9.14 UPDATING ELEMENTS IN A DICTIONARY

You can also update a dictionary by modifying existing key-value pair or by merging another dictionary with an existing one.

Syntax:

```
<Dictionary>[<key>]=<value>
```

If key is there in the dictionary, then it will update the value of that particular key in the dictionary.

For example,

```
>>> Dict = {'Teena': 18,'Riya':12,'Alya':22,'Ravi':25}
>>> Dict['Riya']=28
>>> print(Dict)
{'Teena': 18, 'Riya': 28, 'Alya': 22, 'Ravi': 25}
>>> Dict['Priya']=60
>>> print(Dict)
{'Teena': 18, 'Riya': 28, 'Alya': 22, 'Ravi': 25, 'Priya': 60}
```

If key is not there in the dictionary, then it will add key-value pair in the dictionary.

Note: While adding a value, if the key value already exists, the value gets updated, otherwise a new key with the value is added at the end of the dictionary.

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

Syntax:

```
Dic_name1.update(dic_name2)
```

Using this, dic_name2 is added to Dic_name1.

For 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}
>>> print(d2)
{4: 40, 5: 50}
```

For example,

```
>>> d1={1: 10, 2:30, 3: 30, 5: 40, 6: 60}
>>> d2={1:10,2:20,3:30}
>>> d1.update(d2)
>>> print(d1)
{1: 10, 2: 20, 3: 30, 5: 40, 6: 60}
>>> print(d2)
{1: 10, 2: 20, 3: 30}
```

Value of key 2 is replaced with 20 in the dictionary d1.

9.15 REMOVING AN ITEM FROM DICTIONARY

We can remove an item from the existing dictionary by using del command or using del command or pop() method;

Note: We can remove an item from the existing dictionary by using del command or using pop() method.

1. **using del command:**

   Syntax:

```
   del dicname[key]
```

   For example,

```
>>> A={"mon":"monday","tue":"tuesday","wed":"wednesday",
     "thu":"thursday"}
>>> del A['wed']
>>> print(A)
{'mon': 'monday', 'tue': 'tuesday', 'thu': 'thursday'}
>>> del A['fri']
Traceback (most recent call last):
   File "<pyshell#3>", line 1, in <module>
      del A['fri']
KeyError: 'fri'
```

   If particular key is not there, then it will result in KeyError.

   If you want to delete the entire dictionary, then give the dictionary name along with del keyword.

   For example,

```
>>>del A
>>>A
Traceback (most recent call last):
  File "<pyshell#31>", line 1, in <module>
    A
NameError: name 'A' is not defined
```

   The dictionary named A no longer exists in the memory.

2. **using pop() method:** pop() method will not only delete the item specified by the key from the dictionary but also return the deleted value.

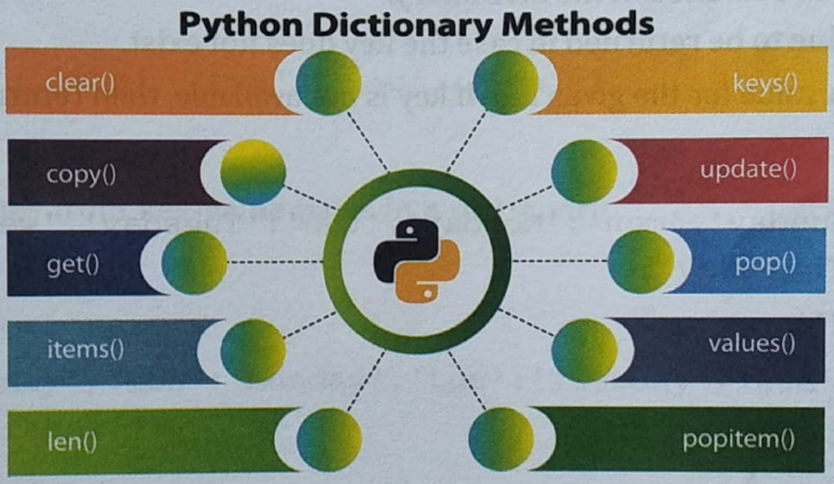
   Syntax:

```
   Dictname.pop(key)
>>> D3={'mon':'Monday','tue':'tuesday','wed':'wednesday'}
>>> D3.pop('tue')
'Tuesday'
>>> print(D3)
{'mon': 'monday', 'wed': 'wednesday'}
```

   It will return the deleted value.

9.17 COMMON DICTIONARY FUNCTIONS AND METHODS

Python provides you with a number of ways to manipulate the dictionary. Let's check out some of the important ones with examples.



len()

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

Syntax:

```
    len(d) # d is dictionary
```

returns number of key-value pairs in the dictionary.

