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

```
1. L1 = ["John", 102, "USA"]
2. L2 = [1, 2, 3, 4, 5, 6]
```

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

```
1. print(type(L1))
2. 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.

Let's check the first statement that lists are the ordered.

```
1. a = [1,2,"Peter",4.50,"Ricky",5,6]
2. b = [1,2,5,"Peter",4.50,"Ricky",6]
3. a ==b
```

Output:

```
False
```

Both lists have consisted of the same elements, but the second list changed the index position of the 5th element that violates the order of lists. When compare both lists it returns the false.

Lists maintain the order of the element for the lifetime. That's why it is the ordered collection of objects.

```
1. a = [1, 2,"Peter", 4.50,"Ricky",5, 6]
2. b = [1, 2,"Peter", 4.50,"Ricky",5, 6]
3. a == b
```

Output:

```
True
```

Let's have a look at the list example in detail.

```
1. emp = ["John", 102, "USA"]
2. Dep1 = ["CS",10]
3. Dep2 = ["IT",11]
4. HOD_CS = [10,"Mr. Holding"]
5. HOD_IT = [11, "Mr. Bewon"]
6. print("printing employee data...")
7. print("Name : %s, ID: %d, Country: %s"%(emp[0],emp[1],emp[2]))
8. print("printing departments...")
9. print("Department 1:\nName: %s, ID: %d\nDepartment 2:\nName: %s, ID: %s"%(De
   p1[0],Dep2[1],Dep2[0],Dep2[1]))
10. print("HOD Details ....")
11. print("CS HOD Name: %s, Id: %d"%(HOD_CS[1],HOD_CS[0]))
12. print("IT HOD Name: %s, Id: %d"%(HOD_IT[1],HOD_IT[0]))
13. print(type(emp),type(Dep1),type(Dep2),type(HOD_CS),type(HOD_IT))
```

Output:

```
printing employee data...
Name : John, ID: 102, Country: USA
printing departments...
Department 1:
```

```
Name: CS, ID: 11
Department 2:
Name: IT, ID: 11
HOD Details ....
CS HOD Name: Mr. Holding, Id: 10
IT HOD Name: Mr. Bewon, Id: 11
<class 'list'> <class 'list'> <class 'list'> <class 'list'> <class 'list'>
```

In the above example, we have created the lists which consist of the employee and department details and printed the corresponding details. Observe the above code to understand the concept of the list better.

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 so on.

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.

1. 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**

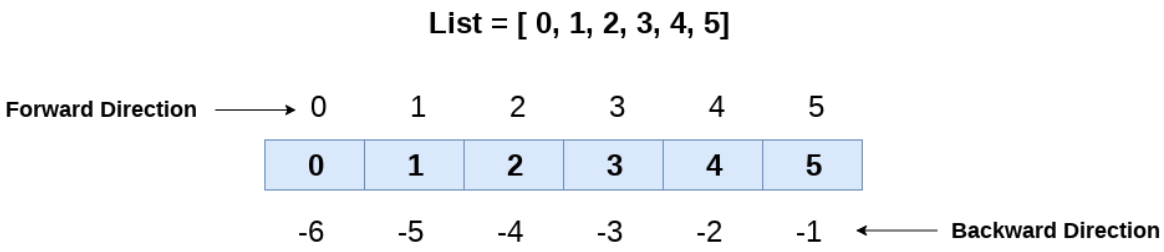
Consider the following example:

```
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])
```

Output:

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

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.

```
1. list = [1,2,3,4,5]
2. print(list[-1])
3. print(list[-3:])
4. print(list[:-1])
5. print(list[-3:-1])
```

Output:

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

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)
```

Output:

```
[1, 2, 3, 4, 5, 6]
[1, 2, 10, 4, 5, 6]
[1, 89, 78, 4, 5, 6]
[1, 89, 78, 4, 5, 25]
```

Remove Specified Item

The `remove()` method removes the specified item.

Example

Remove "banana":

```
thislist = ["apple", "banana", "cherry"]
thislist.remove("banana")
print(thislist)
```

Remove Specified Index

The `pop()` method removes the specified index.

Example

Remove the second item:

```
thislist = ["apple", "banana", "cherry"]
thislist.pop(1)
print(thislist)
```

If you do not specify the index, the `pop()` method removes the last item.

