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

```
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

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/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

```
#!/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 element(s) you are deleting or the remove() method if you do not know.

For example

```
#!/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.

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,	1 2 3	Iteration

Print Page