- **Lists**
  - **Lists are mutable sequences of python.**
  - **We can change list elements.**
  - **It can contain values of mixed datatypes.**
- **Craeting List**

  Lists are formed by placing comma –separated list of expressions in square brackets.

  Ex:- [2,3,5]

  ['a','e','i']
- **Types of Lists**

  1. The empty list :- It is equivalent of 0 or ' '.

  An empty list can be created as :- L=list( )

  2. Long lists :- A long list contains many elements.

  Ex:- [1,2,3,4,5,6,7,6,56,4,4,5,6,6,7,7,,72,3,5,8,7,6,5

  3,6,4,7,5,89,6,9,5,5,9,]

  3. Nested lists :- A list can have an element in it, which is itself is a list. Such a list is called nested list.

ex:- l = [1,2,[5,6],7]

Creating list from existing sequences

It can be done by l=list(<sequence>).

```
Ex:- L =list('hello')
     print(L)
```

Output :- ['h','e','l','l','o']

eval( ) function

- **It can be used to evaluate and return the result of an expression given as string.**

Ex:- eval('2' + '5') # to be corrected

=> 25

- **Length :- Function lel(L) returns the number of items(count) in the list L.**

Ex:- l= [1,2,3]

len(l) => 3

- **Indexing a list/ Accessing individual element**

  L[i] returns the item at index I (the first item has index 0), and
  L[I,j] return a new list, containing the objects at indexes between I and j (excluding index j).
  Ex:- l= [1,2,3,4,5,6]
  l[0:3] => [1,2,3]
  l[1:5:2] => [2,4]

Membership operators

'in' – it tells if an element is present in the list or not.
'not in ' – just opposite of 'in'
L=[1,2,3]
2 in L => True

Concatenation and replication operators

+ :- It adds one list to the end of other.
l1=[1,2,3] l2 = [6,7]
l1 + l2 => [1,2,3,6,7]

* :- It repeats a list.
l1 *3 =>[1,2,3,1,2,3,1,2,3]

Traversing a list

To traverse a list 'for' loop is used.
Syntax :- for <item> in <list>
process each item here

```
L=[1,2,3]
 for x in L :
        print(x)
```

Output :- 1
2
3

Comparing lists

L1 = [1,2,3]
L2 = [1,2,3]
L3 = [2,4,6]
L1 ==L2 => True
L1 == L3 => False

Appending element to a list

L.append(item)

ex:- L = [1,2,3]

L.append(4)

print(L) => [1,2,3,4]

Updating element of a list

L[index]= <new value>

ex:- L = [1,2,3]

L[2] = 10

print(L) => [1,2,10]

Deleting element from a list

L.pop(index) index is optional, if skipped last element is deleted.

ex:- L = [1,2,3]

L.pop( ) => 3

L.pop(1) => 2

Making true copy of a list

ex:- L = [1,2,3]

M = L # now L and M will point to same list

print(L) => [1,2,10]

ex:- L = [1,2,3]

M = L # now L and M will point to same list

To create copy of list :-

L= [1,2,3]

M = list(L)

L[1] = 10

print(L) => [1,10,3]

print(M) => [1,10,3]

Joining Lists

Joining two lists is very easy just like we perform addition, literally :)The concatenation operator +, when used with two lists, joins two lists.

Eg

>>>lst=[1,3,5]

>>>lst=[6,7,8]

>>>lst1+lst2

[1,3,5,6,7,8] // the +operator concatenates two lists and creates a new list

The + operator when used with lists requires that both the operands must be of list types. We can not add a number or any other value to alist. For example , following expressions will result into error:

list+number

list+string

>>> lst=[10,12,14]

>>>lst+2 // Type error : can only concatenate list(not"int") to list

>>>lst+"abc" // type error: can only concatenate list(not"str") to list

Repeating or Replicating Lists

Like strings we can use * operator to replicate a list specified number of times

Eg

```
>>> lst1=[1,3,5]
>>>lst*3
[1,3,5, 1,3,5, 1,3,5,]
```

Like strings, we can only use integer with a * operator when trying to replicate a list

The Len Function

It returns the length of its argument list i.e it returns the number of element in the passed list. The len() function is a Python standard library function.

Eg

```
>>>len([1,2,3])
3
>>>list1=[4,2,[9,1],(4,5)]
>>>len(list1)
```

4

The List Function

Returns a list created from the passed argument, which should be a sequence type (string, List, tuple etc)

If no argument is passed , it will create an empty list. It works as per the following .