```
>>> d1={1: 10, 2: 30, 3: 30, 5: 40, 6: 60}
>>> len(d1)
5
```

clear()

It removes all items from the particular dictionary.

Syntax:

```
    d.clear() # d is dictionary
```

For example,

```
>>> D={1:'one',2:'two',3:'three'4:'four'}
>>> print(D)
{1: 'one', 2: 'two', 3: 'three', 4: 'four'}
>>> D.clear()
>>> print(D)
{}
```

get() method

The get() method returns a value for the given key. If key is not available, then returns default value None.

Syntax:

Following is the syntax for **get()** method:

```
    dict.get(key, default = None)
```

key: This is the key to be searched in the dictionary.

default: This is the value to be returned in case the key does not exist.

This method returns a value for the given key. If key is not available, then returns default value None.

For example,

```
>>> D1={'sun':'Sunday','mon':'Monday','tue':'Tuesday','wed':'wednesday',
        'thu':'Thursday'}
>>> D1.get ('wed')
'Wednesday'
>>> print(D1.get('fri'))
None
```

will return None as key does not exist

You can specify your own message also :

Here we have specified our own message but if the key exists, then it will return its value, not customized.

```
>>> D1.get('fri','never')
'never'
>>> D1.get('mon','never')
'Monday'
```

Note: items() returns a list of tuples having key-value pairs.

items()

It returns the content of dictionary as a list of tuples having key-value pairs.

Syntax:

```
D.items()  #D dictionary
```

Note: It is different from print command because in print command, dictionary values are written in {} and key-value pair is separated by ':'

For example,

```
>>> D={'sun':'Sunday','mon':'Monday','tue':'Tuesday','wed':'Wednesday',
      'thu':'Thursday'}
>>> D.items()
dict_items([('sun', 'Sunday'), ('mon', 'Monday'), ('tue', 'Tuesday'),
           ('wed', 'Wednesday'), ('thu', 'Thursday')])
```

keys()

It returns a list of the key values in a dictionary.

Syntax:

```
D.keys()  #D dictionary
```

For example,

```
>>> D={'sun':'Sunday','mon':'Monday','tue':'Tuesday','wed' :'Wednesday',
      'thu':'Thursday'}
>>> D.keys()
dict_items(['sun', 'mon', 'tue', 'wed', 'thu'])
```

values()

It returns a list of values from key-value pairs in a dictionary.

Syntax:

```
D.values()  #D dictionary
>>> D={'sun':'Sunday','mon':'Monday','tue':'Tuesday','wed':'Wednesday',
      'thu':'Thursday'}
>>> D.values()
dict_values([('Sunday', 'Monday', 'Tuesday', 'Wednesday', 'Thursday'])
```

copy()

We cannot copy a dictionary by using assignment ('=') operator as it will create a reference to the same dictionary variable and modifications will also be reflected in both dictionaries. We can use built-in function copy() to create a new dictionary and modifications done on base dictionary will not be reflected.

Syntax:

