Python Writeup Section – A

Python - is general-purpose programming language that can be used effectively to build almost any kind of application that does not need direct access to computers hardware.

History - Python was developed in 1990 by Guido von Rossum

Features of Python

- Python is simple & easy to learn.
- Python supports object-oriented programming(Abstraction, Encapsulation, Inheritance & Polymorphism).
- Python is not compiled and it is designed to be interpreted.
- Python is free and opensource.
- Python provides runtime feedback which is helpful for novice programmers.
- Python has large set of free libraries which provide extended functionality.
- Python has GUI programming support.
- Python is integrated language, easily integrated with c, c++ etc.

Application of Python

- Web Development
- Machine Learning and Artificial Intelligence
- Data Science
- Game Development
- Audio and Visual Applications
- Desktop GUI
- Web Scraping Application
- Business Applications

Basic Elements of Python:

- Python program (aka **script**) is sequence of definition and commands.
- A **command** or **statement** instructs the interpreter to do something.
- These commands are executed by python interpreter in something called as **shell**.
- Note – Python statements don't end with semicolon
- #(hash) is used for single line comment.
- ””” (3 double/single quotes) are used for multiline comments.
- Manual compiling of the code is not needed in Python
- When you run python code, it is first compiled and then interpreted line by line. The compile part gets deleted as soon as the code gets executed

Python IDLE – software used for Python Programming.

- Python IDLE comes with python package.
- IDLE stands for **Integrated Development and Learning Environment**
- ANACODA and Canopy are the popular IDE's of Python.

Assignment – 1 : Fibonacci Series

Accepting user Input

- Python3 has input() function that can be used to get input from a user. It takes string as argument and displays it as prompt in the shell.
- It waits for user to type something, followed by hitting enter. Typed line is treated as string and becomes the value returned by the function.
  Ex: name = input("Enter your name")
- As input function takes input in string format, there is need for **type conversion** (type casts) if one is operating on int or float values
  Ex: num = int(input("Enter a number")) #converting to int

- **Indentation** is semantically meaningful in python. It is used to identify block of code.
- Other programming languages use some sort of bracketing symbols {} to denote block.

Assignment – 2 : Quadratic Equation

Branching Program

- In python, branching or conditional statement has following syntax

```
if Boolean expression:
        true: block of code
else:
        false: block of code
```

- Branching program work on condition, which has 3 parts
    a) Test, i.e. expression that evaluates to True or False
    b) Block of code that is executed if test evaluates to True
    c) & optional block of code that is executed if test evaluates to false

1. **Conditional Execution**–simple if statement

```
If(x%2 == 0):
        print(“Even”)
```

2. **Alternate Execution** – if with else statement

```
if(x%2 == 0):
        print(“Even”)
else:
        print(“Even”)
```

3. **Chained Execution** – if elif else statement (elif ladder)

```
If(x%5 == 0):
        print(“Number divisible by 5”)
elif(x%3 == 0):
        print(“Number divisible by 3”)
elif(x%2 ==0):
        print(“Number divisible by 2”)
else:
        print(“Odd Number”)
```

4. **Nested Execution**– if within another if statement

```
if(x%2 == 0):
        if(x%3 == 0):
                print('Divisible by 2 and 3')
        else:
                print('Divisible by 2 and not by 3')
```

Assignment – 3 : Sum of Natural Numbers

Iteration

- When we want program to do the same thing many times, we use iteration (called looping). It begins with condition and repeats till condition evaluates to true.
- There are two forms of Iteration – for and while statements.
- It is sometimes convenient to exit a loop without executing all the iterations in the loop. By using **break** statement this can be achieved.
- The **continue** statement will allow us to skip one iteration on specific condition.
- while(True) condition can be used if we want to simulate do-while or exit control loop approach. Allowing the iteration to be performed at least once.
- While(True) acts as an infinite loop and needs to be manually exited or break.

```
while(condition):
        statements….
```

