Adding and updating multiple items of dictionary

update() method of dictionary add or update multiple items of dictionary.

Syntax: dictionary-name.update(iterable)
Iterable should generate two values

1. Key
2. Value

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

```
>>> A={1:10,2:20,3:30,4:40,5:50}
>>> print(A)
{1: 10, 2: 20, 3: 30, 4: 40, 5: 50}
>>> A.update({6:60,7:70})
>>> print(A)
{1: 10, 2: 20, 3: 30, 4: 40, 5: 50, 6: 60, 7: 70}
>>> A.update({8:80,9:90,1:99,2:88})
>>> print(A)
{1: 99, 2: 88, 3: 30, 4: 40, 5: 50, 6: 60, 7: 70, 8: 80, 9: 90}
```

fromkeys(iterable,value=None)
This method create new dictionary using keys generated by iterable

Syntax: variable-name=dict.fromkeys(iterable,value=None)

Example:

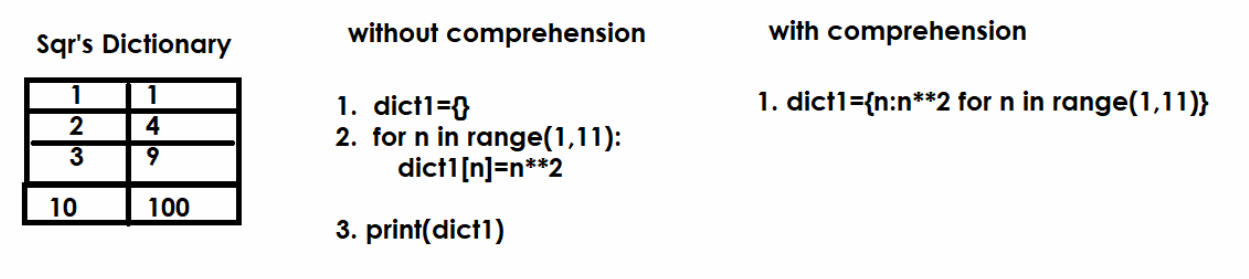
```
>>> B=dict.fromkeys([1,2,3])
>>> print(B)
{1: None, 2: None, 3: None}
>>> C=dict.fromkeys([2020,2021,2022,2023,2024],0)
>>> print(C)
{2020: 0, 2021: 0, 2022: 0, 2023: 0, 2024: 0}
>>> C[2020]=45000
>>> print(C)
{2020: 45000, 2021: 0, 2022: 0, 2023: 0, 2024: 0}
>>> C[2021]=80000
>>> print(C)
{2020: 45000, 2021: 80000, 2022: 0, 2023: 0, 2024: 0}
```

```
>>> players=dict.fromkeys(['rohit','virat','surya'])
>>> print(players)
{'rohit': None, 'virat': None, 'surya': None}
>>> players['rohit']=100
>>> print(players)
{'rohit': 100, 'virat': None, 'surya': None}
```

Dictionary Comprehension

Comprehension is one of elegant or easy method of creating list, set and dictionary writing in single statement or single line

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




```
>>> sdict={n:n**2 for n in range(1,11)}
>>> print(sdict)
{1: 1, 2: 4, 3: 9, 4: 16, 5: 25, 6: 36, 7: 49, 8: 64, 9: 81, 10: 100}
```

Example:

```
# Write a program read the scores of n players
# each player is having name and score

n=int(input("How many players?"))
players={input("Name:"):int(input("Score:"))
        for i in range(n)}

for name,score in players.items():
    print(f'{name}-->{score}')
```

Output

```
How many players?2
Name:rohit
```

Score:100
Name:virat
Score:120
rohit-->100
virat-->120

Example:

```
persons={'naresh':56,
         'suresh':45,
         'ramesh':52,
         'kishore':60,
         'rajesh':30,
         'raman':51}
for name,age in persons.items():
    print(f'{name}-->{age}')
persons1={name:age for name,age in persons.items() if age<50}
persons2={name:age for name,age in persons.items() if age>=50}

print(persons1)
print(persons2)
```

Output

```
naresh-->56
suresh-->45
ramesh-->52
kishore-->60
rajesh-->30
raman-->51
{'suresh': 45, 'rajesh': 30}
{'naresh': 56, 'ramesh': 52, 'kishore': 60, 'raman': 51}
```

