Python - List

In Python, the list is a collection of items of different data types. It is an ordered sequence of items. A list object contains one or more items, not necessarily of the same type, which are separated by comma and enclosed in square brackets [].

Syntax:
list = [value1, value2, value3,...valueN]

The following declares a list type variable.

```
>>> names=["Jeff", "Bill", "Steve", "Mohan"]
```

A list can also contain elements of different types.

```
>>> orderItem=[1, "Jeff", "Computer", 75.50, True]
```

The above list orderItem includes five elements. Each individual element in the sequence is accessed by the index in the square brackets []. An index starts with zero, as shown below.

```
>>> orderItem=[1, "Jeff", "Computer", 75.50, True]
>>> orderItem[0]
1
>>> orderItem[1]
'Jeff'
>>> orderItem[2]
'Computer'
>>> orderItem[3]
75.50
>>> orderItem[4]
True
```

The list object is mutable. It is possible to modify its contents, which will modify the value in the memory.

For instance, item at index 2 in orderItem can be modified as shown below.

```
>>> orderItem=[1, "Jeff", "Computer", 75.50, True]
>>> orderItem[2]="Laptop"
>>> orderItem
[1, "Jeff", "Laptop", 75.50, True]
```

It will throw an error "index out of range" if the element at the specified index does not exist.

```
>>> orderItem=[1, "Jeff", "Computer", 75.50, True]
>>> orderItem[5]
Traceback (most recent call last):
File "<stdin>", line 1, in <module>
IndexError: list index out of range
```

Lists are generally used to store homogenous collections, i.e. items of similar types.

```
>>> languages=['Python', 'Java', 'C#', 'PHP']
>>> temperatures=[56, 65, 78]
```

Use the *del* keyword to delete the list object.

```
>>> del languages
>>> languages
Traceback (most recent call last):
File "<stdin>", line 1, in <module>
NameError: name 'languages' is not defined
```

List Operators

Like the string, the list is also a sequence. Hence, the operators used with strings are also available for use with the list (and tuple also).

Operator	Description	Example

+ Concatenation	Returns a list containing all the elements of the first and the second list.	>>> L1=[1,2,3]
		>>> L2=[4,5,6]
		>>> L1+L2
		[1, 2, 3, 4, 5, 6]

* Repetition	Concatenates multiple copies of the same list.	>>> L1*4 [1, 2, 3, 1, 2, 3, 1, 2, 3, 1, 2, 3]

[] slice	Returns the item at the given index. A negative index counts the position from the right side.	>>> L1=[1, 2, 3, 4, 5, 6]
		>>> L1[3]
		4
		>>> L1[-2]
		5

[:] - Range slice	Fetches items in the range specified by the two index operands separated by : symbol.
	If the first operand is omitted, the range starts from the zero index. If the second operand is omitted, the range goes up to the end of the list.
		>>> L1=[1, 2, 3, 4, 5, 6]
		>>> L1[1:4]
		[2, 3, 4]
		>>> L1[3:]

		[4, 5, 6] >>> L1[:3] [1, 2, 3]

in	Returns true if an item exists in the given list.	>>> L1=[1, 2, 3, 4, 5, 6]
		>>> 4 in L1
		True
		>>> 10 in L1
		False

not in	Returns true if an item does not exist in the given list.	>>> L1=[1, 2, 3, 4, 5, 6]
		>>> 5 not in L1
		False
		>>> 10 not in L1
		True

Built-in List Methods

len()

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

```
>>> L1=[12,45,43,8,35]
>>> len(L1)
5
```

max()

The max() method returns the largest number, if the list contains numbers. If the list contains strings, the one that comes last in alphabetical order will be returned.

```
>>> L1=[12,45,43,8,35]
>>> max(L1)
45
>>> L2=['Python', 'Java', 'C++']
>>> max(L2)
'Python'
```

min()

The min() method returns the smallest number, if the list contains numbers. If the list contains strings, the one that comes first in alphabetical order will be returned.

```
>>> L1=[12, 45, 43, 8, 35]
>>> min(L1)
8
>>> L2=['Python', 'Java', 'C++']
>>> min(L2)
'C++'
```

Note that if a list object contains strings as well as numeric items, max() and min() functions throw an error.

```
>>> L1=[1,'aa',12.22]
>>> max(L1)
Traceback (most recent call last):
File "<stdin>", line 1, in <module>
TypeError: '>' not supported between instances of 'str' and 'int'
```

append()

Adds an item at the end of the list.

```
>>> L2=['Python', 'Java', 'C++']
>>> L2.append('PHP')
>>> L2
['Python', 'Java', 'C++', 'PHP']
```

insert()

Inserts an item in a list at the specified index.

```
>>> L2=['Python', 'Java', 'C++']
>>> L2.insert(1,'Perl')
>>> L2
['Python', 'Perl', 'Java', 'C++']
```

remove()

Removes a specified object from the list.

```
>>> L2=['Python', 'Perl', 'Java', 'C++']
>>> L2.remove('Java')
>>> L2
['Python', 'Perl', 'C++']
```

pop()

Removes and returns the last object in the list.

```
>>> L2=['Python', 'Perl', 'Java', 'C++']
>>> L2.pop()
'C++'
>>> L2
['Python', 'Perl', 'Java']
```

reverse()

Reverses the order of the items in a list.

```
>>> L2=['Python', 'Perl', 'Java', 'C++']
>>> L2.reverse()
>>> L2
['C++', 'Java', 'Perl', 'Python']
```

sort()

Rearranges the items in the list according to the alphabetical order. Default is the ascending order. For descending order, put reverse=True as an argument in the function bracket.

```
>>> L2=['Python', 'C++', 'Java', 'Ruby']
>>> L2.sort()
>>> L2
['C++', 'Java', 'Python', 'Ruby']
>>> L2.sort(reverse=True)
>>> L2
['Ruby', 'Python', 'Java', 'C++']
```

The following utility functions help in converting one sequence data type to another.

list()

Converts a tuple or string to a list object.

```
>>> t2=('python', 'java', 'C++')
>>> list(t2)
['python', 'java', 'C++']
>>> s1="Altran"
>>> list(s1)
[‘A’,’l’,’t’,’r’,’a’,’n’]
```

tuple()

Converts a list or string to a tuple object.

```
>>> L2=['C++', 'Java', 'Python', 'Ruby']
>>> tuple(L2)
('C++', 'Java', 'Python', 'Ruby')
>>> s1="Altran"
>>> tuple(s1)
(‘A’,’l’,’t’,’r’,’a’,’n’)
```