List functions or methods:

1. len():

- This function counts number of elements in the list.

Example:

```
>>> List1 = [10,20,'Python',30,True,'Narayana',10,3+4j]
>>> len(List1)                    # 8
```

2. count(element):

- This function counts the no.of occurrences of specific element in the list.

Example:

```
>>> List1 = [10,20,'Python',30,True,'Narayana',10,3+4j]
>>> List1.count(10)               2
```

Example:

```
>>> lst = [1,2,3,True,False,1,0,False]
>>> lst.count(1)                  3
```

Example:

```
>>> lst = [1,2,3,True,False,1,0,False,1+9j,1.1]
>>> lst.count(0)                  3
>>> lst.count(False)              3
```

3. index(object, index_value):

- This function finds the index value for specific element.

Example:

```
>>> List1 = [10,20,'Python',30,True,'Narayana',10,3+4j]
>>> List1.index(10)               0
```

Example:

```
>>> lst = [1,2,3,True,False,1,0,False,1+9j,1.1]
>>> lst.index(1,1)                3
```

Example:

```
>>> List1 = [10,20,'Python',30,True,'Narayana',10,3+4j]
>>> List1.index(10,1)             # 6
```

Index on nested list:

```
>>> lst  = [100, True, 1, 2, 3, 4, 5, 0, [8, 9], 10, 20, 4]
>>> lst[8].index(9)               # 1
>>> lst[8].index(8)               # 0
```

```
        0    1    2  3    4  5  6    7     8       9     10     11

lst =  [  100,  True,  1,  2,   3,  4,  5,    0,    [8, 9],   10,    20,     4   ]

        -12   -11   -10  -9  -8  -7  -6    -5      -4      -3     -2     -1
```

4. append():

- This function adds new element at the end of the existing list.

Example:

```
>>>List1 = [10,20,'Python',30,True,'Srinivas',10,3+4j]
>>> List1.append(50)
>>> print(List1)   # [10, 20, 'Python', 30, True, 'Srinivas', 10, (3+4j), 50]
```

Example:

```
>>> List1 = [10,20,'Python',30,True,'Srinivas',10,3+4j]
>>> List1.append(60)
>>> print(List1)   # [10, 20, 'Python', 30, True, 'Srinivas', 10, (3+4j), 60]
```

NOTE:

- We can also add multiple elements in the list by using append method but those multiple elements work like nested list or sub list in the existing list and also getting single index value for this sub list

```
>>> List1 = [10,20,'Python',30,True,'Srinivas',10,3+4j]
>>> List1.append([0,1,2])
>>> print(List1)
 [10, 20, 'Python', 30, True, 'Srinivas', 10, (3+4j), [0, 1, 2]]
```

5. extend():

- This function adds multiple elements at the end of the existing list as multiple elements

  Example:

  ```
  >>>List1=[10,20,'Python',30,True,'Srinivas',10,3+4j]
  >>> List1.extend([70,80,90])
  >>> print(List1)
  [10, 20, 'Python', 30, True, 'Srinivas', 10, (3+4j), 70, 80,90]
  ```

6. insert(index_position, object):

- This function adds an element at any required index place in the existing list.

  Example:

  ```
  >>>List1=[10,20,'Python',30,True,'Srinivas',10,3+4j]
  >>> List1.insert(2,100)
  >>> print(List1)
  [10, 20, 100, 'Python', 30, True, 'Srinivas', 10, (3+4j)]
  ```

  Example:

  ```
  >>> List1=[10,20,'Python',30,True,'Srinivas',10,3+4j]
  >>> List1.insert(3,'Durga')
  >>> print(List1)
  [10, 20, 'Python', 'Durga', 30, True, 'Srinivas', 10, (3+4j)]
  ```

- We can also add multiple elements in the list at required place by using insert method but those multiple elements work like nested list or sub list in the existing list and it contains single index value.

  ```
  >>> List1=[10,20,'Python',30,True,'Narayana',10,3+4j]
  >>> List1.insert(0,[0,1,2,3])
  >>> print(List1)
  [[0, 1, 2, 3], 10, 20, 'Python', 30, True, 'Srinivas', 10, (3+4j)]
  ```

7. remove(object):

- This function removes specific element in the existing list.
- This function allows one argument and that should be element name.
- search for removing element from 0 index onwords and removing first occurence.

