Chapter 8

A.

1. B
2. B
3. D
4. C
5. D
6. Options are wrong correct answer is ['A',2,3]
7. A
8. A
9. B
10. B

B.

1. True
2. False
3. True
4. False
5. False
6. False
7. True
8. False
9. True
10. True
11. False
12. True
13. True
14. True
15. True
16. True
17. True
18. True
19. True
20. True

C.

1. List can be created using constructor and without using constructor.
2. The different ways through which list is created is as follows

```
#Creating Empty List
>>>L1= list()
>>>L2= []
#Creating List containing elements without constructor
>>>L1=[1,2,3]
#Creating List containing elements using constructor
>>>L1=list ([1,2,3])
```

3. The slicing operator returns a subset of a list called **slice** by specifying two indices, i.e. **start** and **end.** The syntax is:

   Name_of_Variable_of_a_List[Start_Index: End_Index]

   Example

   ```
   >>> L1= ([10, 20, 30, 40, 50]) #Create a List with 5 Different

   Elements

   >>> L1[1:4]
   20,30,40
   ```

   The L1[1:4] returns the subset of the list starting from index the start index 1 to one index less than that of the end index, i.e. 4-1 = 3.

   ```
   >>> L1=([10,20,30,40,50]) #Create a List with 5 Different
   Elements
   >>> L1[2:5]
   [30, 40, 50]
   ```

4. The advantage of using step size in list is programmer can retrieve all the elements occurring after particular interval by specifying step size in a list. The syntax to specify step size is as follows

   List_Name=[Start_Index:End_Index:Step_Size]

   Example:

   ```
   >>> L1=[1,2,3,4]
   ```

```
>>> L1[::2] # Retrieving alternate elements
[1, 3]
```

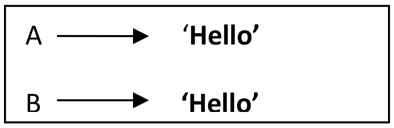
5. Constructor of list is only the inbuild function used to create list. It is created by making use of **list()** function.

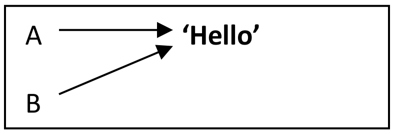
6. Various operators such as +, *, is, in and del operators are used on list to perform different operations. Where + is used to join two lists, * operator is used to replicate elements of a list and del operator is used to remove elements from the list.

7. The use of is operator is used to check if two variables refer to the same object. The following example illustrate the use of is operators.

   Consider two statements
   A = 'Hello'
   B = 'Hello'

We know that, both A and B refers to a string, but we don't know whether they refer to the same string. There are two possible states and they are given as follows.





In the first case A and B refer to the two different objects that have same values. In second case they refer to the same object. Thus, to check whether two variables refer to the same object, programmer can use **is** operator.

8. The del operator is used to delete the elements from the list.

Example:

```
>>> L=[12,1,3,112]
>>> del L[2]
>>> L
```

```
[12, 1, 112]
>>> del L      #Delete List L
```

9. Python supports the concept of list comprehensions. Using list comprehension in coding helps to start writing codes in shorter and effective way. Thus in general code with list comprehensions is much faster as compared to the loop control statements and which results in faster execution.

10. The **count(x)** function is used to count the number of elements present in the list. Following example illustrate the use of count() on list.

Example:

```
>>> List1=['A','B','C','A','B','Z']
>>> List1
['A', 'B', 'C', 'A', 'B', 'Z']
>>> List1.count('A')
2    # Thus, ‘A’  has appeared 2 times in List1
```

11. Elements of List can be converted in two different ways i.e. using in build **reverse()** function on List and second way of creating is by slicing operation.

Example:

```
#Using Reverse() method to reverse the contents of List

>>> L=[10, 20, 30,40,50]
>>> print(L)
[10, 20, 30, 40, 50]
>>> L.reverse()
>>> L
[50, 40, 30, 20, 10]

#List Slicing operation used to reverse the contents of list
>>> L=[10, 20, 30,40,50]
>>> L[:-1]
[50, 40, 30, 20]
```

12. The list is used to convert from string to a list of characters.

Example:

```
>>> p='python'
>>> p
'python'
>>> L=list(p)
>>> L
['p', 'y', 't', 'h', 'o', 'n']
```

13. The elements to list can be added by using append() method or elements can be added directly.

Example:

Adding 10,20, 30, 40 and 50 to the List L using append()

```
>>> L=[]
>>> L=list()
>>> L
[]
>>> L.append(10)
>>> L
[10]
>>> L.append(20)
>>> L
[10, 20]
>>> L.append(30)
>>> L.append(40)
>>> L.append(50)
>>> L
[10, 20, 30, 40, 50]
```

or all the five elements can be added to the list as follows

```
 >>> L = [10,20,30,40,50]
 >>>Print(L)
 >>>[10, 20, 30, 40, 50]
```

Example:

#Reverse using in build reverse () function

```
>>> List1=[1,2,3,4]
>>> List1.reverse ()
>>> List1
[4, 3, 2, 1]
```

#Without using in build function

```
>>> List1=[1,2,3,4]
>>> List1[::]
[4, 3, 2, 1]
```

14. String can be converted into a list of single characters by making use of in build function called as ‘list’ on string.

