PYTHON REVISION TOUR – II(Cont…)

DICTIONARY

DICTIONARIES IN PYTHON

Dictionary - Key :Value Pairs

- are collection or bunch of values in a single variable.
- These are collection of key-value pairs.
- It associates key to values.
- ***Dictionaries are mutable, unordered collections with elements in the form of a key:value pair that associate keys to value.***

Creating a Dictionary

- Dictionary can be created by including the key : value pair in curly braces.
- –Syntax: <dictionary name>=,<key>:<value>,< key>:<value>, …….-
- –Example: dictionary by the name **teachers** that stores **name of teachers** as **key** and **subjects being taught** by them as **value** of respective key
- teachers={“VS":"Computer Science",“SAR": “Physics", “SS":"Maths",“BS":“Chemistry"}
- Keys of dictionary must be of immutable type.

key : value pair	**key**	**value**
“VS”:”Computer Science”	VS	Computer Science
”SAR”: “Physics”	SAR	Physics
”SS”:”Maths”	SS	Maths
”BS”:”Chemistry”	BS	Chemistry

Accessing elements of a Dictionary

Dictionary is accessed through its key.

Syntax: <dictionary name>[<key>]

Example:

```
>>> teachers= {"VS": "Computer Science", "SAR": "Physics", "SS":"Maths",“ BS":"Chemistry"}
>>> teachers["SS"]
```

Output:

'Maths'

Mentioning dictionary name without any square bracket displays the entire content of the dictionary.

Example:

```
>>> teachers
```

Output:

{'VS': 'Computer Science', 'SAR': 'Physics', 'SS': 'Maths', 'BS': 'Chemistry'}

Characteristics of a Dictionary

- **Unordered Set:** order that the keys are added doesn't necessarily reflect what order they may be reported back.
- **Not a Sequence**
- **Indexed by Keys, Not Numbers**
- **Keys must be Unique:** More than one entry per key not allowed. Which means no duplicate key is allowed. When duplicate keys encountered during assignment, the last assignment wins.
- **Mutable:** which means they can be changed
- **Internally Stored as Mapping:** is a mapping of unique keys to values

Traversing a Dictionary

- Traversing means accessing each element of a collection.
- For loop helps to traverse each elements of dictionary as per following syntax:

for <item> in <dictionary>:

process each item here

Example:

```
>>> for key in teachers:
        print(key," : ", teachers[key])
```

Output:

VS : Computer Science

SAR : Physics

SS : Maths

BS : Chemistry

Dictionaries are un-ordered set of elements, the printed order of elements is not same as the order you stored the elements in.

Accessing Keys or Values Simultaneously

- To see all keys in **dictionary** we write <dictionary>.keys( )

Example:

>>> teachers.keys()

Output:

dict_keys(['VS', 'SAR', 'SS', 'BS'])

- To see all values in **dictionary** we write <dictionary>.values( )

Example:

>>> teachers.values()

Output:

- dict_values(['Computer Science', 'Physics', 'Maths', 'Chemistry'])
- We can convert the sequence returned by keys( ) and values( ) function by using list( )

Example:

>>> list(teachers.keys())

Output:

['VS', 'SAR', 'SS', 'BS']

Adding Elements to Dictionary

```
released["iphone"] = 2007
released["iphone3G"] = 2008
released
```

```
{'iphone': 2007, 'iphone3G': 2008}
```

```
released = { "iphone" : 2007, "iphone 3G" : 2008, "iphone 3GS" :2009,
             "iphone 4" : 2010, "iphone 4S" : 2011, "iphone 5" : 2012 }
released
```

```
{'iphone': 2007,
 'iphone 3G': 2008,
 'iphone 3GS': 2009,
 'iphone 4': 2010,
 'iphone 4S': 2011,
 'iphone 5': 2012}
```

```
released["iphone5S"] = 2014
released
```

```
{'iphone': 2007,
 'iphone 3G': 2008,
 'iphone 3GS': 2009,
 'iphone 4': 2010,
 'iphone 4S': 2011,
 'iphone 5': 2012,
 'iphone5S': 2014}
```

Updating Existing Elements in a Dictionary

- Change the value of an existing key using assignment.