Example:

```
>>> List1 = [10,20,'Python',30,True,'Srinivas',10,3+4j]
>>>List1.remove(10)
>>> print(List1)
[20, 'Python', 30, True, 'Srinivas', 10, (3+4j)]
```

Example:

```
>>> List1 = [10,20,'Python',30,True,'Srinivas',10,3+4j]
>>> List1.remove(True)
>>> print(List1)
[10, 20, 'Python', 30, 'Srinivas', 10, (3+4j)]
```

Example:

```
>>> l = [1, 2, 3, 1, 3]
>>> l
[1, 2, 3, 1, 3]
>>> l.remove(1)   #  only first occurence element deleted
>>> l
[2, 3, 1, 3]
>>>  l.remove(6)
```

ValueError: list.remove(x): x not in list

Note : if value is not in given list then it returns ValueError

```
>>> l = [10, 30, 10]
>>> l.remove()
```

TypeError: remove() takes exactly one argument (0 given)

8. pop(index):

- This function removes specific element from given list.
- This function also allows only one argument and that should be index value for an element.
- If index value not given then by default last element removed automatically from given list and also returns this removed value.

Example:

```
>>> List1 = [10,20,'Python',30,True,'Srinivas',10,3+4j]
>>> List1.pop(1)       # 20
>>> print(List1)   # [10, 'Python', 30, True,'Srinivas', 10, (3+4j)]
```

Example:

```
>>> List1 = [10,20,'Python',30,True,'Srinivas',10,3+4j]
>>> List1.pop(2)          # 'Python'
>>> print(List1)          # [10, 20, 30, True, 'Srinivas', 10, (3+4j)]
```

Example:

```
>>> l = [10,30,60,10,40]
>>> l.pop()
40
>>> l
[10, 30, 60, 10]
>>> l.pop(2)   #  index position 2 value removed and returns also.
60
```

```
>>> l
[10, 30, 10]
```

Example:

```
>>> l = [10, 30, 10]
>>> id(l)
67974328
>>> l2 = l.pop()  #  removed value store in l2.
>>> l
[10, 30]
>>> id(l)
67974328
>>> l2
10
>>> id(l2)
1834313024
```

9. reverse():

- This function reverses the existing list elements.

Example:

```
>>> List1 = [10,20,'Python',30,True,'Srinivas',10,3+4j]
>>> List1.reverse()
>>> print(List1)   # [(3+4j), 10, 'Srinivas', True, 30, 'Python', 20, 10]
```

Example:

```
>>> l = [10,40,60,20,30,40]
>>> l2 = l.reverse()      # reverse the elements in given list only but not assign to l2
>>> l2
>>> # no output returns l2.
>>> l
[40, 30, 20, 60, 40, 10]
>>>
```

10. reversed(object) :

- This method reverse the given list elements but returns the list_reverseiterator object

Example:

```
>>> l = [40, 30, 20, 60, 40, 10]
>>> reversed(l)
<list_reverseiterator object at 0x040EAE10>
```

---->> To get elements from this list_reverseiterator object then use list() method.

```
>>> list(reversed(l))
[10, 40, 60, 20, 30, 40]
```

10. copy():

- This function copies the existing list into new variable.

Example:

```
>>> List1 = [10,20,'Python',30,True,'Srinivas',10,3+4j]
>>> x  =  List1.copy()
>>> print(List1)   # [10, 20, 'Python', 30, True, 'Srinivas', 10, (3+4j)]
>>> print(x)          # [10, 20, 'Python', 30, True, 'Srinivas', 10, (3+4j)]
```

- After copying the given list elements into another list gets new object reference.

Example:

```
>>> l = [40, 30, 20, 60, 40, 10]
>>> l2 = l.copy()
>>> l2
[40, 30, 20, 60, 40, 10]
>>> id(l)
67976448
>>> id(l2)
63174056
```

11.clear():

- This function clears the entire elements from the given list.

Example:

```
>>> List1 = [10,20,'Python',30,True,'Srinivas',10,3+4j]
>>> List1.clear()
>>> print(List1)                    []
>>> del List1
```

12.max():

- This functions finds the maximum value in the given list

Example:

```
>>>List2=[10,20,30,40]
>>> max(List2)                    40
```

13.min():

- This function finds the minimum value in the given list

Example:

```
>>> List2=[10,20,30,40]
>>> min(List2)                    10
```

14. sort():

- This function sorts the elements of given list.

```
>>> lst=[1,9,5,11,2]
>>> lst.sort()
>>> print(lst)                    [1, 2, 5, 9, 11]
>>> l1=[1,2,5,3,7,4,2,True]
>>> l1.sort()
>>>print(l1)                     [1, True, 2, 2, 3, 4, 5, 7]
```

Note: sort() method sorts the elements in given list only but not assign to new variable.

