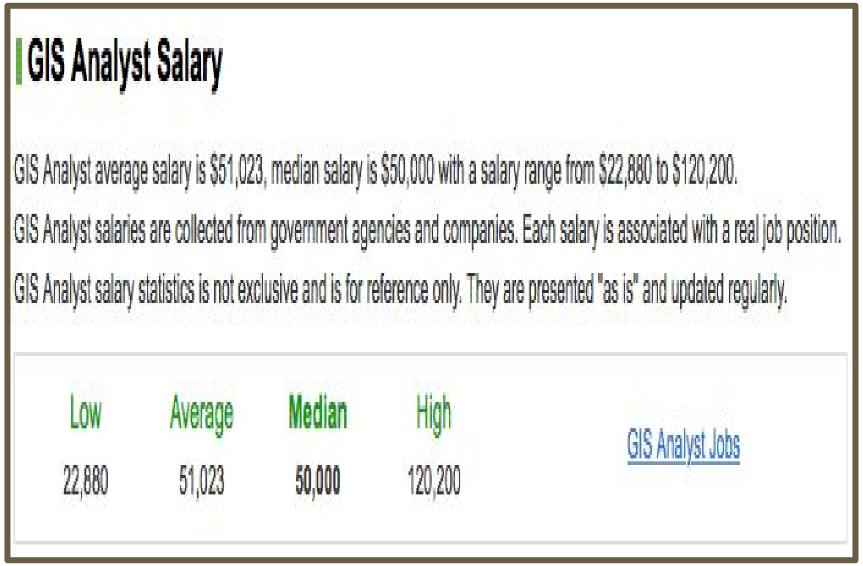
Microsoft® .NET

Programming Languages

Command	Python	JavaScript
Define a variable named x	x = 3	var x = 3;
Print x value on the screen	print(x)	alert(x);
Check if x is not equal to zero	if x <> 0:	if (x != 0)

Programming Benefits for GIS

1. Higher salaries and demand for GIS developers (desktop/web/mobile) in the GIS related market.

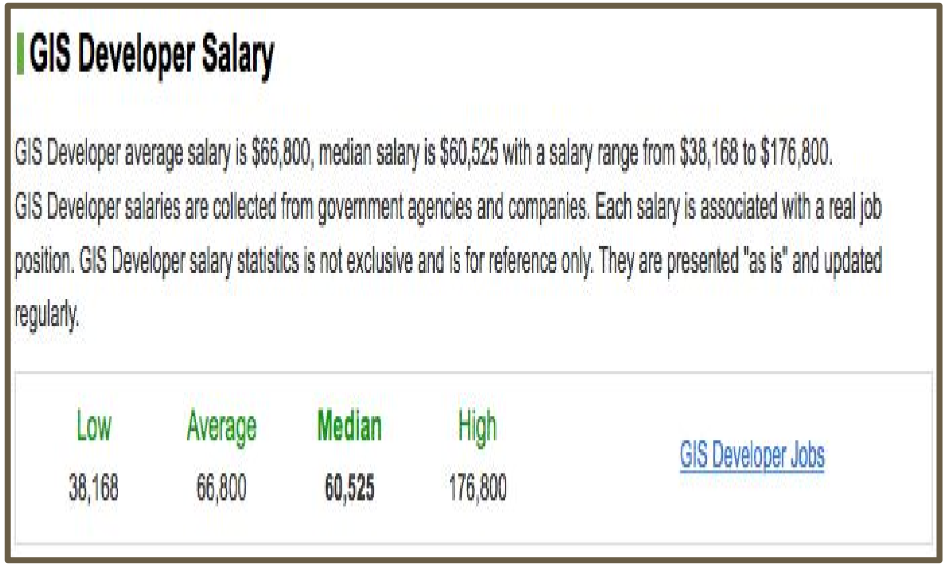


GIS Analyst Salary

GIS Analyst average salary is $51,023, median salary is $50,000 with a salary range from $22,880 to $120,200.

GIS Analyst salaries are collected from government agencies and companies. Each salary is associated with a real job position. GIS Analyst salary statistics is not exclusive and is for reference only. They are presented "as is" and updated regularly.

Low	Average	Median	High	
22,880	51,023	**50,000**	120,200	GIS Analyst Jobs



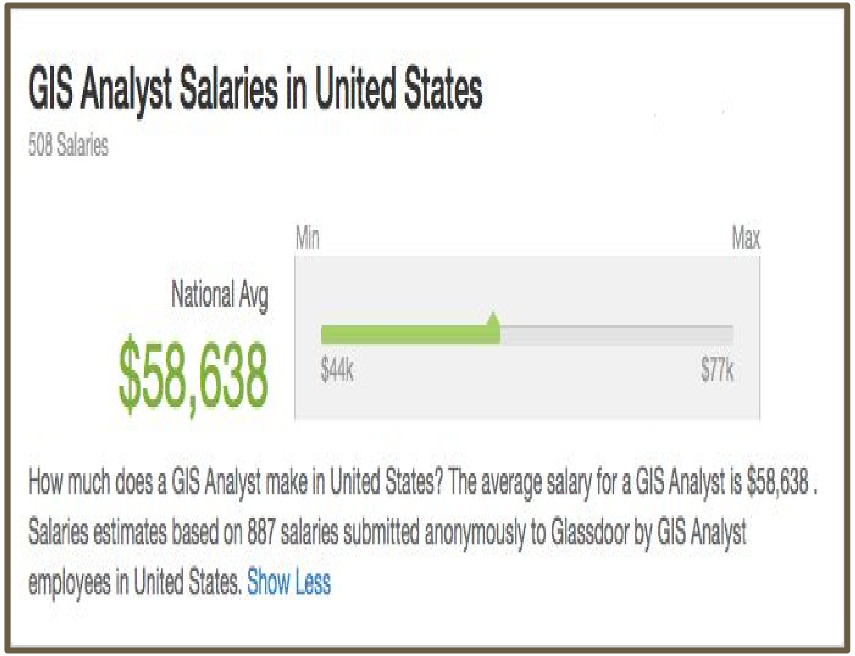
GIS Developer Salary

GIS Developer average salary is $66,800, median salary is $60,525 with a salary range from $38,168 to $176,800.

GIS Developer salaries are collected from government agencies and companies. Each salary is associated with a real job position. GIS Developer salary statistics is not exclusive and is for reference only. They are presented "as is" and updated regularly.

