Python List

A list in Python is used to store the sequence of various types of data.

Python lists are mutable type its mean we can modify its element after it created.

However, Python consists of six data-types that are capable to store the sequences, but the most common and reliable type is the list.

A list can be defined as a collection of values or items of different types.

The items in the list are separated with the comma (,) and enclosed with the square brackets []

A list can be define as below

```
L1 = ["John", 102, "USA"]

L2 = [1, 2, 3, 4, 5, 6]
```

If we try to print the type of L1, L2, and L3 using type() function then it will come out to be a list.

```
print(type(L1))

print(type(L2))
```


Output:

```
<class 'list'>

<class 'list'>
```

Characteristics of Lists :

The list has the following characteristics:

•The lists are ordered.

•The element of the list can access by index.

•The lists are the mutable type.

•The lists are mutable types.

•A list can store the number of various elements.

List indexing and splitting

The indexing is processed in the same way as it happens with the strings. The elements of the list can be accessed by using the slice operator [].

The index starts from 0 and goes to length - 1. The first element of the list is stored at the 0th index, the second element of the list is stored at the 1st index,and soon.

List = [0, 1, 2, 3, 4, 5]

0	1	2	3	4	5

List[0] = 0

List[1] = 1

List[2] = 2

List[3] = 3

List[4] = 4

List[5] = 5

List[0:] = [0,1,2,3,4,5]

List[:] = [0,1,2,3,4,5]

List[2:4] = [2, 3]

List[1:3] = [1, 2]

List[:4] = [0, 1, 2, 3]

We can get the sub-list of the list using the following syntax.

list_varible(start:stop:step)

•The start denotes the starting index position of the list.

•The stop denotes the last index position of the list.

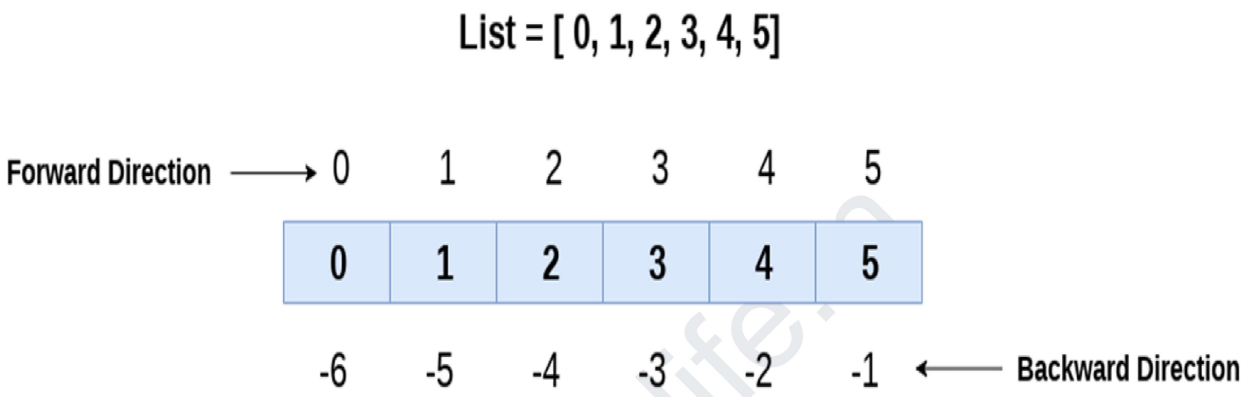
•The step is used to skip the nth element within a start:stop

Consider the following example:

```
list = [1,2,3,4,5,6,7]
print(list[0])
print(list[1])
print(list[2])
print(list[3])
# Slicing the elements
print(list[0:6])
# By default the index value is 0 so its starts from the 0th element and go for index -1.

print(list[:])
print(list[2:5])
print(list[1:6:2])
```

Unlike other languages, Python provides the flexibility to use the negative indexing also. The negative indices are counted from the right. The last element (rightmost) of the list has the index -1; its adjacent left element is present at the index -2 and so on until the left-most elements are encountered.



Let's have a look at the following example where we will use negative indexing to access the elements of the list.

```
list = [1,2,3,4,5]

print(list[-1])

print(list[-3:])

print(list[:-1])

print(list[-3:-1])
```

As we discussed above, we can get an element by using negative indexing. In the above code, the first print statement returned the rightmost element of the list. The second print statement returned the sub-list, and so on.

Updating List values

Lists are the most versatile data structures in Python since they are mutable, and their values can be updated by using the slice and assignment operator.

Python also provides append() and insert() methods, which can be used to add values to the list.

Consider the following example to update the values inside the list

```
list = [1, 2, 3, 4, 5, 6]

print(list)

# It will assign value to the value to the second index

list[2] = 10

print(list)

# Adding multiple-element

list[1:3] = [89, 78]

print(list)

# It will add value at the end of the list

list[-1] = 25

print(list)
```

Iterating a List

A list can be iterated by using a for - in loop. A simple list containing four strings, which can be iterated as follows.

```
list = ["John", "David", "James", "Jonathan"]

for i in list:

    # The i variable will iterate over the elements of the List and contains each element in each iteration.

   print(i)
```

Adding elements to the list

Python provides append() function which is used to add an element to the list. However, the append() function can only add value to the end of the list.

