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.

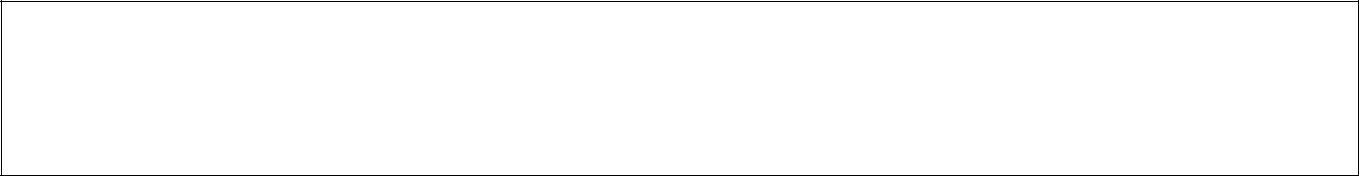
There are certain things you can do with all the sequence types. These operations include indexing, slicing, adding, multiplying, and checking for membership. In addition, Python has built-in functions for finding the length of a sequence and for finding its largest and smallest elements.

**Python Lists**



The list is the most versatile datatype 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 the 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 ];

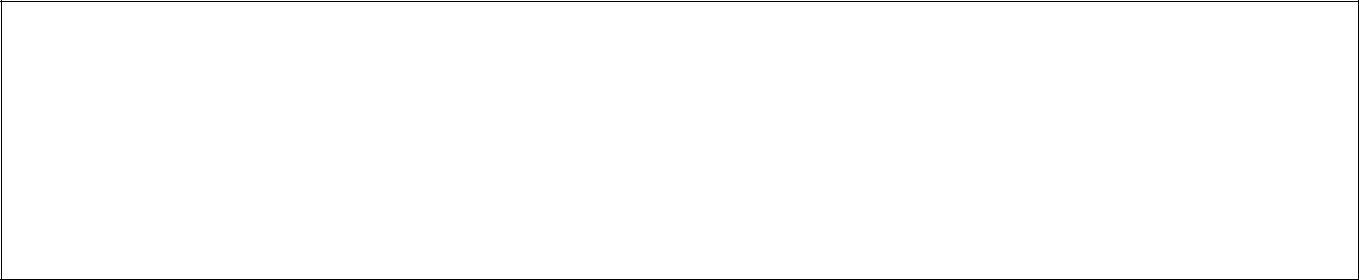
list3 = ["a", "b", "c", "d"];

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-



#!/usr/bin/python3

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]

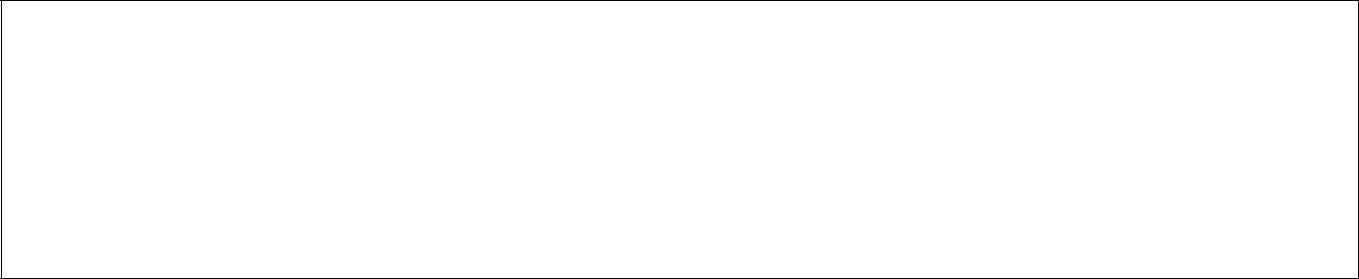
143

Python 3

**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-



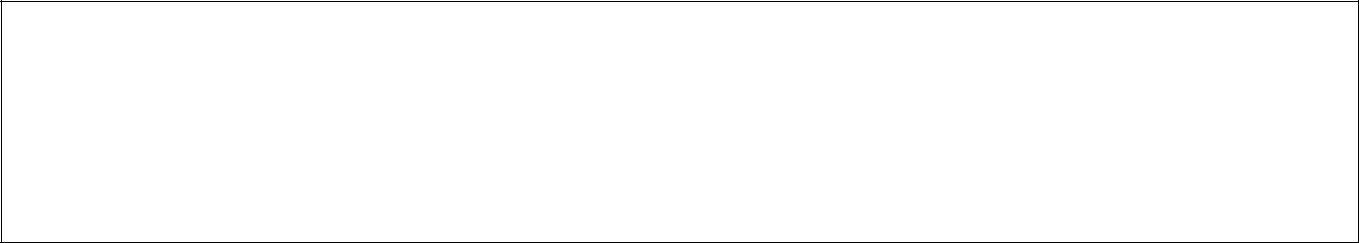
#!/usr/bin/python3

list = ['physics', 'chemistry', 1997, 2000] print ("Value available at index 2 : ", list[2]) list[2] = 2001

print ("New value available at index 2 : ", list[2])

