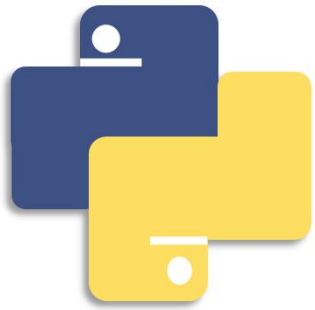

Python Programming - VI

LIST and It's Operations

CONTENTS

LIST:

- Accessing Values In lists
- Updating Lists
- Deleting List Elements
- Basic List Operations

Accessing values in List

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

Example:

```
list1 = ['physics', 'chemistry', 1997, 2000];

print "list1[0]: ", list1[0]
```

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

```
list1[0]: physics
```

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.

Example:

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

print "Value available at index 2 : "

print list[2];

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
```

Deleting List Elements

To remove a list element, you can use either the del statement if you know exactly which element(s) you are deleting.

Example:

```
list1 = ['physics', 'chemistry', 1997, 2000];

print list1;

del list1[2];

print list1;
```

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

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

['physics', 'chemistry', 2000]
```

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 Expressions	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

• Indexing, Slicing, and Matrixes

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

Assume the following input:

```
L = ['physics', 'Physics', 'PHYSICS']
```

Python Expresion	Results	Description
L[2]	'PHYSICS'	Offsets start at zero
L[-2]	'Physics'	Negative: count from the right
L[1:]	['Physics', 'PHYSICS']	Slicing fetches sections

Built-in List Functions and Methods

Python includes the following list functions:

Sr. No.	Function with Description
1	cmp(list1, list2) Compares elements of both lists.
2	len(list) Gives the total length of the list.
3	max(list) Returns item from the list with max value.
4	min(list) Returns item from the list with min value.
5	list(seq) Converts a tuple into list.

Let us go through the functions in detail:

- **Cmp(list1, list2)**

Description:

The method **cmp()** compares elements of two lists.

Syntax:

```
cmp(list1,list2)
```

Example:

```
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)**

  Description:

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

  Syntax:

```
len(List)
```

  This method returns the number of elements in the list.

Example:

```
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)**

Description:

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

Syntax:

```
max(list)
```

Example:

```
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)**

Description:

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

Syntax:

```
min(list)
```

Example:

```
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.append(obj)**

Description:

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

Syntax

Following is the syntax for append() method:

list.append(obj)

Example:

```
aList = [123, 'xyz', 'zara', 'abc'];
aList.append( 2009 );
print "Updated List : ", aList;
```

Output:

```
Updated List :
[123, 'xyz', 'zara', 'abc', 2009]
```

- **list.count(obj)**

Description:

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

Syntax:

Following is the syntax for count() method:

```
list.count(obj)
```

Example:

```
aList = [123, 'xyz', 'zara', 'abc', 123];
print "Count for 123 : ", aList.count(123);
print "Count for zara : ", aList.count('zara');
```

Output:

```
Count for 123 :2
Count for zara :1
```

- **list.extend(seq)**

Description:

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

Syntax:

Following is the syntax for extend() method:

list.extend(seq)

Example:

```
aList = [123, 'xyz', 'zara', 'abc', 123];
bList = [2009, 'manni'];
aList.extend(bList)
print "Extended List : ", aList ;
```

Output:

```
Extended List :[123, 'xyz', 'zara', 'abc', 123, 2009, 'manni']
```

- **list.index(obj)**

Description:

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

Syntax:

Following is the syntax for index() method:

Example:

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

print "Index for xyz : ", aList.index( 'xyz' ) ;
print "Index for zara : ", aList.index( 'zara' ) ;
```

Output:

```
Index for xyz :1
Index for zara :2
```

- **list.insert(index,obj)**

Description:

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

Syntax:

Following is the syntax for insert() method:

list.insert(index, obj)

Example:

```
aList = [123, 'xyz', 'zara', 'abc']
aList.insert( 3, 2009)
print "Final List : ", aList
```

Output:

```
Final List : [123, 'xyz', 'zara', 2009, 'abc']
```

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

Description:

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

Syntax:

Following is the syntax for pop() method:

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

Example:

```
aList = [123, 'xyz', 'zara', 'abc'];
print "A List : ", aList.pop();
print "B List : ", aList.pop(2);
```

Output:

```
A List : abc
B List : zara
```

- **List.reverse()**

Description:

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

Syntax:

Following is the syntax for reverse() method:

list.reverse()

Example:

```
aList = [123, 'xyz', 'zara', 'abc', 'xyz'];
aList.reverse();
print "List : ", aList;
```

Output:

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

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

Description:

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

Syntax:

Following is the syntax for reverse() method:

list.reverse()

Example:

```
aList = [123, 'xyz', 'zara', 'abc', 'xyz']
aList.reverse()
print "List : ", aList
print aList.sort()
```

Output:

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

THANK YOU

@mitu_skillologies

/mITuSkillologies

@mitu_group

/company/mitu-skillologies