Example:

```
>>> l = [10,40,60,30,20]
>>> l2 = l.sort()
>>> l2
>>>
>>> l
[10, 20, 30, 40, 60]
```

Note: by default this function sorts in ascending order, we can also get in descending order by setting True for reverse.

Example:

```
>>> lst=[1,9,5,11,2]
>>> lst.sort(reverse=True)
>>> print(lst)                    [11, 9, 5, 2, 1]
```

Example:

```
>>> l1=[1,2,5,3,7,4,2,True]
```

```
>>> l1.sort(reverse=True)
>>>print(l1)                    [7, 5, 4, 3, 2, 2, 1, True]
```

del command:

- This command is used to remove any specific element in the list or to remove entire list object permanently.

Removing specific element:

```
>>> List1 = [10,20,'Python',30,True,'Srinivas',10,3+4j]
>>> del List1[0]
>>> print(List1)   # [20, 'Python', 30, True, 'Srinivas', 10, (3+4j)]
>>> List1 = [10,20,'Python',30,True,'Srinivas',10,3+4j]
>>> del List1[4]
>>> print(List1)                [10, 20, 'Python', 30, 'Srinivas', 10, (3+4j)]
```

Removing entire list object:

Example:

```
>>> List1 = [10,20,'Python',30,True,'Srinivas',10,3+4j]
>>> del List1                   # deleting lst
>>> print(List1)          # after deleting
```

NameError: name 'lst' is not defined

Nested List:

- Python supports Nested lists also, it means a list contains another lists.

```
>>> List1 = [10,'Python',5.5]
>>> List2 = [20,30,'Srinivas',3+4j]
>>> List3 = [1,True,2,'Durga']
>>> print(List1)                    [10, 'Python', 5.5]
>>> print(List2)                    [20, 30, 'Srinivas', (3+4j)]
>>> print(List3)                    [1, True, 2, 'Durga']
>>> NestList=[List1,List2,List3]     #creating a list by using existing lists.
>>> print(NestList)
[[10, 'Python', 5.5], [20, 30, 'Srinivas', (3+4j)], [1, True, 2, Durga']]
>>> type(NestList)                          <class 'list'>
>>> print(NestList[0])                      [10, 'Python', 5.5]
>>> print(NestList[1])                      [20, 30, 'Srinivas', (3+4j)]
>>> print(NestList[2])                      [1, True, 2, 'Durga']
```

```
>>>print(NestList[0][0])          10
>>>print(NestList[1][0])          20
>>>print(NestList[2][0])          1
```

Converting a String into List :

```
>>> Str1 = 'Python is very simple and easy language'
>>> print(Str1)                    # Python is very simple and easy language
>>> type(Str1)                     # <class 'str'>
>>> List1 = Str1.split()
>>> print(List1)      # ['Python', 'is', 'very', 'simple', 'and', 'easy', 'language']
>>> type(List1)         # <class 'list'>
```

Converting a List to String :

```
>>>List1=['Python', 'is', 'very', 'simple', 'and', 'easy', 'language']
>>> print(List1)     # ['Python', 'is', 'very', 'simple', 'and', 'easy', 'language']
>>> type(List1)              <class 'list'>
>>> Str2=" ".join(List1)
>>> print(Str2)              Python is very simple and easy language
>>> type(Str2)               <class 'str'>
```

The main difference between String and List is mutation.

1. String is immutable whereas List is mutable.
2. Mutable objects can be altered whereas immutable objects can’t be altered.

List Packing :

A list can be created by using a group of variables, it is called list packing

```
>>> a=10
>>> b=20
>>> c=True
>>> d='Py'
>>> list1=[a,b,c,d]
>>> print(list1)                   [10, 20, True, 'Py']
>>> type(list1)                    <class 'list'>
```

List Unpacking :

1. List unpacking allows to extract list elements automatically.
2. List unpacking is the list of variables on the left has the same number of elements as the length of the list.

```
>>> lst=[10,20,'Python',True]
>>> print(lst)              [10, 20, 'Python', True]
>>> type(lst)               <class 'list'>
>>> a,b,c,d=lst             #list unpacking
>>> print(a)                10
>>> type(a)                 <class 'int'>
>>> print(b)                20
>>> type(b)                 <class 'int'>
>>> print(c)                Python
>>> type(c)                 <class 'str'>
>>> print(d)                True
>>> type(d)                 <class 'bool'>
```