What Is A Dictionary In Python?

It is a collection data type just like a list or a set, but there are certain features that make python dictionary unique. A dictionary in python is not ordered and is changeable as well. We can make changes in a dictionary unlike sets or strings which are immutable in nature. Dictionary contains key-value pairs like a map that we have in other programming languages. A dictionary has indexes. Since the value of the keys we declare in a dictionary are always unique, we can use them as indexes to access the elements in a dictionary.

Also, a dictionary does not have any duplicate members. Although the value elements in the key value pairs can contain duplicate values.

Why Use A Dictionary In Python?

First of all, it is not like any other object or data type in python programming language. A dictionary has key value pairs resembling a map. It is often used for unordered data with distinct key values.

It is similar to a real life dictionary. We have distinct keys which we can use to fetch the values specified in them. In case of dictionary, even though there are no duplicate members, we can mention duplicate members in the value elements. Now that we know, why we use a dictionary, lets try to understand how it is different from lists in python.

Lists vs Dictionary

Lists	Dictionary
Ordered	Not ordered
Access elements use index values	Access elements use keys as index values
Collection of elements	Collection of key value pairs
Allows duplicate members	No duplicate members
Preferred for ordered data	Preferred for data with unique key values

How To Implement A Dictionary?

To declare a dictionary in python, we use the curly brackets. The keys and values are separated with a colon and the pairs are separated with a comma.

```
mydictionary = { 'key1' : 'value1' , 'key2': 'value2' , 'key3': 'value3'}
print(mydictionary)
```

Operations In A Python Dictionary

Accessing an element

```
mydictionary = { 1: 'edureka' , 2: 'python' , 3: 'data science'}
mydictionary[1]
#this will get the key value pair with the key 1.
mydictionary.get(1)
#this is another function which will serve the same purpose.
```

Replacing an element

```
mydictionary = { 1: 'edureka', 2: 'python' , 3: 'data science'}
mydictionary[3] = 'artificial intelligence'
print(mydictionary)
#this will replace the value at key 3 to artificial intelligence.
```

Removing an element

```
mydictionary = { 1:'edureka' , 2 : 'python', 3: 'data science'}
del mydictionary[3]
print(mydictionary)
#this will remove the key value pair from the dictionary with the specified key.
```

Other Operations

Following are the operations we have for dictionary in python:

- clear()
- copy()
- values()
- update()
- fromkeys()
- get()
- items()
- keys()
- pop()
- popitem()
- setdefault()

clear( ) – removes all the elements from the dictionary.

```
a = { 1: 2 , 2: 3 , 3: 5}
a.clear()
print(a)
#you will get a empty dictionary as the output.
```

copy( ) – returns a copy of the dictionary.

```
a = {1:2, 2: 3, 3: 4}
b = a.copy()
print(b)
#b will be a copy of the dictionary a.
```

values( ) – returns all the values in a dictionary.

```
a = {1: 2, 2: 3, 3:4}
a.values( )
#this will get you the list of all the values in the dictionary.
```

update( ) – it updates the values of the dictionary with the specified key value pairs.

```
a = {1 : 2, 2: 3, 3: 5}
a.update({4: 6})
#this will update the dictionary with the specified key value pair.
```

fromkeys( ) – returns a dictionary with the specified keys and values.

```
a = {1: 'edureka' , 2: 'data science'}
b = {1: 2, 2: 3, 3: 'python'}
a.Fromkeys(b)
#this will get the dictionary with the specified keys and values.
```

items( ) – returns the list for a tuple of each key value pair in the dictionary.

```
a = {1: 'edureka', 2; 'python'}
a.items()
#this will get the list of tuple for each key value pair.
```

keys( ) – returns a list containing all the keys in the dictionary.

```
a = { 1: 'edureka' , 2 : 'python' , 3 : 'data science'}
a.keys()
#this will get the list of all the keys from the dictionary.
```

pop( ) – removes the element with the specified key.

```
a = { 1: 'edureka' , 2: 'data science' , 3: 'python' }
a.pop(3)
#this will remove the value 'python' from the dictionary.
```

popitem( ) – removes the last inserted key values pair from the dictionary.

```
a = { 1: 'edureka' , 2: 'python' , 3: 'data science'}
a.popitem()
#this will remove the last inserted key value pair from the dictionary.
```

setdefault( ) – returns the value of the specified key, if not present insert the key with the specified value.

```
a = { 1: 'edureka' , 2: 'python' }
a.setdefault(1, 'edureka')
```

Dict( ) constructor

Dictionary constructor is used to declare a dictionary in python.

```
a = dict( 1= 'edureka' , 2= 'python' , 3= 'data science')
print(a)
#this will declare a dictionary with the name a and specified key value pairs.
```