**Note:** The append() method is discussed in the subsequent section.

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



Value available at index 2 :

1997

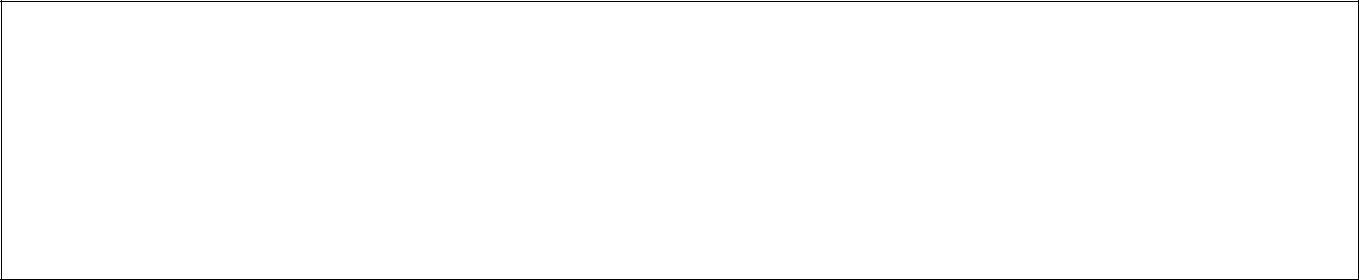
New value available at index 2 :

2001

**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. You can use the remove() method if you do not know exactly which items to delete. For example-



#!/usr/bin/python3

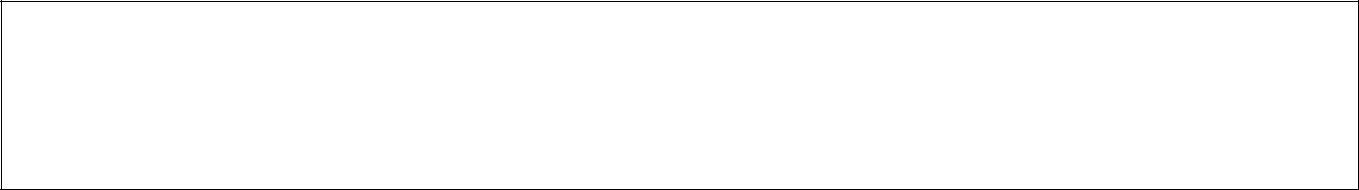
list = ['physics', 'chemistry', 1997, 2000]

print (list)

del list[2]

print ("After deleting value at index 2 : ", list)

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



['physics', 'chemistry', 1997, 2000]

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

**Note:** remove() method is discussed in subsequent section.

**Basic List Operations**



Lists respond to the + and \* operators much like strings; they mean concatenation and repetition here too, except that the result is a new list, not a string.

In fact, lists respond to all of the general sequence operations we used on strings in the prior chapter.

144

|  |  | Python 3 |
| --- | --- | --- |
|  |  |  |
| **Python Expression** | **Results** | **Description** |
|  |  |  |
|  |  |  |
| len([1, 2, 3]) | 3 | Length |
|  |  |  |
| [1, 2, 3] + [4, 5, 6] | [1, 2, 3, 4, 5, 6] | Concatenation |
|  |  |  |
| ['Hi!'] \* 4 | ['Hi!', 'Hi!', 'Hi!', 'Hi!'] | Repetition |
|  |  |  |
| 3 in [1, 2, 3] | True | Membership |
|  |  |  |
| for x in [1,2,3] : print (x,end=' | 1 2 3 | Iteration |
| ') |  |  |
|  |  |  |

**Indexing, Slicing and Matrixes**



Since lists are sequences, indexing and slicing work the same way for lists as they do for strings.

Assuming the following input



L=['C++'', 'Java', 'Python']

| **Python Expression** | **Results** | **Description** |
| --- | --- | --- |
|  |  |  |
|  |  |  |
| L[2] | 'Python' | Offsets start at zero |
|  |  |  |
| L[-2] | 'Java' | Negative: count from the |
|  |  | right |
|  |  |  |
| L[1:] | ['Java', 'Python'] | Slicing fetches sections |
|  |  |  |

**Built-in List Functions & Methods**



Python includes the following list functions-



**SN** **Function with Description**

1. **cmp(list1, list2)**

No longer available in Python 3.

