# Python Dictionary

**Summary**: in this tutorial, you'll learn about Python Dictionary that allows you to organize related information.

## Introduction to the Python Dictionary type

A Python dictionary is a collection of key-value pairs where each key is associated with a value.

A value in the key-value pair can be a number (https://www.pythontutorial.net/python-basics/python-numbers/) , a string (https://www.pythontutorial.net/python-basics/python-string/) , a list (https://www.pythontutorial.net/python-basics/python-list/) , a tuple (https://www.pythontutorial.net/python-basics/python-tuples/) , or even another dictionary. In fact, you can use a value of any valid type in Python as the value in the key-value pair.

A key in the key-value pair must be immutable. In other words, the key cannot be changed, for example, a number, a string, a tuple, etc.

Python uses the curly braces `{}` to define a dictionary. Inside the curly braces, you can place zero, one, or many key-value pairs.

The following example defines an empty dictionary:

```
empty_dict = {}
```

Typically, you define an empty dictionary before a loop, either for loop (https://www.pythontutorial.net/python-basics/python-for-loop-list/) or while loop (https://www.pythontutorial.net/python-basics/python-while/) . And inside the loop, you add key-value pairs to the dictionary.

To find the type of a dictionary, you use the `type()` function as follows:

```
empty_dict = {}

print(type(empty_dict))
```

Ouptut:

```
<class 'dict'>
```

The following example defines a dictionary with some key-value pairs:

```
person = {
    'first_name': 'John',
    'last_name': 'Doe',
    'age': 25,
    'favorite_colors': ['blue', 'green'],
    'active': True
}
```

The `person` dictionary has five key-value pairs that represent the first name, last name, age, favorite colors, and active status.

## Accessing values in a Dictionary

To access a value by key from a dictionary, you can use the square bracket notation or the `get()` method.

### 1) Using square bracket notation

To access a value associated with a key, you place the key inside square brackets:

```
dict[key]
```

The following shows how to get the values associated with the key `first_name` and `last_name` in the `person` dictionary:

```
person = {
    'first_name': 'John',
    'last_name': 'Doe',
    'age': 25,
    'favorite_colors': ['blue', 'green'],
    'active': True
}
print(person['first_name'])
print(person['last_name'])
```

Output:

```
John
Doe
```

## 2) Using the get() method

If you attempt to access a key that doesn't exist, you'll get an error. For example:

```
person = {
    'first_name': 'John',
    'last_name': 'Doe',
    'age': 25,
    'favorite_colors': ['blue', 'green'],
    'active': True
}
```

```
ssn = person['ssn']
```

Error:

```
Traceback (most recent call last):
  File "dictionary.py", line 15, in <module>
    ssn = person['ssn']
KeyError: 'ssn'
```

To avoid this error, you can use the `get()` method of the dictionary:

```
person = {
    'first_name': 'John',
    'last_name': 'Doe',
    'age': 25,
    'favorite_colors': ['blue', 'green'],
    'active': True
}

ssn = person.get('ssn')
print(ssn)
```

Ouput:

```
None
```

If the key doesn’t exist, the `get()` method returns `None` instead of throwing a `KeyError` . Note that `None` means no value exists.

The `get()` method also returns a default value when the key doesn’t exist by passing the default value to its second argument.

The following example returns the `'000-00-0000'` string if the `ssn` key doesn't exist in the `person` dictionary:

```
person = {
    'first_name': 'John',
    'last_name': 'Doe',
    'age': 25,
    'favorite_colors': ['blue', 'green'],
    'active': True
}

ssn = person.get('ssn', '000-00-0000')
print(ssn)
```

Output:

```
000-00-0000
```

## Adding new key-value pairs

Since a dictionary has a dynamic structure, you can add new key-value pairs to it at any time.

To add a new key-value pair to a dictionary, you specify the name of the dictionary followed by the new key in square brackets along with the new value.

