

LIST

Presented By
M.Malarmathi
AP/IT



List

- A list is a sequence of values
- The values in a list are called elements or sometimes items. they can be any type
- list contains a string, a float, an integer.

Syntax

- elements in square brackets [...]
 - [10, 20, 30, 40]
 - ['a', 'b', 'c']



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

Example

```
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]
```

- OUTPUT

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

```
list2[1:5]: [2, 3, 4, 5]
```



Updating Lists

- Update single or multiple elements of lists by left-hand side of the assignment operator or append method
- Example

```
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]
```

OUTPUT

```
Value available at index 2 : 1997
```

```
New value available at index 2 : 2001
```



Delete List Elements



- To remove a list element, use del statement
- Example

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

```
print list1
```

```
del list1[2]
```

```
print "After deleting value at index 2 : "
```

```
print list1
```

OUTPUT

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

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



List operations

- Lists respond to the + and * operators much like strings; they mean concatenation and repetition.
- The + operator concatenates lists:

```
>>> a = [1, 2, 3]
>>> b = [4, 5, 6]
>>> c = a + b
>>> print c
[1, 2, 3, 4, 5, 6]
```
- Similarly, the * operator repeats a list a given number of times:

```
>>> [0] * 4
[0, 0, 0, 0]
>>> [1, 2, 3] * 3
[1, 2, 3, 1, 2, 3, 1, 2, 3]
```
- The first example repeats [0] four times. The second example repeats the list [1, 2, 3] three times.



List slices



- The slice operator also works on lists:

```
>>> t = ['a', 'b', 'c', 'd', 'e', 'f']  
>>> t[1:3]  
['b', 'c']  
>>> t[:4]  
['a', 'b', 'c', 'd']  
>>> t[3:]  
['d', 'e', 'f']
```

- If you omit the first index, the slice starts at the beginning. If you omit the second, the slice goes to the end. So if you omit both, the slice is a copy of the whole list.

```
>>> t[:]  
['a', 'b', 'c', 'd', 'e', 'f']
```

- Since lists are mutable, it is often useful to make a copy before performing operations that fold, spindle or mutilate lists.

- A slice operator on the left side of an assignment can update multiple elements:

```
>>> t = ['a', 'b', 'c', 'd', 'e', 'f']  
>>> t[1:3] = ['x', 'y']  
>>> print t  
['a', 'x', 'y', 'd', 'e', 'f']
```



- `>>> l=[1,2,3,4]`
- `>>> l[1:]+l[:1]`
- `[2, 3, 4, 1]`
- `>>> l=[1,2,3,4]`
- `>>> l[2:]+l[:2]`
- `[3, 4, 1, 2]`
- `>>> l[-1:]+l[:-1]`
- `[4, 1, 2, 3]`



List methods

Sr.No.	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
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



Append():

adds a new element to the end of a list

```
>>> t = ['a', 'b', 'c']  
>>> t.append('d')  
>>> print t  
['a', 'b', 'c', 'd']
```

- Extend():

extend takes a list as an argument and appends all of the elements

```
>>> t1 = ['a', 'b', 'c']  
>>> t2 = ['d', 'e']  
>>> t1.extend(t2)  
>>> print t1  
['a', 'b', 'c', 'd', 'e']
```



Sort():

arranges the elements of the list from low to high:

```
>>> t = ['d', 'c', 'e', 'b', 'a']  
>>> t.sort()  
>>> print t  
['a', 'b', 'c', 'd', 'e']
```

- Count():

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

```
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





Index():
returns index

```
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

- Insert(index,obj)

inserts the given element at the given index

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



Pop()

removes and returns last object

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

```
print "A List : ", aList.pop()
```

```
print "B List : ", aList.pop(2)
```

O/P

A List : abc

B List : zara

- Remove()

removes the given object from the list.

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

```
aList.remove('xyz')
```

```
print "List : ", aList
```

```
aList.remove('abc')
```

```
print "List : ", aList
```

O/P

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

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





Reverse()



reverse the given object from the list.

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

```
aList.reverse()
```






```
print "List : ", aList
```

O/P

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



Built-in functions in list

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.



Thank You