Chapter 3

Dictionaries

After studying this lesson, the students will be able to

- *understand the need of dictionaries;*
- *solve problems by using dictionaries;*
- *get clear idea about dictionaries functions; and*
- *understand the difference between list and dictionary.*

What is dictionary?

A dictionary is like a list, but more in general. In a list, index value is an integer, while in a dictionary index value can be any other data type and are called keys. The key will be used as a string as it is easy to recall. A dictionary is an extremely useful data storage construct for storing and retrieving all key value pairs, where each element is accessed (or indexed) by a unique key. However, dictionary keys are not in sequences and hence maintain no left-to right order.

Key-value pair

We can refer to a dictionary as a mapping between a set of indices (which are called keys) and a set of values. Each key maps a value. The association of a key and a value is called a **key-value pair**.

Syntax:

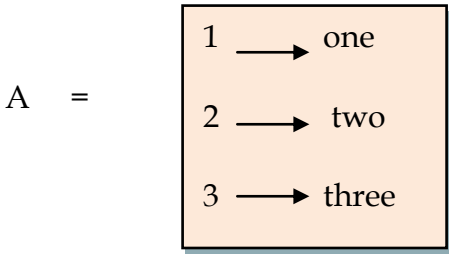
my_dict = {'key1': 'value1','key2': 'value2','key3': 'value3'...'keyn': 'valuen'}

Note: Dictionary is created by using curly brackets(ie. {}).

Example

```
>>> A={1:"one",2:"two",3:"three"}
>>> print A
{1: 'one', 2: 'two', 3: 'three'}
```

In the above example, we have created a list that maps from numbers to English words, so the keys values are in numbers and values are in strings.



Map between keys and values

Example

```
>>>computer={'input':'keybord','output':'mouse','language':'python','os':'windows-8',}
>>> print computer
{'input': 'keyboard', 'os': 'windows-8', 'language': 'python', 'output': 'mouse'}
>>>
```

In the above example, we have created a list that maps from computer related things with example, so here the keys and values are in strings. The order of the key-value pairs is not in same order (ie. input and output orders are not same). We can get different order of items in different computers. Thus, the order of items in a dictionary is unpredictable.

Example

```
>>>
D={'sun':'Sunday','mon':'Monday','tue':'Tuesday','wed':'Wednesday','thu':'Thursday','fri':'Friday','sat':'Saturday'}
>>> print D
{'wed': 'Wednesday', 'sun': 'Sunday', 'thu': 'Thursday', 'tue': 'Tuesday', 'mon': 'Monday', 'fri': 'Friday', 'sat': 'Saturday'}
```

Creation, initializing and accessing the elements in a Dictionary

The function dict ( ) is used to create a new dictionary with no items. This function is called built-in function. We can also create dictionary using {}.

```
>>> D=dict()
>>> print D
{}
```

{} represents empty string. To add an item to the dictionary (empty string), we can use square brackets for accessing and initializing dictionary values.

Example

```
>>> H=dict()
>>> H["one"]="keyboard"
>>> H["two"]="Mouse"
>>> H["three"]="printer"
>>> H["Four"]="scanner"
>>> print H
{'Four': 'scanner', 'three': 'printer', 'two': 'Mouse', 'one': 'keyboard'}
>>>
```

Traversing a dictionary

Let us visit each element of the dictionary to display its values on screen. This can be done by using 'for-loop'.

Example

Code

```
H={'Four': 'scanner', 'three': 'printer', 'two': 'Mouse', 'one': 'keyboard'}
for i in H:
print i,":", H[i],"   ",
```

Output

```
>>>
Four: scanner     one: keyboard     three: printer     two: Mouse
>>>
```

OR

Code

```
H = {'Four': 'scanner', 'three': 'printer', 'two': 'Mouse', 'one': 'keyboard'}
print "i value","\t","H[i] value"
for i in H:
print i,"\t", H[i]
```

Output

```
i value   H[i] value
Four      scanner
one       keyboard
three     printer
two       Mouse
```

As said previously, the order of items in a dictionary is unpredictable.

