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

Ex:

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.

Note: There are some [list methods](https://www.w3schools.com/python/python_lists_methods.asp) that will change the order, but in general: the order of the items will not change.

**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.

Ex:

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

output:

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

## List Length

To determine how many items a list has, use the len() function

Ex:

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

output:

3

## List Items - Data Types

List items can be of any data type

String ,int ,bool  
list1 = ["apple", "banana", "cherry"]  
list2 = [1, 5, 7, 9, 3]  
list3 = [True, False, False]  
  
print(list1)  
print(list2)  
print(list3)

output:

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

[1, 5, 7, 9, 3]

[True, False, False]

Ex1 :A list with strings, integers and boolean values  
list1 = ["abc", 34, True, 40, "male"]  
  
print(list1)

Output:

['abc', 34, True, 40, 'male']

# **Python - Access List Items**

## Access Items

List items are indexed and you can access them by referring to the index number

Ex :

thislist = ["apple", "banana", "cherry"]  
print(thislist[1])

output:

banana

### **Negative Indexing**

Negative indexing means start from the end

-1 refers to the last item, -2 refers to the second last item etc.

Ex :

thislist = ["apple", "banana", "cherry"]  
print(thislist[-1])

output:

cherry

### **Range of Indexes**

You can specify a range of indexes by specifying where to start and where to end the range.

When specifying a range, the return value will be a new list with the specified items.

Ex :

thislist = ["apple", "banana", "cherry", "orange", "kiwi", "melon", "mango"]  
print(thislist[2:5])  
  
#This will return the items from position 2 to 5.  
  
#Remember that the first item is position 0,  
#and note that the item in position 5 is NOT included

output:

['cherry', 'orange', 'kiwi']

Ex1:

thislist = ["apple", "banana", "cherry", "orange", "kiwi", "melon", "mango"]  
print(thislist[:4])

output:

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

Ex2:

thislist = ["apple", "banana", "cherry", "orange", "kiwi", "melon", "mango"]  
print(thislist[2:])

output:

['cherry', 'orange', 'kiwi', 'melon', 'mango']

### **Range of Negative Indexes**

Specify negative indexes if you want to start the search from the end of the list

Ex:

thislist = ["apple", "banana", "cherry", "orange", "kiwi", "melon", "mango"]  
print(thislist[-4:-1])

output:

['orange', 'kiwi', 'melon']

## Check if Item Exists

To determine if a specified item is present in a list use the in keyword:

Ex:

thislist = ["apple", "banana", "cherry"]  
if "apple" in thislist:  
 print("Yes, 'apple' is in the fruits list")

output:

Yes, 'apple' is in the fruits list

# **Python - Change List Items**

## Change Item Value

To change the value of a specific item, refer to the index number.

Ex:

thislist = ["apple", "banana", "cherry"]  
thislist[1] = "blackcurrant"  
  
print(thislist)

output:

['apple', 'blackcurrant', 'cherry']

## Change a Range of Item Values

To change the value of items within a specific range, define a list with the new values, and refer to the range of index numbers where you want to insert the new values:

Ex :

thislist = ["apple", "banana", "cherry", "orange", "kiwi", "mango"]  
  
thislist[1:3] = ["blackcurrant", "watermelon"]  
  
print(thislist)

output:

['apple', 'blackcurrant', 'watermelon', 'orange', 'kiwi', 'mango']

Ex: Change the second value by replacing it with *two* new values.

#change the second values.  
thislist = ["apple", "banana", "cherry"]  
  
thislist[1:2] = ["blackcurrant", "watermelon"]  
  
print(thislist)

Output:

['apple', 'blackcurrant', 'watermelon', 'cherry']

Ex: Change the second and third value by replacing it with one value  
thislist = ["apple", "banana", "cherry"]  
  
thislist[1:3] = ["watermelon"]  
  
print(thislist)

Output:

['apple', 'watermelon']

## Insert Items

To insert a new list item, without replacing any of the existing values, we can use the insert() method.

The insert() method inserts an item at the specified index:

Ex:

thislist = ["apple", "banana", "cherry"]  
  
thislist.insert(2, "watermelon")  
  
print(thislist)

output:

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

# Python - Add List Items

## Append Items

To add an item to the end of the list, use the append() method:

Ex:

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

output:

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

## Extend List:

To append elements from another list to the current list, use the extend() method.