```
Dict_new=Dict_old.copy()
```

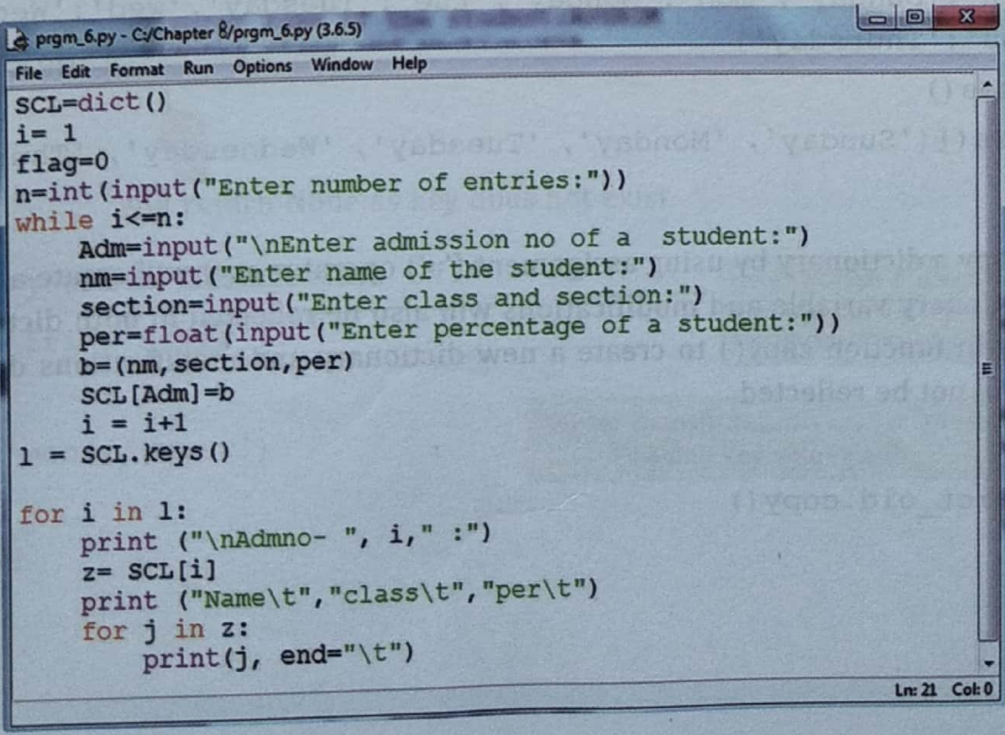
For example,

```
>>> d1={"January":31,"February":28 }
>>> d2=d1                                   Assigning a dictionary
>>> d1
{'January': 31, 'February': 28 }
>>> d2
{'January': 31, 'February': 28 }
>>> d2["March"]=31
>>> d2
{'January': 31, 'February': 28, 'March': 31}     Changes made to d2 is
>>> d1                                           reflected in d1
{'January': 31, 'February': 28, 'March': 31 }
>>> d3=d1.copy()                            Copying a dictionary
>>> d3
{'January': 31, 'February': 28, 'March': 31}
>>> d1["April"]=30                               Changes made in d1 is
>>> d3                                           not reflected in d3
{'January': 31, 'February': 28, 'March': 31 }
>>> d1
{'January': 31, 'February': 28, 'March': 31, 'April': 30}
```

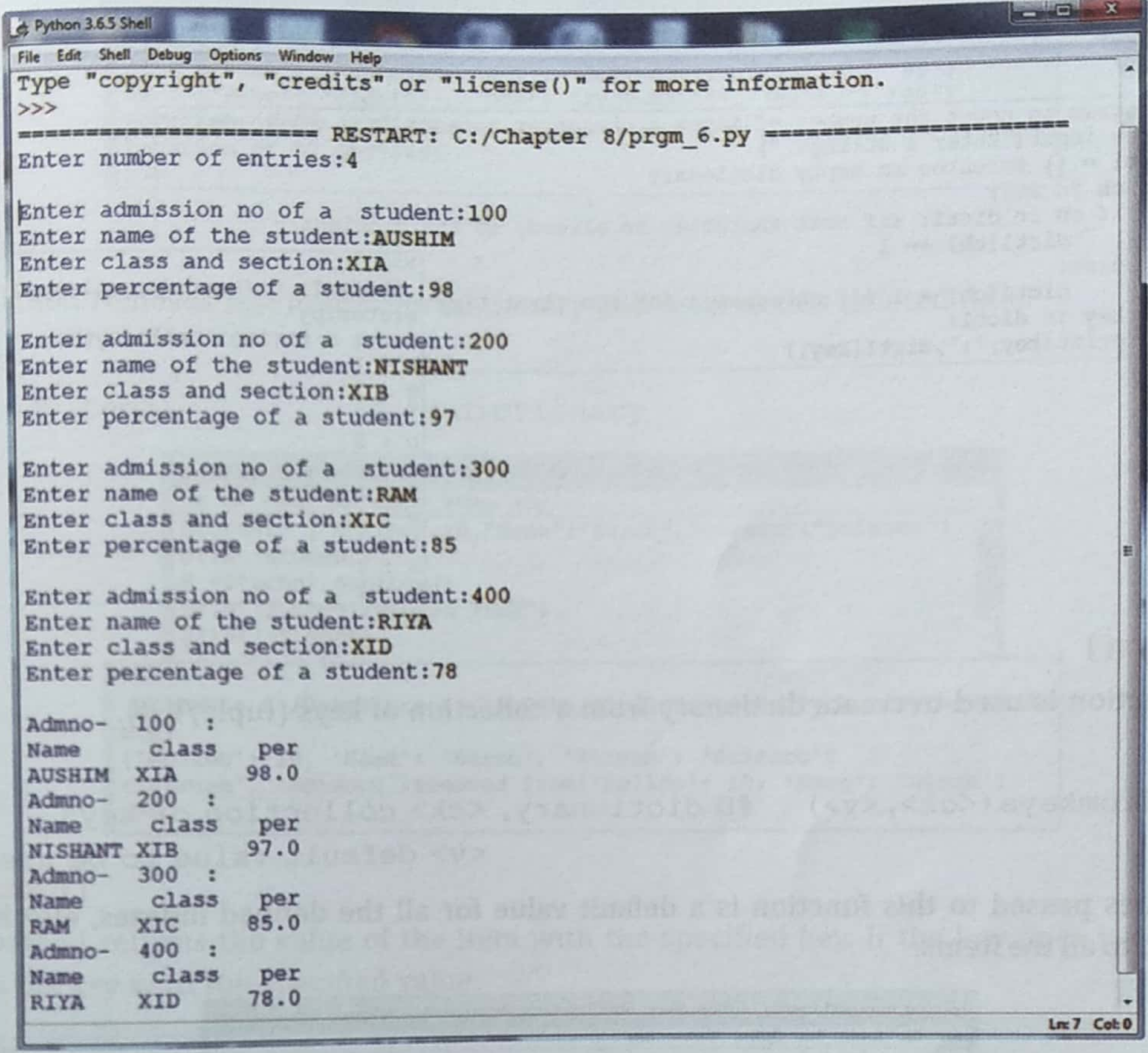
Practical Implementation–16

WAP to store students' information like admission number, roll number, name and marks in a dictionary and display information on the basis of admission number.

Code:


prgm_6.py - C:/Chapter 8/prgm_6.py (3.6.5)
File Edit Format Run Options Window Help
Ln: 21 Col: 0

