Lecture 2

Data Science Course with Python

Previous Assignment

What did You learn??

Advantages of Python??

Limitations of Python??

Java vs Python??

Programming with Python

Loops
Lists
List operations
List Methods
Functions
Importing Modules
Statistics
Exceptions
Hands on Lab

While loops

- Loop Runs ‘while’ the condition holds **True**
- Loop Terminates when condition evaluates to **False**
- A loop is executed repeatedly. This is called iteration.

```
>>> i=1
>>> While i<=5:
>>>       print(i)
>>>       i=i+1
>>> print(“Finished!”)
```

While loops

```
O/P:        1
            2
            3
            4
            5
            Finished!
```

break statement

- To exit from a while loop prematurely use **break** statement.
- Loop Terminates when condition evaluates to **False.**

```
>>> i=5
>>> while True:                  //infinite loop
>>>    print(i)
>>>    i=i-1
>>>    if i<=2:
>>>       break
>>> print("Loop Terminates")
```

break statement

```
O/P:        5
            4
            3
            Loop Terminates
```

continue statement

- Continue statement jumps back to the top of the loop.
- Continue statement stops the current iteration.

```
>>> age =15
>>> while True:
>>>    age =age+1
>>>    if age==18:
>>>       print(“Get a Degree”)
>>>       continue          //skipping rest & Jump to top
>>>    if age==23:
>>>       print(“Get a Job”)
>>>       break
```

Lists

- Lists are used to store an indexed list of items.
- List items can be accessed by using its index in square brackets.

```
>>> words = [“Hello”, “world”, “!”]
>>> print(words[0])
>>>  print(words[1])
>>>  print(words[2])
```

O/P:

```
Hello
World
!
```

Empty lists

```
>>> empty_list = []
>>> print(empty_list)
>>>
```

O/P:

```
[]
```

Reassigning Lists

```
>>> numbers = [7,7,7,7,7]

numbers[2]=5

print(numbers)

O/P:

[7,7,5,7,7]
```

List Operations

- Lists can contain items of different types.
- Lists can also be nested within other lists.
- Lists of Lists are used to represent multidimensional arrays.

```
>>> number= 3
>>> things =[“string”,0,[1,2,number],4.56]
>>> print(things[1])
>>> print(things[2])
>>> print(things[2][2])
O/P:
0
[1,2,3]
3
```

List Operations

- Lists can be added and multiplied just like strings.

```
>>> numbers= [1,2,3]
>>> print(numbers + [4,5,6])
>>> print(numbers*3)
```

O/P:

```
[1,2,3,4,5,6]
[1,2,3,1,2,3,1,2,3]
```

List Operations

- Checking list items using in operator.
- It is also used to detect a substring within a string.

```
>>> World= [“New Zealand”, ”Brazil”, ”Greenland”]
>>> print(“Brazil” in World)
>>> print(“New” in World[1])
>>> print(“Asia” in World)
>>> print (not “Russia” in World)
```

O/P:

```
True
False
False
True
```

List Methods

- Using the append method

```
>>> nums = [1,2,3]
>>> nums.append(4)          // nums=nums+[4]
>>> print(nums)

O/P:
[1,2,3,4]
```

List Methods

- Using the insert method

```
>>> words = [“Python”, “fun”]
>>> index =1
>>> words.insert(index, “is”)
>>> print(words)

O/P:
[‘Python’, ‘is’,’fun’]
```

List Methods

- Using the index method

```
>>> letters = [‘p’, ‘q’,’r’,’s’,’t’]
>>> print(letters.index(‘r’))
>>> print(letters.index(‘p’))
>>> print(letters.index(‘c’))
```

O/P:

```
2
0
Value Error: c not in list
```

List Functions

- Using the len function

```
>>> nums = [1,2,7,5,6,9,3,4]
>>> print(len(nums))

O/P:
8
```