Low	Average	Median	High	
38,168	66,800	**60,525**	176,800	GIS Developer Jobs

Programming Benefits for GIS



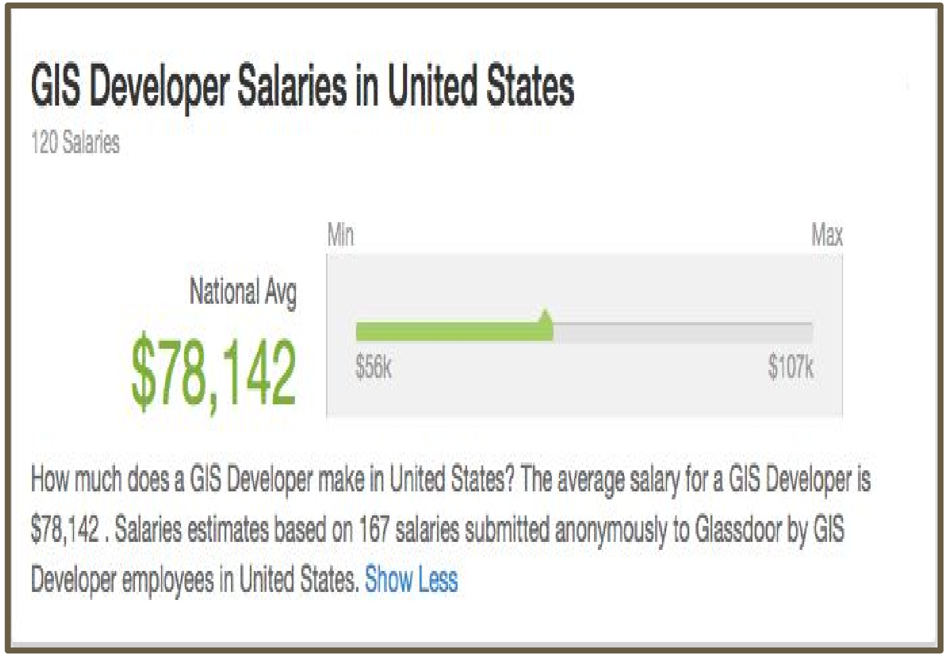
GIS Analyst Salaries in United States

508 Salaries

National Avg

$58,638

How much does a GIS Analyst make in United States? The average salary for a GIS Analyst is $58,638 . Salaries estimates based on 887 salaries submitted anonymously to Glassdoor by GIS Analyst employees in United States. Show Less



GIS Developer Salaries in United States

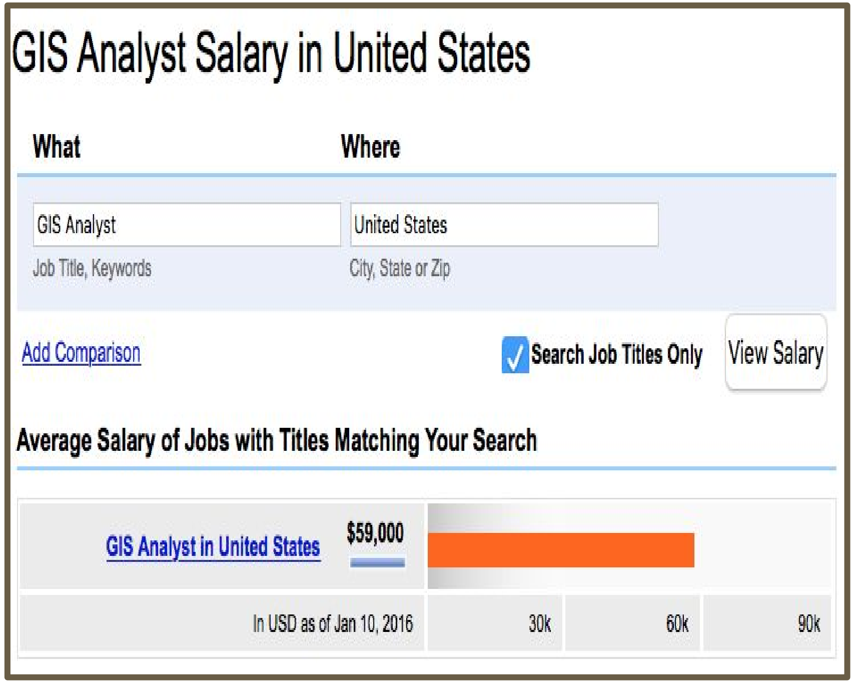
120 Salaries

National Avg

$78,142

How much does a GIS Developer make in United States? The average salary for a GIS Developer is $78,142 . Salaries estimates based on 167 salaries submitted anonymously to Glassdoor by GIS Developer employees in United States. Show Less

Programming Benefits for GIS



GIS Analyst Salary in United States

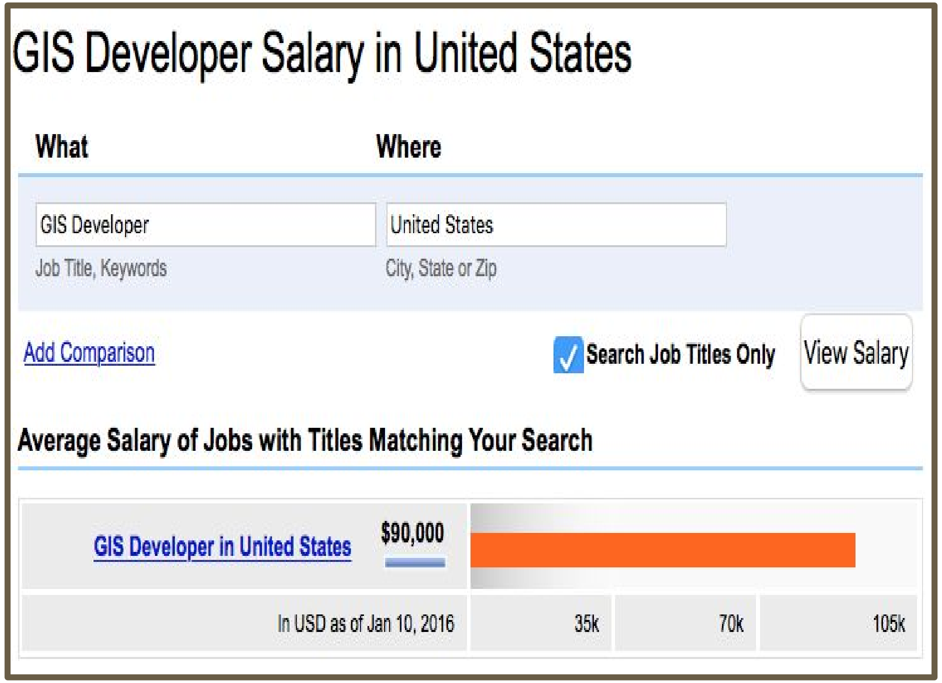
What: GIS Analyst (Job Title, Keywords)

Where: United States (City, State or Zip)

Average Salary of Jobs with Titles Matching Your Search

GIS Analyst in United States $59,000

In USD as of Jan 10, 2016



GIS Developer Salary in United States

What: GIS Developer (Job Title, Keywords)

Where: United States (City, State or Zip)

