Python Basics: Outline

Module-I

Introduction, Syntax, Variables, Operators, Loops, Conditions, Arrays

Module-II

Functions, Default Value Arguments , Arrays, Exception Handling, File I/O

Module-III

Basic Statistics and List processing using Numpy

Input/output

Output:

```
print(“Hello World”)
```

Input:

```
variable_name = input("Enter your Name")
```

Comment:

```
# This is Single Line Comment

"""
A multiline comment
"""
```

Operators

Arithmetic Operators
+, -, *, /, %, //, **

Comparison Operators
>, <, ==, !=, >=, <=

Logical Operators
and, or, not

Assignment Operators
=, +=, -=, /=, *=, %=, //=, **=

Control structures

```
if x < 0:
        something
else:
        somethingdifferent
```

```
if x < 0:
        something
elif x > 0:
        somethingdifferent
else:
        yetanotherthing
```

Control structures

For Loop:

```
for i in range(0, 10):
        print( i )
```

While Loop:

```
x = 10
while x < 10 :
        print( x )
        x += i
```

```
0
1
2
3
4
5
6
7
8
9
```

Arrays: Lists

list1 = [32, 27, 64, 18] – One Dimentional

Declaration:

list1 = [0 for i in range(10)]

list2 = [[1,2] , [3,4]] – Two Dimentional

Declaration:

list2 =[[0 for j in range(3)] for i in range(5)]

Functions:

del, cmp(list1,list2), len(list1), max(list1), min(list1), append(obj), insert(i,obj), count(obj), pop(), remove(obj)

Indexing, Slicing, and Matrixes

List1[2], list1[-2], list1[0:4], list1[2:]

Arrays: tuples

A tuple is a sequence of immutable Python objects which means you cannot update or change the values of tuple elements

tuple1 = (32, 7, 64, 1 8)

Functions:

del, cmp(list1,list2), len(list1), max(list1), min(list1)

Arrays: Others

Dictionary:

Each key is separated from its value by a colon (:), the items are separated by commas, and the whole thing is enclosed in curly braces.

dict = { ‘Name’ : ‘Ali’, ‘Age’ : 23, ‘Department’ : ‘BSCS’ }

print (dict [‘Name’])

Sets:

A set contains an unordered collection of unique and immutable objects

Strings

Python does not support a character type

Assume string variable **a** holds 'Hello' and variable **b** holds 'Python', then

- a + b will give HelloPython
- a * 2 will give HelloHello
- a[1] will give e

Functions:

isalpha(), isalnum(), isdigit(), islower(), isnumeric(), isupper(), max(), min()

Exercise

Write a program that calculates the user's body mass index (BMI) and categorizes it as underweight, normal, overweight, or obese, based on the table from the United States Centers for Disease Control:

BMI	Weight Status
Below 18.5	Underweight
18.5 – 24.9	Normal
25.0-29.9	Overweight
30.0 and above	Obese

To calculate BMI based on weight in pounds (lb) and height in inches (in), use this formula (rounded to tenths):

$$\text{BMI} = \frac{\text{mass(lb)}}{(\text{height(in)})^2} \times 703$$

Prompt the user to enter weight in pounds and height in inches

```
[SAMPLE RUN]
Enter your weight in whole pounds: 110
Enter your height in whole inches: 60
You have a BMI of 21.5, and your weight status is normal.
```

Exercise

Write a program to compute **quotient** and **remainder** of a number without using division ('/') operator and modulo ('%') operator

Write a program that needs to ask the user for her or his email address in the format *firstname.lastname@bahria.edu.pk* OR *firstname.lastname@gmail.com*. The application takes as input this email address, parses the email and replies to the user with first name, last name and host name.

```
Please enter your email address (firstname.lastname@bahria.edu.pk):
khalid.amin@bahria.edu.pk
First Name: Khalid
Last Name: Amin
Host Name: bahria.edu.pk
```

Module - II
Contents

- **Functions Without Return Type and No Argument**
- **Functions Without Return Type and with arguments**
- **Functions With Return Type but no argument**
- **Functions With Return Type and Arguments**
- **Functions With Default value Arguments**
- **Function Overloading**
- **Recursion**
- **Exception Handling**
- **File I/O**

