Python Dictionary Method

1. **update()**

Python update() method updates the dictionary with the key and value pairs. It inserts key/value if it is not present. Itupdates key/value if it is already present in the dictionary.

Example 1:-

It is a simple example to update the dictionary by passing key/value pair. This method updates the dictionary. See anexample below.

```
# Python dictionary update() Method
 # Creating a dictionary
enventory={'Fan': 200, 'Bulb':150, 'Led':1000}
print("Inventory:",enventory)
# Calling Method
enventory.update({'AC':50})
print("Updated inventory:",enventory)
```

Output:

```
Inventory: {'Fan': 200, 'Bulb': 150, 'Led': 1000}
Updated inventory: {'Fan': 200, 'Bulb': 150, 'Led': 1000, 'AC': 50}
```

Example 2:-

If element (key/value) pair is already presents in the dictionary, it will overwrite it. See an example below.

```
# Python dictionary update() Method # Creating a dictionary
enventory = {'Fan': 200, 'Bulb':150, 'Led':1000,'AC':50}
print("Inventory:",enventory)
 # Calling Method
enventory.update({'AC':250})
print("Updated inventory:",enventory)
enventory.update({'AC':150})
print("Updated inventory:",enventory)
```

Output:

```
Inventory: {'Fan': 200, 'Bulb': 150, 'Led': 1000, 'AC': 50}
Updated inventory: {'Fan': 200, 'Bulb': 150, 'Led': 1000, 'AC': 250}
Updated inventory: {'Fan': 200, 'Bulb': 150, 'Led': 1000, 'AC': 150}
```

Example 3:-

The update() method also allows iterable key/value pairs as parameter. Example below two values arepassed to the dictionary and it is updated.

```
# Python dictionary update() Method # Creating a dictionary
inventory = {'Fan': 200, 'Bulb':150, 'Led':1000}
print("Inventory:",inventory)
# Calling Method
inventory.update(cooler=50,switches=1000)
print("Updated inventory:",inventory)
```

Output:

```
Inventory: {'Fan': 200, 'Bulb': 150, 'Led': 1000}

Updated inventory: {'Fan': 200, 'Bulb': 150, 'Led': 1000, 'cooler': 50, 'switches': 1000}
```

2. **clear():-**

The clear() method removes all the elements from a dictionary.

Syntax

dictionary.clear()

ParameterValues: No parameters

Example:

```
car={"brand":"Maruti","model":"Maruti 800", "year": 1964}

car.clear()
print(car)
```

Output: {}

copy():- The copy() method returns a copy of the specified dictionary.

Syntax:

dictionary.copy()

ParameterValues: No parameters

Example:

```
car={"brand":"Maruti","model":"Maruti 800", "year": 1964}

x=car.copy()
print(x)
```

Output:

```
{'brand':'Maruti','model':'Maruti 800','year':1964}
```

3. **fromkeys():-** The fromkeys() method returns a dictionary with the specified keys and the specified value.

Syntax:

dict.fromkeys(keys,value)

Parameter Values:

Parameter	Description
Keys	Required. Specifying the keys of the new dictionary
Value	Optional. The value for all keys. Default value is None

Example:

```
x=('key1','key2','key3')

y = 0

thisdict=dict.fromkeys(x,y)

print(thisdict)
```

Output:

```
['key1':0,'key2':0,'key3':0]
```

4. **get():-**

The get() method returns the value of the item with the specified key.

Syntax:

dictionary.get(keyname,value)

Parameter Values:

Parameter	Description
Keyname	Required. The key name of the item you want to return the value from
Value	Optional. A value to return if the specified key does not exist. Default value None

Example:

```
car={"brand":"Maruti","model":"Maruti 800", "year": 1964}

x=car.get("model")

print(x)
```

Output:

```
Maruti 800
```

5. **items():-**

The items() method returns a view object. The view object contains the key-value pairs of the dictionary, as tuples in a list.

The view object will reflect any changes done to the dictionary.

Syntax:

dictionary.items()

ParameterValues: No parameters

Example:

```
car={"brand":"Maruti","model":"Maruti 800", "year": 1964}

x = car.items()

car["year"]=2018

print(x)
```

Output:

```
dict_items([('brand','Maruti'),('model','Maruti 800'),('year',2018)])
```

6. **keys():-**

The keys() method returns a view object. The view object contains the keys of the dictionary, as a list.

The view object will reflect any changes done to the dictionary.

Syntax:-

dictionary.keys()

ParameterValues: Noparameters

Example:

```
car={"brand":"Maruti","model":"Maruti 800", "year": 1964}

x=car.keys()
print(x)
```

Output:

```
dict_keys(['brand','model','year'])
```

7. **pop():-**

The pop() method removes the specified item from the dictionary.

The value of the removed item is the return value of the pop() method, see example below.

Syntax:-

dictionary.pop(keyname,defaultvalue)

Parameter Values

Parameter	Description
Keyname	Required. The key name of the item you want to remove
Default value	Optional. A value to return if the specified key do not exist. If this parameter is not specified, and the no item with the specified key is found, an error is raised

Example:

```
car={"brand":"Maruti","model":"Maruti 800", "year": 1964}

car.pop("model")

print(car)
```

Output:

```
{'brand':'Maruti','year':1964}
```

8. **popitem():-**

The pop item() method removes the item that was last inserted into the dictionary. In versions before 3.7, the popitem() method removes a random item.

The removed item is the return value of the popitem() method,as a tuple.

Syntax:-

```
dictionary.popitem()
```

Parameter Values: Noparameters

Example:

```
car={"brand":"Maruti","model":"Maruti 800", "year": 1964}
car.pop("model")
print("Remaing Items : ",car)
x=car.popitem()
print("Poped Item : ",x)
```

Output:

```
Remaing Items :  {'brand': 'Maruti', 'year': 1964}

Poped Item :  ('year', 1964)
```

9. **setdefault():-**

The setdefault() method returns the value of the item with the specified key.

If the key does not exist, insert the key, with the specified value.

Syntax:-

```
dictionary.setdefault(keyname,value)
```

Parameter Values:

Parameter	Description
Keyname	Required. The key name of the item you want to return the value from
Value	• Optional. • If the key exist, this parameter has no effect. • If the key does not exist, this value becomes the key's value. • Default value is None

Example:

```
car={"brand":"Maruti","model":"Maruti 800", "year": 1964}

x=car.setdefault("color","White")

print(x)
```

Output:

```
White
```

10. values():-

The values() method returns a view object. The view object contains the values of the dictionary, as a list.

The view object will reflect any changes done to the dictionary.

Syntax:-

```
dictionary.values()
```

Parameter Values: No parameters

Example:

```
car={"brand":"Maruti","model":"Maruti 800", "year":1964}

x=car.values()

print(x)
```

Output: dict_values(['Maruti','Maruti 800',1964])

<u>Nested Dictionary in Python:-</u>

In Python, a nested dictionary is a dictionary inside a dictionary. Basically, it is a collection ofdictionaries kept inside a single dictionary.

Nested dictionaries are one of the ways to represent and store structured information (similar tosome ‘records’ in other languages).
Below given is a simple example of a nested dictionary.

```
my_dict = { 'dict1': {'key_A': 'value_A'},'dict2': {'key_B': 'value_B'}}
```