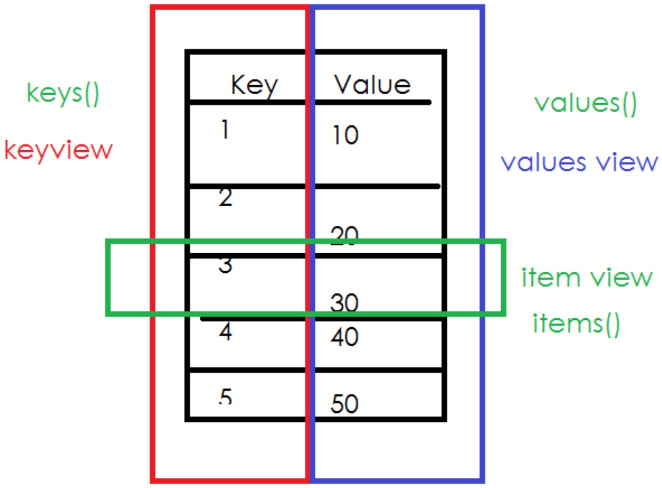
Using for loop

for loop read keys from dictionary.

```
>>> dict1={1:10,2:20,3:30,4:40,5:50}
>>> for x in dict1:
...     print(x)
...
...
1
2
3
4
5
>>> for x in dict1:
...     print(x,dict1[x])
...
...
1 10
2 20
3 30
4 40
5 50
```

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.



Key	Value
1	10
2	20
3	30
4	40
5	50

```
>>> emp_dict={'naresh':50000,
         'suresh':65000,
         'kishore':35000,
         'ramesh':65000}
>>> names=emp_dict.keys()
>>> print(names)
dict_keys(['naresh', 'suresh', 'kishore', 'ramesh'])
>>> for name in names:
    print(name)


naresh
suresh
kishore
ramesh
>>> emp_dict['kiran']=90000
>>> print(emp_dict)
{'naresh': 50000, 'suresh': 65000, 'kishore': 35000, 'ramesh': 65000, 'kiran':
90000}
>>> print(names)
dict_keys(['naresh', 'suresh', 'kishore', 'ramesh', 'kiran'])
>>> salaries=emp_dict.values()
>>> print(salaries)
dict_values([50000, 65000, 35000, 65000, 90000])
>>> for salary in salaries:
...     print(salary)
```

```
...
...
50000
65000
35000
65000
90000
>>> emp_dict['amar']=40000
>>> print(emp_dict)
{'naresh': 50000, 'suresh': 65000, 'kishore': 35000, 'ramesh': 65000, 'kiran':
90000, 'amar': 40000}
>>> print(salaries)
dict_values([50000, 65000, 35000, 65000, 90000, 40000])
>>> employees=emp_dict.items()
>>> print(employees)
dict_items([('naresh', 50000), ('suresh', 65000), ('kishore', 35000),
('ramesh', 65000), ('kiran', 90000), ('amar', 40000)])
>>> for name,sal in employees:
...     print(name,sal)
...
...
naresh 50000
suresh 65000
kishore 35000
ramesh 65000
kiran 90000
```

get(key[, default])
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.

```
sales_dict={2010:450000,
...             2011:540000,
...             2012:670000,
...             2013:340000}
>>> sales=sales_dict[2010]
>>> print(sales)
450000
>>> sales=sales_dict[2013]
>>> print(sales)
```

```
340000
>>> sales=sales_dict[2015]
Traceback (most recent call last):
  File "<pyshell#33>", line 1, in <module>
    sales=sales_dict[2015]
KeyError: 2015
>>> sales=sales_dict.get(2015)
>>> print(sales)
None
>>> sales=sales_dict.get(2010)
>>> print(sales)
450000
```

setdefault(*key*[, *default*])
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={1:10,2:20,3:30}
>>> print(dict1)
{1: 10, 2: 20, 3: 30}
>>> print(dict1[1])
10
>>> print(dict1[2])
20
>>> print(dict1[3])
30
>>> value=dict1.setdefault(2,100)
>>> print(value)
20
>>> value=dict1.setdefault(4,100)
>>> print(value)
100
>>> print(dict1)
{1: 10, 2: 20, 3: 30, 4: 100}
>>> value=dict1.setdefault(5)
>>> print(value)
None
>>> print(dict1)
{1: 10, 2: 20, 3: 30, 4: 100, 5: None}
```

del d[key]
Remove d[key] from *d*. Raises a KeyError if *key* is not in the map.

```
>>> del dict1[1]
>>> print(dict1)
{2: 20, 3: 30, 4: 40, 5: 50}
>>> del dict1[5]
>>> print(dict1)
{2: 20, 3: 30, 4: 40}
>>> del dict1[1]
Traceback (most recent call last):
  File "<pyshell#61>", line 1, in <module>
    del dict1[1]
KeyError: 1
```

clear()
Remove all items from the dictionary.