Consider the following example in which, we are taking the elements of the list from the user and printing the list on the console.

```
#Declaring the empty list

l =[]

#Number of elements will be entered by the user

n = int(input("Enter the number of elements in the list:"))

# for loop to take the input

for i in range(0,n):

# The input is taken from the user and added to the list as the item

l.append(input("Enter the item:"))

print("printing the list items..")

# traversal loop to print the list items

for i in l:
```

```
print(i, end = "  ")
```

Removing elements from the list

Python provides the remove() function which is used to remove the element from the list. Consider the following example to understand this concept.

Example -

```
list = [0,1,2,3,4]

print("printing original list: ");

for i in list:

    print(i,end=" ")

list.remove(2)

print("\nprinting the list after the removal of first element...")

for i in list:

    print(i,end=" ")
```


Python List Built-in functions

Python provides the following built-in functions, which can be used with the lists.

Method	Description
append()	Adds an element at the end of the list
clear()	Removes all the elements from the list
copy()	Returns a copy of the list
count()	Returns the number of elements with the specified value
extend()	Add the elements of a list (or any iterable), to the end of the current list
index()	Returns the index of the first element with the specified value
insert()	Adds an element at the specified position
pop()	Removes the element at the specified position
remove()	Removes the first item with the specified value
reverse()	Reverses the order of the list
sort()	Sorts the list

append()
Used for appending and adding elements to List. It is used to add elements to the last position of the List in Python.

Syntax:
list.append (element)

```
# Adds List Element as value of List.
List = ['Mathematics', 'chemistry', 1997, 2000]
List.append(20544)
print(List)
```

Output:
['Mathematics', 'chemistry', 1997, 2000, 20544]

insert()
Inserts an element at the specified position.

Syntax:
list.insert(<position, element)

Note: Position mentioned should be within the range of List, as in this case between 0 and 4, elsewise would throw IndexError.

```
List = ['Mathematics', 'chemistry', 1997, 2000]
# Insert at index 2 value 10087
List.insert(2,10087)
print(List)
```

Output:

['Mathematics', 'chemistry', 10087, 1997, 2000, 20544]

extend()
Adds contents to List2 to the end of List1.

Syntax
List1.extend(List2)

```
List1 = [1, 2, 3]
List2 = [2, 3, 4, 5]

# Add List2 to List1
List1.extend(List2)
```

```
print(List1)

# Add List1 to List2 now
List2.extend(List1)
print(List2)
```

Output:

```
[1, 2, 3, 2, 3, 4, 5]
[2, 3, 4, 5, 1, 2, 3, 2, 3, 4, 5]
```

Other functions of List

sum()

Calculates the sum of all the elements of the List.

Syntax

```
sum(List)
```

```
List = [1, 2, 3, 4, 5]
print(sum(List))
```

Output:

```
15
```

What happens if a numeric value is not used a parameter?

The sum is calculated only for Numeric values, elsewise throws TypeError.

See example:

```
List = ['gfg', 'abc', 3]
print(sum(List))
```

Output:

```
Traceback (most recent call last):
  File "", line 1, in
    sum(List)
TypeError: unsupported operand type(s) for +: 'int' and 'str'
```

count()

Calculates total occurrence of a given element of List.

Syntax:
List.count(element)

```
List = [1, 2, 3, 1, 2, 1, 2, 3, 2, 1]
print(List.count(1))
```
Output:
4

length
Calculates the total length of the List.
Syntax:
len(list_name)

```
List = [1, 2, 3, 1, 2, 1, 2, 3, 2, 1]
print(len(List))
```
Output:
10

index()
Returns the index of the first occurrence. The start and End index are not necessary parameters.

Syntax:
List.index(element[,start[,end]])


```
List = [1, 2, 3, 1, 2, 1, 2, 3, 2, 1]
print(List.index(2))
```
Output:
1
Another example:
```
List = [1, 2, 3, 1, 2, 1, 2, 3, 2, 1]
print(List.index(2,2))
```
Output:
4

min()
Calculates minimum of all the elements of List.

Syntax:
Deletion of List Elements

To Delete one or more elements, i.e. remove an element, many built-in functions can be used, such as pop() & remove() and keywords such as del.

pop()
The index is not a necessary parameter, if not mentioned takes the last index.

Syntax:

list.pop([index])

Note: Index must be in range of the List, elsewise IndexErrors occurs.

```
List = [2.3, 4.445, 3, 5.33, 1.054, 2.5]
print(List.pop())
```
Output:

2.5

```
List = [2.3, 4.445, 3, 5.33, 1.054, 2.5]
print(List.pop(0))
```
Output:

2.3
del()
Element to be deleted is mentioned using list name and index.

Syntax:

del list.[index]

```
List = [2.3, 4.445, 3, 5.33, 1.054, 2.5]
del List[0]
print(List)
```

Output:
[4.445, 3, 5.33, 1.054, 2.5]

remove()
Element to be deleted is mentioned using list name and element.

Syntax
list.remove(element)

```
List = [2.3, 4.445, 3, 5.33, 1.054, 2.5]
List.remove(3)
print(List)
```

Output:

[2.3, 4.445, 5.33, 1.054, 2.5]