# Python Lists

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

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:

Example[Get your own Python Server](https://www.w3schools.com/spaces/)

Create a List:

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

## 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:

Example

Lists allow duplicate values:

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

Access Items

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

Example[Get your own Python Server](https://www.w3schools.com/spaces/)

Print the second item of the list:

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

Negative Indexing

Negative indexing means start 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:

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

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

Example

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

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

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

Example

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

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

Change Item Value

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

Example[Get your own Python Server](https://www.w3schools.com/spaces/)

Change the second item:

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

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:

Example

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

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

Append Items

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

Example[Get your own Python Server](https://www.w3schools.com/spaces/)

Using the append() method to append an item:

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

Insert Items

To insert a list item at a specified index, use the insert() method.

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

Example

Insert an item as the second position:

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

or thislist[1]=”Orange” it will edit the same item not inset item

Remove Specified Item

The remove() method removes the specified item.

Example

Remove "banana":

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

Remove Specified Index

The pop() method removes the specified index.

Example

Remove the second item:

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

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

Example

Remove the last item:

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

The del keyword also removes the specified index:

Example

Remove the first item:

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

## Loop Through a List

Example[Get your own Python Server](https://www.w3schools.com/spaces/)

Print all items in the list, one by one:

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

Example

Print all items by referring to their index number:

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

## List Comprehension

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

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

## 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)

Example

Sort the list numerically:

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

## Copy a List

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

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

## Join Two Lists

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

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

|  |  |
| --- | --- |
| **Method** | **Description** |
| [append()](https://www.w3schools.com/python/ref_list_append.asp) | Adds an element at the end of the list.  fruits = ['apple', 'banana', 'cherry'] fruits.append("orange") |
| [clear()](https://www.w3schools.com/python/ref_list_clear.asp) | Removes all the elements from the list. fruits.clear() |
| [copy()](https://www.w3schools.com/python/ref_list_copy.asp) | Returns a copy of the list.  fruits = ['apple', 'banana', 'cherry', 'orange'] x = fruits.copy() |
| [count()](https://www.w3schools.com/python/ref_list_count.asp) | Returns the number of elements with the specified value.  fruits = ['apple', 'banana', 'cherry']  x = fruits.count("cherry") |
| [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.  fruits = ['apple', 'banana', 'cherry']  cars = ['Ford', 'BMW', 'Volvo']  fruits.extend(cars) |
| [index()](https://www.w3schools.com/python/ref_list_index.asp) | Returns the index of the first element with the specified value.  fruits = ['apple', 'banana', 'cherry']  x = fruits.index("cherry") |
| [insert()](https://www.w3schools.com/python/ref_list_insert.asp) | Adds an element at the specified position.  fruits = ['apple', 'banana', 'cherry']  fruits.insert(1, "orange") |
| [pop()](https://www.w3schools.com/python/ref_list_pop.asp) | Removes the element at the specified position.  fruits = ['apple', 'banana', 'cherry']  fruits.pop(1) |
| [remove()](https://www.w3schools.com/python/ref_list_remove.asp) | Removes the item with the specified value.  fruits = ['apple', 'banana', 'cherry']  fruits.remove("banana") |
| [reverse()](https://www.w3schools.com/python/ref_list_reverse.asp) | Reverses the order of the list  fruits = ['apple', 'banana', 'cherry']  fruits.reverse() |
| [sort()](https://www.w3schools.com/python/ref_list_sort.asp) | Sorts the list.  cars = ['Ford', 'BMW', 'Volvo']  cars.sort() #Sort the list alphabetically: |

Tuple

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

Tuple is one of 4 built-in data types in Python used to store collections of data, the other 3 are [List](https://www.w3schools.com/python/python_lists.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.

A tuple is a collection which is ordered and **unchangeable**.

Tuples are written with round brackets.

Example

Create a Tuple:

thistuple = ("apple", "banana", "cherry")  
print(thistuple)

Tuple Items

Tuple items are ordered, unchangeable, and allow duplicate values.

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

Ordered

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

Unchangeable

Tuples are unchangeable, meaning that we cannot change, add or remove items after the tuple has been created.