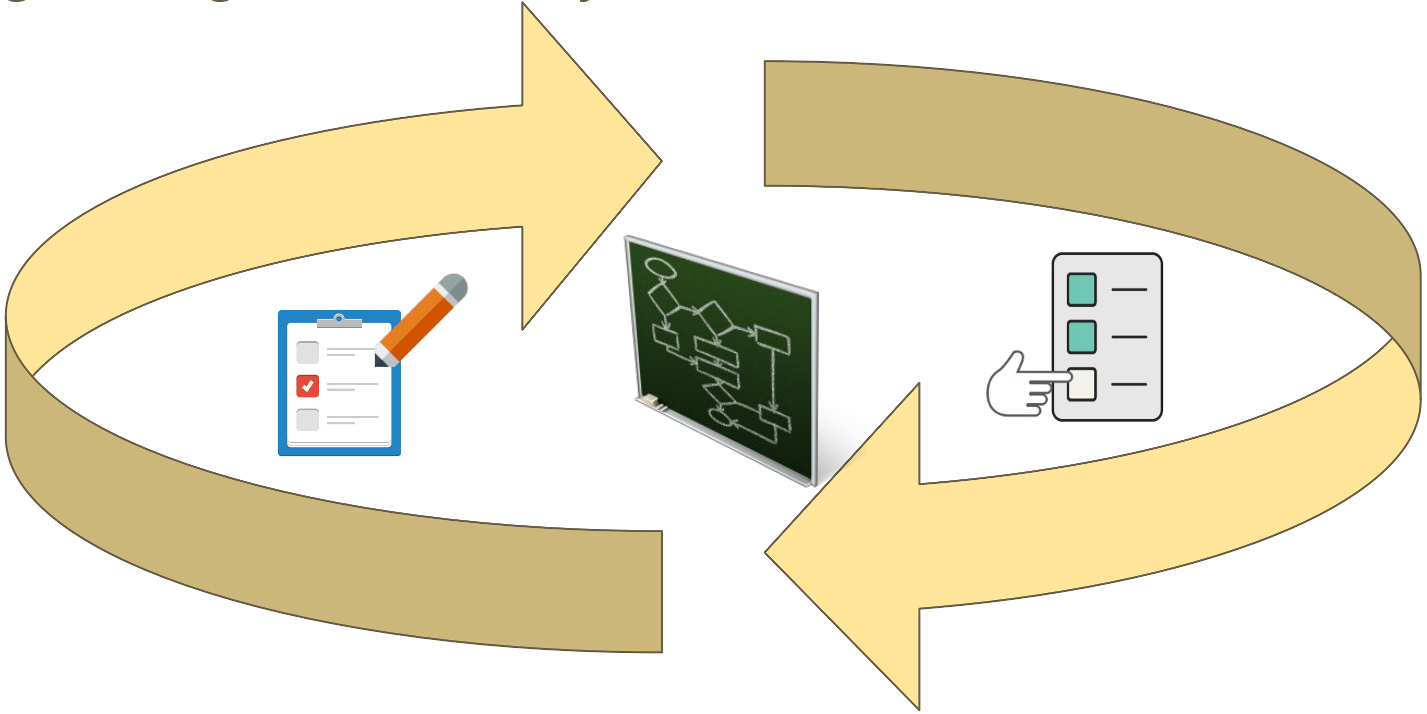
Average Salary of Jobs with Titles Matching Your Search

GIS Developer in United States $90,000

In USD as of Jan 10, 2016

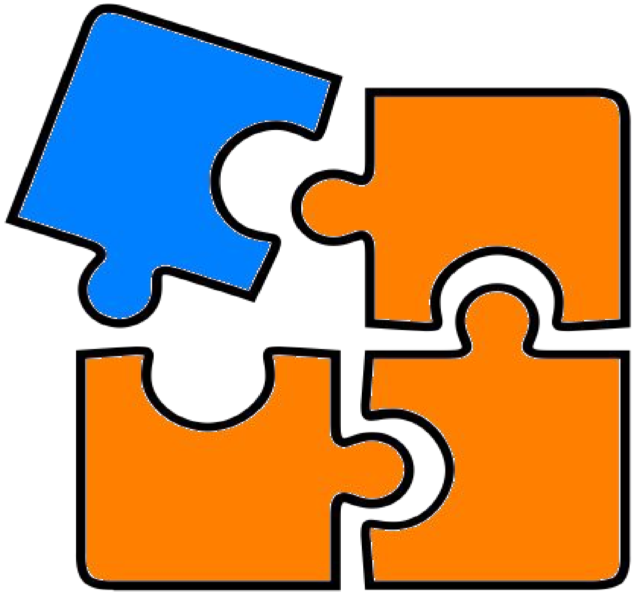
Programming Benefits for GIS

2. Programming automates daily workflows.



Programming Benefits for GIS

3. It extends an application's capabilities and/or functionalities



About Python

- Python is a an OOP "Object Oriented Programming" language.
- It was founded by Guido Van Rossum.
- It is considered an easy language to learn.
- It is used for desktop, web, and mobile development.
- It is OS cross-platform, which means it can run in different operating systems.

Python for GIS professionals

- For a GIS professional, you can also use Python for desktop, web, and mobile development.
- The following are some examples of Python packages and frameworks:
    - **arcpy for ArcGIS**
    - pyqgis for QGIS
    - geoDjango is a web framework
    - shapely library
    - pyshp library
    - PySAL library
- This course will cover only **arcpy for arcGIS**.

arcpy package

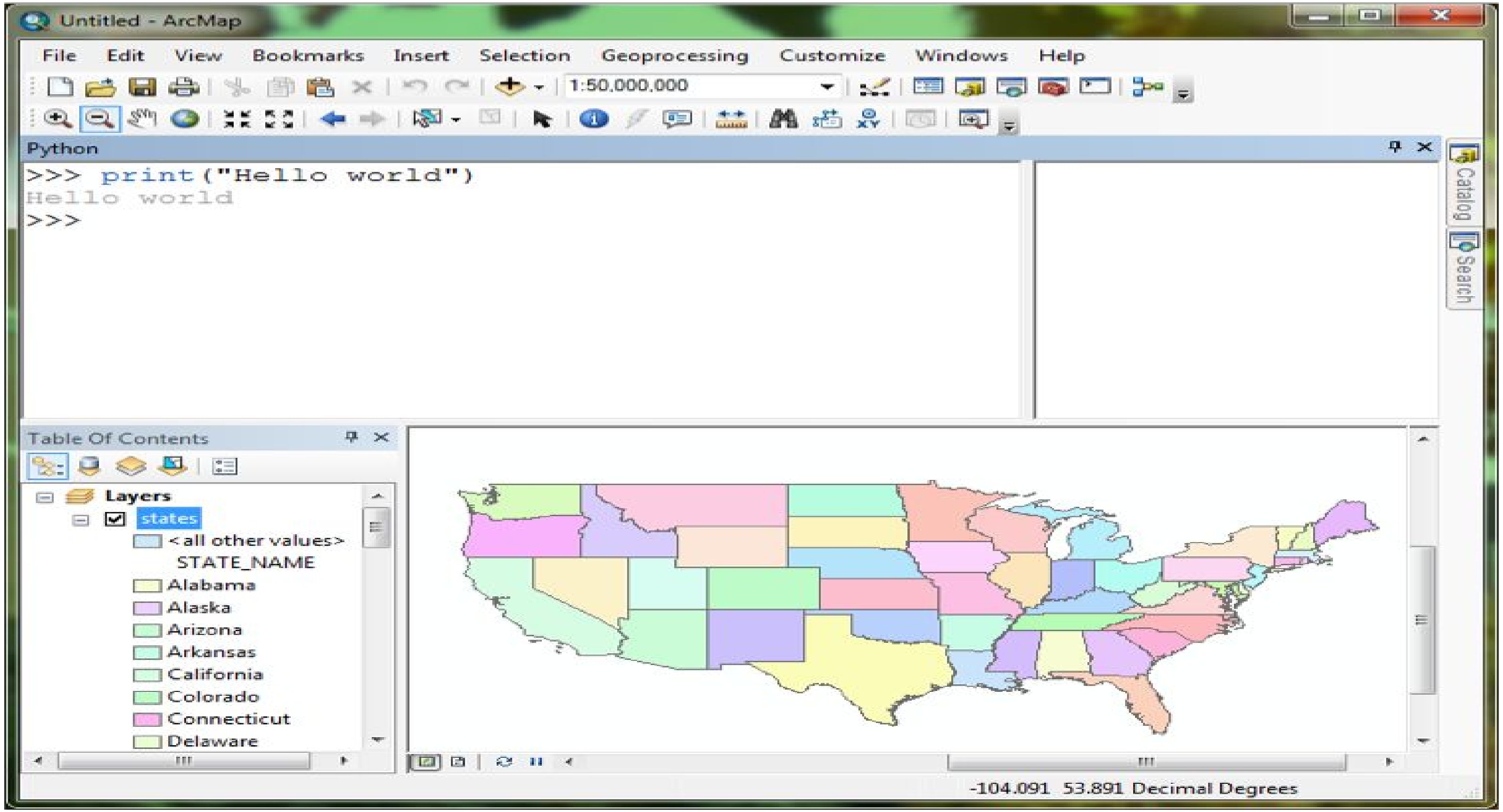
- You have to have ArcGIS installed because arcpy needs the ESRI license.
- Python and arcpy can run in IDLE, ArcMap, or ArcCatalog.
- When coding in ArcMap or ArcCatalog, you do not have to ask Python to use arcpy. This is done for you automatically.
- In IDLE, you have to tell Python that you will use arcpy.
- Always use the IDLE installed with ArcGIS. It can be found in the ArcGIS folder under the start menu. Sometimes, you will have multiple IDLE installations on your machine. For instance, SPSS installs IDLE for statistical coding.

