for loop

for loop iterate or read keys from dictionary.

Syntax:

```
for variable in dictionary-name:
        statement-1
        statement-2
```

Example:

```
person={'naresh':45,
        'suresh':50,
        'ramesh':70,
        'kishore':40,
        'rajesh':50}

# Reading with for loop
for name in person:
   print(name)

# Reading key and value
for name in person:
   print(f'{name}--->{person[name]}')
```

Output

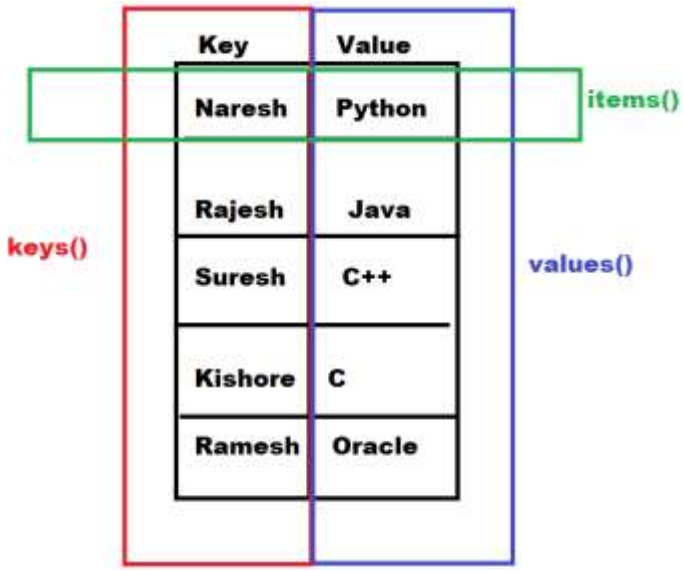
```
naresh
suresh
ramesh
kishore
rajesh
naresh--->45
suresh--->50
ramesh--->70
kishore--->40
rajesh--->50
```

Dictionary Methods

get(key, default=None)
Return the value for key if key is in the dictionary, else default.
If default is not given, it defaults to None, so that this method never raises a KeyError.

```
>>> dict1={1:10,2:20,3:30,4:40,5:50}
>>> print(dict1)
{1: 10, 2: 20, 3: 30, 4: 40, 5: 50}
>>> dict1[2]
20
>>> dict1[3]
30
>>> dict1[8]
Traceback (most recent call last):
  File "<pyshell#4>", line 1, in <module>
    dict1[8]
KeyError: 8
>>> dict1.get(2)
20
>>> dict1.get(5)
50
>>> dict1.get(8)
>>> dict1.get(8,0)
0
>>> dict1.get(8,100)
100
```

Dictionary View objects
The objects returned by dict.keys(), dict.values() and dict.items() are view objects. They provide a dynamic view on the dictionary's entries, which means that when the dictionary changes, the view reflects these changes.



Example:

```
emp_data={'naresh':50000,
          'suresh':54000,
          'ramesh':65000,
          'kishore':76000}


names=emp_data.keys()
print(names)
salaries=emp_data.values()
print(salaries)
employees=emp_data.items()
print(employees)

for name,sal in employees:
    print(f'{name}--->{sal}')
```

Output

```
dict_keys(['naresh', 'suresh', 'ramesh', 'kishore'])
dict_values([50000, 54000, 65000, 76000])
dict_items([('naresh', 50000), ('suresh', 54000), ('ramesh', 65000),
('kishore', 76000)])
naresh--->50000
suresh--->54000
```

ramesh--->65000
kishore--->76000

reversed(dictview)
Return a reverse iterator over the keys, values or items of the dictionary. The view will be iterated in reverse order of the insertion.

Example:

```
emp_data={'naresh':50000,
        'suresh':54000,
        'ramesh':65000,
        'kishore':76000}


names=emp_data.keys()
salaries=emp_data.values()
employees=emp_data.items()

rev_names=reversed(names)
rev_salaries=reversed(salaries)
rev_employees=reversed(employees)

for name in rev_names:
    print(name)

for sal in rev_salaries:
    print(sal)

for name,sal in rev_employees:
    print(name,sal)
```

