Data Structures - I

Insertion in a linear list

1) Unordered list

   Simply add the element at the end

2) Ordered list

   Element to be added at the right position without altering the order, to achieve this rest of the elements to be shifted.

3) Using method in Python

   insert() method

   bisect() module

Insertion in a linear list(Unsorted list)

```
# insertion in unsorted list

List = [34,56,12,8,82,90,16,18]

print('Original List ',List)

element = int(input('Enter the element to be inserted '))

List.append(element)

print('List after insertion ',List)
```

Insertion in a linear list(Sorted list)

```
# insertion in sorted list
def search(List,element):
    for i in range(0,len(List)):
        if element<=List[i]:
            return i
    return -1
List = [34,56,12,8,82,90,16,18]
print('Original List ',List)
element = int(input('Enter the element to be inserted '))
List.sort()
pos=search(List,element)
if pos==-1 :
    List.append(element)
else:
    List = List[:pos]+[element]+List[pos:]
print('List after insertion ',List)
```

Insertion in a linear list(Sorted list using insert())

```
# insertion in sorted list  using insert()
def search(List,element):
    for i in range(0,len(List)):
        if element<=List[i]:
            return i
    return -1
List = [34,56,12,8,82,90,16,18]
print('Original List ',List)
element = int(input('Enter the element to be inserted '))
List.sort()
pos=search(List,element)
if pos==-1 :
    List.append(element)
else:
    List.insert(pos,element)
print('List after insertion ',List)
```

Insertion in a linear list(Sorted list using bisect module)

```
import bisect

List = [34,56,12,8,82,90,16,18]
print('Original List ',List)
element = int(input('Enter the element to be inserted '))
List.sort()
pos = bisect.bisect(List,element)
bisect.insort(List,element)
print('List after insertion ',List)
```

Deletion in a linear list

```
# deletion  sorted list
def search(List,element):
    for i in range(0,len(List)):
        if element==List[i]:
            return i
    return -1
List = [34,56,12,8,82,90,16,18]
print('Original List ',List)
element = int(input('Enter the element to be deleted '))
List.sort()
pos=search(List,element)
if pos==-1 :
    print('Search element is not found ')
else:
    del List[pos]
print('List after deletion ',List)
```

List Comprehensions

1) List = [] for i in range(1,10): List.append(i) print(List)	List = [i for i in range(1,10)] print(List)
Output	[1, 2, 3, 4, 5, 6, 7, 8, 9]

2)
List = []
for i in range(10,100,5):
  if i%4 == 0:
  List.append(i)
print(List)
	List = [i for i in range(10,100,5) if i%4 == 0]
	print(List)

Output	[20, 40, 60, 80]

List Comprehensions

3)
a=[]
for x in [20,12,4,5,16,3]:
    if x<10:
        a.append(x)
    else:
        a.append(x*2)
print(a)
	a = [x if x<10 else x*2 for x in [20,12,4,5,16,3]]
	
	print(a)

Output	[40, 24, 4, 5, 32, 3]

List Comprehensions

1) Create a list using for loop and list comprehension method to store the list with double the values of numbers 1 to 10
2) Suppose List =[31,15,17,39,36,24,27] are the elements of a list, create a new list called VList which will contain only the elements which are multiples of 3 in the given list (write using for loop as well as using list comprehension method)

List Comprehensions

1) Create a list using for loop and list comprehension method to store the list with double the values of numbers 1 to 10

a=[] for x in range(1,11): a.append(x*2) print(a)	a =[x*2 for x in range(1,11)] print(a)
Output	[2, 4, 6, 8, 10, 12, 14, 16, 18, 20]

List Comprehensions

2) Suppose List =[31,15,17,39,36,24,27] are the elements of a list, create a new list called VList which will contain only the elements which are multiples of 3 in the given list (write using for loop as well as using list comprehension method)

List =[31,15,17,39,36,24,27]
List1=[]
for x in List:
  if x%3 == 0:
  List1.append(x)
List=List1
print(List)
	List =[31,15,17,39,36,24,27]
	List = [x for x in List if x%3 == 0]
	print(List)

Output	[15, 39, 36, 24, 27]

List Comprehensions

A List Comprehension is a concise description of a list that shorthands the list creating for loop in the form of a single statement.

