Reading the content of dictionary using dictionary methods

1. get()
2. setdefault()
3. keys()
4. values()
5. items()

get() method

Syntax: dictionary-name.get(key,d=None)

This method returns value of given key
If key exists, it returns value
If key not exists, it returns default value(None)
The advantage of get method, it never raises any KeyError

```
>>> d1={1:10,2:20,3:30,4:40}
>>> print(d1)
{1: 10, 2: 20, 3: 30, 4: 40}
>>> x=d1.get(1)
>>> print(x)
10
>>> y=d1.get(4)
>>> print(y)
40
>>> z=d1.get(6)
>>> print(z)
None
>>> p=d1.get(6,60)
>>> print(p)
60
>>> q=d1.get(3,100)
>>> print(q)
30
>>> d1[1]
10
>>> d1[3]
30
>>> d1[6]
Traceback (most recent call last):
  File "<pyshell#14>", line 1, in <module>
    d1[6]
KeyError: 6
```

Example:

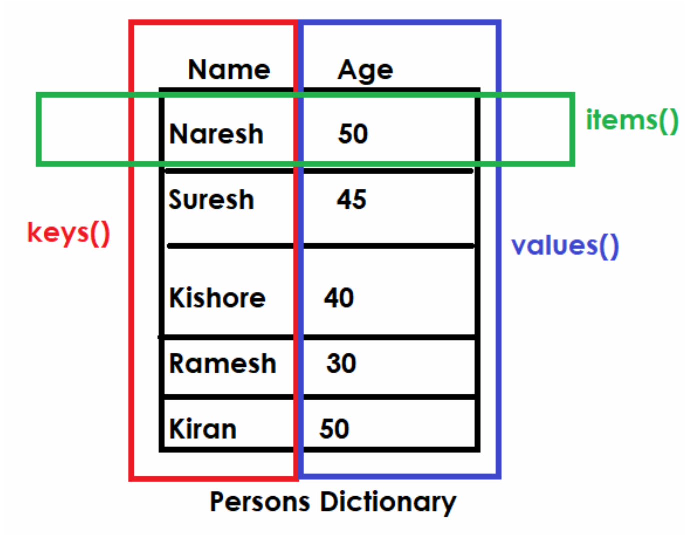
```
# Login Application

users={'nit':'n123',
       'ram':'r321',
       'raj':'r456'}

print("***Login***")
user=input("UserName :")
pwd=input("Password :")
if users.get(user)==pwd:
    print(f'{user} welcome')
else:
    print("invalid username or password")
```

Output

```
***Login***
UserName :nit
Password :n123
nit welcome
>>>
***Login***
UserName :nit
Password :xyz
invalid username or password
>>>
```

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.



```
>>> d1=dict(zip(range(1,6),range(10,60,10)))
>>> print(d1)
{1: 10, 2: 20, 3: 30, 4: 40, 5: 50}
>>> k=d1.keys()
>>> print(k)
dict_keys([1, 2, 3, 4, 5])
>>> v=d1.values()
>>> print(v)
dict_values([10, 20, 30, 40, 50])
>>> a=d1.items()
>>> print(a)
dict_items([(1, 10), (2, 20), (3, 30), (4, 40), (5, 50)])
>>> del d1[1]
>>> print(k)
dict_keys([2, 3, 4, 5])
>>> print(v)
dict_values([20, 30, 40, 50])
>>> print(a)
dict_items([(2, 20), (3, 30), (4, 40), (5, 50)])
```

Example:

```
sales={'naresh':56000,
       'suresh':45000,
       'ramesh':34000,
       'rajesh':78000}

s=sales.values()
```

```
print(s)
total=sum(s)
print(f'Total Sales {total}')
max_sales=max(s)
print(max_sales)
min_sales=min(s)
print(min_sales)
```

Output

```
dict_values([56000, 45000, 34000, 78000])
Total Sales 213000
78000
34000
```

setdefault() method

setdefault() method performs two operations

1. reading value of given key, if exists
2. if given key not exists, it add key with given default value

```
>>> d1={}
>>> print(d1)
{}
>>> x=d1.setdefault(1)
>>> print(x)
None
>>> print(d1)
{1: None}
>>> y=d1.setdefault(2,20)
>>> print(y)
20
>>> print(d1)
{1: None, 2: 20}
```

reversed(iterable)

This function returns reversed iterator object, which iterate or read keys in reverse direction.

```
d1=dict(zip(range(1,6),range(10,60,10)))
print(d1)
a=reversed(d1)
```

```
for k in a:
    print(k,d1[k])
```

Dictionary is a mutable collection

How to add an item within dictionary?

Syntax:
dictionary-name[key]=value

This syntax performs two operations
1. Adding an item
2. Updating value of given key

If key not exists within dictionary, it add given key and value
If key exists within dictionary, it updates value of key