**Mutable operations of dictionary**
Dictionary is mutable collection, after creating dictionary we can add, update and delete items

**Adding or updating item of dictionary**
dictionary-name[key]=value

- ⇨ If the given key is not exists in dictionary, add key and value
- ⇨ If the given key is exists in dictionary, it will replace value/update value of key

```
>>> d1={}
>>> print(d1)
{}
>>> d1[1]=100
>>> d1[2]=200
>>> d1[3]=300
>>> print(d1)
{1: 100, 2: 200, 3: 300}
>>> d1[1]=10
>>> print(d1)
{1: 10, 2: 200, 3: 300}
>>> d1[4]=100
>>> print(d1)
{1: 10, 2: 200, 3: 300, 4: 100}
>>>
```

**Deleting items from dictionary**
Deleting items from dictionary are done using different approaches

1. del keyword
2. Clear()
3. Popitem()
4. Pop(key)

**del keyword**
**Syntax: del <dictionary-name>[key]**
If key is not exists within dictionary, it raises KeyError

```
>>> d2={2018:35000,2019:45000,2020:54000}
>>> print(d2)
{2018: 35000, 2019: 45000, 2020: 54000}
```

```
>>> del d2[2018]
>>> print(d2)
{2019: 45000, 2020: 54000}
>>> del d2[2018]
Traceback (most recent call last):
  File "<pyshell#14>", line 1, in <module>
    del d2[2018]
KeyError: 2018
>>> print(d2)
{2019: 45000, 2020: 54000}
>>>
```

****Shopping cart****

1. Add products
2. Remove products
3. Update products
4. View products
5. Exit

Enter your option:

```
cart={}
while True:
    print("****shopping cart****")
    print("1. Add")
    print("2. Update")
    print("3. Delete")
    print("4. View")
    print("5. Exit")
    opt=int(input("enter your option"))
    if opt==1:
        pname=input("ProductName :")
        if pname in cart:
            print(f'{pname} exists in cart')
        else:
            qty=int(input("Qty:"))
            cart[pname]=qty
            print("product added...")
    elif opt==2:
        pname=input("ProductName :")
        if pname in cart:
```

```
            qty=int(input("New Qty:"))
            cart[pname]=qty
            print("Qty is updated..")
        else:
            print(f'{pname} not exists in cart')
    elif opt==3:
        pname=input("ProductName :")
        if pname in cart:
            del cart[pname]
            print("Product deleted..")
        else:
            print(f'{pname} not exists in cart')
    elif opt==4:
        for pname,qty in cart.items():
            print(f'{pname}\t{qty}')
    elif opt==5:
        break
```

**clear() method**

this method empty dictionary or remove all the items from 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)
{}
>>>
```

**popitem()**

Remove and return a (key, value) pair from the dictionary. Pairs are returned in LIFO order.

```
>>> course_dict={'java':4000,'python':5000,'cpp':2000}
>>> i=course_dict.popitem()
>>> print(course_dict)
{'java': 4000, 'python': 5000}
>>> print(i)
```

```
('cpp', 2000)
>>> j=course_dict.popitem()
>>> print(course_dict)
{'java': 4000}
>>> print(j)
('python', 5000)
>>>
```

**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.

```
>>> d1={1:100,2:200,3:300,4:400,5:500}
>>> print(d1)
{1: 100, 2: 200, 3: 300, 4: 400, 5: 500}
>>> value=d1.pop(1)
>>> print(d1)
{2: 200, 3: 300, 4: 400, 5: 500}
>>> print(value)
100
>>> value=d1.pop(1)
Traceback (most recent call last):
  File "<pyshell#32>", line 1, in <module>
    value=d1.pop(1)
KeyError: 1
>>> value=d1.pop(1,None)
>>> print(value)
None
>>>
```

**update([*other*])**
Update the dictionary with the key/value pairs from *other*, overwriting existing keys. Return None.

**Example:**

```
d1={1:100,2:200,3:300}
print(d1)
d2={4:400,5:500,6:600,7:700}
print(d2)
d1.update(d2)
```

```
print(d1)
d3={1:10,5:50,8:800,9:900}
d1.update(d3)
print(d1)
```
**Output:**
```
{1: 100, 2: 200, 3: 300}
{4: 400, 5: 500, 6: 600, 7: 700}
{1: 100, 2: 200, 3: 300, 4: 400, 5: 500, 6: 600, 7: 700}
{1: 10, 2: 200, 3: 300, 4: 400, 5: 50, 6: 600, 7: 700, 8: 800, 9: 900}
>>>
```

**Dictionary comprehension**

Dictionary comprehension allows creating dictionary using for loop and test condition

Syntax1: {key:value for variable in iterable}
Syntax2: {key:value for variable in iterable if test}

Example:
```
# write a program to create a dictionary with student marks details
# each student is having rollno,name and 2 subject marks
# read the details of n students

n=int(input("enter how many students"))
#without comprehension
stud_dict={}
for i in range(n):
    rollno=int(input("enter rollno"))
    name=input("enter name")
    sub1=int(input("subject1"))
    sub2=int(input("subject2"))
    stud_dict[rollno]=[name,sub1,sub2]

print(stud_dict)

# with comprehension
stud_dict={int(input("Rollno:")):[input("Name"),int(input("sub1")),int(input("su
b2"))] for i in range(n)}
print(stud_dict)
```

**Output:**

```
enter how many students2
enter rollno1
enter namesuresh
subject160
subject270
enter rollno2
enter namekishore
subject190
subject299
{1: ['suresh', 60, 70], 2: ['kishore', 90, 99]}
Rollno:1
Namekishore
sub190
sub299
Rollno:2
Namekiran
sub190
sub277
{1: ['kishore', 90, 99], 2: ['kiran', 90, 77]}
>>>
```

**Example:**

```
# create a dictionary key as number and value as sqr of that number
#without comprehension
dict1={}
for n in range(1,11):
    dict1[n]=n**2

print(dict1)
# with comprehension
dict2={n:n**2 for n in range(1,11)}
print(dict2)
```

**Output:**

```
{1: 1, 2: 4, 3: 9, 4: 16, 5: 25, 6: 36, 7: 49, 8: 64, 9: 81, 10: 100}
{1: 1, 2: 4, 3: 9, 4: 16, 5: 25, 6: 36, 7: 49, 8: 64, 9: 81, 10: 100}
>>>
```