Python Dictionary

- Dictionary is an unordered set of key and value pair.
- It is an container that contains data, enclosed within curly braces.
- The pair i.e., key and value is known as item.
- The key and the value is separated by a colon(:). This pair is known as item. Items are separated from each other by a comma(,). Different items are enclosed within a curly brace and this forms Dictionary.
- Dictionary is mutable i.e., value can be updated.
- Key must be unique and immutable. Value is accessed by key. Value can be updated while key cannot be changed.
- Dictionary is known as **Associative array** since the Key works as Index and they are decided by the user.

eg:

```
data={100:'Ravi' ,101:'Vijay' ,102:'Rahul'}
print data
```

Output:

```
>>>
{100: 'Ravi', 101: 'Vijay', 102: 'Rahul'}
>>>
```

```
plant={}
plant[1]='Ravi'
plant[2]='Manoj'
plant['name']='Hari'
plant[4]='Om'
print plant[2]
print plant['name']
print plant[1]
print plant
```

Output:

```
>>>
Manoj
Hari
Ravi
{1: 'Ravi', 2: 'Manoj', 4: 'Om', 'name': 'Hari'}
>>>
```

Accessing Values

Since Index is not defined, a Dictionaries value can be accessed by their keys.

Syntax:

```
                [key]
Eg:
data1={'Id':100, 'Name':'Suresh',
'Profession':'Developer'}
data2={'Id':101, 'Name':'Ramesh',
'Profession':'Trainer'}
print "Id of 1st employer is",data1['Id']
print "Id of 2nd employer is",data2['Id']
print "Name of 1st employer:",data1['Name']
print "Profession of 2nd employer:",data2['Profession']

Output:
>>>
Id of 1st employer is 100
Id of 2nd employer is 101
Name of 1st employer is Suresh
Profession of 2nd employer is Trainer
```

Updation

The item i.e., key-value pair can be updated. Updating means new item can be added. The values can be modified.

Eg:

```
data1={'Id':100, 'Name':'Suresh',
'Profession':'Developer'}
data2={'Id':101, 'Name':'Ramesh',
'Profession':'Trainer'}
data1['Profession']='Manager'
data2['Salary']=20000
data1['Salary']=15000
print data1
print data2
```

Output:

```
>>>
{'Salary': 15000, 'Profession': 'Manager','Id': 100,
'Name': 'Suresh'}
{'Salary': 20000, 'Profession': 'Trainer', 'Id': 101,
'Name': 'Ramesh'}
>>>
```

Deletion

del statement is used for performing deletion operation.

An item can be deleted from a dictionary using the key.

Syntax:

```
del  [key]
```

Whole of the dictionary can also be deleted using the del statement.

```
Eg:
data={100:'Ram', 101:'Suraj', 102:'Alok'}
del data[102]
print data
del data
print data   #will show an error since dictionary is
deleted.
```

```
Output:
>>>
{100: 'Ram', 101: 'Suraj'}
Traceback (most recent call last):File
"C:/Python27/dict.py", line 5, in
```

Dictionary Functions:

Functions	Description
len(dictionary)	Gives number of items in a dictionary.
cmp(dictionary1,dictionary2)	Compares the two dictionaries.
str(dictionary)	Gives the string representation of a string.

1) len(dictionary):

Eg:

```
data={100:'Ram', 101:'Suraj', 102:'Alok'}
print data
print len(data)
Output:
>>>
{100: 'Ram', 101: 'Suraj', 102: 'Alok'}
3
>>>
```

2) cmp(dictionary1,dictionary2):

The comparison is done on the basis of key and value.

```
If, dictionary1 == dictionary2, returns 0.
      dictionary1 < dictionary2, returns -1.
     dictionary1 > dictionary2, returns 1.
Eg:
data1={100:'Ram', 101:'Suraj', 102:'Alok'}
data2={103:'abc', 104:'xyz', 105:'mno'}
data3={'Id':10, 'First':'Aman','Second':'Sharma'}
data4={100:'Ram', 101:'Suraj', 102:'Alok'}
print cmp(data1,data2)
print cmp(data1,data4)
print cmp(data3,data2)
Output:
>>>
-1
0
1
```

