Assessment – 3 – Python Basics:

(2)

1) How do you insert an object/value at a given index in Python List?
2) Why do we need break and continue statements in Python?
3) Explain identity operators in Python?
4) What function would you use to toggle case for a Python string?
5) What is recursion / recursive function explain with example?
6) What does the function zip() do?
7) What is a doc string, How can I display it in python ?
8) How many arguments can the range() function take?
9) What is oops concept, class and objects?
10) How can I list files in a given directory in python?

(5)

Answer any 6 of the below:

1) Explain what is inheritance and polymorphism with an example?
2) Given a list, write a Python program to swap first and last element of the list?
3) Write a program to find mean, median and mode of a list?
4) Write Python program to display all the prime numbers within an interval?
5) Python program to find the factorial of a number provided by the user?
6) Write a python calculator program using functions, that takes 2 numbers and a mathematical sign as input and returns the result of the operation?
7) Write a program to calculate the electricity bill (accept number of units from user) according to the following criteria: Unit Price no First 100 units No Charge Next 100 units Rs 5 per unit After 200 units Rs 10 per unit (For example, if input unit is 350 than total bill amount is Rs2000) ?
8) Write a program to determine whether the number is Armstrong number or not?

Swaroop N.C
PI - B3
28/03/2022

Assessment 3: Python Basics

5 Mark Qn: PROGRAMS.

2) # swap first & last element of the list.

```
#take list input
n= int(input("Enter no. of elements in list")
l=[ ]
for i in range(0,n):
    l.append(int(input("Enter element")))
print("List l =", l)
#swapping
l[0], l[-1] = l[-1], l[0]
print("After swap list l =", l)
```

5

43/50

Output Console:

```
Enter no. of elements in list 4.
Enter element 1
Enter element 2
Enter element 3
Enter element 4
List l = [1, 2, 3, 4]
After swap list l = [4, 2, 3, 1]
```

5M
2-5
3-
04-0
6-3

3) # program to find Mean, Median, mode:

```
#take list input.
n= int (input ("Enter no. of elements in dataset"))
l=[ ]
for i in range (0,n):
        l.append (float(input("Enter scores")))
print("Dataset l",l)

#mean.
Mean = sum(l)/len(l):
print ("Mean is", Mean)

#median
if (n%.2 == 1) :        #odd case.
        i = n//2
        Median = l[i].
else :                  # even case.
        l = n//2
        Median = (l[i-1] + l[i])/2
print ("Median is", Median)

# Mode
from collections import Counter
d = dict (Counter(l))
lval = d.values( )
Mode = d.getkey (max(lval))
print ("Mode is", Mode)
```

5

Output

```
Enter no. of elements in dataset 6
Enter scores 1
Enter scores 2
Enter scores 3
Enter scores 4
Enter scores 5
Enter scores 5
Dataset l [1,2,3,4,5,5]
Mean is 3.33
Median is 3.5
Mode is 5
```

4) # prime no. in an interval.

```
minn = int(input("Enter lower range "))
maxn = int(input("Enter upper range "))
lp=[ ]
for i in range(minn, maxn+1):
    if(i>1):
        for x in range(2,i):
            if(i%x==0):
                break
        else:
            lp.append(i)
print("Prime No. in given range is ",lp)
```

2 marks given

this makes 1 not print range is (1 to 10)

you can put it in if(minn>1) after lp=[ ]

Output

```
Enter lower range: 3
Enter upper range : 10.
Prime No. in given range is [3,5,7]
```

1
10
(2,3,5,7)
1

5) #factorial program

```
def FacRec(x):
    if (x>1):
        res = x * FacRec(x-1)
    else:
        res = 1
    return (res)

n = int(input("Enter no. to find factorial"))
f = FacRec(n)
print("factorial of",n,"is",f)
```

5

Output

Enter no to find factorial 5
factorial of 5 is 120

```
def add(x,y):
    return(x+y)
def sub(x,y):
    return(x-y)
def mul(x,y):
    return(x*y)
def div(x,y):
    return(x/y)

~~sum=add~~
a=int(input("Enter num1"))
b=int(input("Enter num2"))
sum=add(a,b)
dif=sub(a,b)
prod=mul(a,b)
quo=div(a,b)
print(a,"+",b,"=",sum)
print(a,"-",b,"=",dif)
print(a,"*",b,"=",prod)
print(a,"/",b,"=",quo)
```

Output

```
Enter num1 6
Enter num2 3
6+3=9
6-3=3
6*3=18
6/3=2
```

```
def cal(x,y,'s')
if s=='+'
else if s=='-'
3
#correct way
def calc(a,op,b):
    if(op=='+'):
        c=a+b
    elif(op=='-'):
        c=a-b
    elif(op=='*'):
        c=a*b
    elif(op=='/')
        c=a/b
    print(a,op,b,"=",c)
    return(c)
a=float(input("Enter a"))
op=str(input("Enter + -*/"))
b=float(input("Enter b"))
c=calc(a,op,b)
```

op

```
Enter a 6.0
Enter + -*/ + '+'
Enter b 7 7.0
6.0 + 7.0 = 13.0
```

7) #electricity bill.

```
x = float(input("Enter Units consumed "))

if (x <= 100):
    Pay = 0
elif (x > 200):
    Pay = (5*100) + (x-200)*10
else:
    Pay = 5*(x-100)
print("for", x, "units consumed, Pay Rs", Pay)
```

5

Output

Enter Units Consumed 350.

for 350 units consumed, Pay Rs.2000

Extra program - Q.8 Armstrong No

```
str = input("Enter 3 digit number")
num = int(str)
ls = str.split()
li = []
for j in ls:
    li.append(int(j))
print(li)
scube = 0
for K in li:
    scube = scube + (K**3)
print(scube)
if (num == scube):
    print(num," is Armstrong Number")
else:
    print(num," is NOT Armstrong Number")
```

#"153"
#153
["1",'5','3']

365
365%100 → 5
#[1,5,3] — 365%100
365//100
65//10
%1000
//100
#153

Change split into int

str = int(input("enter 3 digit number"))

#1365

nh = n//100
nt = (n%100)//10
nu = n%10

365//100 → 3
365%100 → 65//10 → 6
365//10 → 5

Output:

Enter 3 digit number 153
[1, 5, 3]
153.
153 is Armstrong Number.