```
>>> d1={'naresh':40,'suresh':50}
>>> print(d1)
{'naresh': 40, 'suresh': 50}
>>> d1.clear()
>>> print(d1)
{}
```

pop(key[, default])
If key is in the dictionary, remove it and return its value, else return default. If default is not given and key is not in the dictionary, a KeyError is raised.
popitem()
Remove and return a (key, value) pair from the dictionary. Pairs are returned in LIFO order.

```
>>> student_dict={'naresh':'A',
              'suresh':'B',
              'kishore':'A',
              'ramesh':'C'}
>>> print(student_dict)
{'naresh': 'A', 'suresh': 'B', 'kishore': 'A', 'ramesh': 'C'}
>>> grade=student_dict.pop('suresh')
>>> print(grade)
```

```
B
>>> print(student_dict)
{'naresh': 'A', 'kishore': 'A', 'ramesh': 'C'}
>>> grade=student_dict.pop('suresh')
Traceback (most recent call last):
  File "<pyshell#74>", line 1, in <module>
    grade=student_dict.pop('suresh')
KeyError: 'suresh'
>>> grade=student_dict.pop('suresh',None)
>>> print(grade)
None
>>> i1=student_dict.popitem()
>>> print(i1)
('ramesh', 'C')
>>> print(student_dict)
{'naresh': 'A', 'kishore': 'A'}
```

update([*other*])

Update the dictionary with the key/value pairs from *other*, overwriting existing keys. Return None.

```
>>> d1={1:10,2:20,3:30,4:40}
>>> d2={5:50,6:60,1:99,3:88}
>>> print(d1)
{1: 10, 2: 20, 3: 30, 4: 40}
>>> print(d2)
{5: 50, 6: 60, 1: 99, 3: 88}
>>> d1.update(d2)
>>> print(d1)
{1: 99, 2: 20, 3: 88, 4: 40, 5: 50, 6: 60}
```

reversed(d)

Return a reverse iterator over the keys of the dictionary. This is a shortcut for reversed(d.keys()).
New in version 3.8.

```
>>> for k in d1.keys():
        print(k)
```

```
1
2
3
4
5
>>> for k in reversed(d1):
          print(k)


5
4
3
2
1
>>> for k in reversed(d1):
...     print(k,d1[k])
...
...
5 50
4 40
3 30
2 20
1 10
```

d | other

Create a new dictionary with the merged keys and values of *d* and *other*, which must both be dictionaries. The values of *other* take priority when *d* and *other* share keys.
New in version 3.9.

```
>>> d1={1:10,2:20,3:30}
>>> d2={4:40,5:50,1:99}
>>> d3=d1|d2
>>> print(d1)
{1: 10, 2: 20, 3: 30}
>>> print(d2)
{4: 40, 5: 50, 1: 99}
>>> print(d3)
{1: 99, 2: 20, 3: 30, 4: 40, 5: 50}
```

d |= other
Update the dictionary *d* with keys and values from *other*, which may be either a mapping or an iterable of key/value pairs. The values of *other* take priority when *d* and *other* share keys.
New in version 3.9.

```
>>> d1={1:10,2:20,3:30}
>>> d2={4:40,5:50,1:99}
>>> d1|=d2
>>> print(d1)
{1: 99, 2: 20, 3: 30, 4: 40, 5: 50}
```

Example:

```
# shoping cart

cart={}
while True:
    print("1.Add Product")
    print("2.Update Product")
    print("3.Delete Product")
    print("4.Search")
    print("5.View Cart")
    print("6.Exit")
    opt=int(input("enter your option"))
    if opt==1:
        pname=input("Product Name")
        if pname in cart:
            print(pname,"exists in cart")
        else:
            qty=int(input("Enter Qty"))
            cart[pname]=qty
            print("product added...")
    elif opt==2:
        pname=input("Product Name")
        if pname in cart:
            qty=int(input("Enter Qty"))
            cart[pname]=qty
            print("Qty updated ....")
        else:
            print(pname,"not exists in cart")
    elif opt==3:
        pname=input("Product Name")
        if pname in cart:
            del cart[pname]
            print("product deleted from cart...")
        else:
            print(pname,"not exists within cart")
    elif opt==4:
        pname=input("Product Name")
        if pname in cart:
            qty=cart[pname]
            print("Qty is ",qty)
        else:
            print("product not exists")
    elif opt==5:
        for pname,qty in cart.items():
            print(pname,qty)
    elif opt==6:
        break
```

Output:

```
1.Add Product
2.Update Product
3.Delete Product
4.Search
5.View Cart
6.Exit
enter your option1
Product Namemouse
Enter Qty10
product added...
1.Add Product
2.Update Product
3.Delete Product
4.Search
5.View Cart
6.Exit
```

```
enter your option1
Product Namekeyboard
Enter Qty5
product added...
1.Add Product
2.Update Product
3.Delete Product
4.Search
5.View Cart
6.Exit
enter your option5
mouse 10
keyboard 5
```

Dictionary Comprehension