145

Python 3



1. **len(list)**

Gives the total length of the list.

1. **max(list)**

Returns item from the list with max value.

1. **min(list)**

Returns item from the list with min value.

1. **list(seq)**

Converts a tuple into list.



Let us understand the use of these functions.

**List len() Method**



**Description**

The **len()** method returns the number of elements in the list.

**Syntax**

Following is the syntax for len() method-



len(list)

**Parameters**

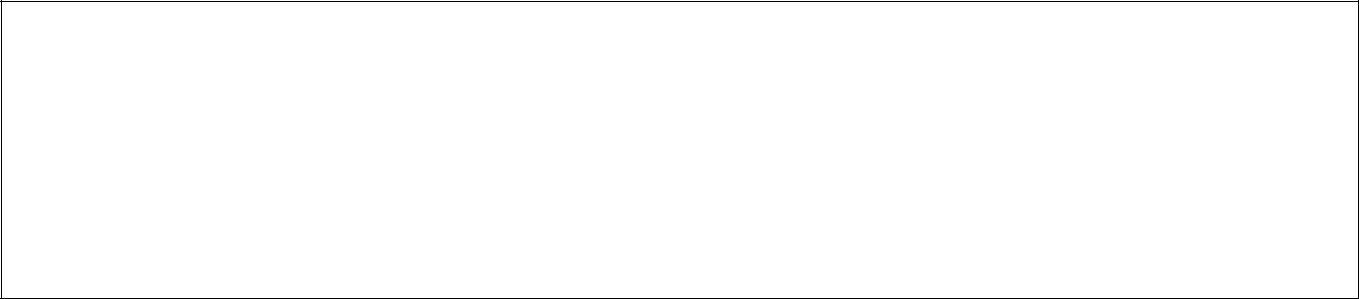
**list** - This is a list for which, number of elements are to be counted.

**Return Value**

This method returns the number of elements in the list.

**Example**

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



#!/usr/bin/python3

list1 = ['physics', 'chemistry', 'maths']

print (len(list1))

list2=list(range(5)) #creates list of numbers between 0-4

print (len(list2))

146

Python 3

When we run above program, it produces following result-



3

5

**List max() Method**



**Description**

The **max()** method returns the elements from the list with maximum value.

**Syntax**

Following is the syntax for max() method-



max(list)

**Parameters**

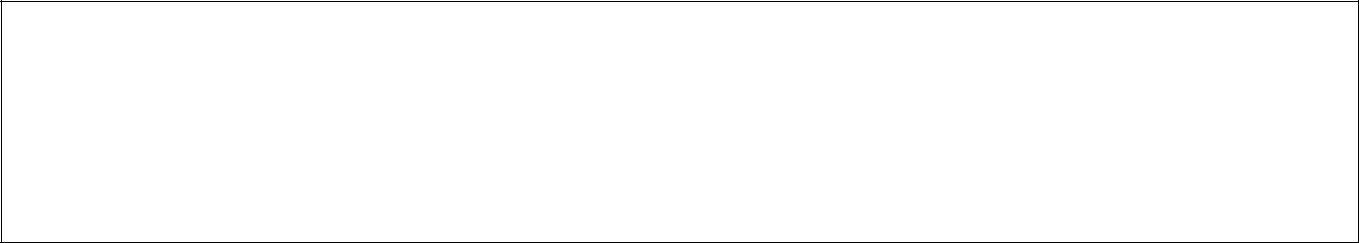
**list** - This is a list from which max valued element are to be returned.

**Return Value**

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

**Example**

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



#!/usr/bin/python3

list1, list2 = ['C++','Java', 'Python'], [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 : Python

Max value element : 700

**List min() Method**



**Description**

The method min() returns the elements from the list with minimum value.