Example:

```
>>>a='abcdef'
>>>list(a)
>>>['a', 'b', 'c', 'd', 'e', 'f']
```

15. Empty List can be created as follows

```
L1= []          //Empty List or
L1=list ()      // Empty List using Constructor
L1=[10,20,30,40,50]   // List with five elements
```

16.
   a. Error
   b. ['a', 'b', 'c', 'd', 'e', 1, 2, 3]
   c. [1, 2, 3, 1, 2, 3]
   d. [1, 2, 3, 1, 2, 3]

17.
   a. 100
   b. 500
   c. 1200
   d. Error
   e. 2

18.

```
   a. [12, 23, 45, 23, 12, 23, 45, 23]
   b.  [12, 23, 45, 23, 12, 23, 45, 23]
   c.  [12, 23, 45, 23, 23]
   d. [12, 23, 45, 23, 12, 23, 45, 23]
   e. [12, 23, 45, 23, 23, 45, 23, 12]

19.

   a.   [10, 23, 5, 56, 78, 90]
   b.   [10, 23, 5, 56]
   c. [10, 23, 5, 56, 78]
   d. [90]
   e. [90, 78, 56, 5, 23, 10]
   f. [10, 23, 5, 56, 78]
   g. [10, 23, 5, 56]



20.
   a.  [90, 89, 7, 45, 12]
   b. [7, 12, 45, 89, 90]
   c. Error
   d. Error
   e. []

21.   can't multiply sequence by non-int of type 'list'

22.
```

```
def rev(List1):
    return(List1[::-1])

List1=[1,2,3,4]
print(rev(List1))
```

Programming Assignments

1.

```
def Replicate_n_times(Lst,n):
    print('List before Replicating ',n,'times:')
    print(Lst)
    Lst = Lst * n
    Lst.sort()
    print('List after Replicating ',n,'times:')
    return Lst
#Sample test:
Lst=[1,2,3,4]
x = Replicate_n_times(Lst,2)
print(x)
```

Output

```
List before Replicating  2 times:
[1, 2, 3, 4]
List after Replicating  2 times:
[1, 1, 2, 2, 3, 3, 4, 4]
```

2.

```
Lst = [1,23,0,9,0,23]
No_dup = []
for i in Lst :
    if i not in No_dup :
        No_dup.append(i)
        print(i ,'occurs',Lst.count(i),'times')
```

Output

```
1 occurs  1 times
23 occurs 2 times
0 occurs  2 times
9 occurs  1 times
```

3.

```
def remove_negative(Lst):
    print('List before Removing Negative Elements:')
    print(Lst)
    for i in Lst :
        if i < 0 :
            Lst.remove(i)
    return Lst
#Sample test:
Lst = [-1, 0,2,-4,12]
x = remove_negative(Lst)
print('List  after removing Negative elements')
print(x)

Output
List before Removing Negative Elements:
[-1, 0, 2, -4, 12]
List  after removing Negative elements
[0, 2, 12]
```

4.

```
def remove_negative(Lst):
    print('List before Removing Negative Elements:')
    print(Lst)
    for i in Lst :
        if i < 0 :
            Lst.remove(i)
    return Lst
#Sample test:
Lst = [-1, 0,2,-4,12]
x = remove_negative(Lst)
print('List  after removing Negative elements')
print(x)

Output
 List Elements Before Duplication
 [1, 2, 3]
 List Elements After Duplication
 [1, 1, 2, 2, 3, 3]
```

5.

```
List1=[3,17,9,2,4,8,97,43,39]
print('List1= ',List1)
lst = []
```

```
print('Prime Numbers from the List are as Follows:')
for a in List1 :
    prime = True
    for i in range(2, a):
        if (a%i == 0):
            prime = False
            lst.append(False)
            break
    if prime:
        lst.append(True)
print(lst)
```

Output

```
 List1=  [3, 17, 9, 2, 4, 8, 97, 43, 39]
 Prime Numbers from the List are as Follows:
 [True, True, False, True, False, False, True, True, False]
```

6.

```
def removeFirstAndLast(numbers):
    a=numbers
    a=a[1:len(a)-1]
    return (a)
Lis1=[10,20,30,40,50]
print(removeFirstAndLast(Lis1))
```

Output:

```
 [20, 30, 40]
```

7.

```
def Extract_Even(numbers):
    a=list()
    for num in numbers:
       if num%2==0:
            a.append(num)
    return (a)
List=[1,2,3,4,5,6]
print(Extract_Even(List))
```

Chapter 8

Output

```
[2,4,6]
```