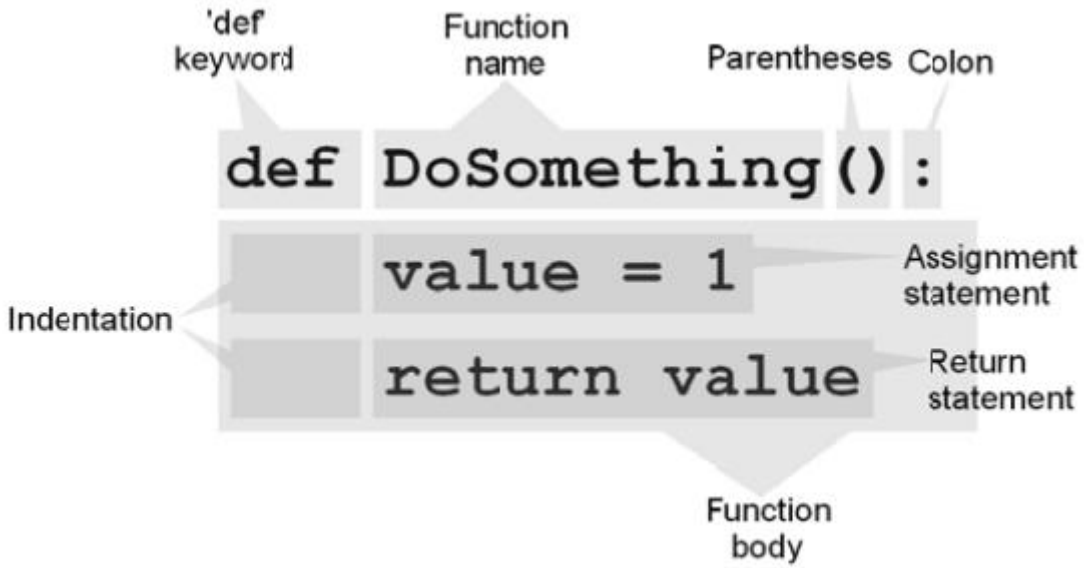
Function

def is a keyword to define a function

***function*_name** is the identifier

parameters

Statement is the function's body



Functions Without Return Type and No Argument

```
def drawLine():
    for i in range(30):
        print("*" , end=' ') # end ='' prints all * in single row
        print()


drawLine()
```

Functions Without Return Type and with arguments

```
def drawLine(n):
    for i in range(n):
        print( "*" , end= ' ' )
        print()


drawLine(30)
```

Functions With Return Type but no argument

```
def getUniversityName():
    return "Bahria University"

print("The University Name is "+ getUniversityName() )
```

Functions With Return Type and Arguments

```
def findSum(b):

    total=0
    for i in range(len(b)):
        total+=b[i]
    return total


a = [2,3,5,8,4,9,7,6,7,8]
sum = findSum(a)
print("The result is ",sum)
```

Function overloading

There is no function overloading in python

Functions With Default value Arguments

```
def sum(a=5,b=10,c=20):
    return a+b+c


print("The sum without passing parameters",sum())
print("The sum with one parameter",sum(10))
print("The sum with two parameters",sum(10,20))
print("The sum with all parameters",sum(10,20,30))
```

Recursion [Task: implement through recursion]

```
def factorial (n):
    sum = 1
    for i in range (1, n+1):
        sum * = i
    return sum


print ( “Factorial of 3 is ”, factorial (3) )
```

Exception handling

try:

    Something

except:

        some_message

```
try:
    num = int(input('enter the number:'))

except ValueError:
    print('not integer')
except:
    print('default ex')
finally:
    print('finally')
```

File i/o [Text Files]

WRITING

```
file = open("biodata.txt", ‘w+')
file.write("[your name] \n [reg no] \n [class]")
file.close()
```

READING

```
file = open("biodata.txt", 'r+')
#read till end of the file
data = file.read()
print(data)
file.close()
```