147

Python 3

**Syntax**

Following is the syntax for min() method-



min(list)

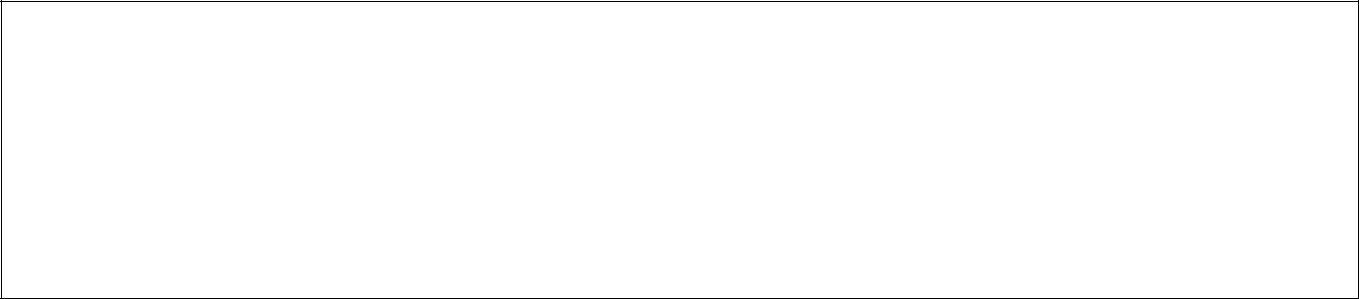
**Parameters**

**list** - This is a list from which min valued element is 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.

#!/usr/bin/python3

list1, list2 = ['C++','Java', 'Python'], [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 : C++

min value element : 200

**List list() Method**



**Description**

The **list()** method takes sequence types and converts them to lists. This is used to convert a given tuple into list.

**Note:** Tuple are very similar to lists with only difference that element values of a tuplecan not be changed and tuple elements are put between parentheses instead of square bracket. This function also converts characters in a string into a list.

**Syntax**

Following is the syntax for list() method-



list( seq )

**Parameters**

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

148

Python 3

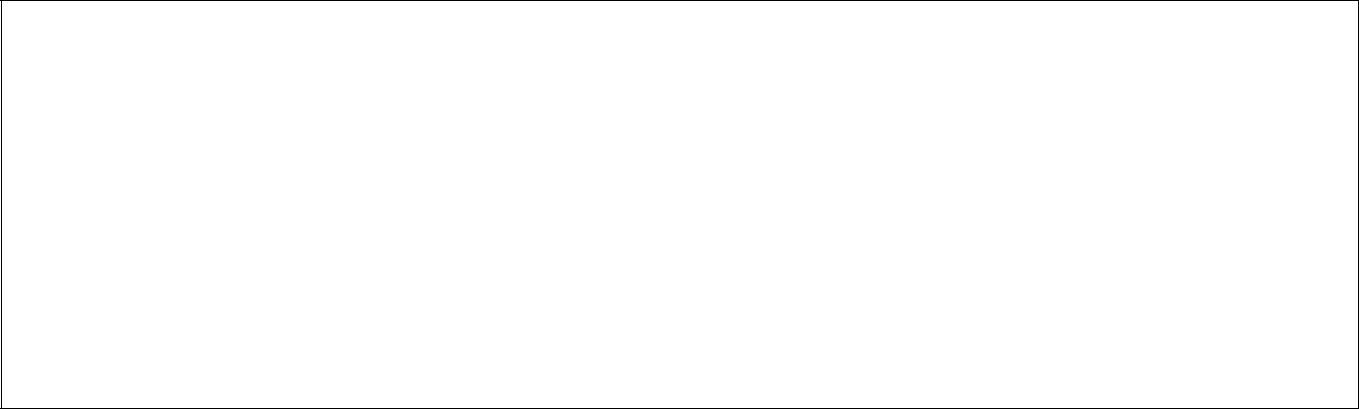
**Return Value**

This method returns the list.

**Example**

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

#!/usr/bin/python3



aTuple = (123, 'C++', 'Java', 'Python')

list1 = list(aTuple)

print ("List elements : ", list1)

str="Hello World"

list2=list(str)

print ("List elements : ", list2)

When we run above program, it produces following result-



List elements : [123, 'C++', 'Java', 'Python']

List elements : ['H', 'e', 'l', 'l', 'o', ' ', 'W', 'o', 'r', 'l', 'd']

Python includes the following list methods-

| **SN** | **Methods with Description** |
| --- | --- |
|  |  |
|  |  |
| 1 | **list.append(obj)** |
|  | Appends object obj to list |
|  |  |
| 2 | **list.count(obj)** |
|  | Returns count of how many times obj occurs in list |
|  |  |
| 3 | **list.extend(seq)** |
|  | Appends the contents of seq to list |
|  |  |
| 4 | **list.index(obj)** |
|  | Returns the lowest index in list that obj appears |
|  |  |
| 5 | **list.insert(index, obj)** |
|  | Inserts object obj into list at offset index |
|  |  |
|  | 149 |

Python 3



1. **list.pop(obj=list[-1])**

Removes and returns last object or obj from list

1. **list.remove(obj)**

Removes object obj from list

1. **list.reverse()**

Reverses objects of list in place

1. **list.sort([func])**

Sorts objects of list, use compare func if given



**List append() Method**



**Description**

The **append()** method appends a passed obj into the existing list.

