Chapter 7-DICTIONARIES

Dictionary

- **Dictionaries are mutable unordered collections with elements in the form of a key : value pairs.**
- **We can change dictionary elements.**
- **It can contain values of mixed datatypes.**

Creating dictionary

Dictionaries are formed by placing key :value pairs
Ex:- {1:'a',2:'b',3:'c'}
{'Name': 'asha','age':17}
# keys of a dictionary must be of immutable types

- **The curly brackets mark the beginning and end of the dictionary.**
- **Each entry(key:Value)consists of a pair separated by a colon—the key and corresponding value is given by writing colon(☺ between them,**
- **The key –value pairs are separated by commas(,).**

```
 rno=[ ]
 mks=[ ]
for a in range(4 ):
  r, m=eval (input("Enter Roll No., Marks:"))
  rno.append(r)
  mks.append(m)
d={rno[ 0]:mks[0],rno[1]:mks[1], rno[2]:mks[2], rno[3]:mks[3]}
print("Created dictionary")
print(d)
```

```
Enter Roll No., Marks :1, 67.5
Enter Roll No., Marks :2, 45.6
Enter Roll No., Marks :3, 78.4
Enter Roll No.,Marks:4,70.5
Created dictionary
```

Accessing Elements of a Dictionary

- While accessing elements from a dictionary , we need key .(but in list , the elements are accessed through their index)
- **In dictionaries, the elements are accessed through the keys defined in the key:value pairs,**
  As per the syntax shown below:

Lookup:-A dictionary operation that takes a key and finds the corresponding value, is called lookup

Eg

```
>>> d={"Vowell":"a","Vowel12":"e","Vowel13":"I","Vowel4":"o","Vowel5":"u"}
>>>d["Vowel1"]
'a'
"Vowel4"
```

Accessing elements using their keys;"Vowel1" and

are the keys used to access corresponding values.

```
>>>d["Vowel4"]
'o'
```

Notice that keys given here are in quotation marks i.e are of string type.

"Traversing a Dictionary"

To traverse a dictionary 'for' loop is used.

```
Syntax :- for <item> in <list>
                    process each item here

D = {1:'a',2:'b',3:'c'}
for x in D
        print(x,' : ',D[x])
Output :-   1 : 'a'
            2 : 'b'
            3 : 'c'
```

Accessing Keys or Values Simultaneously

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

```
>>>
d={“Vowell”:”a”,”Vowel2”:”e”,”Vowel3”:”I”,”Vowel4”:”o”,”Vowel5”:”u”}

>>>d.keys( )

dict_keys([‘Vowel5’,’Vowel4’,’Vowel3’,’Vowel2’,’Vowel1’])

>>>d.values( )

Dict_values([‘u’,’o’,’I’,’e’,’a’])
```

- **We can convert the sequence returned by Keys ( ) and values( ) functions by using list( ) as shown below:**
- **>>>List(d.keys())**
  [([‘Vowel5’,’Vowel4’,’Vowel3’,’Vowel2’,’Vowel1’]
  >>>list(d.values( ))

  [‘u’,’o’,’i’,’e’,’a’]

Characteristics of a dictionary

- **Unordered Set**
  - ✓ **A dictionary is a unordered set of Key:value pairs.**
  - ✓ **We can not tell the order or position of Key:Value pairs in a dictionary as there is no index associated.**
- **Not a sequence**
  - ✓ **Unlike a string, list and tuple, a dictionary is not a sequence because it is unordered set of element.**
  - ✓ **The sequences are indexed by a range of ordinal numbers. Hence , they are ordered , but a dictionary is an unordered collection.**
- **Indexed by Keys, Not Numbers**
- **Keys must be Unique**
  - ✓ **Each of the Keys within a dictionary must be unique.**
  - ✓ **Keys are used to identify values in a dictionary, there cannot be duplicate keys in a dictionary.**

- **Dictionaries are Mutable**
- **Internally Stored as Mappings**

“Multiple Ways of Creating Dictionaries”