Hello world! Program

- The Hello world! program is the most basic program in any programming language.
- Basically what we are trying to do is to print the sentence Hello World! on the screen.
- In Python, we use the command → print(...)

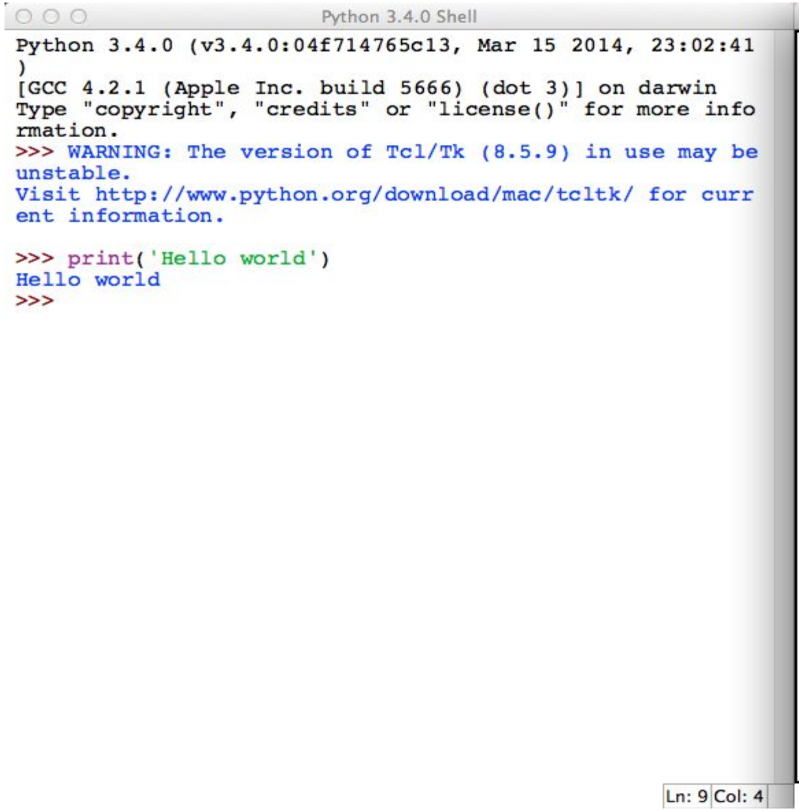
```
print('Hello world!')
```

Python in ArcMap

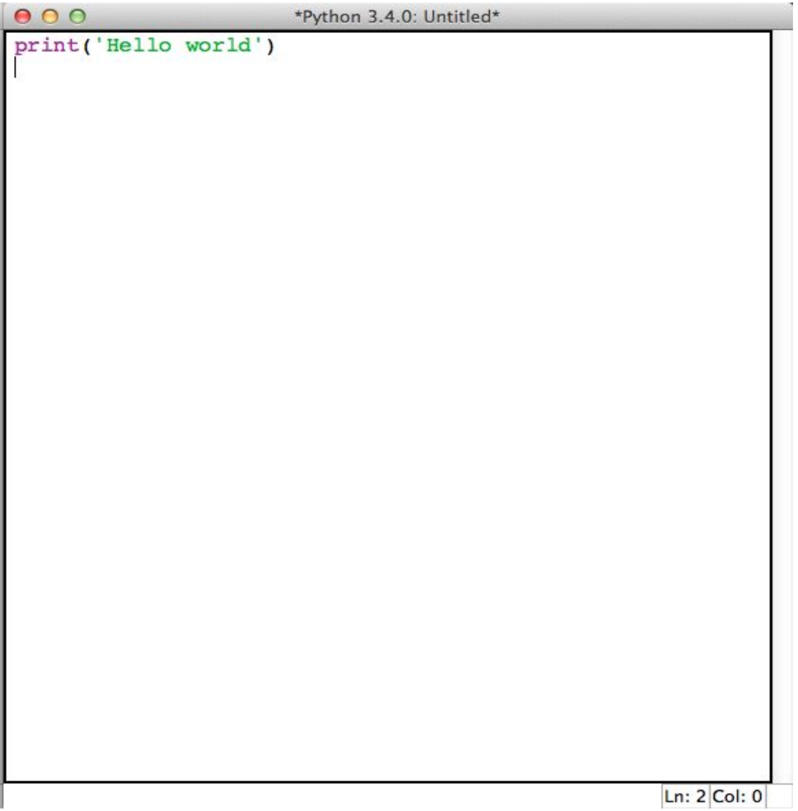


```
>>> print("Hello world")
Hello world
>>>
```

Python in IDLE

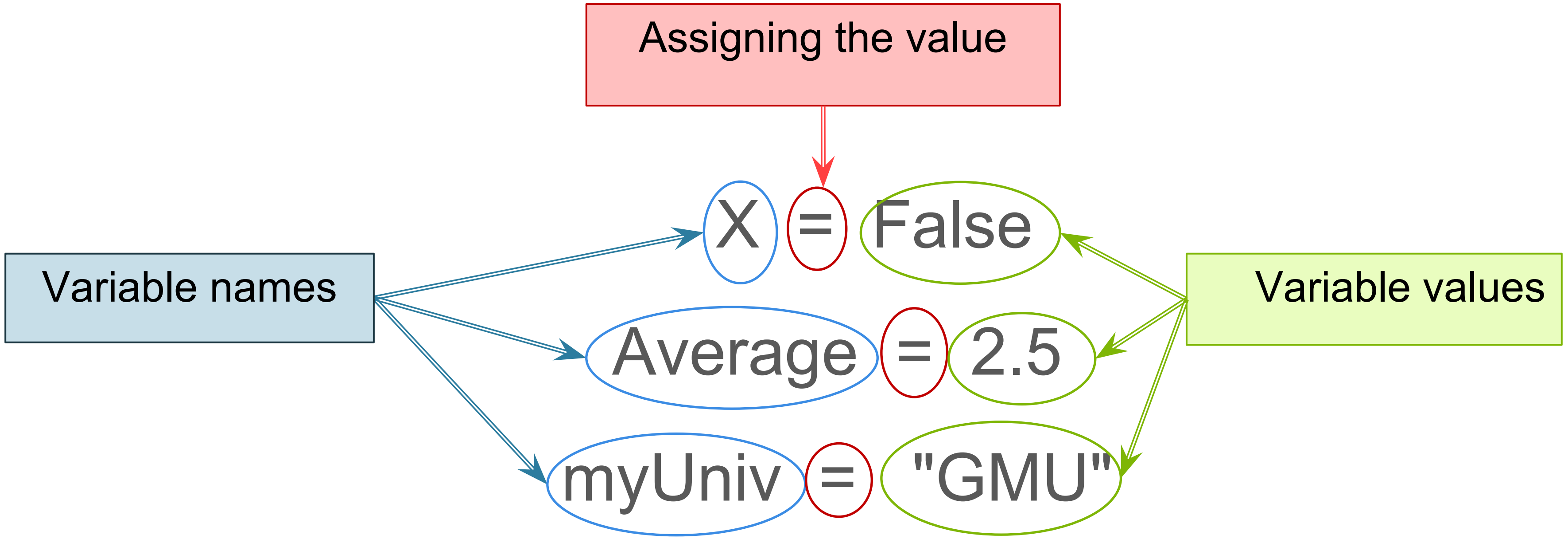


```
Python 3.4.0 (v3.4.0:04f714765c13, Mar 15 2014, 23:02:41
)
[GCC 4.2.1 (Apple Inc. build 5666) (dot 3)] on darwin
Type "copyright", "credits" or "license()" for more info
rmation.
>>> WARNING: The version of Tcl/Tk (8.5.9) in use may be
unstable.
Visit http://www.python.org/download/mac/tcltk/ for curr
ent information.

>>> print('Hello world')
Hello world
>>>
```



```
print('Hello world')
```