**Syntax**

Following is the syntax for append() method-



list.append(obj)

**Parameters**

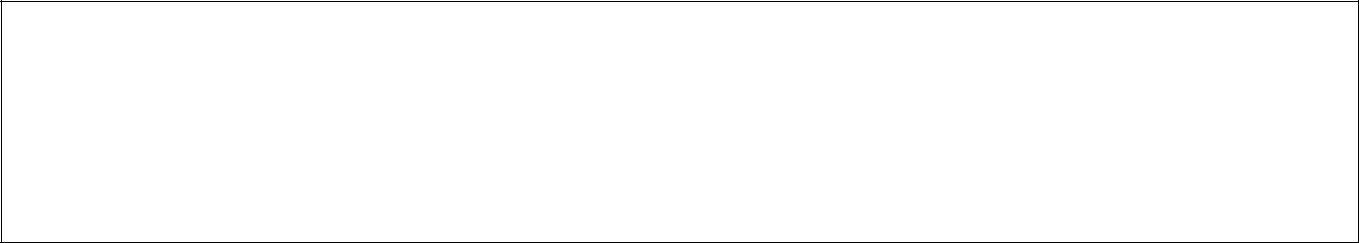
**obj** - This is the object to be appended in the list.

**Return Value**

This method does not return any value but updates existing list.

**Example**

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



#!/usr/bin/python3

list1 = ['C++', 'Java', 'Python']

list1.append('C#')

print ("updated list : ", list1)

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



updated list : ['C++', 'Java', 'Python', 'C#']

150

Python 3

**List count() Method**



**Description**

The **count()** method returns count of how many times obj occurs in list.

**Syntax**

Following is the syntax for count() method-



list.count(obj)

**Parameters**

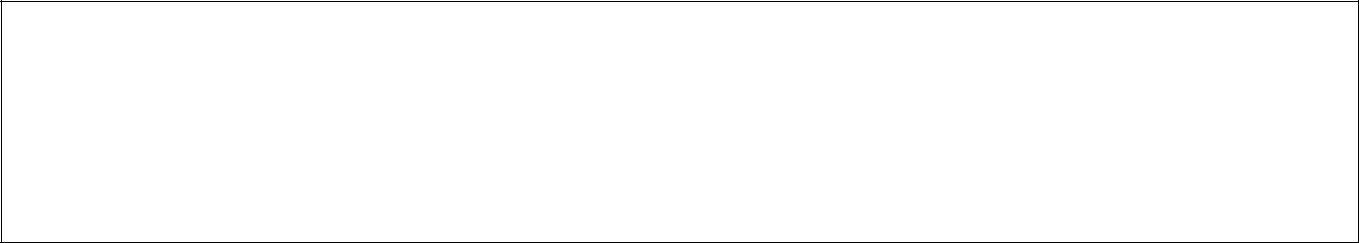
**obj** - This is the object to be counted in the list.

**Return Value**

This method returns count of how many times obj occurs in list.

**Example**

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



#!/usr/bin/python3

aList = [123, 'xyz', 'zara', 'abc', 123];

print ("Count for 123 : ", aList.count(123)) print ("Count for zara : ", aList.count('zara'))

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



Count for 123 : 2

Count for zara : 1

**List extend() Method**



**Description**

The **extend()** method appends the contents of seq to list.

**Syntax**

Following is the syntax for extend() method-

151

Python 3



list.extend(seq)

