Python List Methods

Python has some list methods that you can use to perform frequency occurring task (related to list) with ease. For example, if you want to add element to a list, you can use append() method.

The page contains all methods of list objects. Also, the page includes built-in functionsthat can take list as a parameter and perform some task. For example, all() function returns True if all elements of an list (iterable) is true. If not, it returns False.

Bottom of Form

| Method | Description |
| --- | --- |
| Python List append() | Add Single Element to The List |
| Python List extend() | Add Elements of a List to Another List |
| Python List insert() | Inserts Element to The List |
| Python List remove() | Removes Element from the List |
| Python List index() | returns smallest index of element in list |
| [Python List count()](https://www.programiz.com/python-programming/methods/list/count) | returns occurrences of element in a list |
| Python List pop() | Removes Element at Given Index |
| Python List reverse() | Reverses a List |
| Python List sort() | sorts elements of a list |
| Python List copy() | Returns Shallow Copy of a List |
| Python List clear() | Removes all Items from the List |
| Python any() | Checks if any Element of an Iterable is True |
| Python all() | returns true when all elements in iterable is true |
| [Python ascii()](https://www.programiz.com/python-programming/methods/built-in/ascii) | Returns String Containing Printable Representation |
| [Python bool()](https://www.programiz.com/python-programming/methods/built-in/bool) | Coverts a Value to Boolean |
| [Python enumerate()](https://www.programiz.com/python-programming/methods/built-in/enumerate) | Returns an Enumerate Object |
| [Python filter()](https://www.programiz.com/python-programming/methods/built-in/filter) | constructs iterator from elements which are true |
| [Python iter()](https://www.programiz.com/python-programming/methods/built-in/iter) | returns iterator for an object |
| [Python list() Function](https://www.programiz.com/python-programming/methods/built-in/list) | creates list in Python |
| [Python len()](https://www.programiz.com/python-programming/methods/built-in/len) | Returns Length of an Object |
| [Python max()](https://www.programiz.com/python-programming/methods/built-in/max) | returns largest element |
| [Python min()](https://www.programiz.com/python-programming/methods/built-in/min) | returns smallest element |
| [Python map()](https://www.programiz.com/python-programming/methods/built-in/map) | Applies Function and Returns a List |
| [Python reversed()](https://www.programiz.com/python-programming/methods/built-in/reversed) | returns reversed iterator of a sequence |
| [Python slice()](https://www.programiz.com/python-programming/methods/built-in/slice) | creates a slice object specified by range() |
| [Python sorted()](https://www.programiz.com/python-programming/methods/built-in/sorted) | returns sorted list from a given iterable |
| [Python sum()](https://www.programiz.com/python-programming/methods/built-in/sum) | Add items of an Iterable |
| [Python zip()](https://www.programiz.com/python-programming/methods/built-in/zip) | Returns an Iterator of Tuples |

## How to create a list?

In Python programming, a list is created by placing all the items (elements) inside a square bracket [ ], separated by commas.

It can have any number of items and they may be of different types (integer, float, string etc.).

# empty list

my\_list = []

# list of integers

my\_list = [1, 2, 3]

# list with mixed datatypes

my\_list = [1, "Hello", 3.4]

Also, a list can even have another list as an item. This is called nested list.

# nested list

my\_list = ["mouse", [8, 4, 6], ['a']]

## How to access elements from a list?

There are various ways in which we can access the elements of a list.

### List Index

We can use the index operator [] to access an item in a list. Index starts from 0. So, a list having 5 elements will have index from 0 to 4.

Trying to access an element other that this will raise an IndexError. The index must be an integer. We can't use float or other types, this will result into TypeError.

