List

A list represents a group of elements.

Lists are very similar to array but there is major difference, an array can store only one type of elements whereas a list can store different type of elements.

Lists are mutable so we can modify it’s element.

A list can store different types of elements which can be modified.

Lists are dynamic which means size is not fixed.

Lists are represented using square bracket [ ].

Ex:- a = [10, 20, -50, 21.3, ‘Geekyshows’]

Ex:- a = [10, 20, 50]

Creating a List

A list is similar to an array that consists of a group of elements or items.

Syntax:- list_name = [element1, element2, …..]

Ex:- a = [10, 20, -50, 21.3, ‘Geekyshows’]

Creating an Empty List

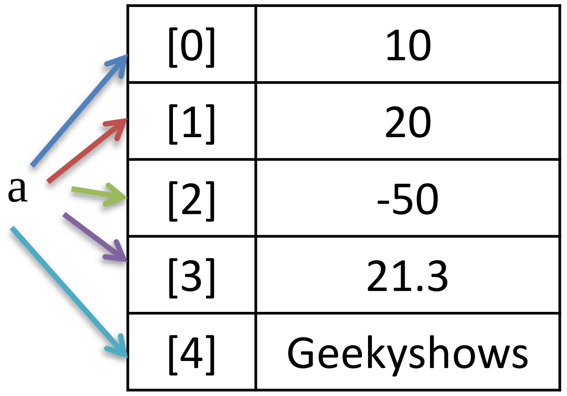
Syntax:- list_name = [ ]

Ex:- a = [ ]

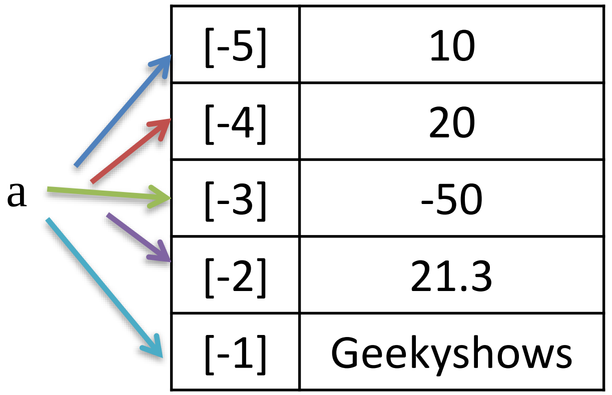
Index

An index represents the position number of an list’s element. The index start from 0 on wards and written inside square braces.

Ex:- a = [10, 20, -50, 21.3, ‘Geekyshows’]



[0]	10
[1]	20
[2]	-50
[3]	21.3
[4]	Geekyshows



[-5]	10
[-4]	20
[-3]	-50
[-2]	21.3
[-1]	Geekyshows

Accessing List’s Element

a = [10, 20, -50, 21.3, ‘Geekyshows’]

print(*a*[0])

print(*a*[1])

print(*a*[2])

print(*a*[3])

print(*a*[4])

10	20	-50	21.3	Geekyshows
a[0]	a[1]	a[2]	a[3]	a[4]

Modifying or Updating Element

Lists are mutable so we can modify it’s element.

a = [10, 20, -50, 21.3, ‘Geekyshows’]

a[1] = 40

10	20 40	-50	21.3	Geekyshows
a[0]	a[1]	a[2]	a[3]	a[4]

Accessing using for loop

```
a = [10, 20, -50, 21.3, ‘Geekyshows’]
```

Without index

```
for element in a:
		print(element)
```

With index

```
n = len(a)
for i in range(n):
		print(a[i])
```

Accessing using while loop

```
a = [10, 20, -50, 21.3, ‘Geekyshows’]

n = len(a)
i = 0
while i < n :
		print(a[i])
		i+=1
```

Deletion

del statement is used to delete an element of list or we can delete entire list using del statement.

a = [10, 20, -50, 21.3, ‘Geekyshows’]

Deleting Element

del a[2]

Deleting Entire List

del a

append ( )

This method is used to add an element at the end of the existing list.

Syntax:-

```
list_name.append(new_element)
```

Getting User input

```
a = []

n = int(input("Enter Number of Elements: "))
for i in range(n):
        a.append(int(input("Enter Element:")))

print("List:")
for element in a:
        print (element)
```

insert( )

This method is used to insert an element in a particular position of the existing list.