**Parameters**

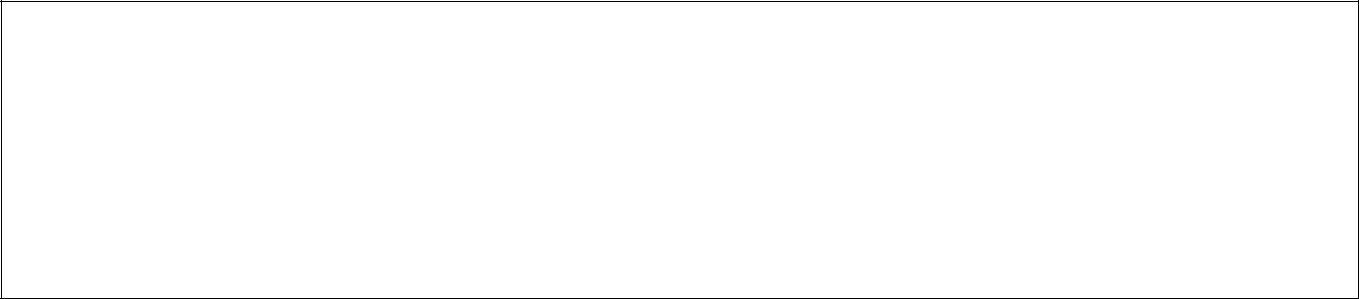
**seq** - This is the list of elements

**Return Value**

This method does not return any value but adds the content to an existing list.

**Example**

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



#!/usr/bin/python3

list1 = ['physics', 'chemistry', 'maths'] list2=list(range(5)) #creates list of numbers between 0-4 list1.extend('Extended List :', list2)

print (list1)

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



Extended List : ['physics', 'chemistry', 'maths', 0, 1, 2, 3, 4]

**List index() Method**



**Description**

The **index()** method returns the lowest index in list that obj appears.

**Syntax**

Following is the syntax for index() method-



list.index(obj)

**Parameters**

**obj** - This is the object to be find out.

**Return Value**

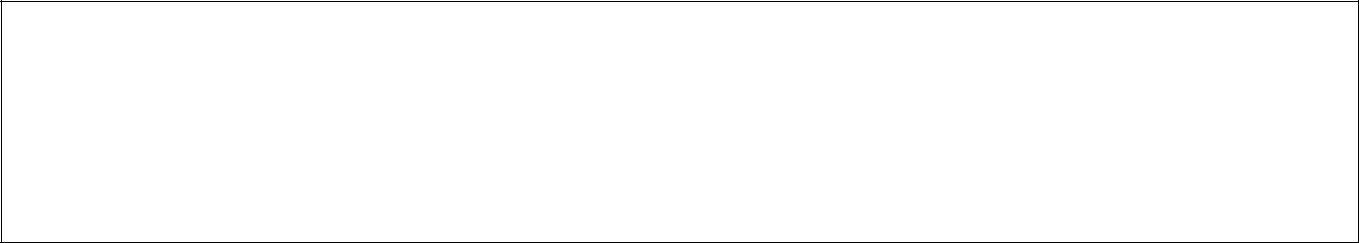
This method returns index of the found object otherwise raises an exception indicating that the value is not found.

**Example**

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

152

Python 3



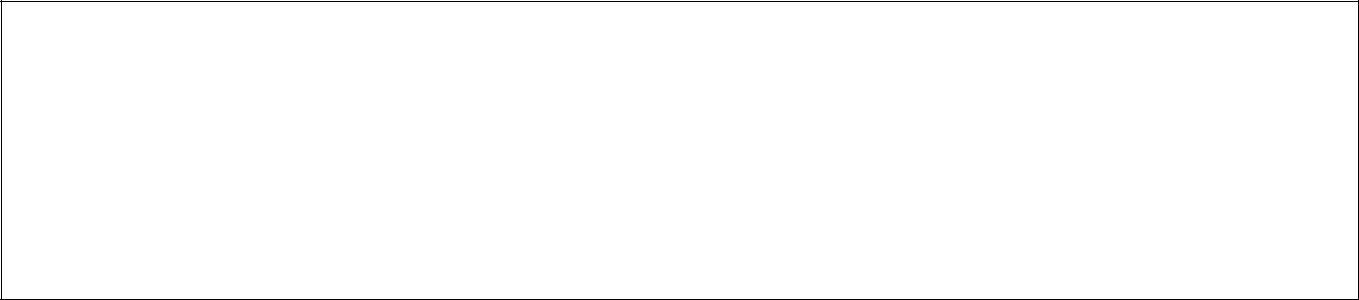
#!/usr/bin/python3

list1 = ['physics', 'chemistry', 'maths']

print ('Index of chemistry', list1.index('chemistry'))

print ('Index of C#', list1.index('C#'))

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



Index of chemistry 1

Traceback (most recent call last):

File "test.py", line 3, in

print ('Index of C#', list1.index('C#'))

ValueError: 'C#' is not in list

**List insert() Method**



**Description**

The **insert() method** inserts object obj into list at offset index.