Ex:

thislist = ["apple", "banana", "cherry"]  
tropical = ["mango", "pineapple", "papaya"]  
  
thislist.extend(tropical)  
  
print(thislist)

output:

['apple', 'banana', 'cherry', 'mango', 'pineapple', 'papaya']

## Add Any Iterable

The extend() method does not have to append lists, you can add any iterable object (tuples, sets, dictionaries etc.).

Ex:

thislist = ["apple", "banana", "cherry"]  
thistuple = ("kiwi", "orange")  
  
thislist.extend(thistuple)  
  
print(thislist)

output:

['apple', 'banana', 'cherry', 'kiwi', 'orange']

# **Python - Remove List Items**

## Remove Specified Item

The remove() method removes the specified item.

Ex:

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

output:

['apple', 'cherry']

## Remove Specified Index

The pop() method removes the specified index.

Ex:

thislist = ["apple", "banana", "cherry"]  
thislist.pop(1)  
print(thislist)

output:

['apple', 'cherry']

Ex: If you do not specify the index, the pop() method removes the last item.

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

output:

['apple', 'banana']

Del:

The del keyword also removes the specified index

del the single element.  
thislist = ["apple", "banana", "cherry"]  
del thislist[0]  
print(thislist)

output:

['banana', 'cherry']

The del keyword can also delete the list completely

Ex:

thislist = ["apple", "banana", "cherry"]  
del thislist  
print(thislist) #this will cause an error because you have succsesfully deleted "thislist".

Output:

NameError: name 'thislist' is not defined.

## Clear the List

The clear() method empties the list.

The list still remains, but it has no content.

Ex: Clear the list content

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

output:

[]

# **Python - Loop Lists**

## Loop Through a List

You can loop through the list items by using a for loop

Ex: Print all items in the list, one by one

thislist = ["apple", "banana", "cherry"]  
for x in thislist:  
 print(x)

output:

apple

banana

cherry

## Loop Through the Index Numbers:

You can also loop through the list items by referring to their index number.

Use the range() and len() functions to create a suitable iterable.

Ex:

#Print all items by referring to their index number  
thislist = ["apple", "banana", "cherry"]  
for i in range(len(thislist)):  
 print(thislist[i])

Output:

apple

banana

cherry

## Using a While Loop

You can loop through the list items by using a while loop.

Use the Len() function to determine the length of the list, then start at 0 and loop your way through the list items by referring to their indexes.

Remember to increase the index by 1 after each iteration

Ex:

thislist = ["apple", "banana", "cherry"]  
i = 0  
while i < len(thislist):  
 print(thislist[i])  
 i = i + 1

output:

apple

banana

cherry

## Looping Using List Comprehension:

List Comprehension offers the shortest syntax for looping through lists:

Ex:

#A short hand for loop that will print all items in a list  
thislist = ["apple", "banana", "cherry"]  
[print(x) for x in thislist]

Output:

apple

banana

cherry

# Python - List Comprehension:

## List Comprehension

List comprehension offers a shorter syntax when you want to create a new list based on the values of an existing list.

Ex:

fruits = ["apple", "banana", "cherry", "kiwi", "mango"]  
newlist = []  
  
for x in fruits:  
 if "a" in x:  
 newlist.append(x)  
  
print(newlist)

output:

['apple', 'banana', 'mango']

## Newlist-The Syntax:

## newlist = [expression for item in iterable if condition == True]

The return value is a new list, leaving the old list unchanged.

### **Condition:**

The condition is like a filter that only accepts the items that valuate to True.

Ex:

fruits = ["apple", "banana", "cherry", "kiwi", "mango"]  
  
newlist = [x for x in fruits if x != "apple"]  
  
print(newlist)

output:

['banana', 'cherry', 'kiwi', 'mango']

**New list-range**:

Ex: newlist = [x for x in range(10)]  
  
print(newlist)

Output:

[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]

Ex 1: Accept only numbers lower than 5

newlist = [x for x in range(10) if x < 5]  
  
print(newlist)

output:

[0, 1, 2, 3, 4]

**New list to upper**:

Ex:

fruits = ["apple", "banana", "cherry", "kiwi", "mango"]  
  
newlist = [x.upper() for x in fruits]  
  
print(newlist)

output:

['APPLE', 'BANANA', 'CHERRY', 'KIWI', 'MANGO']

# **Python - Sort Lists:**

## Sort List Alphanumerically

List objects have a sort() method that will sort the list alphanumerically, ascending, by default.