Other Useful Functions and Methods

- max(list): Returns the list item with the maximum value
- min(list): Returns the list item with minimum value
- list.count(obj): Returns a count of many times an item occurs in a list.
- list.remove(obj): Removes an object from a list
- list.reverse(): Reverses objects in a list

Range Function

- Range function creates a sequential list of numbers.

```
>>> numbers = list(range(10))
>>> print(numbers)
>>> nums = list(range(3,8))
>>> print(nums)
>>> digits = list(range(3,30,3)
>>> print(digits)
O/P:

[0,1,2,3,4,5,6,7,8,9]        //same as range(0,10)

[3,4,5,6,7]

[3,6,9,12,15,18,21,24,27]
```

Using Loop Constructs to Access List Items

- Using For Loop

```
>>> Weather
=[“Winter”,”Spring”,”Summer”,”Fall”,”Autumn”]
>>> for i_weather in Weather:
>>>     print(i_weather + “ 2018”)
O/P:

        Winter 2018
        Spring 2018
        Summer 2018
        Fall 2018
        Autumn 2018
```

Using Loop Constructs to Access List Items

- Using For Loop

```
>>> for i in range(5):
>>>     print(”Day”+str(i))
O/P:

        Day0
        Day1
        Day2
        Day3
        Day4
```

Accessing List using For Loop

- Using For Loop

```
>>> for i in range(10,50,10):
>>>     print(”Job ”+str(i))
O/P:

        Job 10
        Job 20
        Job 30
        Job 40
```

Accessing List using For Loop

- Using For Loop

```
>>> for i in range(10,50,10):
>>>     print(”Job ”+str(i))
O/P:

        Job 10
        Job 20
        Job 30
        Job 40
```

Quiz

1 What is the o/p of this code?

```
NumList =
[1,1,2,3,5,8,13]

print(NumList[NumList[
4]])
```

2 What does this code do?

```
for i in range(10):
    if not i%2==0:
        print(i+1)
```

- Print all odd numbers between 1 and 9
- Print all even numbers between 2 and 10

3 How many lines will this code print?

```
While False:

    print("Looping..")
```

Quiz

1 What is the o/p of this code?

```
NumList =
[1,1,2,3,5,8,13]

print(NumList[NumList[
4]]
```

8

2 What does this code do?

```
for i in range(10):
if not i%2==0:
    print(i+1)
```

- **Print all even numbers between 2 and 10**

3 How many lines will this code print?

```
While False:
```

Zero

Quiz

4 Fill in the blanks to print first element of the list if it contains even number of elements

```
list = [1,2,3,4]

if ___(list)%2==0_

    print(list[__])
```

5 What does this code output?

```
letters= [‘x’, ’y’, ’z’]
letters.insert( 1, ’w’ )
print(letters[2])
```

6 Fill the blanks to iterate over the list using a for loop and print its values

```
list=[1,2,3]

___var___list:

    print(___)
```

Quiz

4 Fill in the blanks to print first element of the list if it contains even number of elements

```
list = [1,2,3,4]

if len(list)%2==0:

    print(list[0])
```

5 What does this code output?

```
letters= ['x', 'y', 'z']
letters.insert( 1, 'w' )
print(letters[2])
```

y

6 Fill the blanks to iterate over the list using a for loop and print its values

```
list=[1,2,3]

for var in list:

    print(var)
```

Functions and Modules

- Python abides the DRY Principle to make the code easier to maintain.
- Using pre-defined functions with Function names, Function Call and Function arguments.

Defining New Functions :

```
>>> def my_func(x,y):          //passing variables as parameters
>>>     x = x+5
>>>     return(x*y)
>>> z= my_func(5,8)                  //passing arguments
>>> print(z)
O/P: 80
```

❖ Referencing Parameter variables outside the scope of the function will result in *NameError: variable not defined.*

Functions and Modules

- Python abides the DRY Principle to make the code easier to maintain.
- Using pre-defined functions with Function names, Function Call and Function arguments.

