UNIT-3

LISTS

mutable

Creating a list using the constructor of the list class :-

```
L1 = list()
L2 = list([10,20,30])
L3 = list(["Apple","Banana","Grapes"])
L4 = list(range(0,6))
L5 = list("xyz")
L6 = list("John","male",25,10.5])
```

Creating a list without using constructor:-

```
L1 = [10, 20, 30]
L2 = ["Apple","Banana","Grapes"]
L3 = ["John","male",25,10.5]
```

Accessing the elements of a List :->

```
L1 = [10, 20, 30, 40, 50]
print(L1[2])        # 30
```

List slicing :-

```
print(L1[2:5])      # (30,40,50)
print(L4[0:5:2])    # 30,50
print(L1[::-1])     # 50,40,30,20
print(L1[-1:0:-1])  # 50,40,30,20
```

Python Inbuilt Functions for Lists :→

len() → Returns the no. of elements in a list
max() → Returns the elements with greatest value
min() → Returns the element with minimum value
sum() → Returns the sum of all the elements of list
shuffle() → shuffles the elements randomly

Exp :-
```
L1 = [10, 20, 30, 40, 50]
print (len(L1))
print (max(L1))
print (min(L1))
print (sum(L1))
```

```
import random
L1 = [10, 20, 30, 40, 50]
print (L1[2])
print (L1)
print (random.shuffle (L1))
print (L1[2])
print (L1)
```

Note: from random import shuffle
print (shuffle (L1))

Exp:
```
L1 = [1, 2, 3]
L2 = [4, 5, 6]
print (L1 + L2)
        # [1, 2, 3, 4, 5, 6]
```

Exp:
```
L1 = [1, 2, 3]
print (3 * L1)
        # [1, 2, 3, 1, 2, 3, 1, 2, 3]
```

Eg:
```
L = [1,2,3,4,5,6]
print(10 in L)      # False
print(10 not in L)  # True
print(4 in L)       # True
```

Identity Operator :-

```
x = "microsoft"
y = "microsoft"
print(x is y)       # True
print(x == y)       # True
```

Note:- Python created only one string object and both x and y refer to the same objects ~~as well~~ However we created two lists with same elements python creates two different object as well

```
x = ["A", "B", "C"]
y = ["A", "B", "C"]

print(x is y)       # False
print(x == y)       # True
```

The del operator

The del operator stands for delete. The del operator is used to remove the elements from list

```
L = [10, 20, 30, 40, 50, 60, 70]
del L[2]
print(L)        # 10, 20, 40, 50, 60, 70
del (L[-1])
print(L)        # 10, 20, 40, 50, 60
del(L[:])
print(L)        # Removed
```

Important

List Comprehension

Eg: Without list comprehension

```
L = [10, 20, 30, 40, 50]
for i in range (0, len(L))
    L[i] = L[i] + 10
print (L)
        # 20, 30, 40, 50, 60
```

With list comprehension

```
L = [10, 20, 30, 40, 50]
L = [n+10 for n in L]
print(L)
```

- print(L) → Output variable
- n → Variable referencing an input sequence
- L → An input sequence

(P) WPP to create a list with elements 1,2,3,9,5 Display even elements of the list using list comprehension

Soln:

```
L = [1,2,3,4,5]
L = [x for x in L if x % 2 == 0]
print(L)
```

(P) Consider the list with mixed type of such L1 = [1, 'x', 4, 5, 6, '2', 9, 'a', 8, 4] create another list using list comprehension which consist of only the entager elements present within the list L1.

```
L1 = [1, 'x', 4, 5, 6, '2', 9, 'a', 8, 4]
L2 = [x for x in L1 if type(x) == int]
print(L2)
```

(*) L = ['X', 'Y', '2']

```
L.append('A')
print(L)    # ['X', 'Y', '2', 'A']
```

(*) clear() #Remove all element

(*) L = ['X', 'Y', '2', 'A', 'A']

```
L.count('A') #2

L1 = L.copy()
L1 = L[:]
```

(*) L = [10, 20, 30, 50, 60, 70]
L.insert(3, 40)
print(L) # [10, 20, 30, 40, 50, 60, 70]

L.pop(1)
print(L) # [10, 30, 40, 50, 60, 70]

pop with index value
↳ It will remove the element by it's index value

L.pop() # [10, 30, 40, 50, 60]

pop without index value
→ it will remove last element of a given list

(*) L = ['A', 'B', 'C', 'D', 'B']
L.remove('B')
print(L) # ['A', 'C', 'D', 'B']

By remove function it will remove first occurance of the Element

L.reverse()
print(L) # ['B', 'D', 'C', 'A']

(*) L = ['G', 'F', 'A', 'C', 'B']
L.sort()
print(L) # ['A', 'B', 'C', 'F', 'G']

Imp

(*)
```
L= ['Red', 3]
L.extend("Green")
print(L)        # ['Red', 3, 'G', 'r', 'e', 'e', 'n']
```

(*)
```
P = "Python"
L = list(P)
print(L)        # ['P','y','t', 'h', 'o', 'n']
```

