

# Python Dictionaries

[< Previous](#)[Next >](#)

## Dictionary

A dictionary is a collection which is unordered, changeable and indexed. In Python dictionaries are written with curly brackets, and they have keys and values.

### Example

Create and print a dictionary:

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

[Try it Yourself »](#)

## Accessing Items

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

## Example

Get the value of the "model" key:

```
x = thisdict["model"]
```

Try it Yourself »

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")
```

Try it Yourself »

---

## Change Values

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

## Example

Change the "year" to 2018:

```
thisdict = {  
    "brand": "Ford",  
    "model": "Mustang",  
    "year": 1964  
}  
thisdict["year"] = 2018
```

Try it Yourself »

---

## 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

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

```
for x in thisdict:  
    print(x)
```

Try it Yourself »

### Example

Print all *values* in the dictionary, one by one:

```
for x in thisdict:  
    print(thisdict[x])
```

Try it Yourself »

### Example

You can also use the **values()** method to return values of a dictionary:

```
for x in thisdict.values():  
    print(x)
```

Try it Yourself »

## Example

Loop through both *keys* and *values*, by using the `items()` method:

```
for x, y in thisdict.items():  
    print(x, y)
```

Try it Yourself »

---

## Check if Key Exists

To determine if a specified key is present in a dictionary use the `in` keyword:

## 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")
```

Try it Yourself »

# Dictionary Length

To determine how many items (key-value pairs) a dictionary has, use the `len()` function.

## Example

Print the number of items in the dictionary:

```
print(len(thisdict))
```

[Try it Yourself »](#)

---

# Adding Items

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

## Example

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

[Try it Yourself »](#)

---

# Removing Items

There are several methods to remove items from a dictionary:

## Example

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

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

Try it Yourself »

## 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)
```

Try it Yourself »

## Example

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

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

Try it Yourself »

## Example

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

```
thisdict = {  
    "brand": "Ford",  
    "model": "Mustang",  
    "year": 1964  
}  
del thisdict  
print(thisdict) #this will cause an error because "thisdict" no longer exists.
```

Try it Yourself »

## Example

The `clear()` method empties the dictionary:

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

Try it Yourself »

## 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()`.

### Example

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

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

Try it Yourself »

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

### Example

Make a copy of a dictionary with the `dict()` function:

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



```
}  
mydict = dict(thisdict)  
print(mydict)
```

Try it Yourself »

---

## Nested Dictionaries

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

### Example

Create a dictionary that contain three dictionaries:

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

Try it Yourself »

Or, if you want to nest three dictionaries that already exists as dictionaries:

### Example

Create three dictionaries, then create one dictionary that will contain the other three dictionaries:

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

Try it Yourself »

---

## The dict() Constructor

It is also possible to use the `dict()` constructor to make a new dictionary:

### Example

```
thisdict = dict(brand="Ford", model="Mustang", year=1964)  
# note that keywords are not string literals  
# note the use of equals rather than colon for the assignment  
print(thisdict)
```

Try it Yourself »

# Dictionary Methods

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

Method	Description
<u>clear()</u>	Removes all the elements from the dictionary
<u>copy()</u>	Returns a copy of the dictionary
<u>fromkeys()</u>	Returns a dictionary with the specified keys and value
<u>get()</u>	Returns the value of the specified key
<u>items()</u>	Returns a list containing a tuple for each key value pair
<u>keys()</u>	Returns a list containing the dictionary's keys
<u>pop()</u>	Removes the element with the specified key
<u>popitem()</u>	Removes the last inserted key-value pair
<u>setdefault()</u>	Returns the value of the specified key. If the key does not exist: insert the key, with the specified value
<u>update()</u>	Updates the dictionary with the specified key-value pairs
<u>values()</u>	Returns a list of all the values in the dictionary

## Test Yourself With Exercises

### Exercise:

Use the `get` method to print the value of the "model" key of the `car` dictionary.