```
SCL=dict()
i= 1
flag=0
n=int(input("Enter number of entries:"))
while i<=n:
    Adm=input("\nEnter admission no of a  student:")
    nm=input("Enter name of the student:")
    section=input("Enter class and section:")
    per=float(input("Enter percentage of a student:"))
    b=(nm,section,per)
    SCL[Adm]=b
    i = i+1
l = SCL.keys()

for i in l:
    print ("\nAdmno- ", i," :")
    z= SCL[i]
    print ("Name\t","class\t","per\t")
    for j in z:
        print(j, end="\t")
```



```
Type "copyright", "credits" or "license()" for more information.
>>>
====================== RESTART: C:/Chapter 8/prgm_6.py ======================
Enter number of entries:4

Enter admission no of a  student:100
Enter name of the student:AUSHIM
Enter class and section:XIA
Enter percentage of a student:98

Enter admission no of a  student:200
Enter name of the student:NISHANT
Enter class and section:XIB
Enter percentage of a student:97

Enter admission no of a  student:300
Enter name of the student:RAM
Enter class and section:XIC
Enter percentage of a student:85

Enter admission no of a  student:400
Enter name of the student:RIYA
Enter class and section:XID
Enter percentage of a student:78

Admno-  100  :
Name     class    per
AUSHIM  XIA     98.0
Admno-  200  :
Name     class    per
NISHANT XIB     97.0
Admno-  300  :
Name     class    per
RAM     XIC     85.0
Admno-  400  :
Name     class    per
RIYA    XID     78.0
```

ractical Implementation–17

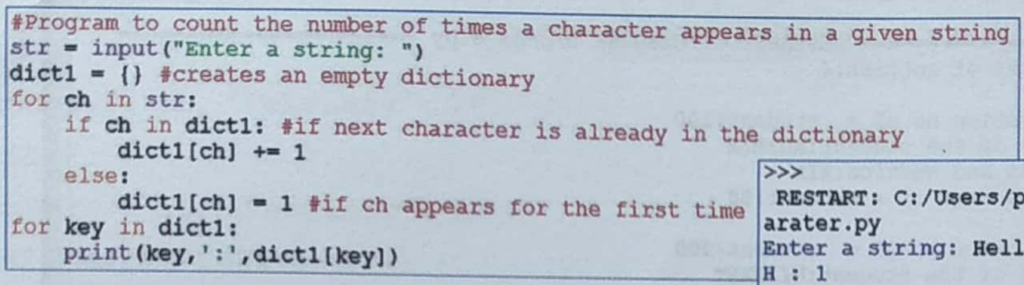
rite a program to enter names of employees and their salaries as input and store them in a dictionary.

