List

Lists are used to store multiple items in a single variable.

Lists are one of 4 built-in data types in Python used to store collections of data, the other 3 are [Tuple](https://www.w3schools.com/python/python_tuples.asp), [Set](https://www.w3schools.com/python/python_sets.asp), and [Dictionary](https://www.w3schools.com/python/python_dictionaries.asp), all with different qualities and usage.

Lists are created using square brackets:

thislist = ["apple", "banana", "cherry"]  
print(thislist)

output: ['apple', 'banana', 'cherry']

**List Items**

List items are ordered, changeable, and allow duplicate values.

List items are indexed, the first item has index [0], the second item has index [1] etc.

**Ordered**

When we say that lists are ordered, it means that the items have a defined order, and that order will not change.

If you add new items to a list, the new items will be placed at the end of the list.

**Changeable**

The list is changeable, meaning that we can change, add, and remove items in a list after it has been created.

**Allow Duplicates**

Since lists are indexed, lists can have items with the same value:

## List Items - Data Types

List items can be of any data type:

### **Example**

String, int and boolean data types:

list1 = ["apple", "banana", "cherry"]  
list2 = [1, 5, 7, 9, 3]  
list3 = [True, False, False]

**A list can contain different data types:**

A list with strings, integers and boolean values:

list1 = ["abc", 34, True, 40, "male"]

What is the data type of a list?

mylist = ["apple", "banana", "cherry"]  
print(type(mylist))

<class 'list'>

## The list() Constructor

It is also possible to use the list() constructor when creating a new list.

### **Example**

Using the list() constructor to make a List:

thislist = list(("apple", "banana", "cherry")) # note the double round-brackets  
print(thislist)

['apple', 'banana', 'cherry']

Python has a set of built-in methods that you can use on lists/arrays.

|  |  |
| --- | --- |
| **Method** | **Description** |
| [append()](https://www.w3schools.com/python/ref_list_append.asp) | Adds an element at the end of the list.  list.append(elmnt), *elmnt* of any type (string, number, object etc.)  fruits = ['apple', 'banana', 'cherry'],fruits.append("orange")  ['apple', 'banana', 'cherry', 'orange']  a = ["apple", "banana", "cherry"]  b = ["Ford", "BMW", "Volvo"]  a.append(b)  ['apple', 'banana', 'cherry', ["Ford", "BMW", "Volvo"]] |
| [clear()](https://www.w3schools.com/python/ref_list_clear.asp) | Removes all the elements from the list  list.clear(), No parameters to be passed  fruits = ["apple", "banana", "cherry"] fruits.clear()  print(fruits) [] |
| [copy()](https://www.w3schools.com/python/ref_list_copy.asp) | Returns a copy of the list  list.copy().No parameters to be passed  fruits = ['apple', 'banana', 'cherry'], x = fruits.copy()  print(x) ['apple', 'banana', 'cherry'] |
| [count()](https://www.w3schools.com/python/ref_list_count.asp) | Returns the number of elements with the specified value  list.count(value), *value* may be any type (string, number, list, tuple, etc.).  fruits = ['apple', 'banana', 'cherry'] x = fruits.count("cherry")  points = [1, 4, 2, 9, 7, 8, 9, 3, 1] x = points.count(9) |
| [extend()](https://www.w3schools.com/python/ref_list_extend.asp) | Add the elements of a list (or any iterable), to the end of the current list  An iterable is any Python object capable of returning its members one  at a time, permitting it to be iterated over in a for-loop.  Familiar examples of iterables include lists, tuples, and strings –  any such sequence can be iterated over in a for-loop.  fruits = ['apple', 'banana', 'cherry']  points = (1, 4, 5, 9)  fruits.extend(points)  ['apple', 'banana', 'cherry', 1, 4, 5, 9] |
| [index()](https://www.w3schools.com/python/ref_list_index.asp) | returns the position at the first occurrence of the specified value.  list.index(elmnt),Any type (string, number, list, etc.). The element to search for  fruits = ['apple', 'banana', 'cherry'] x = fruits.index("cherry")  print(x) 2  fruits = [4, 55, 64, 32, 16, 32] x = fruits.index(32)  print(x) 3 |
| [insert()](https://www.w3schools.com/python/ref_list_insert.asp) | Adds an element at the specified position  list.insert(pos, elmnt) *pos*: position to insert the value,  *elmnt*:  element of any type (string, number, object etc.)  fruits = ['apple', 'banana', 'cherry'] fruits.insert(1, "orange")  print(fruits) ['apple', 'orange', 'banana', 'cherry'] |
| [pop()](https://www.w3schools.com/python/ref_list_pop.asp) | Removes the element at the specified position  list.pop(pos) specifying the position of the element you want to remove  fruits = ['apple', 'banana', 'cherry'] fruits.pop(1)  print(fruits) ['apple', 'cherry'] |
| [remove()](https://www.w3schools.com/python/ref_list_remove.asp) | removes the first occurrence of the element with the specified value.  list.remove(elmnt) elmnt Any type (string, number, list etc.)  fruits = ['apple', 'banana', 'cherry'] fruits.remove("banana")  print(fruits) ['apple', 'cherry'] |
| [reverse()](https://www.w3schools.com/python/ref_list_reverse.asp) | Reverses the order of the list  list.reverse() No parameters  fruits = ['apple', 'banana', 'cherry'] fruits.reverse()  print(fruits) ['cherry', 'banana', 'apple'] |
| [sort()](https://www.w3schools.com/python/ref_list_sort.asp) | Sorts the list  list.sort(reverse=True|False, key=myFunc)  reverse=True will sort the list descending. Default is reverse=False  cars = ['Ford', 'BMW', 'Volvo'] cars.sort() print(cars)  ['BMW', 'Ford', 'Volvo'] |