Variables


Assigning the value
Variable names
Variable values
X = False
Average = 2.5
myUniv = "GMU"

Variables : Naming Validity

Variable Name	Validity
averageGrade	Correct
AverageGrade	Correct
average_grade	Correct
AVERAGE_GRADE	Correct
Average-grade	Wrong
2average_grade	Wrong
averageGrade2	Correct
_averagegrade	Correct
Average%grade	Wrong
!average_grade	Wrong

Variables : Naming Recommendations

- Have meaningful names
- Try to make the names shorter
- Use comments to describe your variables as well as all syntax

Variable Name	Recommendation
averageGrade	Recommended
TheAverageGradeOfStudents	Not recommended
average_grade	Recommended
A_G	Not recommended
_a_grade	Not recommended
AvErAGEGraDE	Not recommended

Variables : Types

Numeric

- integer, a whole number with no decimal value → examples: 2, 0, -1, 679, -51
- float, a number with a decimal value → 2.0, 0.0, -1.0, 679.0, -51.0, 1.23, 0.001

Textual

- string, a sequence of alphanumeric and special characters. The string value can be wrapped in double quotes or single quotes → examples, 'GMU', "GMU", 'GMU2', "GMU2", '#GMU',"#GMU"

Boolean

- A value of True or False. This is used to evaluate expressions and conditions.

Variables : Numeric

x = 9
y = 3
z = 4.0

Operation	Symbol	Example
Addition	+	x + y => 12, y + z => 7.0
Subtraction	-	y - z => -1.0, x - y => 6
Multiplication	*	y * z => 12.0, y * x => 27
Division	/	y / x => **0.33**, z / 0 => ERROR
Remainder	%	x % y => 0, z % x => 4.0
To the power	**	z ** y => 64.0

Variables : Numeric

x = 9
y = 3
z = 4.0

Example	Result
x * y + x	36
x + y * z	21.0
x + y * z + x	30.0
(x + y) * z + x	57.0

Variables : Boolean

x = True
y = False

Example	Result
x and y	False
not x and y	False
y or y	False
x or x	True
x or y	True

Variables : Boolean

```
x = 1
y = 2
z = 2.0
```

Example	Result
x == y	False
x > y	False
y > x	True
y >= x	True
y == z	True

Variables : Textual "String"

univName = 'GMU'
numberOne =1
collegeName = 'College of Science'
departName = 'GGS'

Example	Result
departName + " is in " + collegeName	'GGS is in College of Science'
departName + ' is in ' + collegeName	'GGS is in College of Science'
departName + " isn't in Research Hall"	"GGS isn't in research Hall"
univName + ' is no.' + str(numberOne)	'GMU is no.1'

Variables : Textual "String"

Method	Description	Example
lower()	Converts all uppercase letters in a string to lowercase.	print('AaABaB'.lower()) => 'aaabbb'
upper()	The opposite of lower()	print('AaBb'.upper()) => 'AABB'

capitalize()	Capitalize the first letter of a string	str1 = 'i like pizza'
		str1 = str1.capitalize()
		print(str1) => 'I like pizza'

find(subStr,..)	Return the index of the search string or -1 if not found	str1 = 'i like pizza' print(str1.find('izz') => 8
count(subStr,..)	Return the number of occurrences of a substring in a string	str1 = 'This is GMU. It is great.' print(str1.count('is')) => 3
replace(old,new,...)	Replaces a substring with a new one	str1 = 'This is GMU. GMU is great.' print(str1.replace('GMU','gmu')) => 'This is gmu. gmu is great'

Variables : Textual "String"

Method	Description	Example

isalnum()	Returns True if all characters are alphanumeric	str1 = 'Fall2009'
		str2 = 'Fall 2009!'
		print(str1.isalnum()) => True
		print(str2.isalnum()) => False

isalpha()	Returns True if all characters are alphabetic	print('Fall2009'.isalpha()) => False
isdigit()	Returns True if String has digits only	print('2009.21'.isdigit()) => False
istitle()	Returns True if the string is in title format based on case-format	print('George Mason'.istitle()) => True print('George MASON'.istitle()) => False
title()	Returns the title format of a string	print('gEorGe mAsoN').title()) => 'George Mason'
lstrip()	Removes leading white spaces	print(' a a '.lstrip()) => 'a a '
rstrip()	Removes trailing white spaces	print(' a a '.rstrip()) => ' a a'
strip()	Performs both rstrip() and lstrip()	print(' a a '.strip()) => 'a a'

Assigning Values to Variables as User Input

- Previously, we assigned the values to variables through code.
- Python gives us the option to assign the values using user input.
- x = input('Please enter a number')
- The code will display the 'Please enter a number' message on the screen and wait for user’s input.
- The user’s input will be assigned to the variable x

Commenting the Code

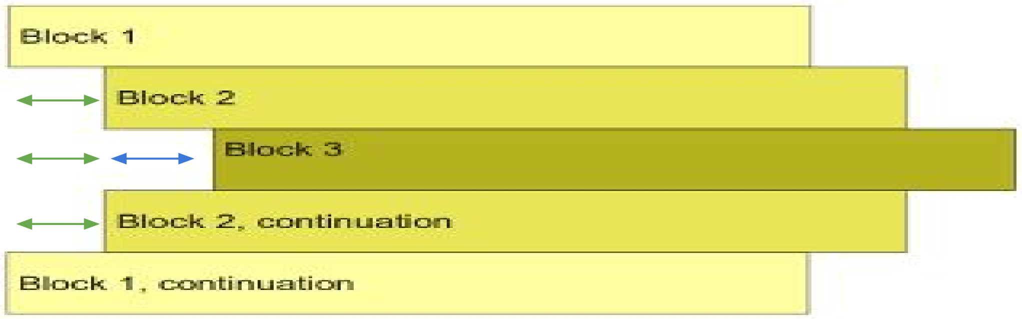
- Commenting the code or documenting the code is very important.
- It explains the code to others or it reminds you about what you did in your old code.
- The # sign comments one line. To comment multiple lines use ''' and then close the comment with '''

```
val1 = 17
val2 = 55
val3 = 101
# The following line prints the average of val1, val2, and val3
print((val1+val2+val3) / 3)
'''
Written by Ahmad Aburizaiza
For educational use
'''
```

