Python Activity -4
Functions

Q1. Write a Python function to find the sum of all numbers in a list Sample List : [8, 2, 3, 0, 7]
Expected Output : 20

```
# way1

def total(n):
    total = 0
    for x in n:
        total += x
    return total

print(total([8, 2, 3, 0, 7]))


# way2
# using sum() method
print(sum([8,2,3,0,7]))
```

```
# way3
# take input from STDIN
lst = []
num = int(input('How many numbers: '))
for n in range(num):
        numbers = int(input('Enter number '))
        lst.append(numbers)
print("Sum of elements in given list is :", sum(lst))
```

root@krosumlabs:~/PY# python A3.py

```
How many numbers: 5   # list size is determined by user
choice
Enter number 10
Enter number 20
Enter number 30
Enter number 40
Enter number 50
('Sum of elements in given list is :', 150)
```

Q2. Write a Python program to reverse a string Sample String : "1234abcd"

Expected Output : "dcba4321"

File:B1.py

```
def string_reverse(str1):

    rstr1 = ''
    index = len(str1)
    while index > 0:
        rstr1 += str1[ index - 1 ]
        index = index - 1
    return rstr1

print(string_reverse('1234abcd'))
```

File: B2.py

```
1 def string_reverse(str1):
2
3     rstr1 = ''
4     index = len(str1)
5     while index > 0:
6         rstr1 += str1[ index - 1 ]
7         index = index - 1
8     return rstr1
9
10
11        var=raw_input("Enter a your input:")
12        print(string_reverse(var))
```

root@krosumlabs:~/PY# python B2.py
Enter a your input:**Hello**
olleH
root@krosumlabs:~/PY# python B2.py
Enter a your input:**Welcome@1234**
4321@emocleW
root@krosumlabs:~/PY#

Q3. Write a Python program to access a function inside a function.

```
root@krosumlabs:~/PY#  cat -n C1.py
     1
     2def f1():
     3      # this is outerfunction block:"
     4      f2() # nested function
     5      rv=f3() # nested function
     6      print("Return value from f3 function:",rv)
     7      print("Exit from f1 block")
     8
     9def f2():
    10            print("This is f2 block:")
    11            print("Exit from f2 block")
    12
    13      def f3():
    14            print("This is f3 block:")
    15            print("Exit from f3 block")
    16            return "VALUE1"
    17
    18
    19      f1()
    20      print("Exit from script")
```

```
root@krosumlabs:~/PY# python C1.py
This is f2 block:
Exit from f2 block
This is f3 block:
Exit from f3 block
('Return value from f3 function:', 'VALUE1')
Exit from f1 block
Exit from script
root@krosumlabs:~/PY#
```

Q4. Write a python program that passes argument as following file names - /etc/passwd, /var/log/boot.log files. Using os.system() display inputfile details.

File: D1.py

```
import os

def display(a1,a2):
        os.system("ls -l "+a1+" "+a2)


display("/etc/passwd","/var/log/boot.log")
```

root@krosumlabs:~/PY# python D1.py

-rw-r--r-- 1 root root 1791 2018-09-26 15:10 /etc/passwd
-rw-r--r-- 1 root root 1730 2019-02-05 09:23 /var/log/boot.log

Q5. Write a function that takes a list and returns a new list that contains all the elements of the first list minus all the duplicates.

```
def dedupe_v1(x):
  y = []
  for i in x:
    if i not in y:
      y.append(i)
  return y

#this one uses sets
def dedupe_v2(x):
    return list(set(x))

a = [1,2,3,4,3,2,1]
print a
print dedupe_v1(a)
print dedupe_v2(a)
```

```
root@krosumlabs:~/PY# python E1.py
[1, 2, 3, 4, 3, 2, 1]
[1, 2, 3, 4]
[1, 2, 3, 4]
root@krosumlabs:~/PY#
```

Q6. Write a function name as connect. pass argument as Database name, DBI, root, password, Portno to connect function using variable length style and display all the inputs to monitor.

```
def Connect(*args): # tuple type structure
        print(args)



print("1st call:")
Connect("MYSQL","DBD","root","PASSWORD",80)
print("2nd call:")
Connect("MYSQL","DBD","root","PASSWORD")
print("3rd call:")
Connect("MYSQL","DBD","root")
print("4th call:")
Connect("MYSQL","DBD")
print("5th call:")
Connect() # empty call
```

root@krosumlabs:~/PY# python F1.py
1st call:
('MYSQL', 'DBD', 'root', 'PASSWORD', 80)
2nd call:
('MYSQL', 'DBD', 'root', 'PASSWORD')
3rd call:
('MYSQL', 'DBD', 'root')
4th call:
('MYSQL', 'DBD')
5th call:
()
root@krosumlabs:~/PY#

Q7. Predict the output of below codes

a. def display(**kwargs):
print(kwargs.keys())

display(DBI=”/usr/bin/oracle”,TABLE=”EMP”,port=80)

a. def display(**kwargs): # Convert to dictionary
print(kwargs.keys()) # dict.keys() -→[list of keys]

display(DBI=”/usr/bin/oracle”, TABLE=”EMP”,port=80)

```
        # |           |                    |         |      |        |
        #  Key     Value               Key       Value Key   Value
        # Keyword arguments
```

Result :-

['DBI', 'TABLE', 'port'] # fetching list of keys

b. def display():
    count=10
    return count+10,”DATA”,”/etc/passwd”

def display():
    count=10
    return count+10,”DATA”,”/etc/passwd”

print(type(display()) # <’type’ **tuple**’> # more than one return type will be tuple type structure

print(display()) # (11,”DATA”,”/etc/passwd”)   # tuple

c. def display():
    global fname
    fname=”/var/log/boot.log”

```
def display():
    global fname  # global is a python keyword – fname
                  will be global variable
    fname=”/var/log/boot.log”

display()

print(“File name:”,fname)   # Filename:/var/log/boot.log
```

d. def display():
    return [“D1”,”D2”,”D3”],{“K1”:”V1”,”K2”:”V2”}

```
print(display())

def display():
    return [“D1”,”D2”,”D3”],{“K1”:”V1”,”K2”:”V2”}
        # more than one return so it will be tuple()

print(display())  #  multidimensional structure
```

([“D1”,”D2”,”D3”],{“K1”:”V1”,”K2”:”V2”})
tuple of list and tuple of dict.