```
#Program to create a dictionary with names of the employees and their salary
num = int(input("Enter the number of employees whose data to be stored: "))
count = 1
employee = dict() #create an empty dictionary
while count <= num:
    name = input("Enter the name of the Employee: ")
    salary = int(input("Enter the salary: "))
    employee[name] = salary
    count += 1
print("\n\nEMPLOYEE_NAME\tSALARY")
for k in employee:
    print(k,'\t\t',employee[k])
```

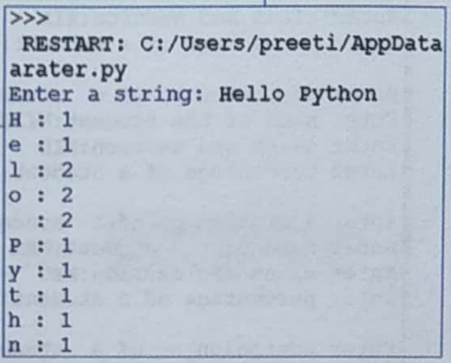
```
>>>
 RESTART: C:/Users/preeti/AppData/Local/Programs/Python/P
t.py
Enter the number of employees whose data to be stored: 2
Enter the name of the Employee: Teena
Enter the salary: 45000
Enter the name of the Employee: Ritu
Enter the salary: 55000


EMPLOYEE_NAME    SALARY
Teena             45000
Ritu              55000
```

Practical Implementation–18

Write a program to count the number of times a character appears in a given string using a dictionary.



```
#Program to count the number of times a character appears in a given string
str = input("Enter a string: ")
dict1 = {} #creates an empty dictionary
for ch in str:
    if ch in dict1: #if next character is already in the dictionary
        dict1[ch] += 1
    else:
        dict1[ch] = 1 #if ch appears for the first time
for key in dict1:
    print(key,':',dict1[key])
```



```
>>>
 RESTART: C:/Users/preeti/AppData
arater.py
Enter a string: Hello Python
H : 1
e : 1
l : 2
o : 2
  : 2
P : 1
y : 1
t : 1
h : 1
n : 1
```

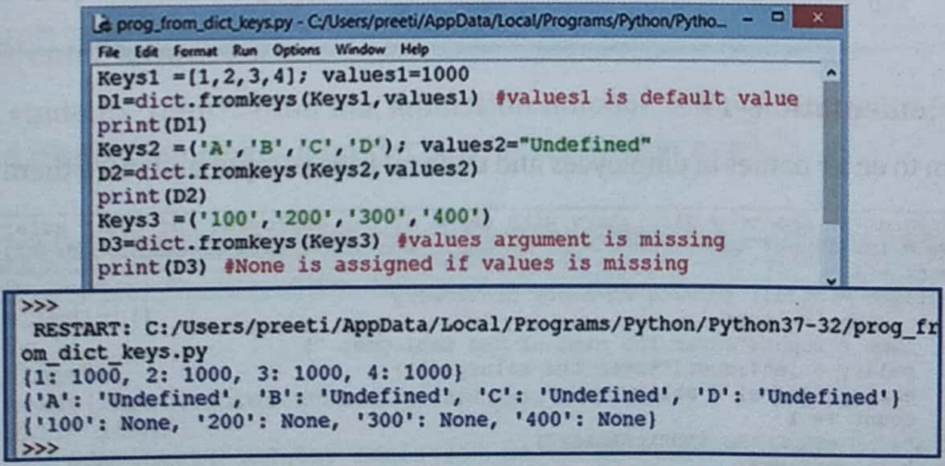
fromkeys()

This function is used to create dictionary from a collection of keys (tuple/list).

Syntax:

```
    D.fromkeys(<ck>,<v>)    #D dictionary, <ck> collection of keys
                                  <v> default value to be assigned
```

The values passed to this function is a default value for all the defined indexes, else None is assigned to all the items.