Output

```
kishore
ramesh
suresh
naresh
```

76000
65000
54000
50000
kishore 76000
ramesh 65000
suresh 54000
naresh 50000

dictview.mapping
Return a types.MappingProxyType that wraps the original dictionary to which the view refers.

```
>> d1=dict(zip(range(1,5),range(10,60,10)))
>>> print(d1)
{1: 10, 2: 20, 3: 30, 4: 40}
>>> dv1=d1.items()
>>> print(dv1)
dict_items([(1, 10), (2, 20), (3, 30), (4, 40)])
>>> dv1.mapping
mappingproxy({1: 10, 2: 20, 3: 30, 4: 40})
>>> dv1.mapping.get(1)
10
>>> dv1.mapping.values()
dict_values([10, 20, 30, 40])
```

iter(dictionary)
This function returns iterator object, which iterates keys from dictionary.

```
>>> d1={1:10,2:20,3:30,4:40,5:50}
>>> a=iter(d1)
>>> value1=next(a)
>>> print(value1)
1
>>> value2=next(a)
>>> print(value2)
```

```
2
>>> for k in a:
...     print(k)
...
...
3
4
5
```

setdefault(key, default=None)
If key is in the dictionary, return its value. If not, insert key with a value of default and return default. default defaults to None.

```
>> dict1={}
>>> x=dict1.setdefault(1)
>>> print(dict1)
{1: None}
>>> print(x)
None
>>> y=dict1.setdefault(2,20)
>>> print(dict1)
{1: None, 2: 20}
>>> print(y)
20
>>> z=dict1.setdefault(2)
>>> print(z)
20
```

Mutable operations of dictionary
After creating dictionary changes can be done.

Adding items inside dictionary
Syntax-1: dictionary-name[key]=value → Add one item if key not exists. If key exists it replace value of key.
Syntax-2: dictionary-name.update(iterable) → Add more than one item inside dictionary

Example:

```
# Write a program to read scores of n players each player having name,score

n=int(input("Enter how many players ?"))
scores={}

for i in range(n):
    name=input("PlayerName :")
    score=int(input("Player Score :"))
    scores[name]=score

total=0
for name,score in scores.items():
    print(f'{name}-->{score}')
    total+=score

print(f'Total Score {total}')
```

Output

```
Enter how many players ?3
PlayerName :rohit
Player Score :60
PlayerName :shubham
Player Score :40
PlayerName :virat
Player Score :10
rohit-->60
shubham-->40
virat-->10
Total Score 110
```

Example:

```
# write a program to read marks details of n students
# each student is having name and 2 subject marks
```

```
n=int(input("Enter how many students?"))
students={}

for i in range(n):
    name=input("Enter Name :")
    marks=[int(input("Enter Marks")) for j in range(2)]
    students[name]=marks

for name,marks in students.items():
    tot=sum(marks)
    avg=tot/2
    print(f'{name}---------->{marks}-------->{tot}------>{avg:.2f}')
```

Output

```
Enter how many students?2
Enter Name :suresh
Enter Marks60
Enter Marks70
Enter Name :ramesh
Enter Marks90
Enter Marks98
suresh---------->[60, 70]-------->130------>65.00
ramesh---------->[90, 98]-------->188------>94.00
```

Example of adding and updating more than one item:

```
>>> d1={1:10,2:20,3:30}
>>> print(d1)
{1: 10, 2: 20, 3: 30}
>>> d2={4:40,5:50,6:60}
>>> d1.update(d2)
>>> print(d1)
{1: 10, 2: 20, 3: 30, 4: 40, 5: 50, 6: 60}
>>> d3={1:100,2:200,7:70,8:80}
>>> d1.update(d3)
>>> print(d1)
{1: 100, 2: 200, 3: 30, 4: 40, 5: 50, 6: 60, 7: 70, 8: 80}
```

Replacing or updating one value
Syntax1: dictionary-name[key]=value
If key exists within dictionary, it update value
If key not exists within dictionary, it add key and value

Example:

```
>>> d1={1:10,2:20,3:30}
>>> print(d1)
{1: 10, 2: 20, 3: 30}
>>> d1[1]=100
>>> print(d1)
{1: 100, 2: 20, 3: 30}
```

Example

```
users_dict={'naresh':'n123',
            'suresh':'m789',
            'ramesh':'ram@45'}

print("Changing Password of User")
name=input("UserName :")
opwd=input("Old Password :")
if name in users_dict and opwd==users_dict[name]:
    npwd=input("New Password :")
    users_dict[name]=npwd
    print("Password Updated...")
    print(users_dict)
else:
    print("Invalid username or password")
```

Output

```
Changing Password of User
UserName :naresh
Old Password :n123
New Password :x123
Password Updated...
{'naresh': 'x123', 'suresh': 'm789', 'ramesh': 'ram@45'}
```