3) str(dictionary):

```
Eg:
data1={100:'Ram', 101:'Suraj', 102:'Alok'}
print str(data1)
Output:
>>>
{100: 'Ram', 101: 'Suraj', 102: 'Alok'}
>>>
```

Dictionary Methods:

Methods	Description
keys()	Return all the keys element of a dictionary.
values()	Return all the values element of a dictionary.
items()	Return all the items(key-value pair) of a dictionary.
update(dictionary2)	It is used to add items of dictionary2 to first dictionary.
clear()	It is used to remove all items of a dictionary. It returns an empty dictionary.
fromkeys(sequence,value1)/ fromkeys(sequence)	It is used to create a new dictionary from the sequence where sequence elements forms the key and all keys share the values ?value1?. In case value1 is not give, it set the values of keys to be none.
copy()	It returns an ordered copy of the data.
has_key(key)	It returns a boolean value. True in case if key is present in the dictionary ,else false.
get(key)	Returns the value of the given key. If key is not present it returns none.

1) keys():

Eg:

```
data1={100:'Ram', 101:'Suraj', 102:'Alok'}
print data1.keys()
```

Output:

```
>>>
[100, 101, 102]
>>>
```

2) values():

Eg:

```
data1={100:'Ram', 101:'Suraj', 102:'Alok'}
print data1.values()
```

Output:

```
>>>
['Ram', 'Suraj', 'Alok']
>>>
```

3) items():

Eg:

```
data1={100:'Ram', 101:'Suraj', 102:'Alok'}
print data1.items()

Output:
>>>
[(100, 'Ram'), (101, 'Suraj'), (102, 'Alok')]
>>>
```

4) update(dictionary2):

Eg:

```
data1={100:'Ram', 101:'Suraj', 102:'Alok'}
data2={103:'Sanjay'}
data1.update(data2)
print data1
print data2

Output:
>>>
{100: 'Ram', 101: 'Suraj', 102: 'Alok', 103: 'Sanjay'}
{103: 'Sanjay'}
>>>
```

5) clear():

Eg:

```
data1={100:'Ram', 101:'Suraj', 102:'Alok'}
print data1
data1.clear()
print data1

Output:
>>>
{100: 'Ram', 101: 'Suraj', 102: 'Alok'}
{}
>>>
```

6) fromkeys(sequence)/ fromkeys(seq,value):

Eg:

```
sequence=('Id' , 'Number' , 'Email')
data={}
data1={}
data=data.fromkeys(sequence)
print data
data1=data1.fromkeys(sequence,100)
print data1

Output:
>>>
{'Email': None, 'Id': None, 'Number': None}
{'Email': 100, 'Id': 100, 'Number': 100}
```

7) copy():

```
data={'Id':100 , 'Name':'Aakash' , 'Age':23}
data1=data.copy()
print data1
Output:
>>>
{'Age': 23, 'Id': 100, 'Name': 'Aakash'}
```

8) has_key(key):

```
data={'Id':100 , 'Name':'Aakash' , 'Age':23}
print data.has_key('Age')
print data.has_key('Email')
Output:
>>>
True
False
```

9) get(key):

```
data={'Id':100 , 'Name':'Aakash' , 'Age':23}
print data.get('Age')
print data.get('Email')
Output:
>>>
23
None
```

Dictionary Membership Test

We can test if a key is in a dictionary or not using the keyword in. Notice that membership test is for keys only, not for values.

```
>>> squares
{1: 1, 3: 9, 5: 25, 7: 49, 9: 81}
>>> 1 in squares
True


>>> 2 not in squares
True


>>> # membership tests for key only not value
>>> 49 in squares
False
```