Assignment – 4 : Multiplication Tables

For Loops

- For loops provide another way to perform looping activity. It can be used to simplify programs containing iterations. The general form is:
  for(variable in sequence):
  	block of code
- The variable followed by for is bound to be first value of sequence.
- The sequence of values bound to variable is generated using **range()** function.
  It takes three arguments. range(start, stop, step), step can be positive/negative value determining increment/decrement.
- Ex: range(5,40,10) produces sequence (5, 15, 25, 35)
  range(40, 5, -10) produces sequence (40, 30, 20, 10)
- If first argument (start value) is omitted, it defaults to 0 & if last argument (step value) is omitted it defaults to 1. Note: stop value is compulsory.
- The for loop is executed until the sequence is exhausted or break statement is executed within the code block.
- Ex: Consider the code

```
x = 4
for(i in range(x)):
        print(i)
It prints
0
1
2
3
```

- Now, think about the code

```
x = 4
for(i in range(x)):
        print(i)
        x = 5
```

- It raises question of whether changing the value of x inside loop affects number of iterations. Answer is it does not. The arguments to the range function in the line with for are evaluated just before the first iteration & not revaluated for subsequent iterations.
- So above code prints 0 1 2 3 although value of x is changed to 5

Assignment – 5 : Prime Numbers

Variable

- Variable is name/identifier that refers to a value
- Example- pi = 3.142, it first binds variable name pi to objects of type float.
- In python, a variable is just a name. **Assignment** statement associates the name to the left of the = symbol with object denoted by expression to the right of =.
- Apt choice of variable names plays an important role in enhancing readability of code.
- Example a = 3.142   pi = 3.142
  Here, we read the statements- variable a is not giving much clarity but variable pi suggest something related to circle. Such apt names are called **mnemonic variables**
- **Rules for declaring Variable –**
  - In python, variable names can be arbitrary long.
  - Variables can contain both letters and numbers. (It cannot start with a number)
  - Variable names are case sensitive (a and A are two different variable names)
  - Special character _(underscore) is allowed. (It is used when we have multiple words)
  - Python keywords(reserve words) cannot be used as variable names.

Assignment – 6 : Sequential Search

Functions

- **Function** is structured sequence of statements written in order to achieve specific task. It is common to say function takes argument and returns a result.

User-defined function

- In python each function is defined as

```
def name_of_function(list of formal parameters):
        body of function
```

- Ex:

```
def maxnum(x, y):
        if(x>y):
                return x
        else:
                return y
a = int(input("Enter a number"))
b = int(input("Enter another number"))
largest = maxnum(a, b)
print("Largest number is ", largest)
```

- **def**is keyword specifying function.**maxnum** is name of the function.
- **x & y** are formal parameters or arguments. a,b are actual parameters or arguments.
- **largest = maxnum(a,b)** is function call.
- Function body is any piece of python code. The**return** statement can be used only within body which returns control back to function call with some value.
- On function call, controls moves from main program to function. Body of function is performed & after return statement control is transferred back to main program.

Fruitful function and void function

- Fruitful function is something which returns a value. Ex – maxnum() as shown above
- Void function is something which doesn't return a value. Ex – minnum() as shown below
-

```
def minnum(x, y):
        if(x<y):
                print(x, " is smallest")
        else:
                print(y, "is smallest")
a = int(input("Enter a number"))
b = int(input("Enter another number"))
minnum(a, b)
```

Assignment – 7 : Calculator

Statement

- **Statement**- is a piece of code that python interpreter can execute.
- A script(python program) usually contains a sequence of statements.
- Example – Assignment statement x = 2.

Expression

- E**xpression-** is combination of values, variables and operators.
- Operators are special symbols that represent computations like addition, subtraction etc.

Example – a = 25, b = 10, c = a + b

Here a, b, c are operands(variables). +, = are operators, 25, 10 are values

Python Arithmetic Operator

Operator	Description	Example
+ Addition	Adds values on either side of the operator.	a + b = 35
- Subtraction	Subtracts right hand operand from left hand operand.	a – b = 15
* Multiplication	Multiplies values on either side of the operator	a * b = 250
/ Division	Divides left hand operand by right hand operand, result is always a floating number	a / b = 2.5
% Modulus	Divides left hand operand by right hand operand and returns remainder	a % b = 5
** Exponent	Performs exponential (power) calculation on operators	a**b =10 to the power 25
//	Integer/ Floor Division - result is the quotient in which the digitsafter the decimal point are removed.	a//b = 2

