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.

Syntax:

```python
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.

```python
Employee = {"Name": "John", "Age": 29, "salary":25000,"Company":"GOOGLE"}
print(type(Employee))
print("printing Employee data .... ")
print(Employee)
```

Output

```
<class 'dict'>
Printing Employee data ....
{'Name': 'John', 'Age': 29, 'salary': 25000, 'Company': 'GOOGLE'}
```

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.

```python
# 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)
```

Output:

```
Empty Dictionary:
{}

Create Dictionary by using dict():
{1: 'Java', 2: 'T', 3: 'Point'}

Dictionary with each item as a pair:
{1: 'Devansh', 2: 'Sharma'}
```

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.

```python
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"])
```

Output:

```
<class 'dict'>
printing Employee data ....
Name : John
Age : 29
Salary : 25000
Company : GOOGLE
```

Python provides us with an alternative to use the get() method to access the dictionary values. It would give the same result as given by the indexing.

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.

Example - 1:

```python
# 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("\nDictionary 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] = 'JavaWorld'
print("\nUpdated key value: ")
print(Dict)
```

Output:

```
Empty Dictionary:
{}

Dictionary after adding 3 elements:
{0: 'Peter', 2: 'Joseph', 3: 'Ricky'}

Dictionary after adding 3 elements:
{0: 'Peter', 2: 'Joseph', 3: 'Ricky', 'Emp_ages': (20, 33, 24)}

Updated key value:
{0: 'Peter', 2: 'Joseph', 3: 'JavaWorld', 'Emp_ages': (20, 33, 24)}
```

Example - 2:

```
Employee = {"Name": "John", "Age": 29, "salary":25000,"Company":"GOOGLE
"}
print(type(Employee))
print("printing Employee data .... ")
print(Employee)
print("Enter the details of the new employee....")
Employee["Name"] = input("Name: ")
Employee["Age"] = int(input("Age: "))
Employee["salary"] = int(input("Salary: "))
Employee["Company"] = input("Company:")
```

```
print("printing the new data");
print(Employee)
```

Output:

```
Empty Dictionary:
{}

Dictionary after adding 3 elements:
{0: 'Peter', 2: 'Joseph', 3: 'Ricky'}

Dictionary after adding 3 elements:
{0: 'Peter', 2: 'Joseph', 3: 'Ricky', 'Emp_ages': (20, 33, 24)}

Updated key value:
{0: 'Peter', 2: 'Joseph', 3: 'JavaWorld', 'Emp_ages': (20, 33, 24)}
```

Deleting elements using del keyword

The items of the dictionary can be deleted by using the **del** keyword as given below.

```
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)
```

Output:

```
<class 'dict'>
printing Employee data ....
{'Name': 'John', 'Age': 29, 'salary': 25000, 'Company': 'GOOGLE'}
Deleting some of the employee data
printing the modified information
```

```
{'Age': 29, 'salary': 25000}
Deleting the dictionary: Employee
Lets try to print it again
NameError: name 'Employee' is not defined
```

The last print statement in the above code, it raised an error because we tried to print the Employee dictionary that already deleted.

- **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: 'JavaWorld', 2: 'Peter', 3: 'Thomas'}
# Deleting a key
# using pop() method
x = Dict.pop(3)
print(Dict)

print(x)
```

Output:

```
{1: 'JavaWorld', 2: 'Peter'}
```

Python also provides a built-in methods popitem() and clear() method for remove elements from the dictionary. The popitem() removes the arbitrary element from a dictionary, whereas the clear() method removes all elements to the whole dictionary.

Iterating Dictionary

A dictionary can be iterated using for loop as given below.

Example 1

# for loop to print all the keys of a dictionary

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

```python
print(x)
```

Output:

```
Name
Age
salary
Company
```

Example 2

#for loop to print all the values of the dictionary

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

Output:

```
John
29
25000
GOOGLE
```

Example - 3

#for loop to print the values of the dictionary by using values() method.

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

Output:

```
John
29
25000
GOOGLE
```

Example 4

#for loop to print the items of the dictionary by using items() method.

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

Output:

```
('Name', 'John')
('Age', 29)
('salary', 25000)
('Company', 'GOOGLE')
```

Properties of Dictionary keys

1. 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.

Consider the following example.

```python
Employee={"Name":"John","Age":29,"Salary":25000,"Company":"GOOGLE","Name":"Jimy"}
for x,y in Employee.items():
    print(x,y)
```

Output:

```
Name John
Age 29
Salary 25000
Company GOOGLE
```

2. 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.

Consider the following example.

```python
Employee = {"Name": "John", "Age": 29, "salary":25000,"Company":"GOOGLE",[100,201,301]:"Department ID"}
for x,y in Employee.items():
    print(x,y)
```

Output:

```
Traceback (most recent call last):
  File "dictionary.py", line 1, in
    Employee = {"Name": "John", "Age": 29,
"salary":25000,"Company":"GOOGLE",[100,201,301]:"Department ID"}
TypeError: unhashable type: 'list'
```

Built-in Dictionary functions

The built-in python dictionary methods along with the description are given below.

SN	Function	Description
1	cmp(dict1, dict2)	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	len(dict)	It is used to calculate the length of the dictionary.
3	str(dict)	It converts the dictionary into the printable string representation.
4	type(variable)	It is used to print the type of the passed variable.

Built-in Dictionary methods

The built-in python dictionary methods along with the description are given below.

SN	Method	Description
1	dic.clear()	It is used to delete all the items of the dictionary.
2	dict.copy()	It returns a shallow copy of the dictionary.
3	dict.fromkeys(iterable, value = None, /)	Create a new dictionary from the iterable with the values equal to value.
4	dict.get(key, default = "None")	It is used to get the value specified for the passed key.

5	dict.has_key(key)	It returns true if the dictionary contains the specified key.
6	dict.items()	It returns all the key-value pairs as a tuple.
7	dict.keys()	It returns all the keys of the dictionary.
8	dict.setdefault(key,default= "None")	It is used to set the key to the default value if the key is not specified in the dictionary
9	dict.update(dict2)	It updates the dictionary by adding the key-value pair of dict2 to this dictionary.
10	dict.values()	It returns all the values of the dictionary.
11	len()	
12	popItem()	
13	pop()	
14	count()	
15	index()	