**Syntax**

Following is the syntax for insert() method-



list.insert(index, obj)

**Parameters**

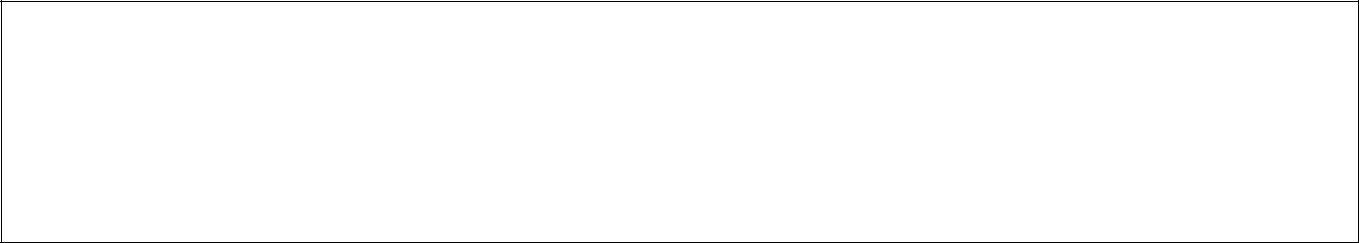
1. **index** - This is the Index where the object obj need to be inserted.
2. **obj** - This is the Object to be inserted into the given list.

**Return Value**

This method does not return any value but it inserts the given element at the given index.

**Example**

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



#!/usr/bin/python3

list1 = ['physics', 'chemistry', 'maths']

list1.insert(1, 'Biology')

print ('Final list : ', list1)

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

153

Python 3



Final list : ['physics', 'Biology', 'chemistry', 'maths']

**List pop() Method**



**Description**

The **pop()** method removes and returns last object or obj from the list.

**Syntax**

Following is the syntax for pop() method-



list.pop(obj=list[-1])

**Parameters**

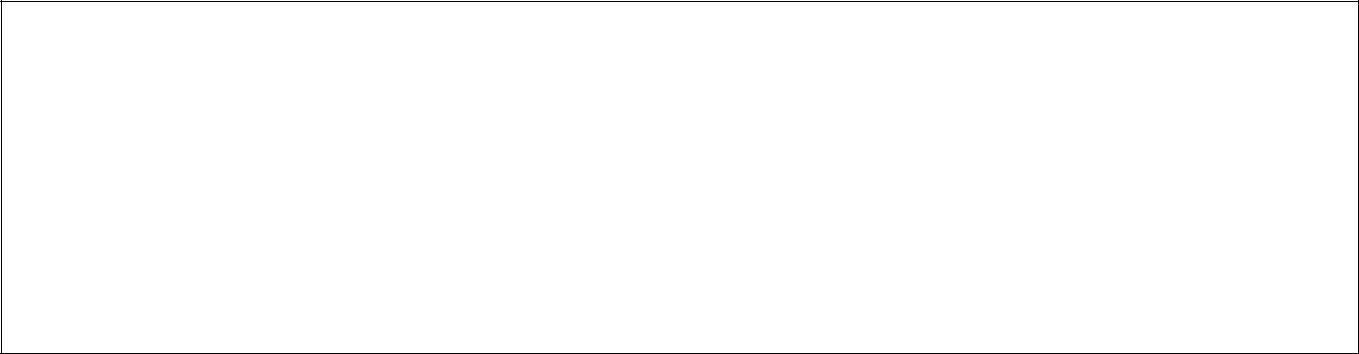
**obj** - This is an optional parameter, index of the object to be removed from the list.

**Return Value**

This method returns the removed object from the list.

**Example**

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



#!/usr/bin/python3

list1 = ['physics', 'Biology', 'chemistry', 'maths']

list1.pop()

print ("list now : ", list1)

list1.pop(1)

print ("list now : ", list1)

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



list now : ['physics', 'Biology', 'chemistry']

list now : ['physics', 'chemistry']

