



# Python Dictionary

In this article, you'll learn everything about Python dictionary; how they are created, accessing, adding and removing elements from them and, various built-in methods.

**Oakland, NJ: Rest in Peace with a \$350,000 Life Insurance Po**

TAP YOUR AGE

18-25

26-35

36-45

46-55

[Calculate Your Coverage](#)

## Table of Contents

- [What is dictionary in Python?](#)
- [How to create a dictionary?](#)
- [How to access elements from a dictionary?](#)
- [How to change or add elements in a dictionary?](#)
- [How to delete or remove elements from a dictionary?](#)
- [Python Dictionary Methods](#)
- [Python Dictionary Comprehension](#)
- [Other Dictionary Operations](#)
  - [Dictionary Membership Test](#)
  - [Iterating Through a Dictionary](#)
  - [Built-in Functions with Dictionary](#)

## What is dictionary in Python?

Python dictionary is an unordered collection of items. While other compound data types have only value as an element, a dictionary has a key: value pair.

Dictionaries are optimized to retrieve values when the key is known.

## How to create a dictionary?

[TUTORIAL](#)[EXAMPLES](#)[BUILT-IN FUNCTIONS](#)

comma.

An item has a key and the corresponding value expressed as a pair, key: value.

While values can be of any data type and can repeat, keys must be of immutable type ([string](#), [number](#) or [tuple](#) with immutable elements) and must be unique.

```
# empty dictionary
my_dict = {}

# dictionary with integer keys
my_dict = {1: 'apple', 2: 'ball'}

# dictionary with mixed keys
my_dict = {'name': 'John', 1: [2, 4, 3]}

# using dict()
my_dict = dict({1:'apple', 2:'ball'})

# from sequence having each item as a pair
my_dict = dict([(1,'apple'), (2,'ball')])
```

As you can see above, we can also create a dictionary using the built-in function `dict()`.

## How to access elements from a dictionary?

While indexing is used with other container types to access values, dictionary uses keys. Key can be used either inside square brackets or with the `get()` method.

The difference while using `get()` is that it returns `None` instead of `KeyError`, if the key is not found.

```
script.py  IPython Shell
1  my_dict = {'name': 'Jack', 'age': 26}
2
3  # Output: Jack
4  print(my_dict['name'])
5
```

```
6 # Output: 26
7 print(my_dict.get('age'))
8
9 # Trying to access keys which doesn't exist throws error
10 # my_dict.get('address')
11 # my_dict['address']
```

TUTORIAL

EXAMPLES

BUILT-IN FUNCTIONS



Run



Powered by DataCamp

When you run the program, the output will be:

```
Jack
26
```

## How to change or add elements in a dictionary?

Dictionary are mutable. We can add new items or change the value of existing items using assignment operator.

If the key is already present, value gets updated, else a new key: value pair is added to the dictionary.

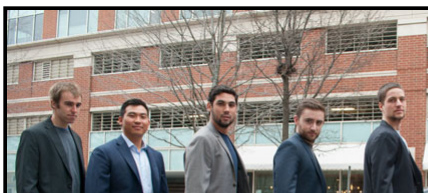
script.py IPython Shell

```
1 my_dict = {'name': 'Jack', 'age': 26}
2
3 # update value
4 my_dict['age'] = 27
5
6 #Output: {'age': 27, 'name': 'Jack'}
7 print(my_dict)
8
9 # add item
10 my_dict['address'] = 'Downtown'
11
12 # Output: {'address': 'Downtown', 'age': 27, 'name': 'Jack'}
13 print(my_dict)
```

Run



Powered by DataCamp



TUTORIAL

EXAMPLES

BUILT-IN FUNCTIONS



**Oakland, NJ: This Unbelievable  
Company Is Disrupting A \$2  
Industry**

When you run the program, the output will be:

```
{'name': 'Jack', 'age': 27}  
{'name': 'Jack', 'age': 27, 'address': 'Downtown'}
```

## How to delete or remove elements from a dictionary?

We can remove a particular item in a dictionary by using the method `pop()`. This method removes an item with the provided key and returns the value.

The method, `popitem()` can be used to remove and return an arbitrary item (key, value) from the dictionary. All the items can be removed at once using the `clear()` method.

We can also use the `del` keyword to remove individual items or the entire dictionary itself.

```
script.py  IPython Shell  
1  # create a dictionary  
2  squares = {1:1, 2:4, 3:9, 4:16, 5:25}  
3  
4  # remove a particular item  
5  # Output: 16  
6  print(squares.pop(4))  
7  
8  # Output: {1: 1, 2: 4, 3: 9, 5: 25}  
9  print(squares)  
10  
11 # remove an arbitrary item  
12 # Output: (1, 1)  
13 print(squares.popitem())  
14  
15 # Output: {2: 4, 3: 9, 5: 25}
```

```
16 print(squares)
17
18 # delete a particular item
19 del squares[5]
20
21 # Output: {2: 4, 3: 9}
```

TUTORIAL    EXAMPLES    BUILT-IN FUNCTIONS



Run



Powered by DataCamp

When you run the program, the output will be:

```
16
{1: 1, 2: 4, 3: 9, 5: 25}
(1, 1)
{2: 4, 3: 9, 5: 25}
{2: 4, 3: 9}
{}
```

## Python Dictionary Methods

Methods that are available with dictionary are tabulated below. Some of them have already been used in the above examples.

