DICTIONARY IN PYTHON

Introduction to Dictionary:

- Dictionary is a sequence data type.
- Dictionary is a mutable data type i.e. it can be changed after being created.
- Dictionary is denoted by curly braces i.e. {}
- Dictionaries are used to store data values in key : value pairs
- Keys of the dictionary are unique.

```
>>> D = {1:10, 2:20, 3:30}
>>> D
{1: 10, 2: 20, 3: 30}
```

Accessing items in a dictionary using keys:

```
>>> D = {1:10, 2:20, 3:30}
>>> D
{1: 10, 2: 20, 3: 30}
```

We can access the values of any key from the dictionary using following syntax:
Dict_Name[key_name]

```
>>> D[1]
10
>>> D[2]
20
>>> D[3]
30
```

If the key is duplicated at the time of dictionary creation then the value assigned at the last will be stored in the key.

```
>>> D = {1:10, 2:20, 3:30, 1:40, 4:50}
>>> D
{1: 40, 2: 20, 3: 30, 4: 50}
>>> D = {'A':10,'B':20,'C':30}
>>> D.get('A')
10
```

Mutability of a dictionary:

Mutability means the values of a key can be updated in the dictionary.
Dict_Name[Key_Name] = New_Value

```
>>> D = {'A':10,'B':20,'C':30}
Change the value of key ‘A’ from 10 to 15.
>>> D['A']=15
>>> D
{'A': 15, 'B': 20, 'C': 30}
```

Adding a new item in dictionary:

Adding an item to the dictionary is done by using a new index key and assigning a value to it:
Dict_Name[New_Key_Name] = Value

```
>>> D = {1:10, 2:20, 3:30}
>>> D
{1: 10, 2: 20, 3: 30}
>>> D[4]=40
>>> D
{1: 10, 2: 20, 3: 30, 4: 40}
```

Traversing a dictionary

- You can loop through a dictionary by using for loop.
- When looping through a dictionary, the return values are the keys of the dictionary, but there are methods to return the values as well.

We can use following methods to traversing a dictionary:

keys()

It returns the keys of the dictionary, as a list.

```
>>> D = {1:10, 2:20, 3:30}
>>> D.keys()
dict_keys([1, 2, 3])
#Print the keys of the dictionary.
D = {1:10, 2:20, 3:30}
for i in D.keys():
     print(i)
```

Output:

```
1
2
3
```

values()

It returns the values of the dictionary, as a list.

```
>>> D = {1:10, 2:20, 3:30}
>>> D.values()
```

```
dict_values([10, 20, 30])

#Print the values of the dictionary.
D = {1:10, 2:20, 3:30}
for i in D.values():
     print(i)
```

Output:
10
20
30

items()
It returns the key-value pairs of the dictionary, as tuples in a list.

```
>>> D = {1:10, 2:20, 3:30}
>>> D.values()
dict_items([(1, 10), (2, 20), (3, 30)])

#Print all keys and values of the dictionary.
D = {1:10, 2:20, 3:30}
for i in D.items():
     print(i)
```

Output:
(1, 10)
(2, 20)
(3, 30)

Built-in functions/methods of dictionary:

1. len()
It returns the number of keys in the list.

```
>>> D = {1:10, 2:20, 3:30}
>>> len(D)
3
```

2. dict()
The dict() function is used to create a dictionary.

```
>>> D = dict(name = "Shiva", age = 26, country = "India")
>>> D
{'name': 'Shiva', 'age': 26, 'country': 'India'}
```

3. get()
The get() method returns the value of the specified key.

```
>>> D = {1:10, 2:20, 3:30}
#Return the value of key ‘1’
>>> D.get(1)
10
#Returns the value of key ‘3’
>>> D.get(3)
30
```

4. update()
The update() method inserts specified elements to the dictionary or merge two dictionaries.
dictionary.update(iterable)

```
>>> D = {1:10, 2:20, 3:30}
>>> D.update({4:40})
>>> D
{1: 10, 2: 20, 3: 30, 4: 40}
```

If key is already exist in the dictionary then the value of the existing key will be update:

```
>>> D.update({3:35})
>>> D
{1: 10, 2: 20, 3: 35, 4: 40}
```

Merge two dictionaries using update() method

```
>>> D1={1:10,2:20}
>>> D1
{1: 10, 2: 20}
>>> D2={3:30,4:45}
>>> D2
{3: 30, 4: 45}
>>> D1.update(D2)
>>> D1
{1: 10, 2: 20, 3: 30, 4: 45}
```

5. del
del is a keyword.
del is used to delete an element from the dictionary using keyname.

```
>>> D = {1:10, 2:20, 3:30}
>>> D
```

```
{1: 10, 2: 20, 3: 30}
>>> del D[1]
>>> D
{2: 20, 3: 30}
```

del can also be used to delete the complete dictionary

```
>>> del D
>>> D
NameError: name 'D' is not defined.
```