Allow Duplicates

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

Example

Tuples allow duplicate values:

thistuple = ("apple", "banana", "cherry", "apple", "cherry")  
print(thistuple)

Access Tuple Items

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

Example[Get your own Python Server](https://www.w3schools.com/spaces/)

Print the second item in the tuple:

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

**Note:** The first item has index 0.

Negative Indexing

Negative indexing means start 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:

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

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_tuple_negative_index)

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:

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

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:

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

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_tuple_loop)

Learn more about for loops in our [Python For Loops](https://www.w3schools.com/python/python_for_loops.asp) Chapter.

Loop Through the Index Numbers

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

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

Example

Print all items by referring to their index number:

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

Join Two Tuples

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

Example[Get your own Python Server](https://www.w3schools.com/spaces/)

Join two tuples:

tuple1 = ("a", "b" , "c")  
tuple2 = (1, 2, 3)  
  
tuple3 = tuple1 + tuple2  
print(tuple3)

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_tuple_concat)

Multiply Tuples

If you want to multiply the content of a tuple a given number of times, you can use the \* operator:

Example

Multiply the fruits tuple by 2:

fruits = ("apple", "banana", "cherry")  
mytuple = fruits \* 2  
  
print(mytuple) it will print a tupple two times

Tuple Methods

Python has two built-in methods that you can use on tuples.

|  |  |
| --- | --- |
| **Method** | **Description** |
| [count()](https://www.w3schools.com/python/ref_tuple_count.asp) | Returns the number of times a specified value occurs in a tuple |
| [index()](https://www.w3schools.com/python/ref_tuple_index.asp) | Searches the tuple for a specified value and returns the position of where it was found |

Example

Return the number of times the value 5 appears in the tuple:

thistuple = (1, 3, 7, 8, 7, 5, 4, 6, 8, 5)  
  
x = thistuple.count(5)  
  
print(x)

thistuple = (1, 3, 7, 8, 7, 5, 4, 6, 8, 5)  
  
x = thistuple.index(8)  
  
print(x)

# Python Dictionaries

Dictionaries are used to store data values in key:value pairs.

A dictionary is a collection which is ordered\*, changeable and do not allow duplicates.

Dictionaries are written with curly brackets, and have keys and values:

thisdict = {  
  "brand": "Ford",  
  "model": "Mustang",  
  "year": 1964  
}

print(thisdict)

## Dictionary Items

Dictionary items are ordered, changeable, and does not allow duplicates.

Dictionary items are presented in key:value pairs, and can be referred to by using the key name.

### Example

Print the "brand" value of the dictionary:

thisdict = {  
  "brand": "Ford",  
  "model": "Mustang",  
  "year": 1964  
}  
print(thisdict["brand"])

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

## Changeable

Dictionaries are changeable, meaning that we can change, add or remove items after the dictionary has been created.

## Duplicates Not Allowed

Dictionaries cannot have two items with the same key:

### Example

Duplicate values will overwrite existing values:

thisdict = {  
  "brand": "Ford",  
  "model": "Mustang",  
  "year": 1964,  
  "year": 2020  
}  
print(thisdict)

## Dictionary Length

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

### Example

Print the number of items in the dictionary:

print(len(thisdict))

## Dictionary Items - Data Types

The values in dictionary items can be of any data type:

### Example

String, int, boolean, and list data types:

thisdict = {  
  "brand": "Ford",  
  "electric": False,  
  "year": 1964,  
  "colors": ["red", "white", "blue"]  
}

### Type of Dictionary

### Example

Print the data type of a dictionary:

thisdict = {  
  "brand": "Ford",  
  "model": "Mustang",  
  "year": 1964  
}  
print(type(thisdict))

## The dict() Constructor

It is also possible to use the dict() constructor to make a dictionary.

### Example

Using the dict() method to make a dictionary:

thisdict = dict(name = "John", age = 36, country = "Norway")  
print(thisdict)

## Accessing Items

You can access the items of a dictionary by referring to its key name, inside square brackets:

### Example[Get your own Python Server](https://www.w3schools.com/spaces/)

Get the value of the "model" key:

thisdict = {  
  "brand": "Ford",  
  "model": "Mustang",  
  "year": 1964  
}  
x = thisdict["model"]

There is also a method called get() that will give you the same result:

### Example

Get the value of the "model" key:

x = thisdict.get("model")

## Get Keys

The keys() method will return a list of all the keys in the dictionary.