Advantages :

1) Code Reduction
2) Faster Code Processing (List Comprehension are executed faster than their equivalent for loop for two reasons)

i) Python will allocate the list's memory first, before adding the elements to it, instead of having to resize on runtime

ii) append() can be avoided, reducing function overhead.

Note : Don't use List Comprehensive when multiple conditions to be checked.

Type B - Application Based Question

1) Create a list SqLst that stores the doubles of elements of another list NSqLst. Following code is trying to achieve this. Will this code work as desired? What will be stored in SqLst after following code?

   NSqLst =[3,5,7,6,8,9,1]

   SqLst = NSqLst * 2

Type B - Application Based Question(Answer)

1) **Output : [3,5,7,6,8,9,1,3,5,7,6,8,9,1]**

2) Change the above code so that SqLst stores the doubled numbers in ascending order.

```
SqLst =[3,5,7,6,8,9,1]

nSqLst = [i*2 for i in SqLst]

nSqLst.sort()

print(nSqLst)
```

Type B - Application Based Question

3) Consider the List L = [1,4,9,16,25,36,49,64,81,100]. Write a code using a list comprehension that takes the list L and makes a new list that has only the even elements of this list in it.

4) Write equivalent list comprehension for the following code:

```
target1=[]

source=[1,4,9,16,25,36,49,64,81,100]

for number in source:

    if number & 1:

        target1.append(number)
```

Type B - Application Based Question(Answer)

3)

```
L=[1,4,9,16,25,36,49,64,81,100]

L=[i for i in L if i%2==0]

print(L)
```

4)

```
target1=[]

source=[1,4,9,16,25,36,49,64,81,100]

target1 = [number for number in source if number&1==1]

print(target1)
```

Type B - Application Based Question

5) Write equivalent for loop for the following list comprehension:

```
Gen = [i/2 for i in [0,9,21,32]]

print(Gen)
```

6) Predict the output of the following code if the input is :

i) 12,3,4,5,7,12,8,23,12 ii) 8,9,2,3,7,8

Code :

```
s=eval(input('Enter a list :'))
n=len(s)
t=s[1:n-1]
print(s[0]==s[n-1] and t.count(s[0])==0)
```

Type B - Application Based Question(Answer)

5) Write equivalent for loop for the following list comprehension:

```
Gen=[]
for i in [0,9,21,32]:
    Gen.append(i/2)
print(Gen)
```

6) Predict the output of the following code if the input is :

i) 12,3,4,5,7,12,8,23,12 ii) 8,9,2,3,7,8

i) False ii) True

Type B - Application Based Question

7) Predict the output

```
def h_t(NLst):
    from_back = NLst.pop()
    from_front = NLst.pop(0)
    NLst.append(from_front)
    NLst.insert(0,from_back)

NLst1 = [[21,12],31]
NLst3 = NLst1.copy()
NLst2 = NLst1
NLst2[-1] = 5
NLst2.insert(1,6)
h_t(NLst1)
print(NLst1[0],NLst1[-1],len(NLst1))
print(NLst2[0],NLst2[-1],len(NLst2))
print(NLst3[0],NLst3[-1],len(NLst3))
```

	0	1	2
NLst1 NLst2	5	6	[21,12]
NLst3	[21,12]	31	

from_back=5
from_front=[21,12]

31 1 3
31 1 3
[21,12] 31 2

5 [21,12] 3
5 [21,12] 3
[21,12] 31 2

Type B - Application Based Question

7)

