




# PYTHON



# PYTHON DICTIONARY



Python Dictionary is used to store the data in a key-value pair format. The dictionary is the data type in Python, which can simulate the real-life data arrangement where some specific value exists for some particular key. It is the mutable data-structure. The dictionary is defined into element Keys and values.

Keys must be a single element

Value can be any type such as list, tuple, integer, etc.

In other words, we can say that a dictionary is the collection of key-value pairs where the value can be any Python object. In contrast, the keys are the immutable Python object, i.e., Numbers, string, or tuple.

# CREATING THE DICTIONARY

The dictionary can be created by using multiple key-value pairs enclosed with the curly brackets {}, and each key is separated from its value by the colon (:).The syntax to define the dictionary is given below.

```
Dict = {"Name": "Tom", "Age": 22}
```

In the above dictionary **Dict**, The keys **Name** and **Age** are the string that is an immutable object.


Let's see an example to create a dictionary and print its content

```
Employee = {"Name": "John", "Age": 29, "salary":25000,"Company":"GOOGLE"}
```

```
print(type(Employee))
```

```
print("printing Employee data .... ")
```

```
print(Employee)
```



Python provides the built-in function **dict()** method which is also used to create dictionary. The empty curly braces `{}` is used to create empty dictionary.

# Creating an empty Dictionary

```
Dict = {}
```

```
print("Empty Dictionary: ")
```

```
print(Dict)
```

# Creating a Dictionary with dict() method

```
Dict = dict({1: 'Java', 2: 'T', 3:'Point'})
```

```
print("\nCreate Dictionary by using dict(): ")
```

```
print(Dict)
```

# Creating a Dictionary with each item as a Pair

```
Dict = dict([(1, 'Devansh'), (2, 'Sharma')])
```

```
print("\nDictionary with each item as a pair: ")
```

```
print(Dict)
```


# ACCESSING THE DICTIONARY VALUES

We have discussed how the data can be accessed in the list and tuple by using the indexing.

However, the values can be accessed in the dictionary by using the keys as keys are unique in the dictionary.

The dictionary values can be accessed in the following way.





```
Employee = {"Name": "John", "Age": 29, "salary":25000,"Company":"GOOGLE"}  
print(type(Employee))  
print("printing Employee data .... ")  
print("Name : %s" %Employee["Name"])  
print("Age : %d" %Employee["Age"])  
print("Salary : %d" %Employee["salary"])  
print("Company : %s" %Employee["Company"])
```

# ADDING DICTIONARY VALUES

The dictionary is a mutable data type, and its values can be updated by using the specific keys. The value can be updated along with key **Dict[key] = value**. The `update()` method is also used to update an existing value.

Note: If the key-value already present in the dictionary, the value gets updated. Otherwise, the new keys added in the dictionary.

Let's see an example to update the dictionary values.



```
# Creating an empty Dictionary
```

```
Dict = {}
```

```
print("Empty Dictionary: ")
```

```
print(Dict)
```

```
# Adding elements to dictionary one at a time
```

```
Dict[0] = 'Peter'
```

```
Dict[2] = 'Joseph'
```

```
Dict[3] = 'Ricky'
```

```
print("Dictionary after adding 3 elements: ")
```

```
print(Dict)
```



```
# Adding set of values with a single Key The Emp_ages doesn't exist to dictionary
```

```
Dict['Emp_ages'] = 20, 33, 24
```

```
print("\\nDictionary after adding 3 elements: ")
```

```
print(Dict)
```

```
# Updating existing Key's Value
```


```
Dict[3] = 'JavaTpoint'
```

```
print("\\nUpdated key value: ")
```

```
print(Dict)
```

# DELETING ELEMENTS USING DEL KEYWORD

```
Employee = {"Name": "John", "Age": 29, "salary":25000,"Company":"GOOGLE"}  
print(type(Employee))  
print("printing Employee data .... ")  
print(Employee)  
print("Deleting some of the employee data")  
del Employee["Name"]  
del Employee["Company"]  
print("printing the modified information ")  
print(Employee)
```



```
print("Deleting the dictionary: Employee");  
del Employee  
print("Lets try to print it again ");  
print(Employee)
```

# USING POP() METHOD

The **pop()** method accepts the key as an argument and remove the associated value. Consider the following example.

```
# Creating a Dictionary
```

```
Dict = {1: 'JavaTpoint', 2: 'Peter', 3: 'Thomas'}
```

```
# Deleting a key
```

```
# using pop() method
```

```
pop_ele = Dict.pop(3)
```

```
print(Dict)
```

# ITERATING DICTIONARY

```
Employee = {"Name": "John", "Age": 29, "salary":25000,"Company":"GOOGLE"}
```

```
for x in Employee:
```

```
    print(x)
```



# FOR LOOP TO PRINT ALL THE VALUES OF THE DICTIONARY

```
Employee = {"Name": "John", "Age": 29, "salary":25000,"Company":"GOOGLE"}
```

```
for x in Employee:
```

```
    print(Employee[x])
```

# FOR LOOP TO PRINT THE VALUES OF THE DICTIONARY BY USING VALUES() METHOD.

```
Employee = {"Name": "John", "Age": 29, "salary":25000,"Company":"GOOGLE"}  
for x in Employee.values():  
    print(x)
```

# FOR LOOP TO PRINT THE ITEMS OF THE DICTIONARY BY USING ITEMS() METHOD.

```
Employee = {"Name": "John", "Age": 29, "salary":25000,"Company":"GOOGLE"}
```

```
for x in Employee.items():
```

```
    print(x)
```


# PROPERTIES OF DICTIONARY KEYS

In the dictionary, we cannot store multiple values for the same keys. If we pass more than one value for a single key, then the value which is last assigned is considered as the value of the key.

```
Employee={"Name":"John","Age":29,"Salary":25000,"Company":"GOOGLE","Name":  
"John"}
```

```
for x,y in Employee.items():
```

```
    print(x,y)
```



In python, the key cannot be any mutable object. We can use numbers, strings, or tuples as the key, but we cannot use any mutable object like the list as the key in the dictionary.

```
Employee = {"Name": "John", "Age": 29, "salary":25000,"Company":"GOOGLE",[10  
0,201,301]:"Department ID"}
```

```
for x,y in Employee.items():
```

```
    print(x,y)
```

# BUILT-IN DICTIONARY FUNCTIONS

Sr.	Function	Description
1	<code>cmp(dict1, dict2)</code>	It compares the items of both the dictionary and returns true if the first dictionary values are greater than the second dictionary, otherwise it returns false.
2	<code>len(dict)</code>	It is used to calculate the length of the dictionary.
3	<code>str(dict)</code>	It converts the dictionary into the printable string representation.
4	<code>type(variable)</code>	It is used to print the type of the passed variable.



**CONTINUE IN NEXT UNIT . . . . .**