# List

A list is a collection which is ordered and changeable. In Python lists are written with square brackets.

### Example

Create a List:

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

## Access Items

You access the list items by referring to the index number:

### Example

Print the second item of the list:

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

### Negative Indexing

Negative indexing means beginning from the end, `-1` refers to the last item, `-2` refers to the second last item etc.

### Example

Print the last item of the list:

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

## 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.

### Example

Return the third, fourth, and fifth item:

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

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

Remember that the first item has index 0.

## Range of Negative Indexes

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

### Example

This example returns the items from index -4 (included) to index -1 (excluded)

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

# Change Item Value

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

### Example

Change the second item:

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

## Loop Through a List

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

### Example

Print all items in the list, one by one:

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

## Check if Item Exists

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

### Example

Check if "apple" is present in the list:

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

## List Length

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

### Example

Print the number of items in the list:

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

## Add Items

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

### Example

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

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

To add an item at the specified index, use the `insert()` method:

### Example

Insert an item as the second position:

```
mylist = ["apple", "banana", "cherry"]
mylist.insert(1, "orange")
print(mylist)
```

## Remove Item

There are several methods to remove items from a list:

### Example

The `remove()` method removes the specified item:

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

### Example

The `pop()` method removes the specified index, (or the last item if index is not specified):

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

### Example

The `del` keyword removes the specified index:

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

### Example

The `del` keyword can also delete the list completely:

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

### Example

The `clear()` method empties the list:

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

## 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()`.

### Example

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

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

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

## Example

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

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

## 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.

## Example

Join two list:

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

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

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

## Example

Append list2 into list1:

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

for x in list2:
  list1.append(x)

print(list1)
```

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

### Example

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)
```

## Tuple

A tuple is a collection which is ordered and **unchangeable**. In Python tuples are written with round brackets.

### Example

Create a Tuple:

```
mytuple = ("apple", "banana", "cherry")
print(mytuple)
```

## Access Tuple Items

You can access tuple items by referring to the index number, inside square brackets:

### Example

Print the second item in the tuple:

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

## Negative Indexing

Negative indexing means beginning from the end, `-1` refers to the last item, `-2` refers to the second last item etc.

### Example

Print the last item of the tuple:

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

## 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 tuple with the specified items.

### Example

Return the third, fourth, and fifth item:

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

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

Remember that the first item has index 0.

## Range of Negative Indexes

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

### Example

This example returns the items from index -4 (included) to index -1 (excluded)

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

## Change Tuple Values

Once a tuple is created, you cannot change its values. Tuples are **unchangeable**, or **immutable** as it also is called.

But there is a workaround. You can convert the tuple into a list, change the list, and convert the list back into a tuple.

### Example

Convert the tuple into a list to be able to change it:

```
x = ("apple", "banana", "cherry")
y = list(x)
y[1] = "kiwi"
x = tuple(y)

print(x)
```

## Loop Through a Tuple

You can loop through the tuple items by using a `for` loop.

### Example

Iterate through the items and print the values:

```
mytuple = ("apple", "banana", "cherry")
for x in mytuple:
  print(x)
```

## Check if Item Exists

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

### Example

Check if "apple" is present in the tuple:

```
mytuple = ("apple", "banana", "cherry")
if "apple" in mytuple:
  print("Yes, 'apple' is in the fruits tuple")
```

## Tuple Length

To determine how many items a tuple has, use the `len()` method:

### Example

Print the number of items in the tuple:

```
mytuple = ("apple", "banana", "cherry")
print(len(mytuple))
```

## Add Items

Once a tuple is created, you cannot add items to it. Tuples are **unchangeable**.

### Example

You cannot add items to a tuple:

```
mytuple = ("apple", "banana", "cherry")
mytuple[3] = "orange" # This will raise an error
print(mytuple)
```

## Remove Items

**Note:** You cannot remove items in a tuple.

Tuples are **unchangeable**, so you cannot remove items from it, but you can delete the tuple completely:

### Example

The `del` keyword can delete the tuple completely:

```
mytuple = ("apple", "banana", "cherry")
del mytuple
print(mytuple) #this will raise an error because the tuple no longer
exists
```

## Join Two Tuples

To join two or more tuples you can use the `+` operator:

### Example

Join two tuples:

```
tuple1 = ("a", "b", "c")
tuple2 = (1, 2, 3)

tuple3 = tuple1 + tuple2
print(tuple3)
```