Chapter – 7

Dictionaries

* Dictionary – Key : Value Pairs

Among the built-in Python data types is a very versatile type called a dictionary. Dictionaries are simply another type of collection in Python, but with a twist. Rather than having an index associated with each data-item, Dictionaries in Python have a "key" and a "value of that key". That is, Python dictionaries are a collection of some key-value pairs.

1. Creating a Dictionary

To create a dictionary, you need to include the key : value pairs in curly braces as per following syntax :—

<dictionary-name> = {<key> : <value>, <key> : <value> ...}

A Dictionary is a Mapping

Dictionaries are actually types of mappings. A mapping type of Python is a mutable type that maps values of one type (the key type) to arbitrary objects (values).

2. Accessing Elements of a Dictionary —

while accessing elements from a dictionary, you need a key.

i. Traversing a Dictionary —

Traversal of a collection means accessing and processing each element of it. Thus traversing a dictionary also means the same and same is the tool for it, i.e., the Python loops.

```
for <item> in <dictionary>:
    process each item here.
```

ii. Accessing Keys or Values Simultaneously —

To see all the keys in a dictionary in one go, you may write <dictionary>.keys() and to see all values in one go, you may write <dictionary>.values(), as shown below (for the same dictionary d created above):—

```
>>> d.keys
dict_keys(["Vowel5", "Vowel4", "Vowel3", "Vowel2",
           Vowel1"])
>>> d.values()
dict_values(["u", "o", "i", "e", "a"])
```

3. Characteristics of a Dictionary —

i. Unordered set

ii. Not a sequence

iii. Indexed by keys, not numbers

iv. Keys must be unique

v. Dictionaries are Mutable

vi. Internally stored as Mappings

* Working with Dictionaries

1. Multiple ways of creating Dictionaries

i. Initializing a Dictionary —

In this method all the key : value pair of a dictionary are written collectively, separated by commas and enclosed in curly braces.

Employee = {'name' : 'John', 'salary' : 10000, 'age' : 243

ii. Adding key:value Pairs to an Empty Dictionary

In this method, firstly an empty dictionary is created and then keys and values are added to it one pair at a time.

→ To create an empty dictionary, there are two ways:

(i) by giving dictionary contents in empty curly braces as shown below:

```
Employees = {}
```

(ii) by using dictionary constructor dict() as shown below:

```
Employee = dict()
```

→ Next step is to add key:value pairs, one at a time

```
<dictionary>[<key>] = <value>
```

iii. Creating a Dictionary from name and value pair

Using the dict() constructor of dictionary, we can also create a new dictionary initialized from specified set of keys and values.

2. **Adding Elements to Dictionary**

We can add new elements (key:value pair) to a dictionary using assignment as per following syntax. But the key ~~~~ being added must not exist in dictionary and must be unique.

<dictionary>[<key>] = <value>

i. **Nesting Dictionaries**

You can even add dictionaries as values inside a dictionary.
For e.g.

```
Employees = {"John": {"age": 25, "salary": 20000}, "Riya":
             {"age": 35, "salary": 50000}}
```

3. **Updating / Modifying Existing Elements in a Dictionary**

~~Updating an element is similar~~
We can change value of an existing key using assignment as per following syntax:

<dictionary>[<key>] = <value>

4. Deleting Elements from a dictionary —

For deleting elements from a dictionary you can ~~use~~ used the del statement as ~~explained below~~ shown below : —

del <dictionary> [<key>]

5. Checking for Existence of a Key —

Usual membership operators in and not in work with dictionaries as well but they can check for the existence of keys only. To check for whether a value is present in a dictionary, we need to write proper code for that.

<key> in <dictionary>
<key> not in <dictionary>

- The in operator will return True if the given key is present in the dictionary, otherwise false.
- The not in operator will return True if the given key is not present in the dictionary otherwise False.

6. Pretty Printing a Dictionary —

In order to display the dictionary in a proper way we import one module "JSON".

JSON = Java Script Object Notation.

For e.g.

```
data = {1:"A", 2:"B", 3:"C"}
import json
print(json.dumps(data, indent=2))
```

⇒

```
{
  1:"A"
  2:"B"
  3:"C"
}
```

* Dictionary Functions and Methods

1. Get Length of the Dictionary —

len(<dictionary>)

- Takes dictionary name as argument and returns an integer.

2. Creating new Dictionary —

The dict() function is used to create dictionary.

dict(mapping)
dict(iterable)
- were the dict() takes a mapping or iterable sequence as an argument and returns a dictionary created from it.

3. Accessing Items, Keys and Values

<dict>.get() to get value of the give key

<dict>.items() to get all the items of the dictionary.

<dict>.keys() to get all the keys of the dictionary.

<dict>.values() to get all the values of the dictionary.

4. Extend / Update dictionary with new key:value Pairs

The syntax to use this method is given below:

<dictionary>.update(<other-dictionary>)

5. Deleting Elements from Dictionary —

del <dict>[<key>] to delete a key from the dictionary.

del <dict> to delete the whole dictionary.

	Function / method	description
1.	len(<D>)	Returns the number of items in the dictionary
2.	<D>.get(key, [default])	Returns corresponding value if the key exists, else it returns Python's Error if default argument is missing otherwise returns default argument.
3.	<D>.items()	Returns a sequence of (key, value) pairs the items.
4.	<D>.keys()	Returns a list sequence of key
5.	<D> values()	Returns a list sequence of values
6.	<D>.update(<other-dictionary>)	Merges key: value pairs from the new dictionary into the original dictionary.
7.	<D>.clear()	Removes all items.