Ex: sort alphabetically.

thislist = ["orange", "mango", "kiwi", "pineapple", "banana"]  
  
thislist.sort()  
  
print(thislist)

output:

['banana', 'kiwi', 'mango', 'orange', 'pineapple']

Ex1:

#sort numerically  
thislist = [100, 50, 65, 82, 23]  
  
thislist.sort()  
  
print(thislist)

Output:

[23, 50, 65, 82, 100]

## Sort Descending:

To sort descending, use the keyword argument reverse = True.

Ex:

#sort list desending order  
thislist = ["orange", "mango", "kiwi", "pineapple", "banana"]  
  
thislist.sort(reverse = True)  
  
print(thislist)

Output:

['pineapple', 'orange', 'mango', 'kiwi', 'banana']

Ex 1:

#sort desnding oreder numeric  
thislist = [100, 50, 65, 82, 23]  
  
thislist.sort(reverse = True)  
  
print(thislist)

Output:

[100, 82, 65, 50, 23]

## Reverse Order

What if you want to reverse the order of a list, regardless of the alphabet?

The reverse() method reverses the current sorting order of the elements

Ex:

#reverse the string of list  
thislist = ["banana", "Orange", "Kiwi", "cherry"]  
  
thislist.reverse()  
  
print(thislist)

Output:

['cherry', 'Kiwi', 'Orange', 'banana']

# **Python - Copy Lists**

## copy a List

You cannot copy a list simply by typing list2 = list1, because: list2 will only be a reference to list1, and changes made in list1 will automatically also be made in list2.

There are ways to make a copy, one way is to use the built-in List method copy()

Ex:

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

output:

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

Ex1:

Make a copy of a list with the list() method

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

output:

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

# Python - Join Lists:

## Join Two Lists

There are several ways to join, or concatenate, two or more lists in Python.

One of the easiest ways are by using the + operator.

Ex:

list1 = ["a", "b", "c"]  
list2 = [1, 2, 3]  
  
list3 = list1 + list2  
print(list3)

output:

['a', 'b', 'c', 1, 2, 3]

Ex 1:

#join use append   
list1 = ["a", "b" , "c"]  
list2 = [1, 2, 3]  
  
for x in list2:  
 list1.append(x)  
  
print(list1)

Output:

['a', 'b', 'c', 1, 2, 3]

Join-Extend():

Ex:

#Use the extend() method to add list2 at the end of list1  
list1 = ["a", "b" , "c"]  
list2 = [1, 2, 3]  
  
list1.extend(list2)  
print(list1)

Output:

['a', 'b', 'c', 1, 2, 3]

# **Python List – Method**

**1.append()** :

## Definition and Usage

The append() method appends an element to the end of the list.

## Syntax

list.append(elmnt)

## Parameter Values

|  |  |
| --- | --- |
| Parameter | Description |
| elmnt | Required. An element of any type (string, number, object etc).) |

Ex:

fruits = ["apple", "banana", "cherry"]  
  
fruits.append("orange")  
  
print("fruits:",fruits)  
  
a = ["apple", "banana", "cherry"]  
  
b = ["Ford", "BMW", "Volvo"]  
  
a.append(b)  
  
print("a:",a)

output:

fruits: ['apple', 'banana', 'cherry', 'orange']

a: ['apple', 'banana', 'cherry', ['Ford', 'BMW', 'Volvo']]

## 2.clear():

## Definition and Usage

The clear() method removes all the elements from a list.

## Syntax

list.clear()

## Parameter Values

No parameters

Ex:

fruits = ["apple", "banana", "cherry"]  
  
fruits.clear()  
  
print(fruits)

output:

[]

## 3.copy():

## Definition and Usage

The copy() method returns a copy of the specified list.