The following example adds a new key-value pair to the `person` dictionary:

```
person['gender'] = 'Famale'
```

## Modifying values in a key-value pair

To modify a value associated with a key, you specify the dictionary name with the key in square brackets and the new value associated with the key:

```
dict[key] = new_value
```

The following example modifies the value associated with the `age` of the `person` dictionary:

```
person = {
    'first_name': 'John',
    'last_name': 'Doe',
    'age': 25,
    'favorite_colors': ['blue', 'green'],
    'active': True
}

person['age'] = 26

print(person)
```

Output:

```
{'first_name': 'John', 'last_name': 'Doe', 'age': 26, 'favorite_colors': ['blue',
```

## Removing key-value pairs

To remove a key-value pair by a key, you use the `del` statement:

```
del dict[key]
```

In this syntax, you specify the dictionary name and the key that you want to remove.

The following example removes the key `'active'` from the `person` dictionary:

```
person = {
    'first_name': 'John',
    'last_name': 'Doe',
    'age': 25,
    'favorite_colors': ['blue', 'green'],
    'active': True
}

del person['active']
print(person)
```

Output:

```
{'first_name': 'John', 'last_name': 'Doe', 'age': 25, 'favorite_colors': ['blue',
```

## Looping through a dictionary

To examine a dictionary, you can use a `for` loop to iterate over its key-value pairs, or keys, or values.

> Note that since Python 3.7, when you loop through a dictionary, you'll get the key-value pairs in the same order that you insert them.

### 1) Looping all key-value pairs in a dictionary

Python dictionary provides a method called `items()` that returns an object which contains a list of key-value pairs as tuples in a list.

For example:

```
person = {
    'first_name': 'John',
    'last_name': 'Doe',
    'age': 25,
    'favorite_colors': ['blue', 'green'],
    'active': True
}

print(person.items())
```

Output:

```
dict_items([('first_name', 'John'), ('last_name', 'Doe'), ('age', 25), ('favorite
```

To iterate over all key-value pairs in a dictionary, you use a `for` loop with two variable `key` and `value` to unpack each tuple (https://www.pythontutorial.net/python-basics/python-unpack-list/) of the list:

```
person = {
    'first_name': 'John',
    'last_name': 'Doe',
    'age': 25,
    'favorite_colors': ['blue', 'green'],
    'active': True
}

for key, value in person.items():
    print(f"{key}: {value}")
```

Output:

```
first_name: John
last_name: Doe
age: 25
```

```
favorite_colors: ['blue', 'green']
active: True
```

Note that you can use any variable name in the `for` loop. They don't have to be the `key` and `value` .

## 2) Looping through all the keys in a dictionary

Sometimes, you just want to loop through all keys in a dictionary. In this case, you can use a `for` loop with the `keys()` method.

The `keys()` method returns an object that contains a list of keys in the dictionary.

For example:

```
person = {
    'first_name': 'John',
    'last_name': 'Doe',
    'age': 25,
    'favorite_colors': ['blue', 'green'],
    'active': True
}

for key in person.keys():
    print(key)
```

Output:

```
first_name
last_name
age
favorite_colors
active
```

In fact, looping through all keys is the default behavior when looping through a dictionary. Therefore, you don't need to use the `keys()` method.

The following code returns the same output like the one in the above example:

```
person = {
    'first_name': 'John',
    'last_name': 'Doe',
    'age': 25,
    'favorite_colors': ['blue', 'green'],
    'active': True
}

for key in person:
    print(key)
```

## 3) Looping through all the values in a dictionary

The `values()` method returns a list of values without any keys.

To loop through all the values in a dictionary, you use a for loop with the `values()` method:

```
person = {
    'first_name': 'John',
    'last_name': 'Doe',
    'age': 25,
    'favorite_colors': ['blue', 'green'],
    'active': True
}

for value in person.values():
    print(value)
```

Output:

```
John
Doe
25
['blue', 'green']
True
```

## Summary

- A Python dictionary is a collection of key-value pairs, where each key has an associated value.
- Use square brackets or `get()` method to access a value by its key.
- Use the `del` statement to remove a key-value pair by the key from the dictionary.
- Use `for` loop to iterate over keys, values, key-value pairs in a dictionary.