File i/o [CSV Files]

```
import csv

file = open('Event2.csv', 'r+')
reader = csv.reader(file)
for row in reader:
    print (row)
```

```
Timestamp,Username,Name,semester,What are Your Expectations?
2016/12/17 9:17:57 PM GMT+5,abc@gmail.com,filza atif,Semester 2,"to get info about graphics
2016/12/17 9:18:59 PM GMT+5,abc@gmail.com,uzair ahmed,Semester 2,To learn more about filing and graphics in c++
2016/12/17 9:21:08 PM GMT+5,abc@gmail.com,bilal ahmed toor,Semester 2,"learn about project
2016/12/17 9:28:19 PM GMT+5,abc@gmail.com,Mehwish Taqi,Semester 1,
2016/12/17 9:28:38 PM GMT+5,abc@gmail.com,Shaheryar Ali,Semester 2,This could teach me new skills and help in programming.
2016/12/17 9:29:22 PM GMT+5,abc@gmail.com,Talha Ahmed,Semester 2,Something different and interesting.
2016/12/17 9:35:41 PM GMT+5,abc@gmail.com,Abdul wagar,Semester 2,good expectation after hardworking
2016/12/17 9:37:01 PM GMT+5,abc@gmail.com,javeeria arshad,Semester 2,High and good
2016/12/17 9:41:03 PM GMT+5,abc@gmail.com,Muhammad Usama Khan,Semester 2,Really interesting session hopefully
2016/12/17 9:41:06 PM GMT+5,abc@gmail.com,Anum Fatima,Semester 2,Good and high
2016/12/17 9:42:40 PM GMT+5,abc@gmail.com,Mehak muhammad sohail,Semester 2,Msmn
2016/12/17 9:47:36 PM GMT+5,abc@gmail.com,kashif shoukat,Semester 1,3 gpa
```

Exercise

Write the following 2 function:
def ComputeOddSum(input):
def ComputeEvenSum(input):
The function **ComputeOddSum** find the sum of all odd numbers less than input.
The function **ComputeEvenSum** find the sum of all even numbers less than input.

Exercise

Write a recursive function to get sum of all number from 1 up to given number.
Example N = 5 Result must be sum (1+2+3+4+5) =15

Write program so that it counts the total number of lines and words in a file.
Your program should print out something like
Lines: 2
Words: 100

BONUS MODULE
Contents

- **Creating Classes**
- **Creating Objects**
- **Inheritance**
- **Overriding**

Creating classes

```
class Employee:

    empCount = 0
    def __init__(self, name, salary):  #constructor
        self.name = name
        self.salary = salary
        Employee.empCount += 1

    def displayCount(self):
        print ("Total Employee ", Employee.empCount)

    def displayEmployee(self):
        print ("Name : ", self.name, ", Salary: ", self.salary)
```

Creating instances

```
"This would create first object of Employee class"
emp1 = Employee(“Ahmed", 2000)
"This would create second object of Employee class"
emp2 = Employee(“Asad", 5000)
```

Something Interesting

```
emp1.age = 7 # Add an 'age' attribute.
emp1.age = 8 # Modify 'age' attribute.
del emp1.age # Delete 'age' attribute.
```

Inheritance

```
class Parent :  # define parent class
    parentAttr = 100
    def __init__(self):
        print ("Calling parent constructor")
    def parentMethod(self):
        print ('Calling parent method')
    def setAttr(self, attr):
        Parent.parentAttr = attr
    def getAttr(self):
        print ("Parent attribute :", Parent.parentAttr)
```

```
class Child(Parent):  # define child class
    def __init__(self):
        print ("Calling child constructor")
    def childMethod(self):
        print ('Calling child method')
```

Overriding

```
class Parent: # define parent class
    def myMethod(self):
        print ('Calling parent method')


class Child(Parent): # define child class
    def myMethod(self):
        print ('Calling child method')


c = Child()     # instance of child
c.myMethod() # child calls overridden method
```

Module – III Contents

- **Numpy**
- **Declaration**
- **Calculation**
- **2D Numpy Array**

Declaration

```
# Create list baseball

baseball = [180, 215, 210, 210, 188, 176, 209, 200]

# Import the numpy package as np

import numpy as np

# Create a Numpy array from baseball: np_baseball

np_baseball = np.array(baseball)

# Print out type of np_baseball

print(type(np_baseball))
```

Calculations - BMI

```
# height and weight are available as a regular lists
# Import numpy
import numpy as np
# Create array from height with correct units:
np_height_m = np.array(height) * 0.0254
# Create array from weight with correct units:
np_weight_kg = np.array(weight) * 0.453592
# Calculate the BMI: bmi
bmi = np_weight_kg/np_height_m**2
# Print out bmi
print(bmi)
```

2d numpy

```
# Create baseball, a list of lists
baseball = [[180, 78.4],   [215, 102.7],    [210, 98.5],  [188, 75.2]]
# Import numpy
import numpy as np
# Create a 2D Numpy array from baseball: np_baseball
np_baseball = np.array(baseball)
# Print out the type of np_baseball
print(type(np_baseball))
# Print out the shape of np_baseball
print(np_baseball.shape)
```

Basic Statistics

```
# Import numpy
import numpy as np
# Create np_height from np_baseball
np_height = np_baseball[:,0]
# Print out the mean of np_height
print(np.mean(np_height))
# Print out the median of np_height
print(np.median(np_height))
```