

Dictionary is having 3 view objects

1. key view object
2. values view object
3. items view object

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
keys() Key View Object	Naresh	9988766554
	Suresh	8877655443
	Ramesh	9988788987
	Kishore	5676589898
	Raman	9988988998
		values() Values View Object
		items() items view object it returns key and value inside tuple

contacts

Example:

```
contacts={'naresh':998878899,  
         'suresh':556676677,  
         'ramesh':556778975,  
         'kishore':776566784}
```

```
names=contacts.keys()  
print(names)  
for name in names:  
    print(name)
```

```
values=contacts.values()  
print(values)  
for value in values:  
    print(value)
```

```
items=contacts.items()
print(items)
for i in items:
    print(i[0],i[1])
```

Output

```
dict_keys(['naresh', 'suresh', 'ramesh', 'kishore'])
naresh
suresh
ramesh
kishore
dict_values([998878899, 556676677, 556778975, 776566784])
998878899
556676677
556778975
776566784
dict_items([('naresh', 998878899), ('suresh', 556676677), ('ramesh',
556778975), ('kishore', 776566784)])
naresh 998878899
suresh 556676677
ramesh 556778975
kishore 776566784
```

setdefault()

setdefault method of dictionary performs 2 operations

1. reading value if key exists
2. adding key and value if key not exists

Syntax:

variable-name=dictionary-name.setdefault(key,value=None)

Example:

```
dict1={1:10,2:20,3:30,4:40,5:50}
value1=dict1.setdefault(1)
print(value1)
value2=dict1.setdefault(5)
```

```
print(value2)
value3=dict1.setdefault(7)
print(dict1,value3,sep="\n")
value4=dict1.setdefault(8,80)
print(value4)
```

Output

```
10
50
{1: 10, 2: 20, 3: 30, 4: 40, 5: 50, 7: None}
None
80
```

reversed()

This function returns reversed iterator on dictionary keys.
This iterator reads keys from dictionary in reverse order (bottom to top)

Example:

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

```
for k in reversed(dict1):
    print(k,dict1[k])
```

Output

```
1 10
2 20
3 30
4 40
5 50
5 50
4 40
3 30
2 20
1 10
```

Mutable Operations of dictionary

Adding item within dictionary

Dictionary provides 2 approaches for adding items.

1. Adding one item
2. Adding multiple items

Syntax1: dictionary-name[key]=value

Syntax2: dictionary-name.update(iterable)

Example:

```
>>> dict1={}
>>> print(dict1)
{}
>>> dict1['naresh']=50
>>> dict1['ramesh']=40
>>> print(dict1)
{'naresh': 50, 'ramesh': 40}
>>> dict1['naresh']=20
>>> print(dict1)
{'naresh': 20, 'ramesh': 40}
>>> d1={1:10,2:20}
>>> print(d1)
{1: 10, 2: 20}
>>> d1.update({3:30,4:40,5:50})
>>> print(d1)
{1: 10, 2: 20, 3: 30, 4: 40, 5: 50}
```

Write a program to create dictionary with

Marks details of n students

```
marks={}
```

```
n=int(input("How many students?"))
```

```
for i in range(n):
```

```
    name=input("Name :")
```

```
    sub=[]
```

```
    for j in range(3):
```

```
        s=int(input("Marks :"))
```

```

        sub.append(s)
marks[name]=sub

for t in marks.items():
    name,sub=t
    tot=sum(sub)
    avg=tot/3
    result="PASS" if sub[0]>=40 and sub[1]>=40 and sub[2]>=40 else
"FAIL"
    print(f'{name}\t{sub}\t{tot}\t{avg:.2f}\t{result}')

```

Output

```

How many students?2
Name :naresh
Marks :60
Marks :70
Marks :80
Name :kishore
Marks :30
Marks :60
Marks :70
naresh      [60, 70, 80]210   70.00PASS
kishore     [30, 60, 70]160   53.33FAIL

```

Replacing or updating values

Replacing or updating values done with same syntax of adding
If key exists, the add syntax perform updating of value

```

dict1=dict(zip(range(1,6),range(10,60,10)))
print(dict1)
{1: 10, 2: 20, 3: 30, 4: 40, 5: 50}
dict1[1]=99
print(dict1)
{1: 99, 2: 20, 3: 30, 4: 40, 5: 50}
>>> dict1[5]=11
>>> print(dict1)
{1: 99, 2: 20, 3: 30, 4: 40, 5: 11}