Code practice

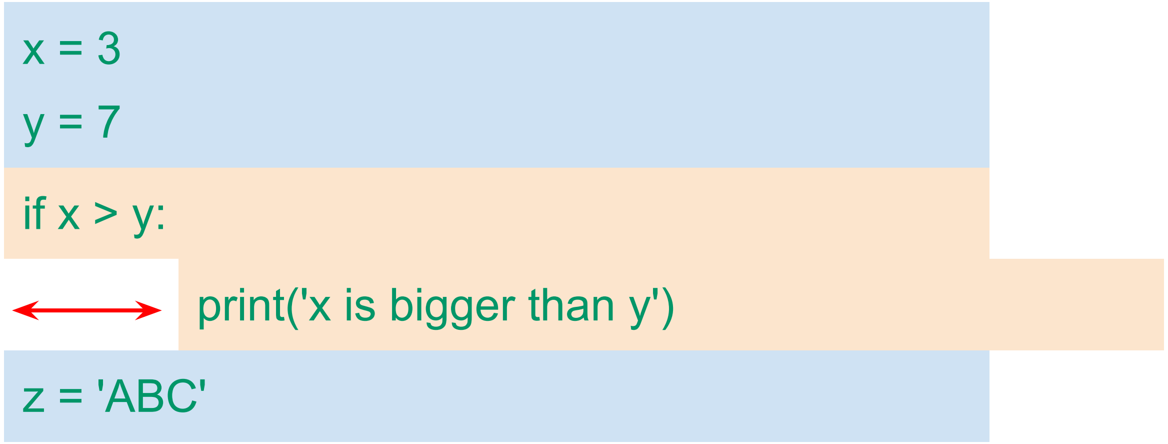
Indentation

- Indentation in Python is similar to using parenthesis **{}** in other programming languages.
- It is used to define blocks of code inside statements such as conditions, functions, loops, classes.
- A block only runs if its parental or hierarchical block or is called or is True.
- For instance, the code in block 2 will not run unless the code in block 1 permits.
- The code in block 3 will run if block 1 permits and then block 2 permits.



Conditions

- Conditional statements are used to run an interior block based on the condition of the statement.
- if statements are the most common conditional statements in programming languages.



```
x = 3
y = 7
if x > y:
    print('x is bigger than y')
z = 'ABC'
```

Conditions

```
month = 10
day = 1
fiscalYear = True
if month == 6:
    print("It is June")

if day <= 5:
    print("It is the beginning of the month")

if fiscalYear:
    print("it is a fiscal year")
```

Conditions

```
raining = True
cold = True

if raining and cold:
    print("Wear a jacket and take an umbrella")

if not raining and cold:
    print("Wear a jacket")

if raining and not cold:
    print("Take an umbrella")

if not raining and not cold:
    print("Enjoy the weather")
```

Conditions

```
randNum = input('Please enter a number: ')
if randNum % 2 == 0:
⟷print( randNum + ' is divisible by 2')

elif randNum % 3 == 0:
⟷print(randNum + ' is divisible by 3')
else:
⟷print(randNum + ' is not divisible neither by 2 nor 3')
```

Lists

- A list is a sequence of data values stored as one variable.
- The data values in a list are called elements.
- Each element is assigned an index.
- In Python, you can create a list of different variable types. It is not recommended but you can do it.

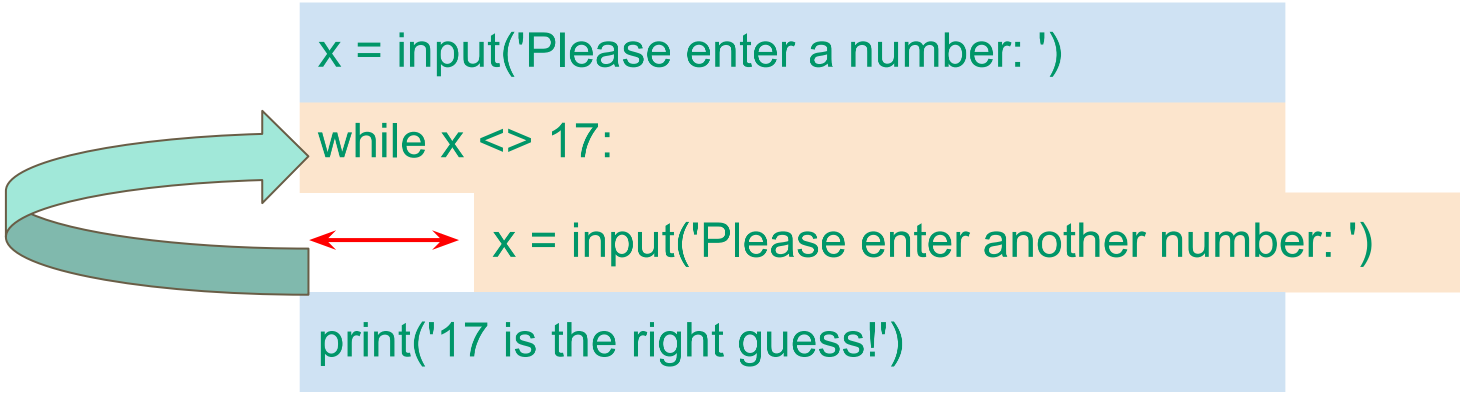
```
intList = [1, 7, 2, 5, 4, 6, 3]
stringList = ['a', 'b', 'abc123', '@TipsForGIS']
mixedList = [1, 'a', 2, '3', 'xy']
```

Lists

```
intList = [2, 7, 1, 5, 4, 6, 3]
stringList = ['a', 'b', 'abc123', '@TipsForGIS']
mixedList = [1, 'a', '22', 75, 'xy']

print(intList[0]) => 2
print(intList[1]) => 7
print(intList[-1]) => 3
del(mixedList[3]) => the element 75 will be deleted.
mixedList.append(101) => adds a new element with a value of 101
intList.sort() => intList will be [1,2,3,4,5,6,7]
print(len(stringList)) => 4
```

Loops

- A loop is control that forces repetition of interior code block(s).
- The for loop is a popular loop in programming.
- The while loop is another popular loop in programming.
- When writing loops, be careful not to write an infinite loop.



```
x = input('Please enter a number: ')
while x <> 17:
    x = input('Please enter another number: ')
print('17 is the right guess!')
```

Loops Comparison

for loop

```
numList = [1,2,3,4,5,6,7,8,9]

for n in numList:
    print(n)
print('Done')
```

while loop

```
n = 1
while n <= 9:
    print(n)
    n = n + 1
print('Done')
```

Conditional Statements Inside Loops

```
numList = [1,2,3,4,5,6,7,8,9]

for n in numList:
    if n % 2 == 0:
        print(str(n) + ' is even')
    else:
        print(str(n) + ' is odd')
print('Done')
```

Code practice

Functions

- A function is used to reuse certain code blocks.
- You can define a function with 0 or more parameters.
- A parameter is a value that you can pass to the function to use it.

```
Def functionName(param1,....):
      line code1
      line code2
      line code3
      .......
```

To call the function → `functionName(x,...)`

Functions

No-return-value function

```
def addTwoIntegers(int1,int2):
        print(int1 + int2)

addTwoIntegers(2,3)
addTwoIntegers(30,40)
```

Return-value function

```
def addTwoIntegers(int1,int2):
        return int1 + int2

a = addTwoIntegers(2,3)
b = addTwoIntegers(30,40)
```

Why Functions?

```
side1 = 3
side2 = 4

largeSide = side1*side1
largeSide = largeSide + side2*side2
largeSide = largeSide ** 0.5

print(largeSide)
```

```
def pythagorean(side1,side2):

    largeSide = side1 * side1
    largeSide = largeSide + side2*side2
    largeSide = largeSide ** 0.5
    return largeSide


print(pythagorean(3,4))
print(pythagorean(1,1))
print(pythagorean(2,7))
```

The Scope

The scope of a variable or an object is where it can be accessed.