Assignment – 8 : String Operations

Operations on String

#Changing Upper and Lower Case Strings

```
string = "hello world"
print(string.upper())
```
Output - HELLO WORLD
```
print(string.lower())
```
Output - hello world
```
print(string.title())
```
Output - Hello World
```
print(string.capitalize())
```
Output - Hello world
```
print(string.swapcase())
```
Output - HELLO WORLD

- **length** of string can be found using len function. Ex: len('abc') is 3
- **Indexing** is used to extract individual characters from a string. Ex: index 'abc'[0] is a
  'abc'[-1] is c & 'abc'[3] is Error: string index out of range.
  Python uses 0 to indicate first element of string & last element will be length-1
- **Slicing** is used to extract substring of arbitrary length.
  If s is string then s[start:end] denotes s starts an index start and ends with index end-1.
  Ex: 'abc'[1:3] is 'bc', 'abc'[:2] is 'ab' (if value before colon is omitted it default to 0)

#Slicing -Use [# : #] to get set of letter

```
word = "Hello World"
print(word[0]) #get one char of the word
```
Output - H
```
print(word[0:3]) #get the first three char
```
Output - Hel
```
print (word[:3]) #get the first three char
```
Output - Hel
```
print (word[-3:]) #get the last three char
```
Output - rld
```
print (word[3:]) #get all but the three first char
```
Output - lo World
```
print (word[:-3]) #get all but the three last character
```
Output - Hello Wo

Assignment – 9 : Selection Sort

Many Values to Multiple Variables

Python allows you to assign values to multiple variables in one line:
```
x, y, z = "Orange", "Banana", "Cherry"
```

One Value to Multiple Variables
```
x = y = z = "Orange"
```

Assignment – 10 : Stack

Stack is implemented using List. Stack follows LIFO – Last In First Out order

- **List** is mutable ordered sequence of values, where each value is identified by an index.
- We use square brackets rather than parentheses to access elements.
- Lists are **mutable**. That means list can be modified after they are created
- **Example**

```
L = [‘I did it all’, 4, ‘you’,22.5]
for i in range(len(L):
    print(L[i]))
```

  produces output

```
I did it all
4
you
22.5
```

1. **L.append(e)** adds the object e to the end of L.

```
#Appending  a list
L = [10,20,30]
L.append(40)
print(L)
```

Output - [10, 20, 30, 40]

2. **L.pop(i)** removes and returns the item at index i in L.
   If i is omitted, it defaults to -1, to remove and return the last element of L.

```
#Poping element from a list
L = [10,20,30,40]
a = L.pop(2)
```

print(a) **Output** – 30

print(L) **Output** – [10,20,40]

Assignment – 11 : File Operations

Files

- File is a place which is used to save data. File can be on any extension (pdf, txt, py etc)
- Python achieves operating-system independence by accessing files through something called a **file-handle**

Opening File

- When we want to read or write a file, we first need to open it.
- fhandle = open("mbox.txt")
  If the open is successful, then OS returns the file handle.
  If the files does not exist, open will fail with FileNotFoundError.

Reading from file

- We can also open file for **reading** using argument 'r' and prints its contents. Python treats file as a sequence of lines, we use for statement to iterate over file's contents.
  fhandle = open('mbox.txt' , 'r')

Writing into file

- We can also open file for **writing** using argument 'w' and write into it.
- fhandle = open('mbox.txt' , 'w')

Appending file

- We can also append the contents of file by using argument 'a'.
  If we don't use appending, then the contents of the mbox will be overwritten.
- fhandle = open('mbox.txt' , 'a')
  name = input("Enter mobile ")
  fhandle.append(name + "\n")
  fhandle.close()

Common functions for accessing files are

- **fh = open(fn, 'w')** fn is a string representing a file name. Creates a file for writing and returns a file handle.
- **fh** = **open(fn, 'r')** Opens an existing file for reading and returns a file handle.
- **fh = open(fn, 'a')** Opens an existing file for appending and returns a file handle.
- **fh.read()** returns a string containing the contents of the file associated with file handle fh.
- **fh.readlines()** returns a list, each element of which is one line of the file associated with the file handle fh.
- **fh.write(s)** write the string s to the end of the file associated with the file handle fh.
- **fh.writeLines(S)** S is a sequence of strings. Writes each element of S to the file associated with the file handle fh.
- **fh.close()** closes the file associated with the file handle fh.