Syntax:-

```
        list_name.insert(position_number, new_element)
```

pop ( )

This method is used to remove last element from the existing list.

Syntax:-

```
list_name.pop( )
```

pop (n)

This method is used to remove an element specified by position number, from the existing list and returns removed element.

Syntax:-

```
        list_name.pop(position_number)
```

remove( )

This method is used to remove first occurrence of given element from the existing list. If it doesn’t found the element, shows valueError.

Syntax:-

```
list_name.remove(element)
```

index( )

This method returns position number of first occurrence of given element in the list. If it doesn’t found the element, shows valueError.

Syntax:-

```
list_name.index(element)
```

reverse ( )

This method is used to reverse the order of elements in the list.

Syntax:-

```
        list_name.reverse( )
```

extend( )

This method is used to append another list or iterable object at the end of the list.

Syntax:-

```
list_name.extend(lst)
```

count( )

This method returns number of occurrence of a specified element in the list.

Syntax:-

```
list_name.count(specified_element)
```

sort( )

This method is used to sort the elements of the list into ascending order.

Syntax:-

```
        list_name.sort()
```

clear( )

This method is used to delete all the elements from the list

Syntax:-

```
        list_name.clear()
```

Slicing on List

Slicing on list can be used to retrieve a piece of the list that contains a group of elements. Slicing is useful to retrieve a range of elements.

Syntax:-

new_list_name = list_name[start:stop:stepsize]

List Concatenation

+ operator is used to do concatenation the list.

Ex:-

```
a = [10, 20, 30]
b = [1, 2, 3]
result = a + b
print(result)
```

Repetition of List

* Operator is used to repeat the elements of list.

Ex:-

```
b = [1, 2, 3]
result = b * 3
print(result)
```

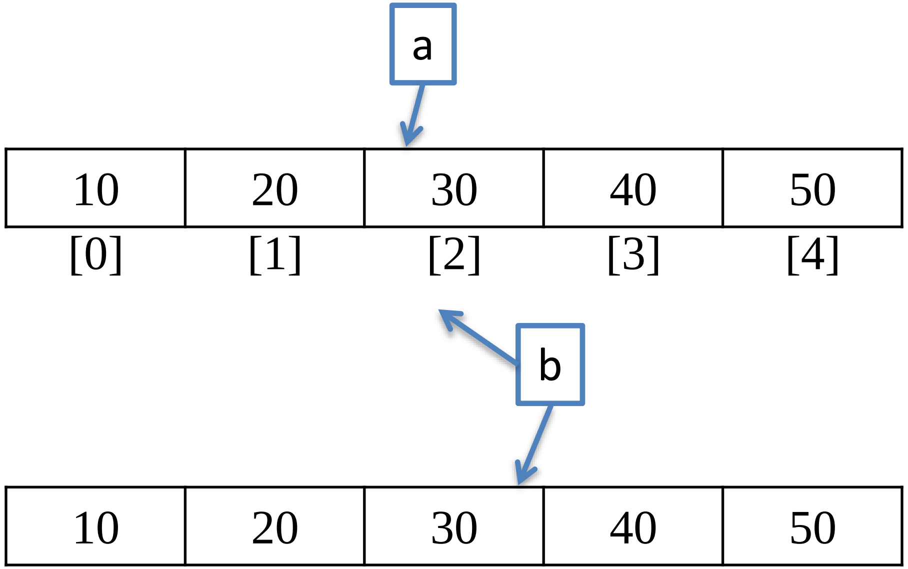
Aliasing List

Aliasing means giving another name to the existing object. It doesn't mean copying.

a = [10, 20, 30, 40, 50]

b = a

Modification in *a* will affect *b* and vice versa.



