

List

- Used to store multiple items in a single variable.
- Created by using `[]` (sq. brackets) separated by comma.

Lists items are

i) Ordered.

ii) Mutable.

iii) Allow duplicate values.

List Methods

i) append()

Add a single element to the end of the list.

append(Δ)

I/P :

0 □ □

→

O/P :

0 □ □ Δ

`s = [1, 'aa', 2, "bb"]`

`s.append(1.25)`

`s.append('Hio')`

`s.append([7, 4])`

`s.append((2, 2))`

`s.append({'a': 20})`

O/P.

1. [1, 'aa', 2, 'bb', 1.25]
2. [1, 'aa', 2, 'bb', 1.25, 'H10']
3. [1, 'aa', 2, 'bb', 1.25, 'H10', [7, 4]]
4. [1, 'aa', 2, 'bb', 1.25, 'H10', [7, 4],
(2, 2)]
5. [1, 'aa', 2, 'bb', 1.25, 'H10', [7, 4],
(2, 2), {'a': 20}]

Note :

List, Tuple, set & Dict can
also be appended to List.

ii) Extend()

- Adds all the elements
of an iterable to the end of
the list i.e., [list, tuple, string etc...]

I/P.

Extend(Δ , 0)

0 □ □ → 0 □ □ Δ 0

s = [1, 2, 3, 4, 5]

s.extend([6, 7])

s.extend((8, 9))

s.extend('End')

s.extend(10)

O/P:

1. [1, 2, 3, 4, 5, [6, 7]]

2. [1, 2, 3, 4, 5, 6, 7, 8, 9]

3. [1, 2, 3, 4, 5, 6, 7, 8, 9, 'E',
'n', 'd']

4. TypeError ('int' is not
Iterable obj).

iii) count()

Returns the no. of elements
with the specified value.

IP ..

count(□)

○ □ □ → 2.

s = [1, 2, 3, 4, 8, 2, 3, 4, 8, 9]

s.count(1) // 1

s.count(2) // 2

s.count(4) // 2

s.count(5) // 0

iv) Index()

Returns the index of the element in the list. If its not there then "value error"

IP.. index (□)

○ □ △ → 1.

s = [1, 2, 5, 7, 9, 11, 13]

s.index(5) // 2

s.index(11) // 5

s.index(1) // 0.

v) Insert()

Adds an element at the specified position.

insert(1, △)

IP..

○ □ □ → ○ △ □ □.

s = [1, 2, 5, 7, 9, 11, 13]

s.insert(5, 'Hello')

s.insert(0, 'agi').

o/p..

[1, 2, 5, 7, 9, 'Hello', 11, 13]

['agi', 1, 2, 5, 7, 9, 'Hello', 11, 13]

vi) pop() :

Removes element at the given index.

Ex :: pop()

□ 0 Δ → □ Δ

S = [1, 2, 3, 4, 5, 6]

S.pop(0) // [2, 3, 4, 5, 6]

S.pop() // [2, 3, 4, 5]

Note ::

If the index is not specified, then last element will be removed by default.

vii) remove() ::

Removes the item with the specified value. i.e., 1st occ. will be removed

Ex ::

remove()

□ 0 0 Δ → □ 0 Δ

S = [1, 2, 5, 2, 7, 9]

S.remove(2) // [1, 5, 2, 7, 9]

S.remove(10) // valueError

Note :: If the element is not in the list, then it will throw error.

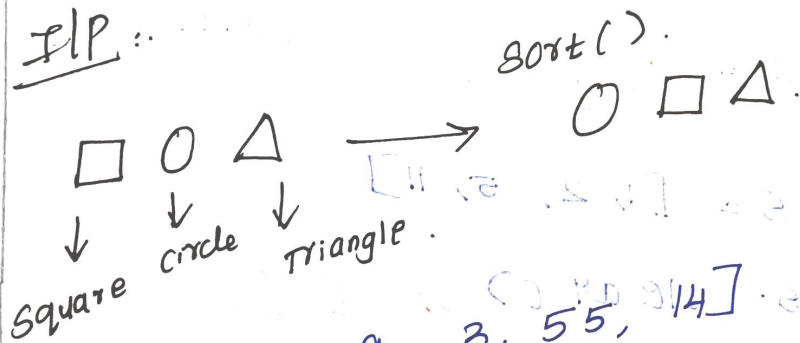
viii) sort()

- sorts the items in list either in ascending / descending order.

- Default: Ascending order.

- For descending (list.sort(reverse=True)).

ILP ::



$S = [91, 11, 9, 3, 55, 14]$



$S.sort()$ // $[3, 9, 11, 14, 55, 91]$

$S.sort(reverse=True)$

// $[91, 55, 14, 11, 9, 3]$

Note: For string, sorting = Dictionary Order.

ix) reverse()
Reverse the order of the list.

I/P:
 $\xrightarrow{\text{reverse()}}$ 

$s = [1, 2, 9, 11]$

$s.reverse()$ // $[11, 9, 2, 1]$



x) clear()
Empties the list

I/P:  $\xrightarrow{\text{clear()}}$

$s = [1, 2, 5, 11]$

$s.clear()$ // $[\]$

xi) copy()
copies the list

I/P:  $\xrightarrow{\text{copy()}}$ 

$s = [1, 2, 3, 4]$

$a = s.copy()$

$\text{print}(a)$ // $[1, 2, 3, 4]$

D/w. between append() &
extend() :

$s = [1, 2, 3, 4]$

$s.append(\{ 'a': 23, 'b': 10 \})$

$s.extend(\{ 'a': 23, 'b': 10 \})$

O/P:

1. $[1, 2, 3, 4, \{ 'a': 23, 'b': 10 \}]$

2. $[1, 2, 3, 4, 'a', 'b']$

$s = [1, 2, 3]$

$s.append([100, 100])$

$s.extend([100, 100])$

O/P: 1. $[1, 2, 3, [100, 100]]$

2. $[1, 2, 3, 100, 100]$

X ————— X