Defining New Functions :

```
>>> def my_func(x,y):          //passing variables as parameters
>>>     x = x+5
>>>     return(x*y)
>>>     print("Unreachable code")  //executions stops after return
>>> z= my_func(5,8)                //passing arguments
>>> print(z)
```

O/P: 80

Modules

- Modules are pieces of codes written by others to fulfill common tasks.
- To use a module, add *import module_name* at the top of your code.
- Using Modules :

```
>>> import math
>>>    num =100
>>>    print(math.sqrt(num))
O/P:10
```

❖ **Accessing sqrt() function defined in math module using period operator**

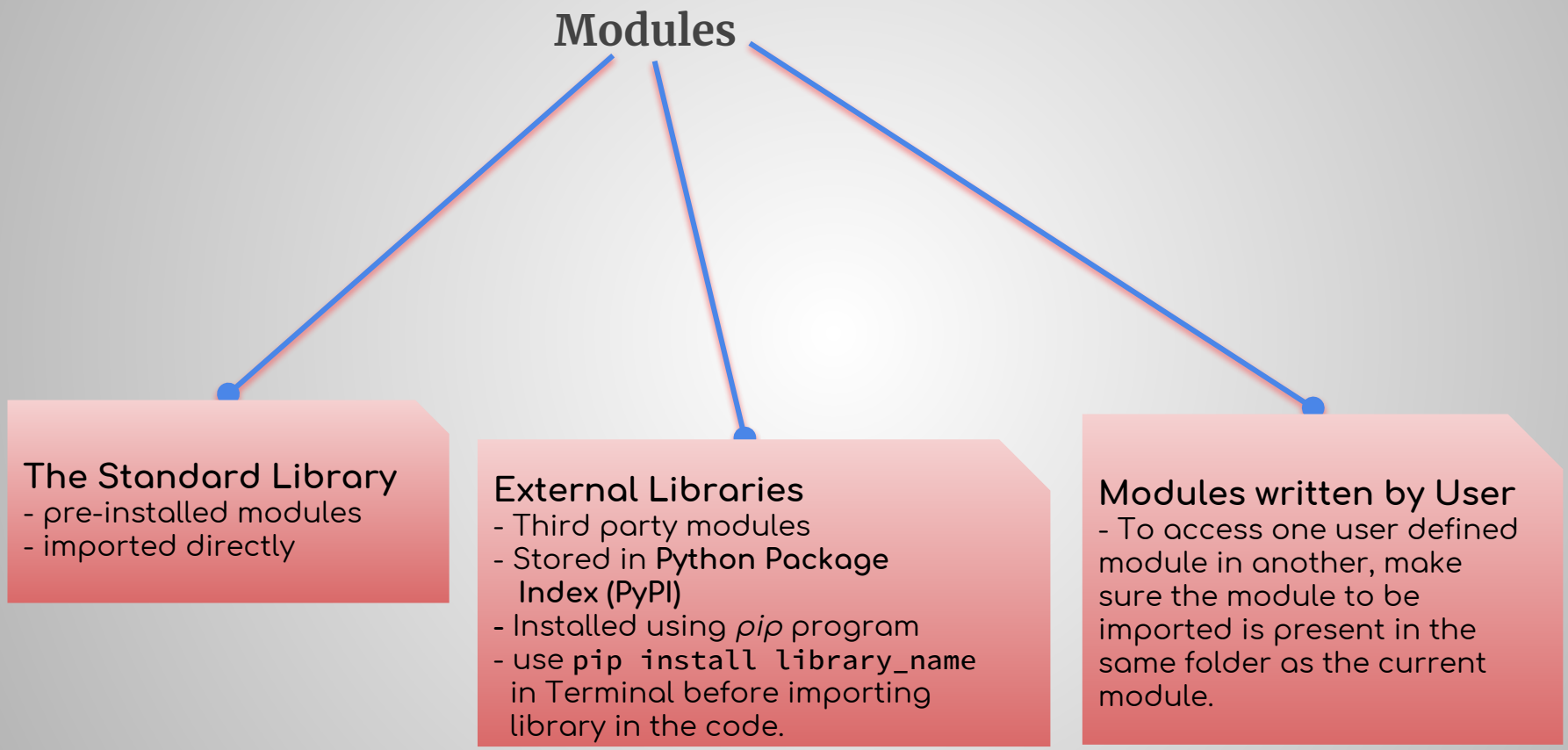
Modules

- Use *from module_name import ** to import all objects from a module.
- Importing a module that is not available causes *ImportError*

