LAB # 08

LISTING

EXERCISE

A. Point out the errors, if any, and paste the output also in the following Python programs.

1. Code

```
Def max_list( list ):
    max = list[ 0 ]
    for a is in list:
        elif a > max:
            max = a
    return max
print(max_list[1, 2, -8, 0])
```

Output

```
>>> %Run 'kist 2.py'

  2
```

Capital D in def function , elif should be replaced by if and print should not be ther in the call function.

2. Code

```
motorcycles = {'honda', 'yamaha', 'suzuki'}
print(motorcycles)
del motorcycles(0)
print(motorcycles)
```

Output:

```
>>> %Run 'error 3.py'

  ['honda', 'yamaha', 'suzuki']
  ['yamaha', 'suzuki']

>>>
```

In list [] are used istead of {}and in line3 index is also called by [].

3. Code

```
Def dupe_v1(x):
  y = []
  for i in x:
    if i not in y:
```

```
      y(append(i))
      return y

a = [1,2,3,4,3,2,1]
print a
print dupe_v1(a)
```

Output:

```
>>> %Run 'error 2.py'

  [1, 2, 3, 4, 3, 2, 1]

>>>
```

Capital D in def function , append function is used as (.append) and brackets are missing in print function in line 9

B. What will be the output of the following programs:

1. Code

```
list1= [1,2,4,5,6,7,8]
print("Negative Slicing:",list1[-4:-1])
x = [1, 2, 3, 4, 5, 6, 7, 8, 9]
print("Odd number:", x[::2])
```

Output

```
>>> %Run list.py

  Negative Slicing: [5, 6, 7]
  Odd number: [1, 3, 5, 7, 9]
```

2. Code

```
def multiply_list(elements):
    t = 1
    for x in elements:
        t*= x
    return t
print(multiply_list([1,2,9]))
```

Output

```
>>> %Run K:

  18

>>>
```

3. Code

```
def add(x,lst=[] ):
    if x not in lst:
        lst.append(x)
        return lst

def main():
    list1 = add(2)
    print(list1)
    list2 = add(3, [11, 12, 13, 14])
    print(list2)
main()
```

Output

```
>>> %Run -K list 2.py

  [2]
  [11, 12, 13, 14, 3]

>>>
```

C. Write Python programs for the following:

1. Write a program that store the names of a few of your friends in a list called 'names'. Print each person's name by accessing each element in the list, one at a time.

SOURCE CODE:

```
print("\t ***Names of my Friends***")
names=["Talal Moin","Abdul Moiz Khan","Shaheer Khan Qureshi","Komal Shakeel","Tooba Shakeel","Asma Hashim Khan"]

for i in  names:
   print (i)
```

OUTPUT:

```
>>> %Run list.py

          ***Names of my Friends***
  Talal Moin
  Abdul Moiz Khan
  Shaheer Khan Qureshi
  Komal Shakeel
  Tooba Shakeel
  Asma Hashim Khan
```

2. Write a program that make a list that includes at least four people you'd like to invite to dinner. Then use your list to print a message to each person, inviting them to dinner. But one of your guest can't make the dinner, so you need to send out a new set of invitations. Delete that person on your list, use ***del statement*** and add one more person at the same specified index, use the ***insert( )*** method. Resend the invitation.

CODE:

```
#Sending Invitation to the guests
print("***\tInvitation to  the Guests\t***\n")
guest_list=["Talal ","Abdul Moiz Khan","Shaheer","Owais Ahmed"]
for i in guest_list:
    print("->",i,"I invite u on dinner at 10:00 clock  ")

#A guest cannot come to the dinner
print("\n\n>>>",guest_list[3],"can't come at dinner <<<\n\n")

#adding a guest to the list and removing the old guest
guest_list[guest_list.index("Owais Ahmed")]="Naveed"
print("***\tResending Invitation to  the Guests\t***\n")
for x in guest_list:
    print("->",x,"I invite u on dinner at 10:00 clock")
```

OUTPUT:

```
>>> %Run 'task 2.py'

***     Invitation to  the Guests:      ***

-> Talal  I invite u on dinner at 10:00 clock
-> Abdul Moiz Khan I invite u on dinner at 10:00 clock
-> Shaheer I invite u on dinner at 10:00 clock
-> Owais Ahmed I invite u on dinner at 10:00 clock


>>> Owais Ahmed can't come at dinner <<<


***     Resending Invitation to  the Guests:    ***

-> Talal  I invite u on dinner at 10:00 clock
-> Abdul Moiz Khan I invite u on dinner at 10:00 clock
-> Shaheer I invite u on dinner at 10:00 clock
-> Naveed I invite u on dinner at 10:00 clock

>>>
```

3. Write a program that take list = [30, 1, 2, 1, 0], what is the list after applying each of the following statements? Assume that each line of code is independent.

- list.append(40)

  Code:

```
list = [30, 1, 2, 1, 0]
list.append(40)
print(list)
```

Output:

```
>>> %Run 'task 2.py'

  [30, 1, 2, 1, 0, 40]
```

- list.remove(1)

Code:

```
list = [30, 1, 2, 1, 0]
list.remove(1)
print(list)
```

Output:

```
>>> %Run 'task 2.py'

  [30, 2, 1, 0]
```

- list.pop(1)

Code:

```
list = [30, 1, 2, 1, 0]
list.pop(1)
print(list)
```

Output:

```
>>> %Run 'task 2.py'

  [30, 2, 1, 0]
```

- list.pop()

Code:

```
list = [30, 1, 2, 1, 0]
list.pop()
print(list)
```

Output:

```
>>> %Run 'task 2.py'

  [30, 1, 2, 1]
```

- list.sort()

Code:

```
list = [30, 1, 2, 1, 0]
list.sort()
print(list)
```

Output:

```
>>> %Run 'task 2.py'

  [0, 1, 1, 2, 30]
```

- list.reverse()

Code:

```
list = [30, 1, 2, 1, 0]
list.reverse()
```

```
print(list)
```

Output:

```
>>> %Run 'task 2.py'

  [0, 1, 2, 1, 30]

>>>
```

4. Write a program to define a function called 'printsquare' with no parameter, take first 7 integer values and compute their square and stored all square values in the list.

CODE:

```
def printsquare():
    a=list()
    for i in range(1,8):
        a.append(i**2)
    print(a)

printsquare()
```

OUTPUT:

```
>>> %Run 'task 2.py'

  *Following are the squares of the first seven integers stored in a list*
  [1, 4, 9, 16, 25, 36, 49]
```