Eg

```
>>> list("hello")
['h','e','l','l','o']              //list created a list from the characters of the passed string
>>>a=list((34,51,110))
>>>a
[34,51,110]
>>>b=list()
>>>b     // List without any argument creates an empty list
[ ]
>>>c=list({'a':101,'b':201})
>>>c
```

['a','b'] // list( ) created a list from the keys of the passed dictionary

The index Function

- it returns the index of first matched list item from the list.

```
eg:- L = [1,2,3,4]
     L.index(2)     => 1
```

Eg

```
>>>L1=[13,18,11,16,18,14]
```

```
>>>L1.index(18)
1
>>>L1.index(33)
Valueerror:33 is not in List
```

The append Function

append( ) - it adds an item to the end of a list.

```
ex:- L = [1,2,3,4]
     L.append(5)
     L => [1,2,3,4,5]
```

```
Eg >>>colours=["red","green","blue"]
   >>>colours.append("yellow")
   >>>colours
["red","green","blue","Yellow"]
```

The append() does not return the new list, just modifies the original. To understand this, consider the following example:

```
>>> lst=[1,2,3]
>>>lst2=1st.append(12)
>>>lst2
>>>lst

[1,2,3,12]
```

The extend Function

extend **- method is also used for adding multiple elements(given in the form of a list) to a list.**

extend( ) - it adds multiple elements to a list.

```
ex:- L1 = [1,2,3,4]      L2 = [10,20]
     L1.extend(L2)
     L          => [1,2,3,4,10,20]
```

Difference between append( ) and extend( )

```
ex:- L1 = [1,2,3,4]      L2 = [10,20]
     L1.append(L2)
     L1          => [1,2,3,4,[10,20]]
     len(L1)     => 5
ex:- L1 = [1,2,3,4]      L2 = [10,20]
     L1.extend(L2)
     L1          => [1,2,3,4,10,20]
     len(L1)     => 6
```

The insert( ) **Function**

insert( ) - it inserts an item at a given position.

```
ex:- L = [1,2,3,4]
L.insert(<pos>, <item>)
L.insert(1,5)
L          => [1,5,2,3,4]
L.insert(-5,20)
L          =>[20,1,2,3,4,]
```

 # If index is less than 0 item will be added in the beginning of the list

pop( ) - it removes the item from the list.

```
ex:- L = [1,2,3,4]
     L.pop(<index>)
     x = L.pop(1)
     L          => [1,3,4]

x          => 2
```

remove( ) - it removes the first occurrence of given item from the list.

```
ex:- L = [1,2,3,4,1]
L.remove(1)
L              => [2,3,4,1]
L.remove(5) => error, element not in list
```

The pop method removes an individual item and returns it, while remove searches for an item, and removes the first matching item from the list.

The Clear ( ) **Function**

This method removes all the items from the list and the list becomes empty list after this function. This function returns nothing . It is used as per following format.

```
List . clear ()
For instance, if we have a list L1 as
>>>L1=[2,3,4,5]
>>>L1.clear( )   //it will remove all the items from list L1.
>>>L1
```

Unlike del<lstname> statement clear ( ) removes only the element and not the list element. After clear ( ), the list object still exists as an empty.

count( ) - it returns count of the item in list. If item not found, it return 0.

ex:- L1 = [1,2,3,4,2]

L1.count(2) => 2

Eg L=[13,18,20,10,23]

```
>>>L1.count(18)
2
>>>L1.count(28)

0
```

reverse( ) - it reverse the order of items of list.

ex:- L = [1,2,3]
L.reverse( )
L => [3,2,1]

Takes no argument, return no list, reverses the list 'in place' and does not return any thing

```
>>> t1
['e','l','q','a','q','p']
>>>t1.reverse( )
>>>t1
['p','q','a','q','l','e']
>>>t2=[3,4,5]
>>>t3=t2.reverse( )
>>>t3
>>>t2

[5,4,4]
```

sort( ) - it sorts the list items. By default in increasing order.

ex:- L = [1,3,2]
L.sort( )
L => [1,2,3]

 # to sort a list in decreasing order :-

L.sort(reverse =True)

L => [3,2,1]

sorted( ) - This function takes the name of the list as an argument and returns a new sorted list with sorted element in it.

Eg

```
>>> val=[17,24,15,30]
>>> sval=sorted (val)
>>>sval
[15,17,24,30]
>>>rsval=sorted (val, reverse=True)
>>>rsval
[30,24,17,15]
```

count( ) - it returns count of the item in list. If item not found, it return 0.

ex:- L1 = [1,2,3,4,2]

L1.count(2) => 2

Eg L=[13,18,20,10,23]

```
>>>L1.count(18)
2
>>>L1.count(28)

0
```