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

Example:

```
In [42]: released
Out[42]: {'iphone': 2007,
          'iphone 3G': 2008,
          'iphone 3GS': 2009,
          'iphone 4': 2010,
          'iphone 4S': 2011,
          'iphone 5': 2012,
          'iphone5S': 2014}

In [43]: released["iphone5S"] = 2015
         released
Out[43]: {'iphone': 2007,
          'iphone 3G': 2008,
          'iphone 3GS': 2009,
          'iphone 4': 2010,
          'iphone 4S': 2011,
          'iphone 5': 2012,
          'iphone5S': 2015}
```

- Note: Make sure key must exist in the dictionary otherwise new entry will be added to the dictionary.

Deleting Elements From a Dictionary

- There are two methods for deleting element from a dictionary

(i) To delete a dictionary element or a dictionary entry, i.e key:value pair we will use del command.

Example:

```
>>> del teachers["SS"]
>>> teachers
{'VS': 'Computer Science', 'SAR': 'Physics', 'BS': 'Chemistry'}
```

The pop( ) Method

(ii) –pop( ) method

Syntax:

<dictionary>.pop(<key>)

This method will not only delete the key:value pair for mentioned key but also return the corresponding value.

Example:

```
>>> teachers.pop("VS")
'Computer Science'
>>> teachers
{'SAR': 'Physics', 'BS': 'Chemistry'}
```

If we try to delete a key which does not exist, python gives KeyError

Example:

```
>>> del teachers["VR"]
Traceback (most recent call last):
  File "<pyshell#26>", line 1, in <module>
    del teachers["VR"]
KeyError: 'VR'
```

Checking of Existence of a Key

- Membership operators **in** and **not in** checks the presence of key in a dictionary. It does not check the presence of value.

Syntax:

<key> **in** <dictionary>

returns true if given key is present in the dictionary, otherwise false.

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

returns true if given key is not present in the dictionary, otherwise false

Example:

```
>>> "BS" in teachers
True
>>> "VS" in teachers
False
>>> "VS" not in teachers
True
```

Checking of Existence of a Value

Example:

```
>>> "Chemistry" in teachers.values()
True

>>> "Maths" in teachers.values()
False
>>> "Maths" not in teachers.values()
True
```

Dictionary Functions and Methods

The len() Method

This function returns the length of the dictionary.

Syntax: len(<dictionary>)

Example:

>>> len(teachers)

Output:

2

The clear() Method

This function removes all the items from the dictionary.

Syntax:

<dictionary>.clear( )

Example:

```
>>> teachers.clear()
>>> teachers
{}
```

The get() Method

This function returns the corresponding value for the key.

Syntax:

<dictionary>.get(key,[default])

Example:

```
>>> teachers.get("SAR")
'Physics‘

>>> teachers.items()
dict_items([('SAR', 'Physics'), ('BS', 'Chemistry')])
```

The key() Method

This function returns all of the keys in the dictionary.

Syntax:

<dictionary>.keys( )

Example:

```
>>> teachers.keys()
dict_keys(['SAR', 'BS'])
```

The values() Method

This function returns all the values from the dictionary as a sequence.

Syntax:

<dictionary>.values( )

Example:

```
>>> teachers.values()
dict_values(['Physics', 'Chemistry'])
```

The update() Method

This function merges key: value pairs from the new dictionary into the original dictionary , adding or replacing as needed.

Syntax:

dictionary>.update(<new dictionary>)

Example:

```
>>> new_teachers= {"LB":"English"}
>>> teachers.update(new_teachers)
>>> teachers
{'SAR': 'Physics', 'BS': 'Chemistry', 'LB': 'English'}
```