6. clear()
The clear( ) method is used to delete all the elements (key:value pairs) from a dictionary.
It does not delete the structure of the dictionary

```
>>> D = {1:10, 2:20, 3:30}
>>> D
{1: 10, 2: 20, 3: 30}
>>> D.clear()
>>> D
{}
```

7. pop()
pop() method is used to delete a specific element from the dictionary and also return the value of the deleted element.

```
>>> D = {1:10, 2:20, 3:30}
>>> D
{1: 10, 2: 20, 3: 30}
>>> D.pop(2)
20
>>> D
{1: 10, 3: 30}
```

8. popitem()
popitem() method is used to delete the last element from the dictionary. It return the (key, value) pair of deleted elements in tuple type.

```
>>> D = {1:10, 2:20, 3:30}
>>> D
{1: 10, 2: 20, 3: 30}
>>> D.popitem()
(3, 30)
```

9. setdefault()

setdefault() method returns the value of the specified key. If the key does not exist in the dictionary it insert the key into dictionary, with the specified value

```
>>> D = {1:10, 2:20, 3:30}
Return the value of key ‘2’
>>> D.setdefault(2)
20
```

Return the value of key ‘3’. If key is already exists in the dictionary, 2nd argument does not effects

```
>>> D.setdefault(3,50)
30
>>> D
{1: 10, 2: 20, 3: 30}
```

Return the value of key ‘4’. If the key is not found in the dictionary, then add the key and specified value and then return the value of the key.

```
>>> D.setdefault(4,40)
40
>>> D
{1: 10, 2: 20, 3: 30, 4: 40}
```

If the key is not found in the dictionary, then add the key and None as the value of key (if value is not passed in the argument). It will not return anything.

```
>>> D.setdefault(5)
>>> D
{1: 10, 2: 20, 3: 30, 4: 40, 5: None}
```

10. max()

max() function returns the largest key from the dictionary. Comparison done on the bases of ASCII values

```
>>> D={1: 10, 2: 20, 3: 30, 4: 40}
>>> max(D)
4
```

If keys are string, here ‘h’ has the largest ASCII value.

```
>>> D={'a':10,'B':20,'h':30}
>>> max(D)
```

```
'h'
```

11. min()
min() function returns the smallest key from the dictionary.

```
>>> D={1: 10, 2: 20, 3: 30, 4: 40}
>>> min(D)
1
```

If keys are string, here ‘B’ has the largest ASCII value.

```
>>> D={'a':10,'B':20,'h':30}
>>> min(D)
'B'
```

12. fromkeys()
The fromkeys() method creates and returns a dictionary. It is useful when we want to create a dictionary with multiple keys which have the same value.
Syntax: dict.fromkeys(keys, value)

```
>>> K=('A','B','C')
>>> V=10
>>> D=dict.fromkeys(K,V)
>>> D
{'A': 10, 'B': 10, 'C': 10}
```

13. copy()
The copy() method returns a copy of the specified dictionary.
dictionary.copy()

```
>>> D={1:10,2:20}
Create a copy of dictionary D and store in D1
>>> D1=D.copy()
>>> D1
{1: 10, 2: 20}
```

14. sorted()
sorted() function sort the dictionary keys in ascending order

```
>>> D={1:10,4:40,3:30,2:20}
Return the keys of dictionary in sorted order
>>> sorted(D)
[1, 2, 3, 4]
Return the keys of dictionary in sorted order
>>> sorted(D.keys())
[1, 2, 3, 4]
```

```
Return the values of dictionary in sorted order
>>> sorted(D.values())
[10, 20, 30, 40]
Return the (key, value) pairs of dictionary in sorted order
>>> sorted(D.items())
[(1, 10), (2, 20), (3, 30), (4, 40)]
```

Suggested Programs:

Program-1 : Write a Program in Python to count the number of times a character appears in a given string using a dictionary.

```
S=input("Enter a string: ")
D={}
for c in S:
      if c in D:
      D[c]+=1
      else:
      D[c]=1
print("Frequency of Characters in String: \n",D)
```

Output:

```
Enter a string: S P SHARMA
Frequency of Characters in String:
 {'S': 2, ' ': 2, 'P': 1, 'H': 1, 'A': 2, 'R': 1, 'M': 1}
```

Program-2 : Write a Program in Python to create a dictionary with names of employees, their salary and access them

```
Emp = { }
while True :
      name = input("Enter employee name : ")
      salary = int(input("Enter employee salary : "))
      Emp[name] = salary
      c = input("Do you want to add more employee details(y/n) :")
      if c not in "yY" :
            break
print(Emp)
```

Output:

```
Enter employee name : Amit

Enter employee salary : 20000

Do you want to add more employee details(y/n) :y

Enter employee name : Sachin

Enter employee salary : 25000

Do you want to add more employee details(y/n) :y

Enter employee name : Saanvi

Enter employee salary : 28000

Do you want to add more employee details(y/n) :n

{'Amit': 20000, 'Sachin': 25000, 'Saanvi': 28000}
```