## Syntax

list.copy()

## Parameter Values

No parameters

Ex:

fruits = ["apple", "banana", "cherry"]  
  
x = fruits.copy()  
  
print(x)

output:

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

## 4.Count() :

## Definition and Usage

The count() method returns the number of elements with the specified value.

## Syntax

list.count(value)

## Parameter Values

|  |  |
| --- | --- |
| Parameter | Description |
| value | Required. Any type (string, number, list, tuple, etc.).  The value to search for. |

Ex :

fruits = ["apple", "banana", "cherry"]  
  
x = fruits.count("cherry")  
  
print("x:",x)  
  
fruits = [1, 4, 2, 9, 7, 8, 9, 3, 1]  
  
y= fruits.count(9)  
  
print("y:",y)

output:

x: 1

y: 2

## 5.extend():

## Definition and Usage

The extend() method adds the specified list elements (or any iterable) to the end of the current list.

## Syntax

list.extend(iterable)

## Parameter Values

|  |  |
| --- | --- |
| Parameter | Description |
| iterable | Required. Any iterable (list, set, tuple, etc.) |

Ex:

fruits = ['apple', 'banana', 'cherry']  
  
cars = ['Ford', 'BMW', 'Volvo']  
  
fruits.extend(cars)  
  
print("fruits:",fruits)  
#ex2  
fruits = ['apple', 'banana', 'cherry']  
  
points = (1, 4, 5, 9)  
  
fruits.extend(points)  
  
print("string:",fruits)

output:

fruits: ['apple', 'banana', 'cherry', 'Ford', 'BMW', 'Volvo']

string: ['apple', 'banana', 'cherry', 1, 4, 5, 9]

## 6.index():

## Definition and Usage

The index() method returns the position at the first occurrence of the specified value.

## Syntax

## list.index(elmnt)

## Parameter Values

|  |  |
| --- | --- |
| Parameter | Description |
| elmnt | Required. Any type (string, number, list, etc.). The element to search for |

Ex :

fruits = ['apple', 'banana', 'cherry']  
  
x = fruits.index("cherry")  
  
print("first:",x)  
  
  
fruits = [4, 55, 64, 32, 16, 32]  
  
x = fruits.index(32)  
  
print("second:",x)

output:

first: 2

second: 3

## 7.insert():

## Definition and Usage

The insert() method inserts the specified value at the specified position.

## Syntax

list.insert(pos, elmnt)

## Parameter Values

|  |  |
| --- | --- |
| Parameter | Description |
| pos | Required. A number specifying in which position to insert  the value |
| elmnt | Required. An element of any type (string, number, object etc) |

Ex :

fruits = ['apple', 'banana', 'cherry']  
  
fruits.insert(1, "orange")  
  
print(fruits)

output:

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

## 8.Pop():

## Definition and Usage

The pop() method removes the element at the specified position.

## Syntax

list.pop(pos)

## Parameter Values

|  |  |
| --- | --- |
| Parameter | Description |
| pos | Optional. A number specifying the position of the element  you want to remove, default value is -1,  which returns the last item |

Ex:

#Remove the second element of the fruit list  
fruits = ['apple', 'banana', 'cherry']  
  
fruits.pop(1)  
  
print(fruits)  
  
#Return the removed element  
fruits = ['apple', 'banana', 'cherry']  
  
x = fruits.pop(1)  
  
print(x)

Output:

['apple', 'cherry']

Banana

## 9.Remove():

## Definition and Usage

The remove() method removes the first occurrence of the element with the specified value.

## Syntax

## list.remove(elmnt)

## Parameter Values

|  |  |
| --- | --- |
| Parameter | Description |
| *elmnt* | Required. Any type (string, number, list etc.)  The element you want to remove element |

Ex :

fruits = ['apple', 'banana', 'cherry']  
  
fruits.remove("banana")  
  
print(fruits)

output:

['apple', 'cherry']

## 9.reverse():

## Definition and Usage

The reverse() method reverses the sorting order of the elements.

## Syntax

list.reverse()

## Parameter Values

No parameters

Ex: fruits = ['apple', 'banana', 'cherry']  
  
fruits.reverse()  
  
print(fruits)

Output:

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