List in Python

Lists are Python's most flexible ordered collection object type. It can also be referred to as a sequence that is an ordered collection of objects that can host objects of any data type, such as Python Numbers, Python Strings and nested lists as well. Lists are one of the most used and versatile Python Data Types. In this module, we will learn all about lists in order to get started with them.
Following is the list of all topics that are going to be covered in this module.

- Creating Lists in Python
- Creating Multi-dimensional Lists in Python
- Python List Comprehension
- Python Lists Extension
- Accessing Lists in Python
- Common List Operations in Python
  - Slicing Python Lists
  - Iterating through Python Lists
  - Update or Add Elements in a Python List
  - Remove elements from list in python
  - Remove duplicates from lists in python
  - Sorting Lists in Python
  - Reverse a list in python
- Python List Functions and Methods

Unlike strings, lists can contain any sort of objects: numbers, strings, and even other lists. Python lists are:

- Ordered collections of arbitrary objects
- Accessed by offset
- Arrays of object references
- Of variable length, heterogeneous, and arbitrarily nestable
- Of the category, mutable sequence
- Data types in which elements are stored in the index basis with starting index as 0
- Enclosed between square brackets '[]'

Creating Multi-dimensional Lists in Python

A list can hold other lists as well which can result in multi-dimensional lists. Next, we will see how to create multi-dimensional lists, one by one.

One-dimensional Lists in Python:

```
init_list = [0]*3
print(init_list)
```

Output:

```
[0, 0, 0]
```

Two-dimensional Lists In Python:

```
two_dim_list = [ [0]*3 ] *3
print(two_dim_list)
```

Output:

```
[[0, 0, 0], [0, 0, 0], [0, 0, 0]]
```

Three-dimensional Lists in Python:

```
two_dim_list = [[ [0]*3 ] *3]*3
print(two_dim_list)
```

Output:

```
[[[0, 0, 0], [0, 0, 0], [0, 0, 0]],
[[0, 0, 0], [0, 0, 0], [0, 0, 0]],
[[0, 0, 0], [0, 0, 0], [0, 0, 0]]]
```

List Extension

Python allows lists to resize in many ways. We can do that just by adding two or more of them.

Example:

```
two_dim = [[0]*3 for i in range(3)]print(two_dim)
[[0, 0, 0], [0, 0, 0], [0, 0, 0]]
two_dim[0][2] = 1
print(two_dim)
```

Output:

```
[[0, 0, 1], [0, 0, 0], [0, 0, 0]]
```

- **extend():**

Alternately, we can do extension by using the extend() method. See the following example:

```
L1 = [‘a’, ‘b’]
L2 = [‘c’, ‘d’]
L1.extend(L2)
print(L1)
```

Output:

```
[‘a’, ‘b’, ‘c’, ‘d’]
```

- **append():**

Next, we can append a value to a list by calling the append() method. See the following example:

```
L1 = [‘a’, ‘b’]
L2 = [‘c’, ‘d’]
L1.extend(L2)
print(L1)
Output:
[‘a’, ‘b’, ‘c’, ‘d’]
```

Accessing Lists in Python

Much similar to strings, we can use the index number to access items in lists as shown below.

Example:

```
list1 = [1,2,3,4,5]
```

Accessing a List Using Reverse Indexing

To access a list in reverse order, we have to use indexing from −1, −2…. Here, −1 represents the last item in the list.

```
print(list1[-1])
print(list1[-3])
Output:
5
3
```

Common List Operations in Python

Following is the list of some of the most common list operations in Python, along with their descriptions and examples.

Slicing Python Lists

Slicing operation is used to print a list up to a specific range. We can use slice operation by including the starting index and ending index of the range that we want to print separated by a colon as shown below:

```
list1[2:4]
output:
[3, 4]
list1[2:-1]
output:
[3, 4]
list1[:2]
output:
[1, 2]
```

Iterating through Python Lists

Iterating is quite simple in lists. We can just use Python for loop to iterate, as shown below:

```
list1 = [1,2,3,4,5]
for element in list1:
print(element)
```

Output:

```
1
2
3
4
5
```

Update or Add Elements in a Python List

We can update a particular item or multiple items of a list by using the slice operation, and then add an element using the append () method as shown below.

Example:

```
list1[4] = ‘number’print(list1)list1[4:7] = [“Apple”,”Mango”,”Banana”]
print(list1)list1.insert(0,33)
print(list1)list1.insert(6,29)
print(list1)
```

Output:

```
[1, 2, 3, 4, ‘number’]
[1, 2, 3, 4, ‘Apple’, ‘Mango’, ‘Banana’]
[33, 1, 2, 3, 4, ‘Apple’, ‘Mango’, ‘Banana’]
[33, 1, 2, 3, 4, ‘Apple’, 29, ‘Mango’, ‘Banana’]
```

Remove elements from list in python

There are three ways of removing elements from lists. We can either use the del keyword to remove a particular element or we can use the remove () method, and the last way is using the pop () method, as shown in the following code block:

```
list1 = [1,2,3,4,5]
del list1[2]
list2 = [1,2,3,4,5]
list2.remove(4)
print(list2)
list3 = [1,2,3,4,5]
print(list3.pop(1))
print(list3)
```

Output:

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

Remove duplicates from lists in python

Here's an example of list where some items are repeating. Let us see how we can remove duplicates from list in python.

```
mylist = ["a", "b", "c", "d", "c"]
mylist = list(dict.fromkeys(mylist))
```

output:

```
["a", "b","c","d"]
```

Reverse a list in python

```
lst = [10, 11, 12, 13, 14, 15]
lst.reverse()
print(lst)
```

output:

```
[15, 14, 13, 12, 11, 10]
```

Sorting Lists in Python

Python list implements the **sort()** method for ordering (in both ascending and descending order) its elements in place.

list1.sort()

Sorting in ascending order:

```
list1 = [1,3,2,4,5,9,6]
list1.sort()
print(list1)
```

output:

```
[1, 2, 3, 4, 5, 6, 9]
```

Sorting in descending order:

```
list1 = [1,3,2,4,5,9,6]
list1.sort(reverse=True)
print(list1)
```

output:

```
[9, 6, 5, 4, 3, 2, 1]
```

Python List Functions and Methods

Let's understand different types of Python functions for lists through the following table that contains a list of different functions with their respective descriptions.

Method	Description
min(list_name)	Returns the minimum value from a list in Python
max(list_name)	Returns the largest value from a list in Python
len(list_name)	Returns the number of elements in a list in Python
cmp(list1,list2)	Compares two lists in Python
list.reverse()	Reverses a list in Python
list.sort	Sorts a list in Python
list(sequence)	Converts the sequence of a list in Python
list.append(value)	Adds a value into a list in Python
list.remove(value)	Removes a value from a list in Python