Dictionary Methods

copy( )

- It will create a shallow copy of a dictionary
- It will not create a new object it will be referring to new object

```
>>>
>>> dict1 = { 101 : 'Production' , 102 : 'Accounts' , 103 : 'Sales & Marketing' , 104 : 'Inventory' }
>>>
>>> dict2=dict1.copy()
>>>
>>> dict2
{101: 'Production', 102: 'Accounts', 103: 'Sales & Marketing', 104: 'Inventory'}
>>> dict2[102]='Designing'
>>>
>>> dict1
{101: 'Production', 102: 'Accounts', 103: 'Sales & Marketing', 104: 'Inventory'}
>>> dict2
{101: 'Production', 102: 'Designing', 103: 'Sales & Marketing', 104: 'Inventory'}
>>> id(dict1[101])
140284086280240
>>> id(dict2[101])
140284086280240
>>>
```

- Copy will give copy of dict1 in dict2 now we can modify , add , update , del etc can be performed

Update(iterable)

- If you want to add more key-value pairs you can call update( )

```
>>>
>>> dict1 = { 101 : 'Production' , 102 : 'Accounts' , 103 : 'Sales & Marketing' , 104 : 'Inventory' }
>>> dict2 = {105 : 'Designing', 106 : 'Packaging'}
>>>
>>> dict1.update(dict2)
>>>
>>> dict1
{101: 'Production', 102: 'Accounts', 103: 'Sales & Marketing', 104: 'Inventory', 105: 'Designing', 106: 'Packaging'}
>>>
>>>
```

Setdefault(key , default)

- This method works like get( ) , if the key is not found then it'll insert a key and its value will be none , if value is given it'll give key - value as output.

```
>>>
>>> dict1 = { 101 : 'Production' , 102 : 'Accounts' , 103 : 'Sales & Marketing' , 104 : 'Inventory' }
>>> dict2 = {105 : 'Designing', 106 : 'Packaging'}
>>>
>>> dict1.update(dict2)
>>> dict1
{101: 'Production', 102: 'Accounts', 103: 'Sales & Marketing', 104: 'Inventory', 105: 'Designing', 106: 'Packaging'}
>>>
>>> dict1.get(102)
'Accounts'
>>> dict1.get(110)
>>> dict1
{101: 'Production', 102: 'Accounts', 103: 'Sales & Marketing', 104: 'Inventory', 105: 'Designing', 106: 'Packaging'}
>>>
>>> dict1.setdefault(102)
'Accounts'
>>> dict1.setdefault(110)
>>> dict1
{101: 'Production', 102: 'Accounts', 103: 'Sales & Marketing', 104: 'Inventory', 105: 'Designing', 106: 'Packaging', 110: None}
>>> dict1.setdefault(111,'Adv')
'Adv'
>>> dict1
{101: 'Production', 102: 'Accounts', 103: 'Sales & Marketing', 104: 'Inventory', 105: 'Designing', 106: 'Packaging', 110: None, 111: 'Adv'}
>>>
```

formkeys(sequence , value)

- You can insert keys from other sequence like list , tuple , set and insert them as key value , it takes only one value and all the keys will have the same value
- Keys are important here

```
>>>
>>> L1 = [11, 22, 33, 44]
>>> dict3={}
>>> dict3.fromkeys(L1)
{11: None, 22: None, 33: None, 44: None}
>>> dict3.fromkeys(L1,100)
{11: 100, 22: 100, 33: 100, 44: 100}
>>> dict3.fromkeys(L1,'hello')
{11: 'hello', 22: 'hello', 33: 'hello', 44: 'hello'}
>>>
```

- For removing elements from dictionary following methods are used

Pop(key , default)

- It will remove the key and value from dictionary and the key should be mentioned
- If key is not present I'll give an error

```
>>>
>>> dict1 = { 101 : 'Production' , 102 : 'Accounts' , 103 : 'Sales & Marketing' , 104 : 'Inventory' }

>>> dict1.pop(104)
'Inventory'
>>> dict1
{101: 'Production', 102: 'Accounts', 103: 'Sales & Marketing'}
>>> dict1.pop(110)
Traceback (most recent call last):
  File "<pyshell#4>", line 1, in <module>
    dict1.pop(110)
KeyError: 110
```

Pop ( )

- I'll pop the last key value from the dictionary

Clear( )

- I'll clear the dictionary

del

- If you want to delete the entire Dictionary use del keyword

```
>>> dict1
{101: 'Production', 102: 'Accounts', 103: 'Sales & Marketing', 110: 'Adv'}
>>>
>>> dict1.popitem()
(110, 'Adv')
>>> dict1.clear()
>>> dict1
{}
>>> del dict1
>>> dict1
Traceback (most recent call last):
  File "<pyshell#14>", line 1, in <module>
    dict1
NameError: name 'dict1' is not defined
>>>
```