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. 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 a 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 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 −

```python
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 −

```python
#!/usr/bin/python

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 −

```python
#!/usr/bin/python

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]
```
Note: append method is discussed in 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 elements you are deleting or the remove method if you do not know. For example –

```python
#!/usr/bin/python
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]

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.

<table>
<thead>
<tr>
<th>Python Expression</th>
<th>Results</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>len[1, 2, 3]</td>
<td>3</td>
<td>Length</td>
</tr>
<tr>
<td>[1, 2, 3] + [4, 5, 6]</td>
<td>[1, 2, 3, 4, 5, 6]</td>
<td>Concatenation</td>
</tr>
<tr>
<td>['Hi!'] * 4</td>
<td>['Hi!', 'Hi!', 'Hi!', 'Hi!']</td>
<td>Repetition</td>
</tr>
<tr>
<td>3 in [1, 2, 3]</td>
<td>True</td>
<td>Membership</td>
</tr>
<tr>
<td>for x in [1, 2, 3]: print x,</td>
<td>1 2 3</td>
<td>Iteration</td>
</tr>
</tbody>
</table>

Indexing, Slicing, and Matrixes

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

Assuming following input –

L = ['spam', 'Spam', 'SPAM!']
<table>
<thead>
<tr>
<th>Python Expression</th>
<th>Results</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>L[2]</td>
<td>'SPAM!'</td>
<td>Offsets start at zero</td>
</tr>
<tr>
<td>L[-2]</td>
<td>'Spam'</td>
<td>Negative: count from the right</td>
</tr>
<tr>
<td>L[1:]</td>
<td>['Spam', 'SPAM!']</td>
<td>Slicing fetches sections</td>
</tr>
</tbody>
</table>

**Built-in List Functions & Methods:**

Python includes the following list functions –

<table>
<thead>
<tr>
<th>SN</th>
<th>Function with Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td><code>cmp(list1, list2)</code></td>
</tr>
<tr>
<td></td>
<td>Compares elements of both lists.</td>
</tr>
<tr>
<td>2</td>
<td><code>len(list)</code></td>
</tr>
<tr>
<td></td>
<td>Gives the total length of the list.</td>
</tr>
<tr>
<td>3</td>
<td><code>max(list)</code></td>
</tr>
<tr>
<td></td>
<td>Returns item from the list with max value.</td>
</tr>
<tr>
<td>4</td>
<td><code>min(list)</code></td>
</tr>
<tr>
<td></td>
<td>Returns item from the list with min value.</td>
</tr>
<tr>
<td>5</td>
<td><code>list(seq)</code></td>
</tr>
<tr>
<td></td>
<td>Converts a tuple into list.</td>
</tr>
</tbody>
</table>

Python includes following list methods

<table>
<thead>
<tr>
<th>SN</th>
<th>Methods with Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td><code>list.append(obj)</code></td>
</tr>
<tr>
<td></td>
<td>Appends object obj to list</td>
</tr>
<tr>
<td>2</td>
<td><code>list.count(obj)</code></td>
</tr>
<tr>
<td></td>
<td>Returns count of how many times obj occurs in list</td>
</tr>
<tr>
<td>3</td>
<td><code>list.extend(seq)</code></td>
</tr>
</tbody>
</table>
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

6

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

Removes and returns last object or obj from list

7

```
list.remove(obj)
```

Removes object obj from list

8

```
list.reverse()
```

Reverses objects of list in place

9

```
list.sort(func)
```

Sorts objects of list, use compare func if given