Creating, initializing values during run time (Dynamic allocation)

We can create a dictionary during run time also by using dict () function. This way of creation is called dynamic allocation. Because, during the run time, memory keys and values are added to the dictionary.

Example

Write a program to input total number of sections and class teachers' name in 11th class and display all information on the output screen.

Code

```
classxi=dict()
n=input("Enter total number of section in xi class")
i=1
while i<=n:
a=raw_input("enter section")
```

```
b=raw_input ("enter class teacher name")
classxi[a]=b
i=i+1
print "Class","\t","Section","\t","teacher name"
for i in classxi:
print "XI","\t",i,"\t",classxi[i]
```

Output

```
>>>
Enter total number of section in xi class3
enter sectionA
enter class teacher nameLeena
enter sectionB
enter class teacher nameMadhu
enter sectionC
enter class teacher nameSurpreeth
Class    Section    teacher name
XI       A          Leena
XI       C          Surpreeth
XI       B          Madhu
>>>
```

Appending values to the dictionary

We can add new elements to the existing dictionary, extend it with single pair of values or join two dictionaries into one. If we want to add only one element to the dictionary, then we should use the following method.

Syntax:

Dictionary name [key]=value

Example

```
>>> a={"mon":"monday","tue":"tuesday","wed":"wednesday"}
>>> a["thu"]="thursday"
>>> print a
{'thu': 'thursday', 'wed': 'wednesday', 'mon': 'monday', 'tue': 'tuesday'}
>>>
```

Merging dictionaries: An update ( )

Two dictionaries can be merged in to one by using update ( ) method. It merges the keys and values of one dictionary into another and overwrites values of the same key.

Syntax:

Dic_name1.update (dic_name2)

Using this dic_name2 is added with Dic_name1.

Example

```
>>> d1={1:10,2:20,3:30}
>>> d2={4:40,5:50}
>>> d1.update(d2)
>>> print d1
{1: 10, 2: 20, 3: 30, 4: 40, 5: 50}
```

Example

```
{1: 10, 2: 30, 3: 30, 5: 40, 6: 60}  # k>>> d1={1:10,2:20,3:30}   # key 2 value is 20
>>> d2={2:30,5:40,6:60}  #key 2 value is 30
>>> d1.update(d2)
>>> print d1
ey 2 value is replaced with 30 in d1
```

Removing an item from dictionary

We can remove item from the existing dictionary by using del key word.

Syntax:

del dicname[key]

Example

```
>>> A={"mon":"monday","tue":"tuesday","wed":"wednesday","thu":"thursday"}
>>> del A["tue"]
>>> print A
{'thu': 'thursday', 'wed': 'wednesday', 'mon': 'monday'}
>>>
```

Dictionary functions and methods

cmp ( )

This is used to check whether the given dictionaries are same or not. If both are same, it will return 'zero', otherwise return 1 or -1. If the first dictionary having more number of items, then it will return 1, otherwise return -1.

Syntax:

cmp(d1,d2) #d1and d2 are dictionary.

returns 0 or 1 or -1

Example

```
>>>
D1={'sun':'Sunday','mon':'Monday','tue':'Tuesday','wed':'Wednesday','thu':'Thursday','fri':'Friday','sat':'Saturday'}
>>>
D2={'sun':'Sunday','mon':'Monday','tue':'Tuesday','wed':'Wednesday','thu':'Thursday','fri':'Friday','sat':'Saturday'}
>>> D3={'mon':'Monday','tue':'Tuesday','wed':'Wednesday'}
>>> cmp(D1,D3)                    #both are not equal
```

```
1
>>> cmp(D1,D2)        #both are  equal
0
>>> cmp(D3,D1)
-1
```

len( )

This method returns number of key-value pairs in the given dictionary.

Syntax:

len(d) #d dictionary

returns number of items in the list.

Example

```
>>> H={'Four': 'scanner', 'three': 'printer', 'two': 'Mouse', 'one': 'keyboard'}
>>> len(H)
4
```

clear ( )

It removes all items from the particular dictionary.