**List remove() Method**



**Parameters**

**obj** - This is the object to be removed from the list.

**Return Value**

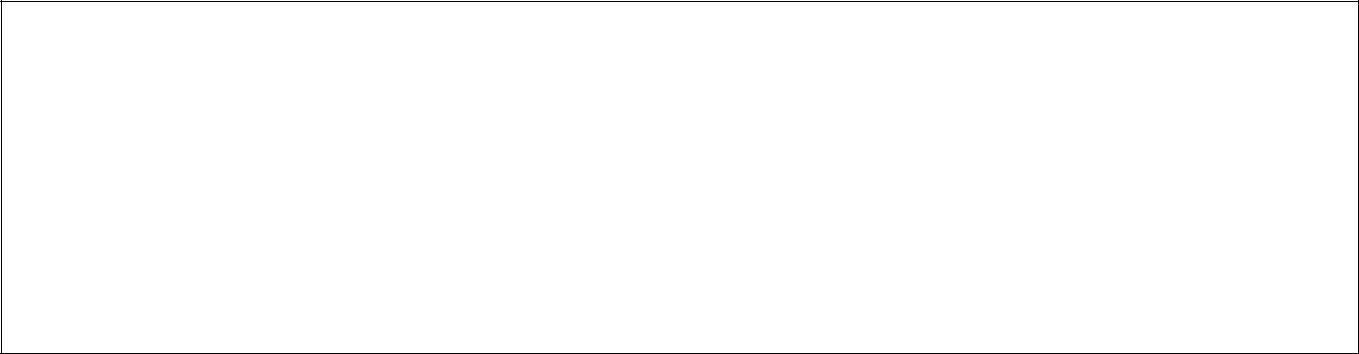
This method does not return any value but removes the given object from the list.

154

Python 3

**Example**

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



#!/usr/bin/python3

list1 = ['physics', 'Biology', 'chemistry', 'maths']

list1.remove('Biology')

print ("list now : ", list1)

list1.remove('maths')

print ("list now : ", list1)

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



list now : ['physics', 'chemistry', 'maths']

list now : ['physics', 'chemistry']

**List reverse() Method**



**Description**

The **reverse()** method reverses objects of list in place.

**Syntax**

Following is the syntax for reverse() method-



list.reverse()

**Parameters**

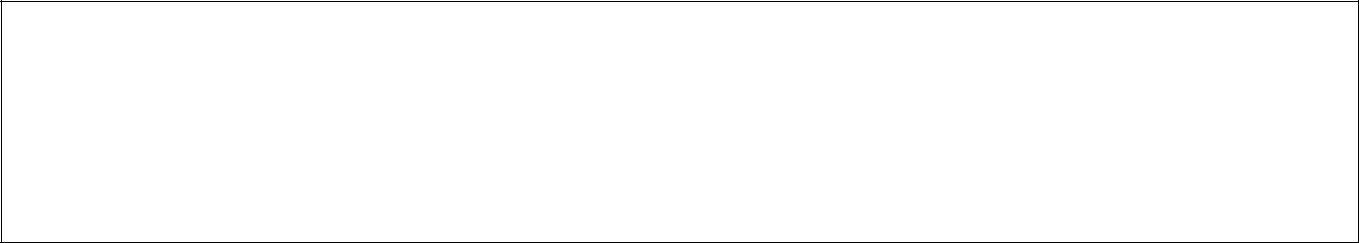
NA

**Return Value**

This method does not return any value but reverse the given object from the list.

**Example**

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



#!/usr/bin/python3

list1 = ['physics', 'Biology', 'chemistry', 'maths']

list1.reverse()

print ("list now : ", list1)

When we run above program, it produces following result-

155

Python 3



list now : ['maths', 'chemistry', 'Biology', 'physics']

**List sort() Method**



**Description**

The **sort()** method sorts objects of list, use compare function if given.

**Syntax**

Following is the syntax for sort() method-



list.sort([func])

**Parameters**

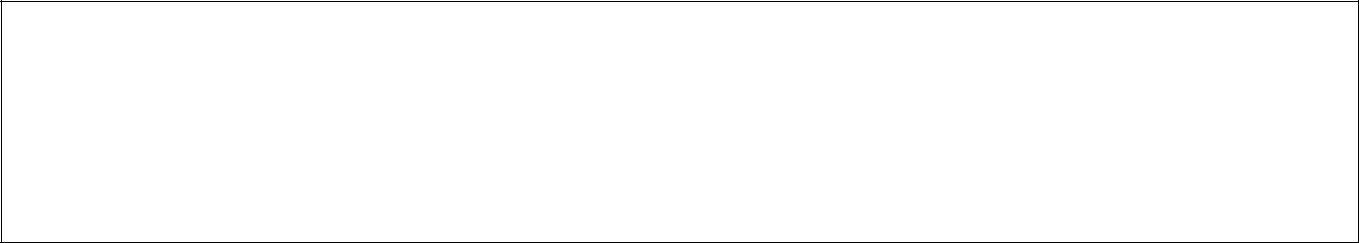
NA

**Return Value**

This method does not return any value but reverses the given object from the list.

**Example**

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



#!/usr/bin/python3

list1 = ['physics', 'Biology', 'chemistry', 'maths']

list1.sort()

print ("list now : ", list1)

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



list now : ['Biology', 'chemistry', 'maths', 'physics']