A tuple is a sequence of immutable Python objects. Tuples are sequences, just like lists. The differences between tuples and lists are, the tuples cannot be changed unlike lists and tuples use parentheses, whereas lists use square brackets.

Creating a tuple is as simple as putting different comma-separated values. Optionally you can put these comma-separated values between parentheses also. For example –

```python
tup1 = ('physics', 'chemistry', 1997, 2000);
tup2 = (1, 2, 3, 4, 5 );
tup3 = "a", "b", "c", "d";
```

The empty tuple is written as two parentheses containing nothing –

```python
tup1 = ();
```

To write a tuple containing a single value you have to include a comma, even though there is only one value –

```python
tup1 = (50,);
```

Like string indices, tuple indices start at 0, and they can be sliced, concatenated, and so on.

### Accessing Values in Tuples:

To access values in tuple, 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
tup1 = ('physics', 'chemistry', 1997, 2000);
tup2 = (1, 2, 3, 4, 5, 6, 7);
print "tup1[0]: ", tup1[0]
print "tup2[1:5]: ", tup2[1:5]
```

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

```
tup1[0]: physics
tup2[1:5]: [2, 3, 4, 5]
```

### Updating Tuples

Tuples are immutable which means you cannot update or change the values of tuple elements. You are able to take portions of existing tuples to create new tuples as the following example demonstrates –

```python
#!/usr/bin/python
tup1 = (12, 34.56);
tup2 = ('abc', 'xyz');

# Following action is not valid for tuples
# tup1[0] = 100;

# So let's create a new tuple as follows
print tup3
```

```
tup3 = tup1 + tup2;
```
When the above code is executed, it produces the following result –

(12, 34.56, 'abc', 'xyz')

Delete Tuple Elements

Removing individual tuple elements is not possible. There is, of course, nothing wrong with putting together another tuple with the undesired elements discarded.

To explicitly remove an entire tuple, just use the `del` statement. For example:

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

Tup = ('physics', 'chemistry', 1997, 2000);
print Tup
del Tup;
print "After deleting Tup : "
print Tup
```

This produces the following result. Note an exception raised, this is because after `del Tup` tuple does not exist any more –

After deleting Tup :
Traceback (most recent call last):
  File "test.py", line 9, in <module>
    print Tup;
NameError: name 'tup' is not defined

Basic Tuples Operations

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

In fact, tuples 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 tuples are sequences, indexing and slicing work the same way for tuples as they do for strings. Assuming following input –

```python
L = ('spam', 'Spam', 'SPAM!')
```

<table>
<thead>
<tr>
<th>Python Expression</th>
<th>Results</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
No Enclosing Delimiters

Any set of multiple objects, comma-separated, written without identifying symbols, i.e., brackets for lists, parentheses for tuples, etc., default to tuples, as indicated in these short examples −

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

print 'abc', -4.24e93, 18+6.6j, 'xyz'
x, y = 1, 2;
print "Value of x, y : ", x, y
```

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

```
abc -4.24e+93 (18+6.6j) xyz
Value of x, y : 1 2
```

Built-in Tuple Functions

Python includes the following tuple functions −

<table>
<thead>
<tr>
<th>SN</th>
<th>Function with Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td><code>cmpTuple1, tuple2</code></td>
</tr>
<tr>
<td></td>
<td>Compares elements of both tuples.</td>
</tr>
<tr>
<td>2</td>
<td><code>lenTuple</code></td>
</tr>
<tr>
<td></td>
<td>Gives the total length of the tuple.</td>
</tr>
<tr>
<td>3</td>
<td><code>maxTuple</code></td>
</tr>
<tr>
<td></td>
<td>Returns item from the tuple with max value.</td>
</tr>
<tr>
<td>4</td>
<td><code>minTuple</code></td>
</tr>
<tr>
<td></td>
<td>Returns item from the tuple with min value.</td>
</tr>
<tr>
<td>5</td>
<td><code>tupleSeq</code></td>
</tr>
<tr>
<td></td>
<td>Converts a list into tuple.</td>
</tr>
</tbody>
</table>