(i) Initializing a Dictionary

- ➨ **D = {1:’a’, 2:’b’, 3:’c’}**

(ii) Adding Key:Value Pairs to an Empty Dictionary

- ➨ **D = dict(1=‘a’, 2=‘b’, 3=‘c’)**

  D => {1:’a’, 2:’b’, 3:’c’}

(iii) Creating a Dictionary from name and value Pairs

- ➨ **D= dict(zip((1,2,3),(‘a’,’b’,’c’)))**

  D => {1:’a’, 2:’b’, 3:’c’}

(iv) Initializing a Dictionary

- ➨ **D = dict((([1,‘a’], [2,‘b’], [3,‘c’])))**

  D => {1:’a’, 2:’b’, 3:’c’}

Adding elements to a dictionary

D = {1:’a’, 2:’b’, 3:’c’}
D[4] = ‘box’
D => {1:’a’, 2:’b’, 3:’c’,4 : ‘box’}

Updating elements of a dictionary

D = {1:’a’, 2:’b’, 3:’c’}
D[2] = 10
D => {1:’a’, 2:10, 3:’c’,4 : ‘box’}

Nesting Dictionaries

- ✓ We can even add dictionaries as Values inside a dictionary.
- ✓
- ✓ We can store a dictionary as a value only, inside a dictionary.

Eg .

Employees={‘John’:{‘age’:25,’salary’:20000},’Diya’:{‘age’:35,’Salary’:50,000}}

for key in Employees:

```
print("Employee", key, ':')
```

The element are being accessed from inner dictionaries, stores

```
print('Age:', str(Employees[key]['age']))

print('Salary:',str(Employees[key]['salary]))
```

Output

```
Employee John:
  Age : 25
  Salary: 20,000

Employee Diya:
  Age : 35
  Salary: 50,000
```

Updating/Modifying Existing Elements in a Dictionary

- ✓ **We can change value of an existing key using assignment as per following syntax:**
- ✓ **Dictionary >[<key>]=<value>**

```
>>>Employee={'name':'John','salary':10000,'age':24}
>>>Employee['salary']=20,000
>>>Employee
{'salary':20000,'age':24,'name':'John'}
```

In Dictionaries , the updation and addition of elements are similar in syntax. But for addition, the key must not exist in the dictionary and for updation, the key must exist in the dictionary.

Deleting Elements from a Dictionary

- ✓ **To delete a dictionary element or a dictionary entry, i.e, a key:value pair, we can use del command.**

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

del D[2]

D     => {1:'a', 3:'c'}

                 OR

 D.pop( 2)

 D     => {1:'a', 3:'c'}
```

Check for Existence of a Key

a) 'in'   :- <key> in <dictionary>

b) 'not in' :- <key> not in <dictionary>

```
D = {1:'a', 2:'b', 3:'c'}
2 in D        =>  True
4 not in D  =>  True
```

The in and not in operator check for membership only in keys of the dictionary and not in values.

Pretty printing a Dictionary

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

       import json

        print(jsob.dumps(D, indent=2) )

        {

           "1"  :  'a',

            "2" : 'b',
```

“3” : ‘c’

}

Dictionary Functions & Methods

a) len( ) :- It returns length of the dictionary.

Ex :- D = {1:’a’, 2:’b’, 3:’c’}

len(D)   => 3

b) clear( ) :- It removes all the elements from the dictionary.

Ex:- D= {1:’a’, 2:’b’, 3:’c’}

D.clear( )

D   => {}

Dictionary Functions & Methods

Dict() :- this function is used to create a new dictionary from iterables and other dictionaries. Passing

no argument to dict( ) will create an empty dictionary{ }.

Eg

Dict(iterable)

Eg data=[[1,67.8],[2,75.5],[3,72.5]]

D1=dict(data)

Print(“Created dictionary is “)

Print(d1)

UPDATE( ) method :-

- **The update( ) method merges key:value pairs from the new dictionary into the original dictionary, adding or replacing as needed.**
- **The items in the new dictionary are added to the old one and override any items already there with the same keys.**

D = {1:’a’, 2:’b’, 3:’c’}

D[2] = 10

D => {1:’a’, 2:10, 3:’c’,4 : ‘box’}

get() and items() function)

get( ) :- It gets the value with the given key from the dictionary.

Ex:- D= {1:’a’, 2:’b’, 3:’c’}

D.get(1) => ‘a’

items( ) :- It returns all the items in the dictionary as a sequence

of (key, value) tuples.

Ex :- D = {1:’a’, 2:’b’, 3:’c’}

```
lis = D.items( )
for x in lis :
        print (x)
```

Output :-

```
(1,'a')
(2,'b')
(3,'c')
```