```
Keys1 =[1,2,3,4]; values1=1000
D1=dict.fromkeys(Keys1,values1) #values1 is default value
print(D1)
Keys2 =('A','B','C','D'); values2="Undefined"
D2=dict.fromkeys(Keys2,values2)
print(D2)
Keys3 =('100','200','300','400')
D3=dict.fromkeys(Keys3) #values argument is missing
print(D3) #None is assigned if values is missing
```

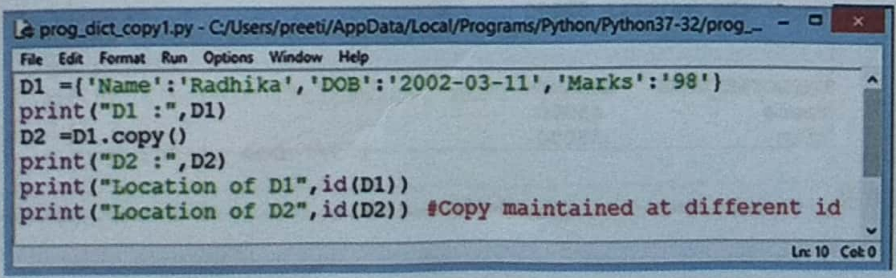
```
>>>
 RESTART: C:/Users/preeti/AppData/Local/Programs/Python/Python37-32/prog_fr
om_dict_keys.py
{1: 1000, 2: 1000, 3: 1000, 4: 1000}
{'A': 'Undefined', 'B': 'Undefined', 'C': 'Undefined', 'D': 'Undefined'}
{'100': None, '200': None, '300': None, '400': None}
>>>
```

copy()

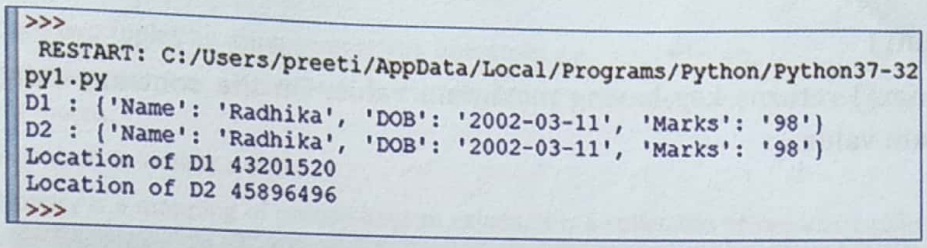
This method creates a copy of the dictionary.

Syntax:

```
    D.copy() #D dictionary
```



```
D1 ={'Name':'Radhika','DOB':'2002-03-11','Marks':'98'}
print("D1 :",D1)
D2 =D1.copy()
print("D2 :",D2)
print("Location of D1",id(D1))
print("Location of D2",id(D2)) #Copy maintained at different id
```



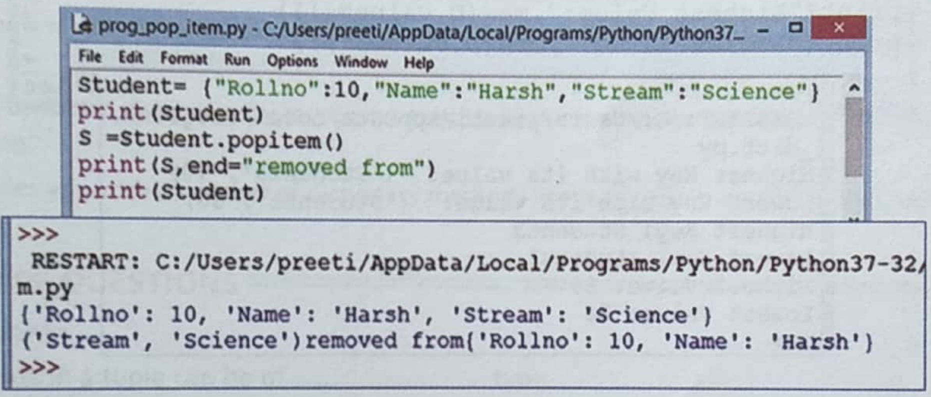
```
>>>
 RESTART: C:/Users/preeti/AppData/Local/Programs/Python/Python37-32
py1.py
D1 : {'Name': 'Radhika', 'DOB': '2002-03-11', 'Marks': '98'}
D2 : {'Name': 'Radhika', 'DOB': '2002-03-11', 'Marks': '98'}
Location of D1 43201520
Location of D2 45896496
>>>
```

popitem()

This method removes last item from dictionary and returns this deleted item.

Syntax:

```
    D.popitem()                    #D dictionary
```


prog_pop_item.py - C:/Users/preeti/AppData/Local/Programs/Python/Python37...
File Edit Format Run Options Window Help

```
Student= {"Rollno":10,"Name":"Harsh","Stream":"Science"}
print(Student)
S =Student.popitem()
print(S,end="removed from")
print(Student)
```