(P) create a list of 5 element pass the list to a function and print the contents of the list inside the function.

```
def print_list(list):
    for i in list:
        print(i, end = " ")
list = [10, 20, 30, 40, 50]
print_list (list)
```

(P) create a list of five element. pass the list to a function and compute the avg of five no's

```
def calculate_avg (list):
    avg = sum(list) / len(list)
    print("avg:", avg)
list = [10, 20, 30, 40, 50]
calculate_avg (list)
```

(P) WPP to split_list(list, n) where list list is split into two parts and the length of 1st first part is given as n

```
Lst = [1, 2, 3, 4, 5, 6]
split_list(Lst, 2)
Lst1 = [1, 2]
Lst2 = [3, 4, 5, 6]
```

soln

```
def split_list(lst, n)
    lst1 = [ : n]
    lst2 = [n : ]
    print(Lst1)
    print(Lst2)
lst = [1, 2, 3, 4, 5, 6]
split_list (lst, 2)
```

(P) WPP to pass a list to a function and return / print reverse list

```
def reverse_list(list):
    list.reverse()
    return list
list = [10, 20, 30, 40, 50]
print(reverse_list(list))
```

(8) write a function that accepts a positive integer k and return a list that contains the first five multiple of k

```
def multiple_value(k):
    L = [ ]
    for i in range(1, 6):
        req = k*i
        L.append(req)
    return L
print(multiple_value(10))
```

(9) write a function that accept two the integer a and b and return a list of all the even numbers between a and b (including a not including b)

```
def find_even_no(a, b):
    L = [ ]
    for i in range(a, b):
        if i % 2 == 0:
            L.append(i)
    return L
print(find_even_no(10, 20))
```

STRINGS

The str class:-

Define string with constructer

```
s1 = str()          # creates an empty string object
s2 = str("Hello")   # creates on a string object for Hello
```

Defining string without constructor

```
s1 = " "
s2 = "Hello"
```

Basic Inbuilt python function for string

```
a = "python"
len(a)   # returns total no. of characters present in string

max(a)   # return largest alphabet from the string according to ASCII value of character

min(a)   # return smallest alphabet from the string according to ASCII value of characters
```

The index [ ] operator:

```
s1 = "Python"
s1[0]  # P
s1[5]  # n
s1[6]  # Index error: string index out of range
```

Accessing characters via Negative index :-

```
s1 = "Python"
s1[-1]    # n
s1[-5]    # y
```

Accessing characters via ~~string~~ slicing

```
s1 = "Python"
print(s1[2:5])    #'tho'
```

Traversing string with for & while loop :-

(P) WPP to traverse all the elements of a string using for loop.

```
I = "India"
for ch in I:
    print(ch, end = " ")
```

```
for ch in range(0, len(I)):
    print(I[ch], end = " ")
```

```
c = 0
while (c<=len(I))
```

(P) Write a program to traverse every second ~~in~~ characters of a string using the for loop

```
s = "I love python programming"
for ch in range(0, len(s), 2):
    print(ch, end = " ")
```

Immutable Strings :→

```
str = "I love Python"
str[0] = "U"
print(str)  # Type Error: 'str' object does not support
              item assignment that's why string
              is immutable data type
```

Note:

If you want to change the existing string the best way is to create a new string that is a variation of the original string

```
str = "I love python"
str = "U" + str[1:]
print(str)  # "U love pythonx"
```

+, *, membership and identity operator :-

+ operator :- [concatenation]

```
s1 = "IIT"
s2 = "Delhi"
print(s1+s2)        # "IIT Delhi"
```

* operator :

```
s = "Delhi"
print(3*s)          # "Delhi Delhi Delhi"
```

membership operator

```
s = "python programming"
print('python' in s)      #True
print('Java' in s)        #False
print('Java' not in s)    #True
print('prog' in s)        #True
```

Identity operator

```
A = "python"
B = "python"
print(A is B)       #True
```

#identity operator give true when both string creates same objects (share same memory location)

(P) WPP to print all the letter which appear in word 1 as well as word 2.

```
word 1 = " USA North America"
word 2 = "USA south America "

for letter in word 1:
    if letter in word 2:
        print (letter, end=" ")
                    # USA ath America
```

Imp

(P) WPP to print all the words which appear in word 1 as well as word 2

```
word 1 = "USA North America"
word 2 = " USA south America"
L1 = word1.split()
L2 = word2.split()
print(L1)        # ['USA', 'North', 'America']
print(L2)        # ['USA', 'south', 'America']
for word in L1:
    if word in L2:
        print(word, end=" ")

                    # " USA, America "
```

String operation

String comparision

① s_1 = "abcd"

s_2 = "ABCD"

```
print (s1 > s2)    #True
print (s1 = s2)    #false
```

② s_1 = "abc"

s_2 = "abc"

```
print(s1 == s2)    #True
```

Testing strings

#1
```
s = "Python programming"
print (s.isalnum())    #false
```

#2
```
s = "1John"
print(s.isalnum())    #True
```

#3
```
s = "1234"
print(s.isdigit())    #True
```

#4
```
s = "1234Cyo"
print (s.isdigit())    #false
```