Dictionary

- **Dictionaries** are a useful data structure for storing data in Python because they are capable of imitating real-world data arrangements where a certain value exists for a given key.
- **Dictionary in Python** is a collection of keys values, used to store data values like a map, which, unlike other data types which hold only a single value as an element.

The data is stored as key-value pairs using a Python dictionary.

a) This data structure is mutable
b) The components of dictionary were made using keys and values.
c) Keys must only have one component.
d) Values can be of any type, including integer, list, and tuple.

❖ Creating a Dictionary

- Curly brackets are the simplest way to generate a Python dictionary, although there are other approaches as well.
- With many key-value pairs surrounded in curly brackets and a colon separating each key from its value, the dictionary can be built. (:). The following provides the syntax for defining the dictionary.

Syntax:

```
Dict = {"Name": "Gayle", "Age": 25}
```

✓ In the above dictionary **Dict**, The keys **Name** and **Age** are the strings which comes under the category of an immutable object.

- Python provides the built-in function **dict()** method which is also used to create the dictionary.

- The empty curly braces {} is used to create empty dictionary.

❖ Accessing the dictionary values

- In order to access the items of a dictionary refer to its key name. Key can be used inside square brackets.
- There is also a method called get() that will also help in accessing the element from a dictionary.This method accepts key as argument and returns the value.

Example:

```
# Python program to demonstrate

# accessing a element from a Dictionary


# Creating a Dictionary

Dict = {1: 'TopperWorld', 'name': 'For', 3: 'TopperWorld'}


# accessing a element using key

print("Accessing a element using key:")

print(Dict['name'])


# accessing a element using key

print("Accessing a element using key:")

print(Dict[1])
```

Output:

```
Accessing a element using key:

For

Accessing a element using key:

TopperWorld
```

❖ Adding Dictionary Values

- The dictionary is a mutable data type, and utilising the right keys allows you to change its values.
- Dict[key] = value and the value can both be modified.
- An existing value can also be updated using the update() method.

Note: The value is updated if the key-value pair is already present in the dictionary. Otherwise, the dictionary's newly added keys.

.

❖ Deleting Elements using del Keyword

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

Example:

```
# Python program to demonstrate deleting Elements using del Keyword

# Creating a Dictionary

Dict = {1: 'Topper World', 'name': 'For', 3: 'Topper World'}

#Deleting some of the Dictionary data

del(Dict[1])

print("Data after deletion Dictionary=")

print(Dict)
```

Output:

```
Data after deletion Dictionary={'name': 'For', 3: 'Topper World'}
```

❖ Deleting Elements using pop() Method

- A dictionary is a group of key-value pairs in Python. You can retrieve, insert, and remove items using this unordered, mutable data type by using their keys.
- The pop() method is one of the ways to get rid of elements from a dictionary.

❖ Iterating Dictionary

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

Example:

```
my_dict = {'a': 1, 'b': 2, 'c': 3}


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

Output:

```
Key: a,   Value: 1
Key: b,    Value: 2
Key: c,    Value: 3
```

❖ 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.
- The key cannot belong to any mutable object in Python. Numbers, strings, or tuples can be used as the key, however mutable objects like lists cannot be used as the key in a dictionary.

❖ Dictionary Methods

Method	Description
dic.clear()	Remove all the elements from the dictionary
dict.copy()	Returns a copy of the dictionary
dict.get(key, default = “None”)	Returns the value of specified key
dict.items()	Returns a list containing a tuple for each key value pair
dict.keys()	Returns a list containing dictionary’s keys
dict.update(dict2)	Updates dictionary with specified key-value pairs
dict.values()	Returns a list of all the values of dictionary
pop()	Remove the element with specified key
popItem()	Removes the last inserted key-value pair
dict.setdefault(key,default= “None”)	set the key to the default value if the key is not specified in the dictionary
dict.has_key(key)	returns true if the dictionary contains the specified key.

dict.get(key, default = “None”)	used to get the value specified for the passed key.

Example:

```
# Creating a dictionary
person = {
    "name": "Alice",
    "age": 30,
    "city": "Wonderland"
}

# Accessing dictionary values
print("Name:", person["name"])
print("Age:", person["age"])
print("City:", person["city"])
```

Output:

```
Name: Alice
Age: 30
City: Wonderland
```