```
>>>
 RESTART: C:/Users/preeti/AppData/Local/Programs/Python/Python37-32/
m.py
{'Rollno': 10, 'Name': 'Harsh', 'Stream': 'Science'}
('Stream', 'Science')removed from{'Rollno': 10, 'Name': 'Harsh'}
>>>
```

setdefault()

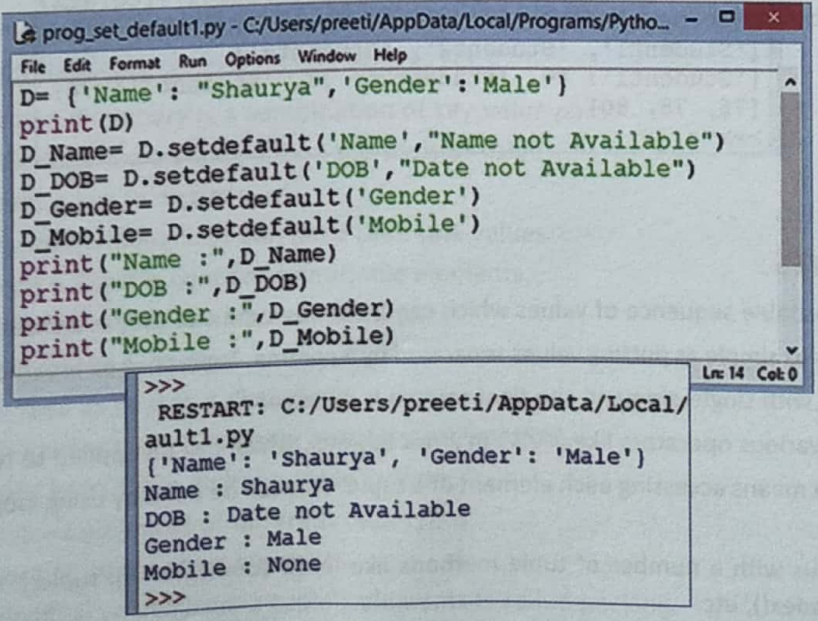
This method returns the value of the item with the specified key. If the key does not exist, it inserts the key with the specified value.

Syntax:

```
    <Value>=<Dict>.setdefault(<key>,<default_value>)
```

The setdefault() function returns:

- Value of the key, if it is in the dictionary
- None, if key is not in the dictionary and default_value is not specified
- Default_value, if key is not in the dictionary and default_value is specified


prog_set_default1.py - C:/Users/preeti/AppData/Local/Programs/Pytho...
File Edit Format Run Options Window Help
Ln: 14 Col: 0

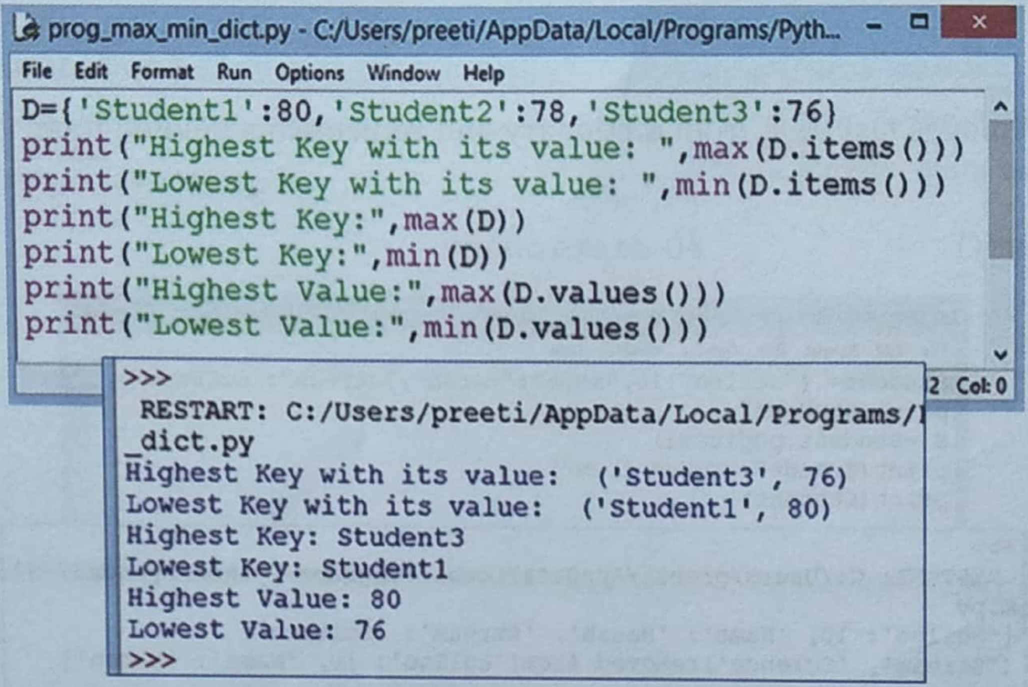
```
D= {'Name': "Shaurya",'Gender':'Male'}
print(D)
D_Name= D.setdefault('Name',"Name not Available")
D_DOB= D.setdefault('DOB',"Date not Available")
D_Gender= D.setdefault('Gender')
D_Mobile= D.setdefault('Mobile')
print("Name :",D_Name)
print("DOB :",D_DOB)
print("Gender :",D_Gender)
print("Mobile :",D_Mobile)
```