Copying List

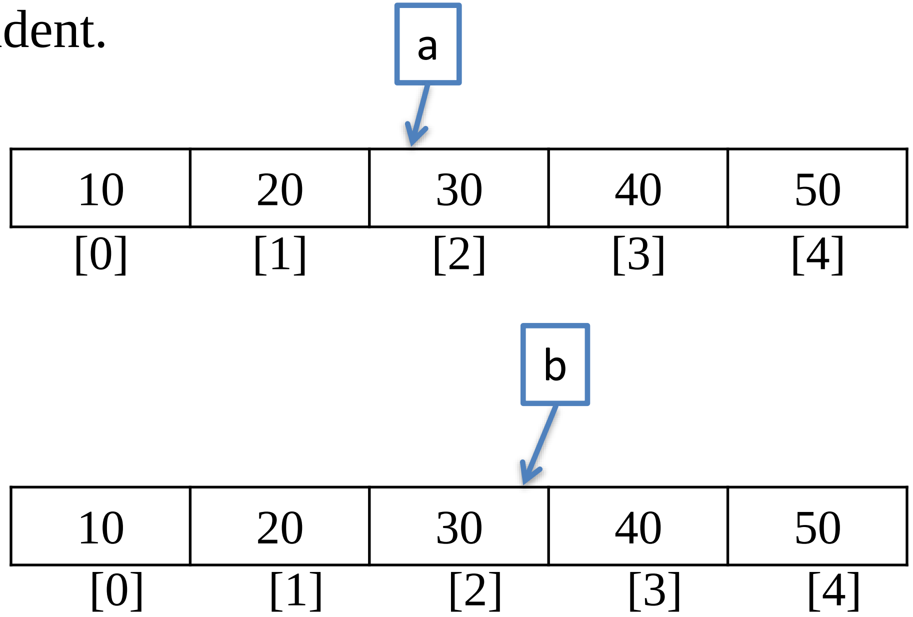
copy( ) method is used to copy all the elements of a list to another list.

When we copy a list a separate copy of all the elements is stored in another list. Both the list are independent.

a = [10, 20, 30, 40, 50]

b = a.copy()

Modification in *a* will not affect *b* and vice versa.



Cloning List

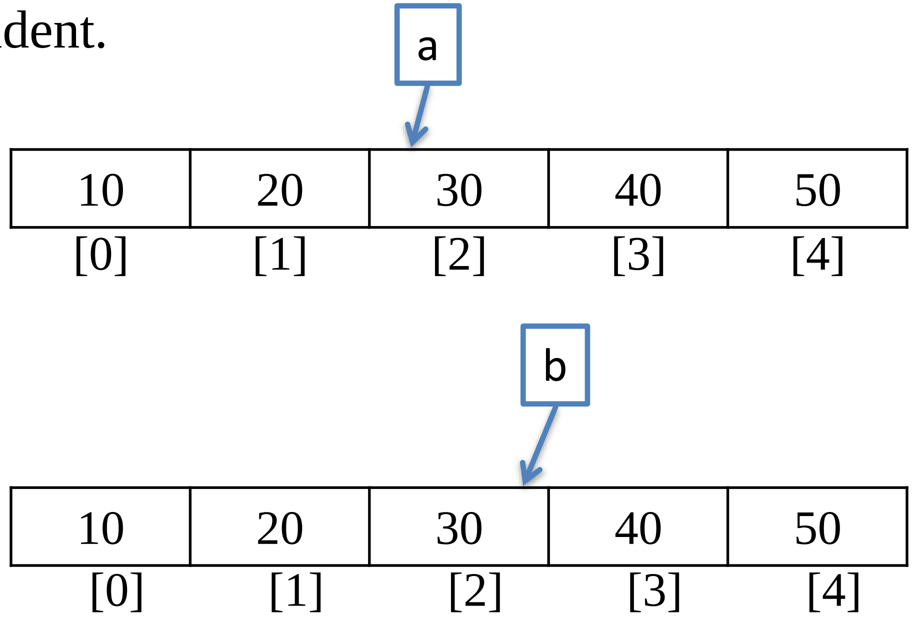
We can clone a list into another list using slicing.

When we clone a list a separate copy of all the elements is stored in another list. Both the list are independent.

a = [10, 20, 30, 40, 50]

b = a[:]

Modification in *a* will not affect *b* and vice versa.


a
10
20
30
40
50
[0]
[1]
[2]
[3]
[4]
b
10
20
30
40
50
[0]
[1]
[2]
[3]
[4]