```

```
>>> dict1[6]=60
>>> print(dict1)
{1: 99, 2: 20, 3: 30, 4: 40, 5: 11, 6: 60}
>>> dict1.update({1:10,2:22,3:33})
>>> print(dict1)
{1: 10, 2: 22, 3: 33, 4: 40, 5: 11, 6: 60}
```

Removing or deleting items from dictionary

Deleting items from dictionary is done in different ways

1. Using del keyword
2. Using clear method
3. Using pop method
4. Using popitem method

Using “del” keyword

Syntax: del dictionary-name[key]

If key exists within dictionary it deletes item

If key not exists within dictionary it raises KeyError

Example

```
courses={'python':6000,
         'java':2000,
         'oracle':1000,
         '.net':3000}
print(courses)
del courses['java']
print(courses)
del courses['.net']
print(courses)
if 'java' in courses:
    del courses['java']
else:
    print("invalid key")
```

Output

```
{'python': 6000, 'java': 2000, 'oracle': 1000, '.net': 3000}  
{'python': 6000, 'oracle': 1000, '.net': 3000}  
{'python': 6000, 'oracle': 1000}  
invalid key
```

using clear()

This method removes all the items from dictionary (OR) empty dictionary

```
>>> dict1=dict(zip(range(1,6),range(10,60,10)))  
>>> print(dict1)  
{1: 10, 2: 20, 3: 30, 4: 40, 5: 50}  
>>> dict1.clear()  
>>> print(dict1)  
{}
```

pop() method

This method performs 2 operations

1. Read value of given key
2. Remove item

variable-name=dictionary-name.pop(key,default)

if key exists, it returns value and remove item from dictionary
if key not exists, it returns default value, if default value not given it raises KeyError

```
>>> dict1=dict(zip(range(1,6),range(10,60,10)))  
>>> print(dict1)  
{1: 10, 2: 20, 3: 30, 4: 40, 5: 50}  
>>> v1=dict1.pop(1)  
>>> print(v1)  
10  
print(dict1)  
{2: 20, 3: 30, 4: 40, 5: 50}  
>>> v2=dict1.pop(5)  
>>> print(v2)
```

```

50
>>> print(v2)
50
>>> print(dict1)
{2: 20, 3: 30, 4: 40}
>>> v3=dict1.pop(1,0)
>>> print(v3)
0
>>> v4=dict1.pop(1)
Traceback (most recent call last):
  File "<pyshell#39>", line 1, in <module>
    v4=dict1.pop(1)
KeyError: 1

```

popitem()

using this method dictionary can be used as stack
 Stack is data structure which follows LIFO (Last In First Out)
 Popitem() perform 2 operations

1. Reading
2. Removing

Before removing last item, it return an item as tuple (key,value)

Syntax:

variable-name=dictionary-name.popitem()

```

>>> dict1=dict(zip("ABCDE",[10,20,30,40,50]))
>>> print(dict1)
{'A': 10, 'B': 20, 'C': 30, 'D': 40, 'E': 50}
>>> item1=dict1.popitem()
>>> print(item1)
('E', 50)
>>> print(dict1)
{'A': 10, 'B': 20, 'C': 30, 'D': 40}
>>> item2=dict1.popitem()
>>> print(item2)
('D', 40)
>>> print(dict1)

```



```
{'A': 10, 'B': 20, 'C': 30}  
>>>
```

Example:

Shoping Cart

```
cart={}
while True:
    print("1.Add Item")
    print("2.Update Item")
    print("3.Remove Item")
    print("4.View Items")
    print("5.Exit")
    opt=int(input("Enter Your Option :"))
    if opt==1:
        name=input("Item Name :")
        if name not in cart:
            qty=int(input("Qty :"))
            cart[name]=qty
            print("Item Added to Cart")
        else:
            print("This item exists in Cart")
    elif opt==2:
        name=input("Item Name :")
        if name in cart:
            qty=int(input("Updated Qty :"))
            cart[name]=qty
            print("Item Updated to Cart")
        else:
            print("Item not Exits")
    elif opt==3:
        name=input("Item Name :")
        if name in cart:
            del cart[name]
            print("Item Removed from Cart")
        else:
            print("Item not Exists")
```

```
elif opt==4:
    if len(cart)==0:
        print("Cart is empty")
    else:
        for name,qty in cart.items():
            print(f'{name}\t{qty}')
elif opt==5:
    break
```

Output

```
1.Add Item
2.Update Item
3.Remove Item
4.View Items
5.Exit
Enter Your Option :1
Item Name :Mouse
Qty :2
Item Added to Cart
1.Add Item
2.Update Item
3.Remove Item
4.View Items
5.Exit
Enter Your Option :1
Item Name :Monitor
```