```
>>>
 RESTART: C:/Users/preeti/AppData/Local/
ault1.py
{'Name': 'Shaurya', 'Gender': 'Male'}
Name : Shaurya
DOB : Date not Available
Gender : Male
Mobile : None
>>>
```

max() and min()

The method max() returns key having maximum value. On the contrary, min() returns key having minimum value.

Syntax:

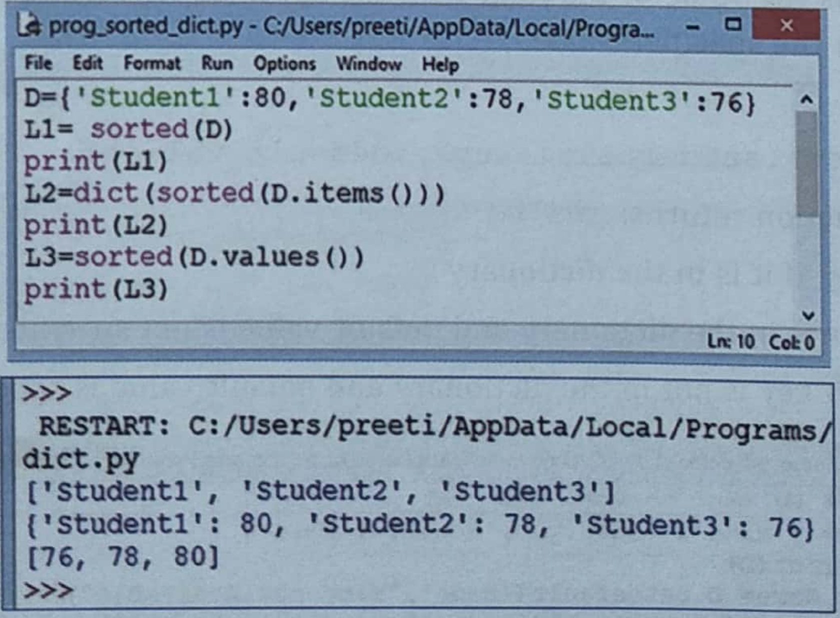

prog_max_min_dict.py - C:/Users/preeti/AppData/Local/Programs/Pyth...
File Edit Format Run Options Window Help
2 Col: 0

```
D={'Student1':80,'Student2':78,'Student3':76}
print("Highest Key with its value: ",max(D.items()))
print("Lowest Key with its value: ",min(D.items()))
print("Highest Key:",max(D))
print("Lowest Key:",min(D))
print("Highest Value:",max(D.values()))
print("Lowest Value:",min(D.values()))
```

```
>>>
 RESTART: C:/Users/preeti/AppData/Local/Programs/
_dict.py
Highest Key with its value:  ('Student3', 76)
Lowest Key with its value:  ('Student1', 80)
Highest Key: Student3
Lowest Key: Student1
Highest Value: 80
Lowest Value: 76
>>>
```

sorted()

This method sorts the elements of a dictionary by its key or value.


prog_sorted_dict.py - C:/Users/preeti/AppData/Local/Progra...
File Edit Format Run Options Window Help
Ln: 10 Col: 0

```
D={'Student1':80,'Student2':78,'Student3':76}
L1= sorted(D)
print(L1)
L2=dict(sorted(D.items()))
print(L2)
L3=sorted(D.values())
print(L3)
```

```
>>>
 RESTART: C:/Users/preeti/AppData/Local/Programs/
dict.py
['Student1', 'Student2', 'Student3']
{'Student1': 80, 'Student2': 78, 'Student3': 76}
[76, 78, 80]
>>>
```