Nested list are accessed using nested indexing.

my\_list = ['p','r','o','b','e']

# Output: p

print(my\_list[0])

# Output: o

print(my\_list[2])

# Output: e

print(my\_list[4])

# Error! Only integer can be used for indexing

# my\_list[4.0]

# Nested List

n\_list = ["Happy", [2,0,1,5]]

# Nested indexing

# Output: a

print(n\_list[0][1])

# Output: 5

print(n\_list[1][3])

### Negative indexing

Python allows negative indexing for its sequences. The index of -1 refers to the last item, -2 to the second last item and so on.

my\_list = ['p','r','o','b','e']

# Output: e

print(my\_list[-1])

# Output: p

print(my\_list[-5])

### How to slice lists in Python?

We can access a range of items in a list by using the slicing operator (colon).

my\_list = ['p','r','o','g','r','a','m','i','z']

# elements 3rd to 5th

print(my\_list[2:5])

# elements beginning to 4th

print(my\_list[:-5])

# elements 6th to end

print(my\_list[5:])

# elements beginning to end

print(my\_list[:])

Slicing can be best visualized by considering the index to be between the elements as shown below. So if we want to access a range, we need two index that will slice that portion from the list.



## How to change or add elements to a list?

List are mutable, meaning, their elements can be changed unlike string or tuple.

We can use assignment operator (=) to change an item or a range of items.

# mistake values

odd = [2, 4, 6, 8]

# change the 1st item

odd[0] = 1

# Output: [1, 4, 6, 8]

print(odd)

# change 2nd to 4th items

odd[1:4] = [3, 5, 7]

# Output: [1, 3, 5, 7]

print(odd)

We can add one item to a list using append() method or add several items using extend()method.

odd = [1, 3, 5]

odd.append(7)

# Output: [1, 3, 5, 7]

print(odd)

odd.extend([9, 11, 13])

# Output: [1, 3, 5, 7, 9, 11, 13]

print(odd)

We can also use + operator to combine two lists. This is also called concatenation.

The \* operator repeats a list for the given number of times.

odd = [1, 3, 5]

# Output: [1, 3, 5, 9, 7, 5]

print(odd + [9, 7, 5])

#Output: ["re", "re", "re"]

print(["re"] \* 3)

Furthermore, we can insert one item at a desired location by using the method insert() or insert multiple items by squeezing it into an empty slice of a list.

odd = [1, 9]

odd.insert(1,3)

# Output: [1, 3, 9]

print(odd)

odd[2:2] = [5, 7]

# Output: [1, 3, 5, 7, 9]

print(odd)

## How to delete or remove elements from a list?

We can delete one or more items from a list using the keyword del. It can even delete the list entirely.

my\_list = ['p','r','o','b','l','e','m']

# delete one item

del my\_list[2]

# Output: ['p', 'r', 'b', 'l', 'e', 'm']

print(my\_list)

# delete multiple items

del my\_list[1:5]

# Output: ['p', 'm']

print(my\_list)

# delete entire list

del my\_list

# Error: List not defined

print(my\_list)

We can use remove() method to remove the given item or pop() method to remove an item at the given index.

The pop() method removes and returns the last item if index is not provided. This helps us implement lists as stacks (first in, last out data structure).

We can also use the clear() method to empty a list.

my\_list = ['p','r','o','b','l','e','m']

my\_list.remove('p')

# Output: ['r', 'o', 'b', 'l', 'e', 'm']

print(my\_list)

# Output: 'o'

print(my\_list.pop(1))

# Output: ['r', 'b', 'l', 'e', 'm']

print(my\_list)

# Output: 'm'

print(my\_list.pop())

# Output: ['r', 'b', 'l', 'e']

print(my\_list)

my\_list.clear()

# Output: []

print(my\_list)

Finally, we can also delete items in a list by assigning an empty list to a slice of elements.

>>> my\_list = ['p','r','o','b','l','e','m']

>>> my\_list[2:3] = []

>>> my\_list

['p', 'r', 'b', 'l', 'e', 'm']

>>> my\_list[2:5] = []

>>> my\_list

['p', 'r', 'm']

## Python List Methods

Methods that are available with list object in Python programming are tabulated below.