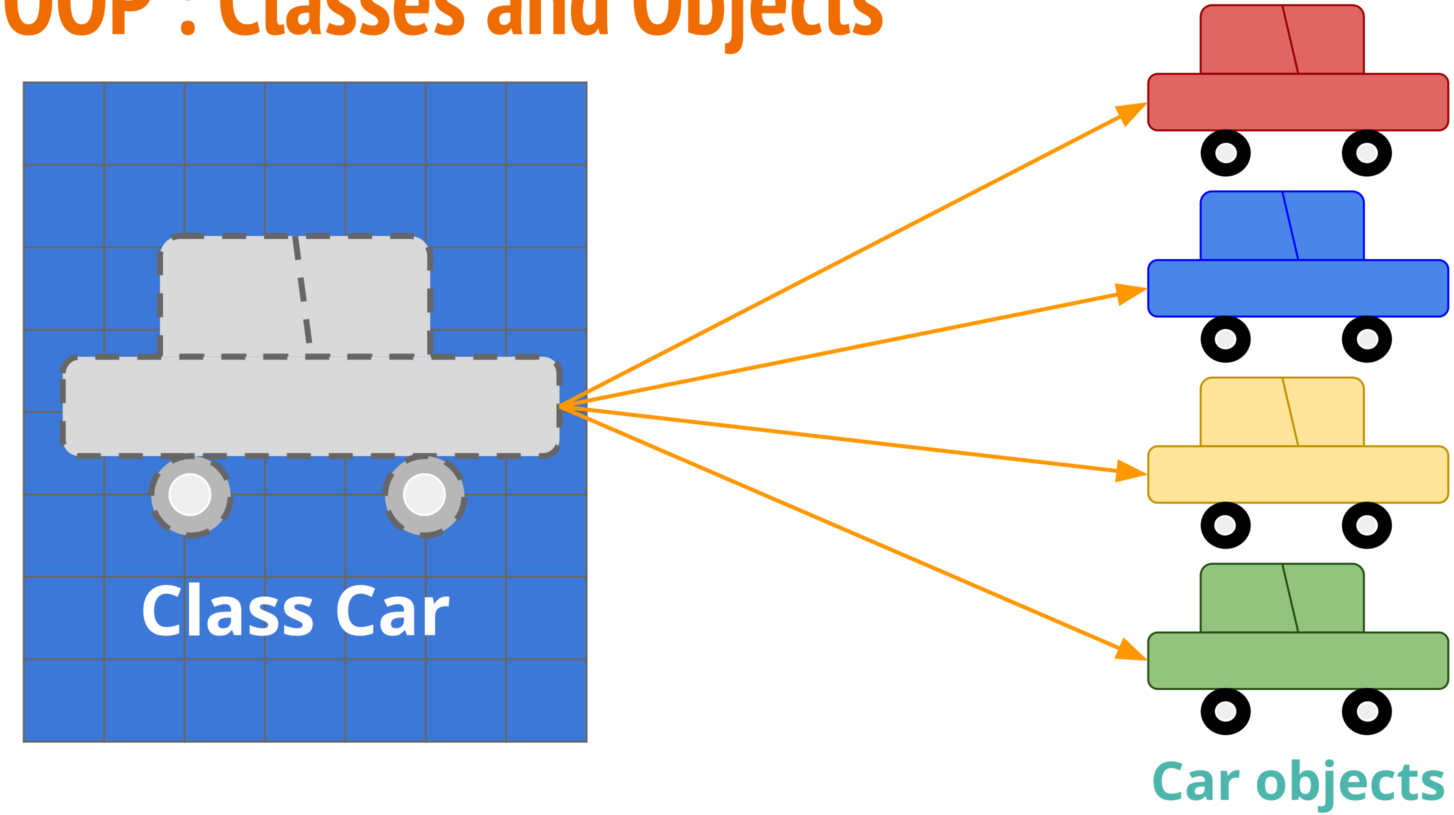
```
x = 5

def func1():
	x = 7
	print(x) => This x is the local x inside func1

func1()
print(x) => This x is the global x outside func1
```

Code practice

OOP : Classes and Objects

- OOP “Object Oriented Programming” is a concept of dealing with objects in programming.
- Objects have attributes and methods “functions or actions”.
- A class “blueprint” is created to produce objects. The class name should start with a capital letter.
- You can create your own classes or classes created by others.
- We are not going to cover class creation. But we need to know how to use predefined classes.

OOP : Classes and Objects



OOP : Classes and Objects

Object	Properties	Methods

	car.name = Fiat
	car.model = 500
	car.weight = 850kg
	car.color = white
		car.start()
		car.drive()
		car.brake()
		car.stop()

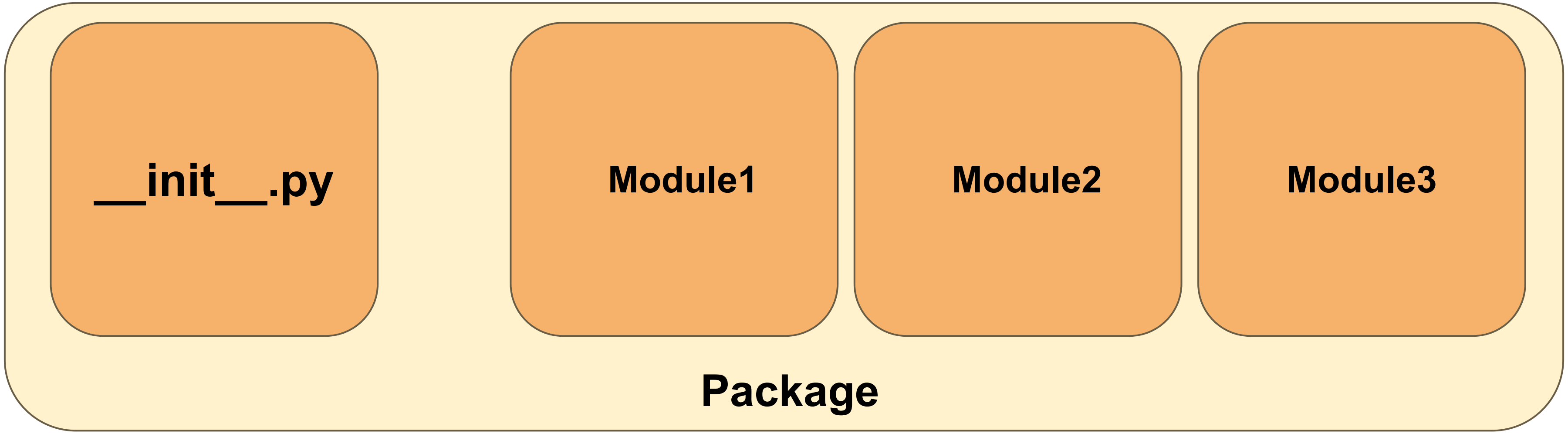
http://www.w3schools.com/js/js_objects.asp