5 [21, 12] 3
5 [21, 12] 3
[21, 12] 31 2

8) Predict the output

```
ages = [11,14,15,17,13,18,25]
print(ages)
Elig = [x for x in ages if x in range(14,18)]
print(Elig)
```

Type B - Application Based Question

8)Output

[11, 14, 15, 17, 13, 18, 25]
[14, 15, 17]
9) Predict the output

```
L1=[x ** 2 for x in range(10) if x%3 == 0]
L2 = L1
L1.append(len(L1))
print(L1)
print(L2)
L2.remove(len(L2)-1)
print(L1)
```

Type B - Application Based Question

9) Output

[0, 9, 36, 81, 4]
[0, 9, 36, 81, 4]
[0, 9, 36, 81]

10) Predict the output

```
def even(n):
    return n%2 == 0

list1=[1,2,3,4,5,6,7,8,9]
ev =[n for n in list1 if n%2 == 0]
evp = [n for n in list1 if even(n)]
print(evp)
```

Type B - Application Based Question

10) [2, 4, 6, 8]

11) Predict the output :

b = [[9,6],[4,5],[7,7]] x = b[:2] x.append(10) print(x)	b = [[9,6],[4,5],[7,7]] x = b[:2] x[1].append(10) print(x)

Type B - Application Based Question

11) Predict the output :

[[9, 6], [4, 5], 10]	[[9, 6], [4, 5, 10]]

Type B - Application Based Question

12. Find the error. Consider the following code, which runs correctly at times but gives error at other times. Find the error and its reason.

```
Lst1 = [23,34,12,77,34,26,28,93,48,69,73,23,19,88]    1
Lst2 = []                                              2
print('List 1 originally is ',Lst1)                    3
ch = int(input('Enter 1/2/3 and predict which operation was performed'))    4
if ch == 1:    5
    Lst1.append(100)    6
    Lst2.append(100)    7
elif ch == 2:                8
    print(Lst1.index(100))      9
    print(Lst2.index(100))      10
elif ch == 3:                   11
    print(Lst1.pop())           12
    print(Lst2.pop())           13
```

Type B - Application Based Question

12. Solution :

```
Lst1 = [23,34,12,77,34,26,28,93,48,69,73,23,19,88]   1
Lst2 = []          2
print('List 1 originally is ',Lst1)   3
ch = int(input('Enter 1/2/3 and predict which operation was performed'))    4
if ch == 1:                          5
    Lst1.append(100)                 6
    Lst2.append(100)                 7
elif ch == 2 and len(Lst2) != 0:                 8
    print(Lst1.index(100))                       9
    print(Lst2.index(100))                       10
elif ch == 3 and len(Lst2) != 0:
    print(Lst1.pop())
    print(Lst2.pop())
```

Type B - Application Based Question

13. Find the error: Consider the following code(s) and predict the errors(s) and output after correcting :

i) y for y in range(100) if y %2 == 0 and if y%5 == 0

```
print(y)
```

ii) (y for y in range(100) if y %2 == 0 and if y%5 == 0)

```
print(y)
```

Type B - Application Based Question

13. Find the error: Consider the following code(s) and predict the errors(s) and output after correcting :

i) y for y in range(100) if y %2 == 0 and if y%5 == 0
  print(y)
y =[y for y in range(100) if y %2 == 0 and y%5 == 0]

Output :

[0, 10, 20, 30, 40, 50, 60, 70, 80, 90]

ii) (y for y in range(100) if y %2 == 0 and if y%5 == 0)

y =[y for y in range(100) if y %2 == 0 and y%5 == 0]

[0, 10, 20, 30, 40, 50, 60, 70, 80, 90]

Type B - Application Based Question

14 Find the errors in the following list comprehension:

[“good” if i<3: else: “better” for i in range(6)]

Also predict the output

Type B - Application Based Question

Correction

```
x = [ "good" if i<3  else "better" for i in range(6)]

print(x)
```

Output :

['good', 'good', 'good', 'better', 'better', 'better']