- Importing Modules Objects :

```
>>> from math import pi,cos,sqrt as square
>>>    print(pi)
>>>    print(square(256))
O/P:   3.14
       16
```

Types of Modules



Using Statistics Module in Python

- Use `Statistics` Module for data analysis using descriptive statistics.
- Statistics Module is part of *The Standard Library*
- Statistics module is not available in Python 2.7. Only Python 3+.
- It supports following statistical techniques -
  - Mean
  - Median
  - Mode
  - Standard Deviation
  - Variance

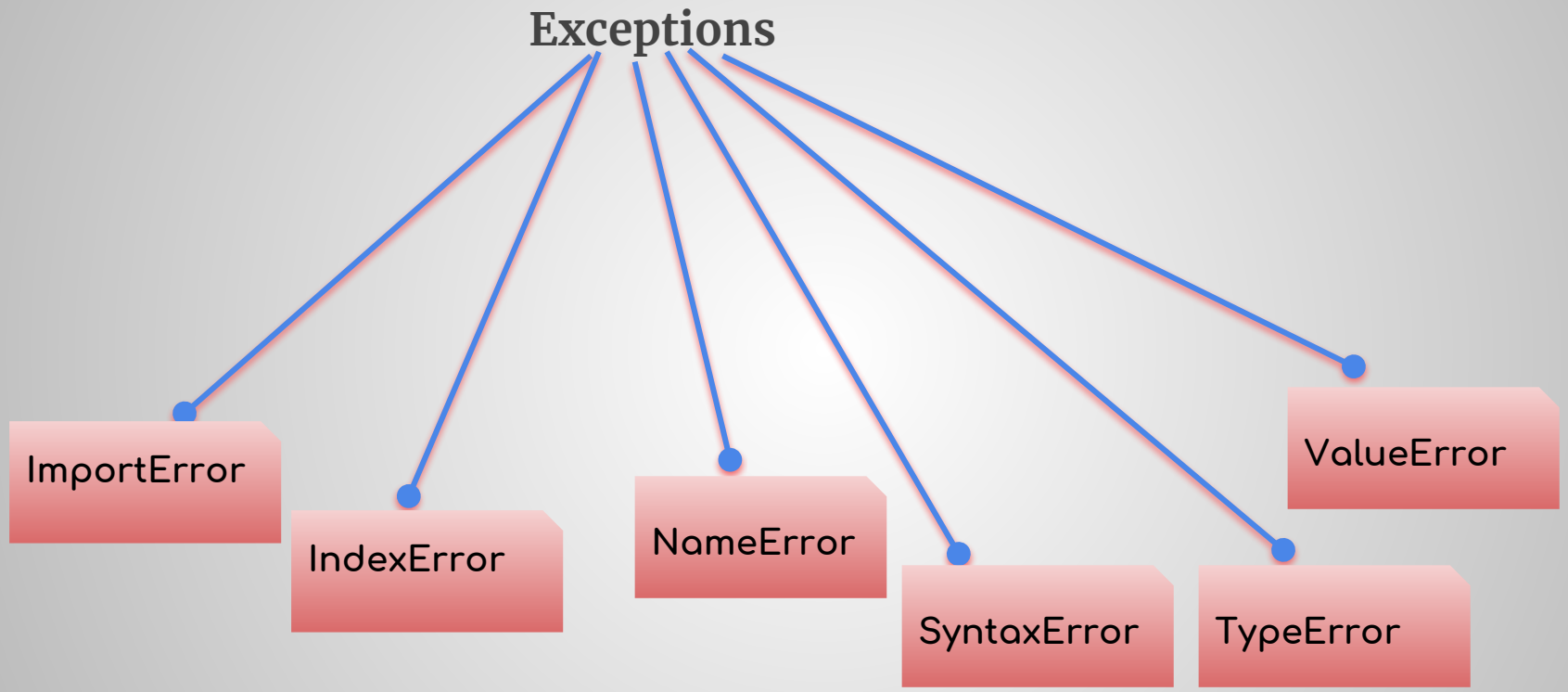
Using Statistics Module in Python

```
>>> import statistics
>>> ListOfNumbers = [5,2,5,6,1,2,6,7,2,6,3,5,5]
>>> x = statistics.mean(ListOfNumbers)
>>> print(x)
4.230769230769231
>>> y = statistics.median(ListOfNumbers)
>>> print(y)
5
>>> z = statistics.mode(ListOfNumbers)
>>> print(z)
5
>>> a = statistics.stdev(ListOfNumbers)
>>> print(a)
1.96
>>> b = statistics.variance(ListOfNumbers)
>>> print(b)
3.8
```

