# Methods of list and tuple

01 August 2024 12:38 PM

List built- in methods:

| Method names | Syntax | Functionality |
|---|---|---|
| count( ) | count(value) | 1. It is used for count ion how many times a given element is repeated<br>2. Count returns integer |
| index( ) | index(value, [start_index],[end_index]) | 1. It is used for returning the index position of given element<br>2. If element is not present it returns error |

Example:

```
>>> l = [11,22,33,44,11,55,66]
>>> l.index(11)
0
>>> l.index(11,1)
4
>>> l.index(11,5)
Traceback (most recent call last):
  File "<pyshell#3>", line 1, in <module>
    l.index(11,5)
ValueError: 11 is not in list
>>> l.count(11)
2
>>> l.count(110)
0
```

| | | |
|---|---|---|
| clear( ) | clear( ) | It is used for deleting all the elements of the given list but not the memory |

```
>>> l = [22,33,44,55,66]
>>> l
[22, 33, 44, 55, 66]
>>> l.clear()
>>> l
[]
```

| append( ) | extend( ) | insert( ) |
|---|---|---|
| Syntax:<br>append(data)<br>Data can be SVDT or CDT | Syntax:<br>extend(data)<br>Data can be only CDT | Syntax:<br>insert(index_position,data)<br>Data can be SVDT or CDT |
| It is used for adding element at last | It is used for adding element at last | It is used for adding element at specified index position |
| If we pass collection as argument then it will add entire collection as one element in list | If we pass collection as argument then it extract individual elements and add them at last | If specified index position is more than length then element will add at last |

Example:

```
>>> l = [11,22,33]
>>> l
[11, 22, 33]
>>> l.append(90)
>>> l
[11, 22, 33, 90]
>>> l.append(67)
>>> l
[11, 22, 33, 90, 67]
>>> l.append('hai')
>>> l
[11, 22, 33, 90, 67, 'hai']
>>> l = [11,22,33]
>>> l.extend(100)
Traceback (most recent call last):
  File "<pyshell#9>", line 1, in <module>
    l.extend(100)
TypeError: 'int' object is not iterable
>>> l.extend('hai')
>>> l
[11, 22, 33, 'h', 'a', 'i']
>>> l.extend(['hai',100])
>>> l
[11, 22, 33, 'h', 'a', 'i', 'hai', 100]
>>> l= [11,22,33]
>>> l
[11, 22, 33]
>>> l.insert(0,100)
>>> l
[100, 11, 22, 33]
>>> l.insert(2,[110,200])
>>> l
[100, 11, [110, 200], 22, 33]
```

Deletion methods of list data type:

| pop( ) | remove( ) |
|---|---|
| Syntax:<br>pop([index_position])<br>Default value for<br>index_position = -1 | Syntax:<br>remove(value) |
| Based on given index_position pop will delete the element | Based on given value remove will delete an element<br>If specified element is present for multiple times also it delete only one time |

| If specified index is more than length it will through an error | If specified value is not present it will throw an error |
|---|---|

```
>>> l = [22,33,44,[100,200]]
>>> l
[22, 33, 44, [100, 200]]
>>> l.pop(1)
33
>>> l
[22, 44, [100, 200]]
>>> l.pop(10000000)
Traceback (most recent call last):
  File "<pyshell#4>", line 1, in <module>
    l.pop(10000000)
IndexError: pop index out of range
>>> l.pop()
[100, 200]
>>> l
[22, 44]
>>> l= [22,33,44,[100,200]]
>>> l.remove(44)
>>> l
[22, 33, [100, 200]]
>>> l.remove([100,200])
>>> l
[22, 33]
```

Difference between the sort and sorted

| sort | sorted |
|---|---|
| It is method of list | It is a normal built-in function |
| Sort method is used only with list data type | Sorted can be used on any collection data type |
| Syntax for ascending order:<br>Objectrefernce.sort( )<br>Syntax for descending order:<br>Objectreference.sort(reverse=true) | Syntax for ascending order:<br>sorted(collection)<br>Syntax for descending order:<br>sorted(collection,reverse=true) |
| Sort method will not return any output | Sorted will return the arranged output in the form of list |
| It modifies in the list memory | Actual memory address data will not be modified |

Example:

```
>>> l = [11,90,78,10,34]
>>> l.sort()
>>> l
[10, 11, 34, 78, 90]
>>> l.sort(reverse=True)
>>> l
[90, 78, 34, 11, 10]
>>> l = [11,'hello',78,'hai',34]
>>> l.sort()
Traceback (most recent call last):
  File "<pyshell#6>", line 1, in <module>
    l.sort()
TypeError: '<' not supported between instances of 'str' and 'int'
>>> l = ['Hello','abc','hai','war']
>>> l.sort()
>>> l
['Hello', 'abc', 'hai', 'war']
>>> ord('A')
65
>>> ord('Z')
90
>>> ord('a')
97
>>> ord('z')
122
>>> sorted('python')
['h', 'n', 'o', 'p', 't', 'y']
>>> l = [11,90,78,10,34]
>>> sorted(l)
[10, 11, 34, 78, 90]
```

**Built-in method of tuple:**

1. index( )
2. count( )

Functionality of tuple index ( ) and count( ) method is same as of in list index( ) and count( ) methods.

```
1 t = (11,22,33,44,11)
2 t.index(11)
```

```
0
```

```
1 t.index(11,1)
```

```
4
```

```
1 t.index(11,1,4)
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Cell In[5], line 1
----> 1 t.index(11,1,4)

ValueError: tuple.index(x): x not in tuple
```

```
1 t.count(11)
```

```
2
```

```
1 t.count(110)
```

```
0
```