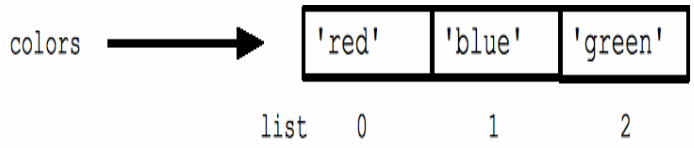
Concept of Python List :-

The most basic data structure in Python is the **sequence**. Each element of a sequence is assigned a number - its position or index. The first index is zero, the second index is one, and so forth.

Python has six built-in types of sequences, but the most common ones are lists and tuples, which we would see in this tutorial.

There are certain things you can do with all sequence types. These operations include indexing, slicing, adding, multiplying, and checking for membership.

```
colors = ['red', 'blue', 'green']
print colors[0]    ## red
print colors[2]    ## green
print len(colors)  ## 3
```



Creating Python Lists

The list is a most versatile data type available in Python which can be written as a list of comma-separated values (items) between square brackets. Important thing about a list is that items in a list need not be of the same type.

Creating a list is as simple as putting different comma-separated values between square brackets. For example −

```
list1 = ['physics', 'chemistry', 1997, 2000];
list2 = [1, 2, 3, 4, 5 ];
list3 = ["a", "b", "c", "d"]
```

Similar to string indices, list indices start at 0, and lists can be sliced, concatenated and so on.

Accessing Values in Lists

To access values in lists, use the square brackets for slicing along with the index or indices to obtain value available at that index.

For example −

```
list1 = ['physics', 'chemistry', 1997, 2000];
list2 = [1, 2, 3, 4, 5, 6, 7 ];
print "list1[0]: ", list1[0]
print "list2[1:5]: ", list2[1:5]
```

When the above code is executed, it produces the following result −

```
list1[0]:  physics
list2[1:5]:  [2, 3, 4, 5]
```

Updating Lists

You can update single or multiple elements of lists by giving the slice on the left-hand side of the assignment operator, and you can add to elements in a list with the append() method.

For example −

```
list = ['physics', 'chemistry', 1997, 2000];
print "Value available at index 2 : "
print list[2]
list[2] = 2001;
print "New value available at index 2 : "
print list[2]
```

Delete List Elements

To remove a list element, you can use either the del statement if you know exactly which element(s) you are deleting or the remove() method if you do not know. For example −

Ex.

```
list1 = ['physics', 'chemistry', 1997, 2000];
print list1
del list1[2];
print "After deleting value at index 2 : "
print list1
```

When the above code is executed, it produces following result −

```
['physics', 'chemistry', 1997, 2000]
After deleting value at index 2 :
['physics', 'chemistry', 2000]
```

- List Methods

Here are some other common list methods.
list.append(elem) -- adds a single element to the end of the list. Common error: does not return the new list, just modifies the original.

list.insert(index, elem) -- inserts the element at the given index, shifting elements to the right.

list.extend(list2) adds the elements in list2 to the end of the list. Using + or += on a list is similar to using extend().

list.index(elem) -- searches for the given element from the start of the list and returns its index. Throws a ValueError if the element does not appear (use "in" to check without a ValueError).

list.remove(elem) -- searches for the first instance of the given element and removes it (throws ValueError if not present)
list.sort() -- sorts the list in place (does not return it). (The sorted() function shown later is preferred.)

list.reverse() -- reverses the list in place (does not return it)

list.pop(index) -- removes and returns the element at the given index. Returns the rightmost element if index is omitted (roughly the opposite of append()).

Ex.

```
list = ['larry', 'curly', 'moe']
list.append('shemp')                    ## append elem at end
list.insert(0, 'xxx')                   ## insert elem at index 0
list.extend(['yyy', 'zzz'])             ## add list of elements at end
print list
        ## ['xxx', 'larry', 'curly', 'moe', 'shemp', 'yyy', 'zzz']
print list.index('curly')                                    ## 2

list.remove('curly')       ## search and remove that element
list.pop(1)              ## removes and returns 'larry'
print list  ## ['xxx', 'moe', 'shemp', 'yyy', 'zzz']
```

Built-in List Functions & Methods

Python includes the following list functions −

- **cmp(list1, list2)**

Compares elements of both lists.

Python list method **cmp()** compares elements of two lists.

Syntax

Following is the syntax for **cmp()** method −

cmp(list1, list2)

Parameters

- **list1** − This is the first list to be compared.
- **list2** − This is the second list to be compared.

Return Value

If elements are of the same type, perform the compare and return the result. If elements are different types, check to see if they are numbers.

- If numbers, perform numeric coercion if necessary and compare.
- If either element is a number, then the other element is "larger" (numbers are "smallest").
- Otherwise, types are sorted alphabetically by name.

```
list1, list2 = [123, 'xyz'], [456, 'abc']
print cmp(list1, list2)
print cmp(list2, list1)
list3 = list2 + [786];
print cmp(list2, list3)
```

- When we run above program, it produces following result −

```
-1
1
-1
```

➢ len(list)

Gives the total length of the list.

Following is the syntax for **len()** method −

len(list)

Parameters

- **list** − This is a list for which number of elements to be counted.

Return Value

This method returns the number of elements in the list.

Example

The following example shows the usage of len() method.

```
list1, list2 = [123, 'xyz', 'zara'], [456, 'abc']
print "First list length : ", len(list1)
print "Second list length : ", len(list2)
```

When we run above program, it produces following result −

```
First list length :  3
Second list length :  2
```

➢ max(list)

Returns item from the list with max value.

Following is the syntax for **max()** method −

```
max(list)
```

Parameters

- **list** − This is a list from which max valued element to be returned.

Return Value

This method returns the elements from the list with maximum value.

Example

The following example shows the usage of max() method.

```
list1, list2 = [123, 'xyz', 'zara', 'abc'], [456, 700, 200]
print "Max value element : ", max(list1)
print "Max value element : ", max(list2)
```

When we run above program, it produces following result −

```
Max value element :  zara
Max value element :  700
```

- min(list)

Returns item from the list with min value.

Following is the syntax for **min()** method −

```
min(list)
```

Parameters

- **list** − This is a list from which min valued element to be returned.

Return Value

This method returns the elements from the list with minimum value.

Example

The following example shows the usage of min() method.

```
list1, list2 = [123, 'xyz', 'zara', 'abc'], [456, 700, 200]
print "min value element : ", min(list1)
```

```
print "min value element : ", min(list2)
```

When we run above program, it produces following result −

```
min value element :  123
min value element :  200
```

- list(seq)

Converts a tuple into list.

list(seq)

Parameters

- **seq** − This is a tuple to be converted into list.

Return Value

This method returns the list.

Example

The following example shows the usage of list() method.

```
aTuple = (123, 'xyz', 'zara', 'abc');
aList = list(aTuple)
print "List elements : ", aList
```

When we run above program, it produces following result −

```
List elements :  [123, 'xyz', 'zara', 'abc']
```

Python list copy() or clone :-

Python list copy() is an inbuilt function that returns a shallow copy of the list. The copy() method is used to copy all the items from one list to another list.

syntax of the Python List Copy() method is the following.

```
newList = list.copy()
```

You have to provide the source list to copy the elements of that list.

See the following example.

```
# app.py


targaryens = ['Aegon', 'Daenerys', 'Aemon', 'Aeris', 'Rhaegar', 'Viserys']
clonedList = targaryens.copy()
print(clonedList)
```

Alias List :-

```
a = [81, 82, 83]
b = a
print(a is b)
```