2 mark - Theory Qn

```
l = [1,2,3,4]
l.insert(2,2.5)
l = [1,2,2.5,3,4]
```

l.insert (index, value)

2

we use. l.insert(index, value) list function to insert a particular value at particular index.

An example has been illustrated above.

2) Break & continue are mostly used in Python traversables like 'for' loop & while loop too.

'break' →

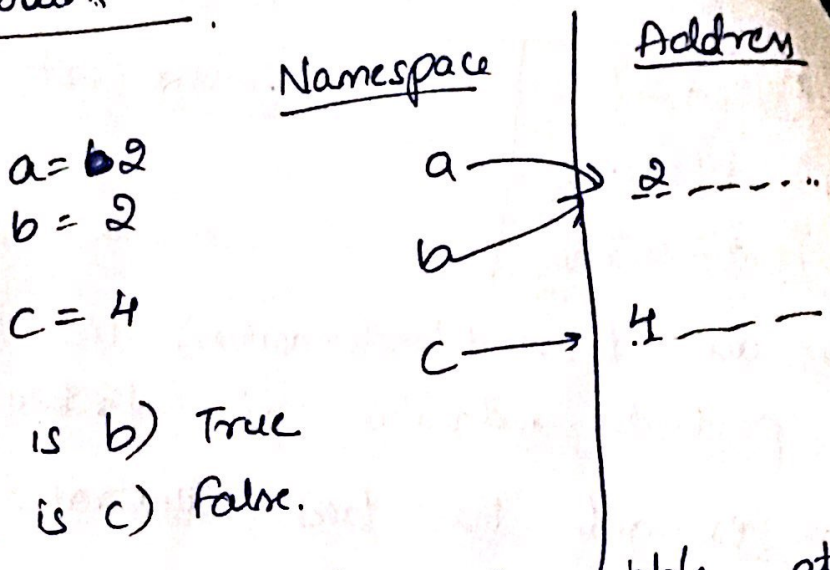
* we mainly use break to break the iterations of for loop in between. process and move control of program next to 'for' block of code.

'continue' →

* it is primarily used to skip a particular iteration in loop statements
* It skips a particular iteration 'x' & move back to iteration 'x+1'

2

3) Identity Operators



(a is b) True
(a is c) false.

In python, rather than storing a variable at particular memory address, a ~~part~~ specific value is placed in memory address, & that value is mapped to. other variables bearing same value.
here in our case a, b.

- Identity operators are Ex (a is b), (a is not b)
- It return true or false boolean, if both. both variables are mapped to same memory address i.e id(a) = id(b) # True / false.

2

4) str = "prime"
str.switchcase() #PRIME is o/p
str.switchcase() #prime is o/p now
we use x.switchcase() to ~~switch~~ toggle case of string ~~between~~ from upper to lower & vice-versa

• Swapcase()

Recursion / Recursive function

→ It is mainly used to call a function within its own definition again

→ It is highly useful in programs such as finding factorial, etc.

ex:

```
def facRec(x):
    if (x>1):
        res = x * facRe(x-1)
    else:
        res = 1
    return (res)

n=5
f = facRec(5)
print(f)
```

output

120

#120 = 5*4*3*2*1.

⑥ Zip()

```
l1= [1,2,3,4]
l2= [5,6,7,8]
l3 = list(zip(l1,l2))
```

l3 = [(5,1),(6,2),(7,3),(8,4)]

→ Zip function is used to bind 2 iterables (list/tuple) into like 'pairs'

→ It is helpful to sort one list as per other list.

- An example has been illustrated above.

7) doc string is a document string, where there info/help about a function.

Ex: def add(a,b):
"""This func adds 2 nums"""
z=a+b
return z

2

add.__doc__()
⇒ This displays string contained within this function between """ ----- """

8) range (start, stop, step)

- range function takes 3 arguments at maximum while start & step are optional
- Ex:1) range(0,10,2)
#o/p is 0,2,4,6,8

2

2) range(5)
#op is 0,1,2,3,4.

10). import os
os.listdir()

2

- by using os.listdir(), we can get list of all files in current working directory.
- we can also pass filepath within braces ( ) to get list of files in a specific directory
os.listdir(filepath)

OOPs

→ Object oriented programming is one of 3 styles of programming, while other 2 being, functional & procedural

→ OOPs prime purpose is to implement real word application & objects

→ It's prime concepts include ~~&~~ inheritence, polymorphism & encapsulation.

Class:

→ It is just like constructors in C or java.

→ Syntax:

```
class Classname ([SuperClass]):
    ——————  } attributes
    ——————  } methods
```

→ It consists of attributes & methods ; Ex: ~~Good~~.
Class Car:

Objects:

→ It represents an instant of class

→ It has some state & existance

→ Ex: 'Red color car.'

```
RedCar = Car()
  ↑       ↑
object   class.
```

2