What is an Exception?

“An event that occurs due to incorrect code or input.”

Types of Exceptions



Quiz?

Which exception is raised by this code?

```
>>> print("7"+4)
```

Quiz?

Which exception is raised by this code?

```
>>> print("7"+4)
```

Type Error

Exception Handling

```
try:
    num1= 7
    num2=0
    print(num1/num2)
    print("Done Calculation")
except ZeroDivisionError :
     print("An error occurred")
     print("Due to zero division")
```

What is the output of this code?

```
try:
    var=10
    print(10/2)

except ZeroDivisionError :
     print("Error")
print("Finished")
```

Multiple except Blocks

```
try:
    var=10
    print(var+ “hello”)
except ZeroDivisionError :
     print(“Zero division Error, Retry.”)

except (ValueError, TypeError) :
     print(“Error occured”)
```

try-except-finally

```
try:
    print("Hello")
    print(1/0)
except ZeroDivisionError :
    print("Divided by zero")
finally :
    print("This code will print no matter what")
```

Raising Exceptions

```
>>>print(1)
>>>raise ValueError
>>>print(2)
```

```
>>> name = “123”
>>> raise NameError(“Invalid name!”)
```

Summary

1. A python program files is saved with extension .py while a Jupyter notebook has extension .ipynb

2. A python variable can store any type of data but must be given an initial value when it is declared.

3. Syntax errors due to incorrect code are recognized by the interpreter before execution of the program.

4. Runtime errors due to exceptions are recognized by the interpreter during execution of the program.

5. Placeholders can be created in the code by inserting the pass keyword where a statement is required syntactically.

Assignment

Program a Simple Calculator

```
While True:
   print(“Options:”)
   print(“Enter sum for plus”)
   print(“Enter sub for minus”)
   print(“Enter mul for multiply”)
   print(“Enter div for division”)
   print(“Enter quit to exit”)
   compute = input(“: ”)
   if compute == “quit”:
       pass
   elif    compute == “sum”:
       pass
   elif compute == “sub”:
       pass
   elif compute == “mul”:
       pass
   elif compute == “div”:
       pass
   else:
       print(“Unknown input”)
```

Assignment

Raise a ValueError if the input is negative

```
num = input(“Enter value:”)
if pass:
    pass(“Negative input”)
```

Further Reading

Internet of Things

&

Smart Home

https://www.explainthatstuff.com/smart-home-automation.html

Hands-on Lab with Jupyter

Instructions

- Launch Jupyter on localhost using Anaconda Navigator.
- Select Python 3 kernel to open notebook file
- Open --
  - PythonBasics.ipynb

  And,
  - PythonExercise.ipynb

Next

Python Programming
Generators
Dictionary
Tuple
Assertions
Files
OOP
Regular Expressions