Simple Class Definition

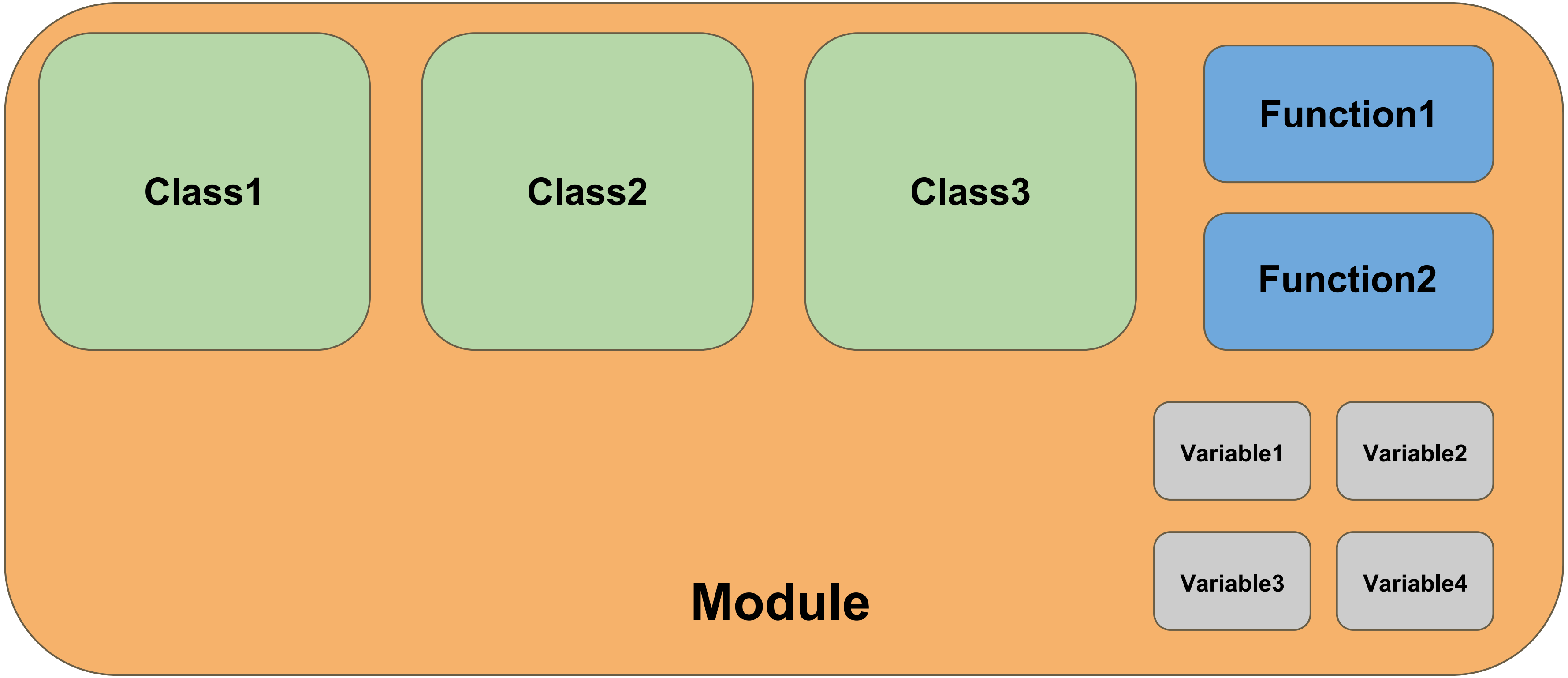
```
class Person:
        def __init__(self,name,age,weight,height):
                self.name = name
                self.age = age
                self.weight = weight
                self.height = height

        def walk(self):
                print(self.name + ‘ is walking’)
        def eat(self):
                print(self.name + ‘ is eating’)
```

OOP

- A module is .py file contains a collection of classes independent functions and/or variable.
- A package is basically a folder of modules + the __init__.py module.
- The __init__.py makes the folder a Python package. It can be left empty.



OOP


Class1
Class2
Class3
Function1
Function2
Variable1
Variable2
Variable3
Variable4
Module

OOP

To import a class from a module, type in:

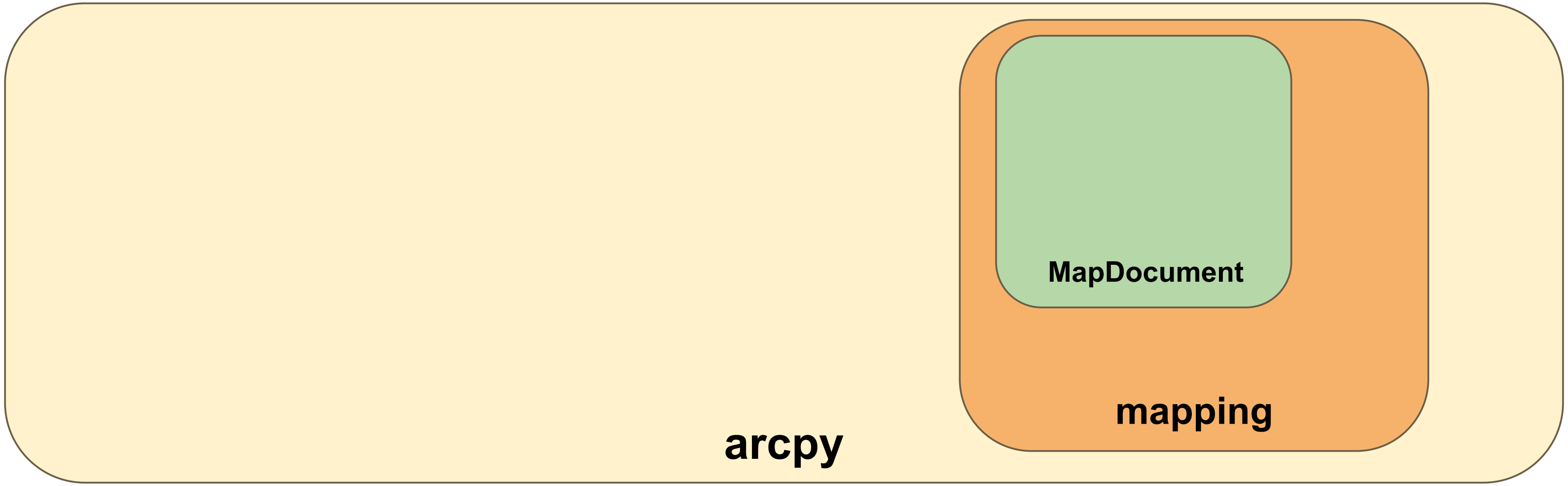
```
from moduleName import ClassName
```
OR
```
from moduleName import*
```
OR
```
import moduleName
```

To import a class from a module from a package, type in:

```
from packageName.moduleName import ClassName
```
OR
```
from packageName.moduleName import*
```
OR
```
import packageName.moduleName
```

OOP : Classes and Objects

A very popular class under arcpy package is MapDocument which resides under the mapping module.



Importing a Class from arcpy

To import the MapDocument class:
→ import arcpy.mapping.MapDocument

Class name	Properties	Methods
MapDocument	title	save()
	author	saveACopy(fileName)
	activeDataFrame	makeThumbnail()
	credits	deleteThumbnail()

Conclusion

Topics not covered in the workshop: break and continue in loops, read/write files, and class creation

Check out my Youtube channel named Tips for GIS
https://www.youtube.com/channel/UCOjxVdT7wKbHKA5PWvFsW3g

Also, check out my Github account for documented code samples
https://github.com/TipsForGIS

Thank you for listening

Part 2 of this workshop will cover the usage of the arcpy package