Introduction of Python Dictionary

Python Dictionary:-

Python Dictionary is used to store the data in a key-value pair format. The dictionary is the data type in Python. 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:

```
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)
```

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.

```
# Creating an empty Dictionary
Dict = {}
print("Empty Dictionary: ")
print(Dict)
# Creating a Dictionary
# with dict() method
Dict = dict({1: 'Python', 2: 'is a', 3:'Programming Language'})
print("\n Create Dictionary by using dict(): ")
print(Dict)
# Creating a Dictionary
# with each item as a Pair
Dict = dict([(1, 'Devansh'), (2, 'Sharma')])
print("\n Dictionary with each item as a pair: ")
print(Dict)
```

Output:

```
Empty Dictionary:
{}

Create Dictionary by using dict():
{1: 'Python', 2: 'is a', 3: 'Programming Language'}

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

Accessing the dictionary values:

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"])
```

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:

```
# 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("\n Dictionary after adding 3 elements:")
print(Dict)
# Updating existing Key's Value
Dict[3] = 'Python'
print("\n Updated 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: 'Python', '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:

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
```