They are accessed as list.method(). Some of the methods have already been used above.

|  |
| --- |
| Python List Methods |
| **append()** - Add an element to the end of the list |
| **extend()** - Add all elements of a list to the another list |
| **insert()** - Insert an item at the defined index |
| **remove()** - Removes an item from the list |
| **pop()** - Removes and returns an element at the given index |
| **clear()** - Removes all items from the list |
| **index()** - Returns the index of the first matched item |
| **count()** - Returns the count of number of items passed as an argument |
| **sort()** - Sort items in a list in ascending order |
| **reverse()** - Reverse the order of items in the list |
| **copy()** - Returns a shallow copy of the list |

Some examples of Python list methods:

my\_list = [3, 8, 1, 6, 0, 8, 4]

# Output: 1

print(my\_list.index(8))

# Output: 2

print(my\_list.count(8))

my\_list.sort()

# Output: [0, 1, 3, 4, 6, 8, 8]

print(my\_list)

my\_list.reverse()

# Output: [8, 8, 6, 4, 3, 1, 0]

print(my\_list)

## List Comprehension: Elegant way to create new List

List comprehension is an elegant and concise way to create new list from an existing list in Python.

List comprehension consists of an expression followed by for statement inside square brackets.

Here is an example to make a list with each item being increasing power of 2.

pow2 = [2 \*\* x for x in range(10)]

# Output: [1, 2, 4, 8, 16, 32, 64, 128, 256, 512]

print(pow2)

This code is equivalent to

pow2 = []

for x in range(10):

pow2.append(2 \*\* x)

A list comprehension can optionally contain more for or if statements. An optional ifstatement can filter out items for the new list. Here are some examples.

>>> pow2 = [2 \*\* x for x in range(10) if x > 5]

>>> pow2

[64, 128, 256, 512]

>>> odd = [x for x in range(20) if x % 2 == 1]

>>> odd

[1, 3, 5, 7, 9, 11, 13, 15, 17, 19]

>>> [x+y for x in ['Python ','C '] for y in ['Language','Programming']]

['Python Language', 'Python Programming', 'C Language', 'C Programming']

## Other List Operations in Python

### List Membership Test

We can test if an item exists in a list or not, using the keyword in.

my\_list = ['p','r','o','b','l','e','m']

# Output: True

print('p' in my\_list)

# Output: False

print('a' in my\_list)

# Output: True

print('c' not in my\_list)

### Iterating Through a List

Using a for loop we can iterate though each item in a list.

for fruit in ['apple','banana','mango']:

print("I like",fruit)

### Built-in Functions with List

Built-in functions like all(), any(), enumerate(), len(), max(), min(), list(), sorted() etc. are commonly used with list to perform different tasks.

|  |  |
| --- | --- |
| Built-in Functions with List | |
| Function | Description |
| [all()](https://www.programiz.com/python-programming/methods/built-in/all) | Return True if all elements of the list are true (or if the list is empty). |
| [any()](https://www.programiz.com/python-programming/methods/built-in/any) | Return True if any element of the list is true. If the list is empty, return False. |
| [enumerate()](https://www.programiz.com/python-programming/methods/built-in/enumerate) | Return an enumerate object. It contains the index and value of all the items of list as a tuple. |
| [len()](https://www.programiz.com/python-programming/methods/built-in/len) | Return the length (the number of items) in the list. |
| [list()](https://www.programiz.com/python-programming/methods/built-in/list) | Convert an iterable (tuple, string, set, dictionary) to a list. |
| [max()](https://www.programiz.com/python-programming/methods/built-in/max) | Return the largest item in the list. |
| [min()](https://www.programiz.com/python-programming/methods/built-in/min) | Return the smallest item in the list |
| [sorted()](https://www.programiz.com/python-programming/methods/built-in/sorted) | Return a new sorted list (does not sort the list itself). |
| [sum()](https://www.programiz.com/python-programming/methods/built-in/sum) | Return the sum of all elements in the list. |