Python Lists

List

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

- Ordered
- Changeable
- Allow Duplicates
- List items can be of any data type

Lists are created using square brackets.

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

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

The list() Constructor

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

```
# Using the list() constructor to make a List:

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

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

List Length

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

```
# Print the number of items in the list:

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

```
3
```

Access Items

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

```
# Print the second item of the list:
```

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

```
banana
```

Negative Indexing

Negative indexing means start from the end.

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

```
# Print the last item of the list:

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

```
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.

```
# Return the third, fourth, and fifth item:

thislist = ["apple", "banana", "cherry", "orange", "kiwi", "melon", "mango"]
print(thislist[2:5])
```

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

Note: The search will start at index 2 (included) and end at index 5 (not included).

By leaving out the start value, the range will start at the first item

```
# This example returns the items from the beginning to, but NOT including, "kiwi":

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

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

By leaving out the end value, the range will go on to the end of the list

```
# This example returns the items from "cherry" to the end:

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

```
['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.

```
# This example returns the items from "orange" (-4) to, but NOT including "mango" (-1):

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

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

Check if Item Exists

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

```
# Check if "apple" is present in the list:

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

```
Yes, 'apple' is in the fruits list
```

Change List Items

Change Item Value

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

```
  # Change the second item:

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

```
['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.

Change the values "banana" and "cherry" with the values "blackcurrant" and "watermelon"

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

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

If you insert more items than you replace, the new items will be inserted where you specified, and the remaining items will move accordingly.

```
# Change the second value by replacing it with two new values:

thislist = ["apple", "banana", "cherry"]
thislist[1:2] = ["blackcurrant", "watermelon"]
print(thislist)
```

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

Note: The length of the list will change when the number of items inserted does not match the number of items replaced.

If you insert less items than you replace, the new items will be inserted where you specified, and the remaining items will move accordingly.

```
# Change the second and third value by replacing it with one value:

thislist = ["apple", "banana", "cherry"]
thislist[1:3] = ["watermelon"]
print(thislist)
```

```
['apple', 'watermelon']
```

Add List Items

Append Items

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

```
  # Using the append() method to append an item:

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

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

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.

```
# Insert "watermelon" as the third item:

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

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

Extend List

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

The elements will be added to the end of the list.

```
# Add the elements of tropical to thislist:

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

```
['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.).

```
# Add elements of a tuple to a list:

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

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

Remove List Items

Remove Specified Item

The remove() method removes the specified item.

```
# Remove "banana":

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

```
['apple', 'cherry']
```

Remove Specified Index

The pop() method removes the specified index.

```
# Remove the second item:

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

```
['apple', 'cherry']
```

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

```
# Remove the last item:

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

```
['apple', 'banana']
```

The del keyword also removes the specified index.

```
# Remove the first item:

thislist = ["apple", "banana", "cherry"]
del thislist[0]
print(thislist)
```

```
['banana', 'cherry']
```

The del keyword can also delete the list completely.

```
  # Delete the entire list:

  thislist = ["apple", "banana", "cherry"]
  del thislist
```

Clear the List

The clear() method empties the list.

The list still remains, but it has no content.

```
# Clear the list content:

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

```
[]
```

Loop Lists

Loop Through a List

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

```
# Print all items in the list, one by one:

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

```
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.

```
# Print all items by referring to their index number:

thislist = ["apple", "banana", "cherry"]
for i in range(len(thislist)):
  print(thislist[i])
```

```
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 refering to their indexes.

Remember to increase the index by 1 after each iteration.

```
# Print all items, using a while loop to go through all the index numbers

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

```
apple
banana
cherry
```

Looping Using List Comprehension

List Comprehension offers the shortest syntax for looping through lists.

```
# A short hand for loop that will print all items in a list:

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

```
apple
banana
cherry
[None, None, None]
```

Sort Lists

Sort List Alphanumerically

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

Sort the list alphabetically

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

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

Sort the list numerically

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

```
[23, 50, 65, 82, 100]
```

Sort Descending

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

Sort the list descending

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

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

Sort the list descending

```
thislist = [100, 50, 65, 82, 23]
thislist.sort(reverse = True)
print(thislist)
```

```
[100, 82, 65, 50, 23]
```

Case Insensitive Sort

By default the sort() method is case sensitive, resulting in all capital letters being sorted before lower case letters.

Case sensitive sorting can give an unexpected result.

```
thislist = ["banana", "Orange", "Kiwi", "cherry"]
thislist.sort()
print(thislist)
```

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

Perform a case-insensitive sort of the list.

```
thislist = ["banana", "Orange", "Kiwi", "cherry"]
thislist.sort(key = str.lower)
print(thislist)
```

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

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.

Reverse the order of the list items

```
thislist = ["banana", "Orange", "Kiwi", "cherry"]
thislist.reverse()
print(thislist)
```

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

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().

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

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

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

Another way to make a copy is to use the built-in method list()

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

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

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

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.

Join two list

```
list1 = ["a", "b", "c"]
list2 = [1, 2, 3]

list3 = list1 + list2
print(list3)
```

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

Another way to join two lists is by appending all the items from list2 into list1, one by one.

Append list2 into list1

```
list1 = ["a", "b" , "c"]
list2 = [1, 2, 3]

for x in list2:
  list1.append(x)

print(list1)
```

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

Or you can use the extend() method, which purpose is to add elements from one list to another list.

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)
```

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

List Metods

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

Method	Description
append()	Adds an element at the end of the list
clear()	Removes all the elements from the list
copy()	Returns a copy of the list
count()	Returns the number of elements with the specified value
extend()	Add the elements of a list (or any iterable), to the end of the current list
index()	Returns the index of the first element with the specified value
insert()	Adds an element at the specified position
pop()	Removes the element at the specified position
remove()	Removes the item with the specified value
reverse()	Reverses the order of the list
sort()	Sorts the list