Syntax:

d.clear( ) #d dictionary

Example

```
>>> D={'mon':'Monday','tue':'Tuesday','wed':'Wednesday'}
>>> print D
{'wed': 'Wednesday', 'mon': 'Monday', 'tue': 'Tuesday'}
>>> D.clear( )
>>> print D
{}
```

get(k, x)

There are two arguments (k, x) passed in 'get( )' method. The first argument is key value, while the second argument is corresponding value. If a dictionary has a given key (k), which is equal to given value (x), it returns the corresponding value (x) of given key (k). However, if the dictionary has no key-value pair for given key (k), this method returns the default values same as given key value. The second argument is optional. If omitted and the dictionary has no key equal to the given key value, then it returns None.

Syntax:

D.get (k, x) #D dictionary, k key and x value

Example

```
>>>
D={'sun':'Sunday','mon':'Monday','tue':'Tuesday','wed':'Wednesday','thu':'Thursda
y','fri':'Friday','sat':'Saturday'}
>>> D.get('wed',"wednesday")        # corresponding value wed
'Wednesday'
>>> D.get("fri","monday")           # default value of fri
'Friday'
>>> D.get("mon")                    # default value of mon
'Monday'
>>> D.get("ttu")                    # None
>>>
```

has_key( )

This function returns 'True', if dictionary has a key, otherwise it returns 'False'.

Syntax:

D.has_key(k) #D dictionary and k key

Example

```
>>>
```

```
D={'sun':'Sunday','mon':'Monday','tue':'Tuesday','wed':'Wednesday','thu':'Thursda
y','fri':'Friday','sat':'Saturday'}
>>> D.has_key("fri")
True
>>> D.has_key("aaa")
False
>>>
```

items( )

It returns the content of dictionary as a list of key and value. The key and value pair will be in the form of a tuple, which is not in any particular order.

Syntax:

D.items() # D dictionary

Example

```
>>>
D={'sun':'Sunday','mon':'Monday','tue':'Tuesday','wed':'Wednesday','thu':'Thursda
y','fri':'Friday','sat':'Saturday'}
>>> D.items()
[('wed', 'Wednesday'), ('sun', 'Sunday'), ('thu', 'Thursday'), ('tue', 'Tuesday'), ('mon',
'Monday'), ('fri', 'Friday'), ('sat', 'Saturday')]
```

Note: items () is different from print command because, in print command dictionary values are written in {}

keys()

It returns a list of the key values in a dictionary, , which is not in any particular order.

Syntax:

D.keys( ) #D dictionary

Example

```
>>>
```

```
D={'sun':'Sunday','mon':'Monday','tue':'Tuesday','wed':'Wednesday','thu':'Thursda
y','fri':'Friday','sat':'Saturday'}
>>> D.keys()
['wed', 'sun', 'thu', 'tue', 'mon', 'fri', 'sat']
>>>
```

values()

It returns a list of values from key-value pairs in a dictionary, which is not in any particular order. However, if we call both the items () and values() method without changing the dictionary's contents between these two (items() and values()), Python guarantees that the order of the two results will be the same.

Syntax:

```
D.values()      #D values
```

Example

```
>>>
D={'sun':'Sunday','mon':'Monday','tue':'Tuesday','wed':'Wednesday','thu':'Thursda
y','fri':'Friday','sat':'Saturday'}
>>> D.values()
['Wednesday', 'Sunday', 'Thursday', 'Tuesday', 'Monday', 'Friday', 'Saturday']
>>> D.items()
[('wed', 'Wednesday'), ('sun', 'Sunday'), ('thu', 'Thursday'), ('tue', 'Tuesday'), ('mon',
'Monday'), ('fri', 'Friday'), ('sat', 'Saturday')]
```

Solved Examples

1. Write a python program to input 'n' names and phone numbers to store it in a dictionary and to input any name and to print the phone number of that particular name.

   Code

```
phonebook=dict()
```