Python Dictionary Methods

Method	Description
<code>clear()</code>	Remove all items form the dictionary.
<code>copy()</code>	Return a shallow copy of the dictionary.
<code>fromkeys( seq [, v ])</code>	Return a new dictionary with keys from <code>seq</code> and value equal to <code>v</code> (defaults to <code>None</code> ).
<code>get( key [, d ])</code>	Return the value of <code>key</code> . If <code>key</code> doesnot exit, return <code>d</code> (defaults to <code>None</code> ).
<code>items()</code>	Return a new view of the dictionary's items (key, value).

<code>keys()</code>	Return a new view of the dictionary's keys.
<code>pop( key [, d ])</code>	Remove the item with <code>key</code> and return its value or <code>d</code> if <code>key</code> is not found. If <code>d</code> is not provided and <code>key</code> is not found, raises <code>KeyError</code> .

TUTORIAL

EXAMPLES

BUILT-IN FUNCTIONS



	the dictionary is empty.
<code>setdefault( key [, d ])</code>	If <code>key</code> is in the dictionary, return its value. If not, insert <code>key</code> with a value of <code>d</code> and return <code>d</code> (defaults to <code>None</code> ).
<code>update([ other ])</code>	Update the dictionary with the key/value pairs from <code>other</code> , overwriting existing keys.
<code>values()</code>	Return a new view of the dictionary's values

Here are a few example use of these methods.

```
script.py  IPython Shell
1  marks = {}.fromkeys(['Math', 'English', 'Science'], 0)
2
3  # Output: {'English': 0, 'Math': 0, 'Science': 0}
4  print(marks)
5
6  for item in marks.items():
7      print(item)
8
9  # Output: ['English', 'Math', 'Science']
10 list(sorted(marks.keys()))
```

Run



Powered by DataCamp

## Python Dictionary Comprehension

Dictionary comprehension is an elegant and concise way to create new dictionary from an iterable in Python.

Dictionary comprehension consists of an expression pair (key: value) followed by `for` statement inside curly braces `{}`.

Here is an example to make a dictionary with each item being a pair of a number and its square.

script.py    IPython Shell

TUTORIAL    EXAMPLES    BUILT-IN FUNCTIONS



```
4   print(squares)
```

Run



Powered by DataCamp

This code is equivalent to

```
squares = {}  
for x in range(6):  
    squares[x] = x*x
```

A dictionary comprehension can optionally contain more [for](#) or [if statements](#).

An optional `if` statement can filter out items to form the new dictionary.

Here are some examples to make dictionary with only odd items.

script.py    IPython Shell

```
1   odd_squares = {x: x*x for x in range(11) if x%2 == 1}  
2  
3   # Output: {1: 1, 3: 9, 5: 25, 7: 49, 9: 81}  
4   print(odd_squares)
```

Run



Powered by DataCamp

TUTORIAL

EXAMPLES

BUILT-IN FUNCTIONS



## Dictionary Membership Test

We can test if a key is in a dictionary or not using the keyword `in`. Notice that membership test is for keys only, not for values.

script.py IPython Shell

```
1 squares = {1: 1, 3: 9, 5: 25, 7: 49, 9: 81}
2
3 # Output: True
4 print(1 in squares)
5
6 # Output: True
7 print(2 not in squares)
8
9 # membership tests for key only not value
10 # Output: False
11 print(49 in squares)
```

Run



Powered by DataCamp

## Iterating Through a Dictionary

Using a `for` loop we can iterate through each key in a dictionary.

script.py IPython Shell

```
1 squares = {1: 1, 3: 9, 5: 25, 7: 49, 9: 81}
2 for i in squares:
3     print(squares[i])
```



Run



Powered by DataCamp

TUTORIAL

EXAMPLES

BUILT-IN FUNCTIONS



Built-in functions like `all()`, `any()`, `len()`, `cmp()`, `sorted()` etc. are commonly used with dictionary to perform different tasks.

### Built-in Functions with Dictionary

Function	Description
<code>all()</code>	Return <code>True</code> if all keys of the dictionary are true (or if the dictionary is empty).
<code>any()</code>	Return <code>True</code> if any key of the dictionary is true. If the dictionary is empty, return <code>False</code> .
<code>len()</code>	Return the length (the number of items) in the dictionary.
<code>cmp()</code>	Compares items of two dictionaries.
<code>sorted()</code>	Return a new sorted list of keys in the dictionary.

Here are some examples that uses built-in functions to work with dictionary.

script.py IPython Shell

```
1 squares = {1: 1, 3: 9, 5: 25, 7: 49, 9: 81}
2
3 # Output: 5
4 print(len(squares))
5
6 # Output: [1, 3, 5, 7, 9]
7 print(sorted(squares))
```

Run



Powered by DataCamp