### Example

Get a list of the keys:

x = thisdict.keys()

### Adding New Keys into Dictionary

### Example

Add a new item to the original dictionary, and see that the keys list gets updated as well:

car = {  
"brand": "Ford",  
"model": "Mustang",  
"year": 1964  
}  
  
x = car.keys()  
  
print(x) #before the change  
  
car["color"] = "white"  
  
print(x) #after the change

## Get Values

The values() method will return a list of all the values of keys in the dictionary.

### Example

Get a list of the values:

x = thisdict.values()

car = {  
"brand": "Ford",  
"model": "Mustang",  
"year": 1964  
}  
  
x = car.values()  
  
print(x) #before the change  
  
car["year"] = 2020  
  
print(x) #after the change

## Get Items

The items() method will return each item in a dictionary, as tuples in a list.

### Example

Get a list of the key:value pairs

x = thisdict.items()

car = {  
"brand": "Ford",  
"model": "Mustang",  
"year": 1964  
}  
  
x = car.items()  
  
print(x) #before the change  
  
car["year"] = 2020  
  
print(x) #after the change

### Check Key Exists or not

### Example

Check if "model" is present in the dictionary:

thisdict = {  
  "brand": "Ford",  
  "model": "Mustang",  
  "year": 1964  
}  
if "model" in thisdict:  
  print("Yes, 'model' is one of the keys in the thisdict dictionary")

## Change Values

You can change the value of a specific item by referring to its key name:

Change the "year" to 2018:

thisdict = {  
  "brand": "Ford",  
  "model": "Mustang",  
  "year": 1964  
}

thisdict["year"] = 2018

## Update Dictionary

The update() method will update the dictionary with the items from the given argument.

### Example

Update the "year" of the car by using the update() method:

thisdict = {  
  "brand": "Ford",  
  "model": "Mustang",  
  "year": 1964  
}  
thisdict.update({"year": 2020})

## Adding Items

Adding an item to the dictionary is done by using a new index key and assigning a value to it:

### Example[Get your own Python Server](https://www.w3schools.com/spaces/)

thisdict = {  
  "brand": "Ford",  
  "model": "Mustang",  
  "year": 1964  
}  
thisdict["color"] = "red"  
print(thisdict)

## Removing Items

There are several methods to remove items from a dictionary:

### Example[Get your own Python Server](https://www.w3schools.com/spaces/)

The pop() method removes the item with the specified key name:

thisdict = {  
  "brand": "Ford",  
  "model": "Mustang",  
  "year": 1964  
}  
thisdict.pop("model")  
print(thisdict)

### Example

The popitem() method removes the last inserted item (in versions before 3.7, a random item is removed instead):

thisdict = {  
  "brand": "Ford",  
  "model": "Mustang",  
  "year": 1964  
}  
thisdict.popitem()  
print(thisdict)

### Example

The del keyword removes the item with the specified key name:

thisdict = {  
  "brand": "Ford",  
  "model": "Mustang",  
  "year": 1964  
}  
del thisdict["model"]  
print(thisdict)

## Loop Through a Dictionary

You can loop through a dictionary by using a for loop.

When looping through a dictionary, the return value are the keys of the dictionary, but there are methods to return the values as well.

### Example[Get your own Python Server](https://www.w3schools.com/spaces/)

Print all key names in the dictionary, one by one:

for x in thisdict:  
  print(x)

## Copy a Dictionary

You cannot copy a dictionary simply by typing dict2 = dict1, because: dict2 will only be a reference to dict1, and changes made in dict1 will automatically also be made in dict2.

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

thisdict = {  
  "brand": "Ford",  
  "model": "Mustang",  
  "year": 1964  
}  
mydict = thisdict.copy()  
print(mydict)

## Nested Dictionaries

A dictionary can contain dictionaries, this is called nested dictionaries.

### Example[Get your own Python Server](https://www.w3schools.com/spaces/)

Create a dictionary that contain three dictionaries:

myfamily = {  
  "child1" : {  
    "name" : "Emil",  
    "year" : 2004  
  },  
  "child2" : {  
    "name" : "Tobias",  
    "year" : 2007  
  },  
  "child3" : {  
    "name" : "Linus",  
    "year" : 2011  
  }  
}