Example

Remove the last item:

```
thislist = ["apple", "banana", "cherry"]
thislist.pop()
print(thislist)
```

The `del` keyword also removes the specified index:

Example

Remove the first item:

```
thislist = ["apple", "banana", "cherry"]
del thislist[0]
print(thislist)
```

The `del` keyword can also delete the list completely.

Example

Delete the entire list:

```
thislist = ["apple", "banana", "cherry"]
del thislist
```

Clear the List

The `clear()` method empties the list.

The list still remains, but it has no content.

Example

Clear the list content:

```
thislist = ["apple", "banana", "cherry"]
thislist.clear()
print(thislist)
```

Python List Operations

The concatenation (+) and repetition (*) operators work in the same way as they were working with the strings.

Let's see how the list responds to various operators.

1. Consider a Lists l1 = [1, 2, 3, 4], **and** l2 = [5, 6, 7, 8] to perform operation.

Operator	Description	Example
Repetition	The repetition operator enables the list elements to be repeated multiple times.	`L1*2 = [1, 2, 3, 4, 1, 2, 3, 4]`
Concatenation	It concatenates the list mentioned on either side of the operator.	`l1+l2 = [1, 2, 3, 4, 5, 6, 7, 8]`
Membership	It returns true if a particular item exists in a particular list otherwise false.	`print(2 in l1) prints True.`

Iteration	The for loop is used to iterate over the list elements.	`for i in l1:`
		`  print(i)`
		Output
		1
		2
		3
		4

Length	It is used to get the length of the list	`len(l1) = 4`

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 e
ach iteration.
    print(i)
```

Output:

```
John
David
James
Jonathan
```

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.

```
1. #Declaring the empty list
2. l =[]
3. #Number of elements will be entered by the user
4. n = int(input("Enter the number of elements in the list:"))
5. # for loop to take the input
6. for i in range(0,n):
7.     # The input is taken from the user and added to the list as the item
8.     l.append(input("Enter the item:"))
9. print("printing the list items..")
10. # traversal loop to print the list items
11. for i in l:
12.     print(i, end = "  ")
```

Output:

```
Enter the number of elements in the list:5
Enter the item:25
Enter the item:46
Enter the item:12
Enter the item:75
Enter the item:42
```

```
printing the list items
25  46  12  75  42
```

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 -

```
1. list = [0,1,2,3,4]
2. print("printing original list: ");
3. for i in list:
4.     print(i,end=" ")
5. list.remove(2)
6. print("\nprinting the list after the removal of first element...")
7. for i in list:
8.     print(i,end=" ")
```

Output:

```
printing original list:
0 1 2 3 4
printing the list after the removal of first element...
0 1 3 4
```

Python List Built-in functions

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

SN	Function	Description	Example
1	cmp(list1, list2)	It compares the elements of both the lists.	This method is not used in the Python 3 and the above versions.

2	len(list)	It is used to calculate the length of the list.	`L1 = [1,2,3,4,5,6,7,8]`
			`print(len(L1))`
			`8`

3	max(list)	It returns the maximum element of the list.	`L1 = [12,34,26,48,72]`
			`print(max(L1))`
			`72`

4	min(list)	It returns the minimum element of the list.	`L1 = [12,34,26,48,72]`
			`print(min(L1))`
			`12`

5	list(seq)	It converts any sequence to the list.	`str = "Johnson"`
			`s = list(str)`
			`print(type(s))`
			`<class list>`

Let's have a look at the few list examples.

Example: 1- Write the program to remove the duplicate element of the list.

```
list1 = [1,2,2,3,55,98,65,65,13,29]
# Declare an empty list that will store unique values
list2 = []
for i in list1:
    if i not in list2:
        list2.append(i)
print(list2)
```

Output:

```
[1, 2, 3, 55, 98, 65, 13, 29]
```

Example:2- Write a program to find the sum of the element in the list.

```
list1 = [3,4,5,9,10,12,24]
sum = 0
for i in list1:
    sum = sum+i
print("The sum is:",sum)
```

Output:

```
The sum is: 67
```

Example: 3- Write the program to find the lists consist of at least one common element.

```
list1 = [1,2,3,4,5,6]
list2 = [7,8,9,2,10]
for x in list1:
    for y in list2:
        if x == y:
            print("The common element is:",x)
```

Output:

```
The common element is: 2
```