Example:

```
marks={'naresh':[40,50],
       'suresh':[70,80],
       'ramesh':[30,80],
       'rajesh':[70,90],
       'kishore':[25,65]}

stud_result={name:[s[0],s[1],"pass" if s[0]>=40 and
                  s[1]>=40 else "fail"] for name,s in marks.items()}
```

```
print(marks)
print(stud_result)
```

Output

```
{'naresh': [40, 50], 'suresh': [70, 80], 'ramesh': [30, 80], 'rajesh': [70, 90],
'kishore': [25, 65]}
{'naresh': [40, 50, 'pass'], 'suresh': [70, 80, 'pass'], 'ramesh': [30, 80, 'fail'],
'rajesh': [70, 90, 'pass'], 'kishore': [25, 65, 'fail']}
```

Example:

```
grade_dict={'naresh':'A',
            'suresh':'B',
            'ramesh':'A',
            'rajesh':'A',
            'raman':'C'}

grade_dictA={name:grade for name,grade in grade_dict.items()
             if grade=='A'}
grade_dictB={name:grade for name,grade in grade_dict.items()
             if grade=='B'}
grade_dictC={name:grade for name,grade in grade_dict.items()
             if grade=='C'}
print(grade_dict)
print(grade_dictA)
print(grade_dictB)
print(grade_dictC)
```

Output

```
{'naresh': 'A', 'suresh': 'B', 'ramesh': 'A', 'rajesh': 'A', 'raman': 'C'}
{'naresh': 'A', 'ramesh': 'A', 'rajesh': 'A'}
{'suresh': 'B'}
{'raman': 'C'}
```

Nested dictionary

A dictionary cannot defined inside dictionary directly
Inside dictionary, a dictionary can be represented as value.

```
>>> d1={1:{'a':10,'b':20},
...     2:{'x':100,'y':200}}
```

```
>>> print(d1)
{1: {'a': 10, 'b': 20}, 2: {'x': 100, 'y': 200}}
>>> marks={1:['naresh',50,60],
       2:['suersh',70,80],
       3:['kishore',90,80]}
>>> marks[1]
['naresh', 50, 60]
>>> marks[1][0]
'naresh'
>>> marks[1][2]
60
>>> marks[3][1]
90
>>> marks={1:{'name':'naresh','sub1':50,'sub2':60},
...        2:{'name':'suresh','sub1':70,'sub2':80},
...        3:{'name':'kishore','sub1':90,'sub2':77}}
>>> marks[1]['name']
'naresh'
>>> marks[1]['sub1']
50
>>> marks[3]['sub2']
77
```

Merging of dictionaries

| This symbol or operator is used for merging of two dictionaries

Syntax: <variable-name>=dictionary1|dictionary2

```
>>> d1={1:10,2:20}
>>> d2={3:30,4:40}
>>> d3=d1|d2
>>> print(d1)
{1: 10, 2: 20}
>>> print(d2)
{3: 30, 4: 40}
>>> print(d3)
{1: 10, 2: 20, 3: 30, 4: 40}
```

```
>>> d3={1:99,3:30,4:44,5:55}
>>> d4=d1|d3
>>> print(d4)
{1: 99, 2: 20, 3: 30, 4: 44, 5: 55}
>>>
```

|= updates dictionary by merging items from another dictionary

```
>>> dict1={1:10,2:20}
>>> dict2={3:30,4:40}
>>> dict1|=dict2
>>> print(dict1)
{1: 10, 2: 20, 3: 30, 4: 40}
```

What is difference between list, set and dictionary?

List	**Set**	**Dictionary**
List is ordered collection	Set is unordered collection	Dictionary is ordered collection
In list data is organized in sequential order	In set data is organized using hashing data structure	In dictionary data is organized as key and value pair (mapping collection)
List allows duplicates	Set does not allows duplicates	Dictionary allows duplicate values but not allows duplicate keys
List allows any type of objects	Set allows only hashable objects or immutable objects	Dictionary keys are immutable and values can be any type
List is index based	Set is non index based	Dictionary is key based
In application development list is used to group individual objects where duplicates are allowed, reading and writing is done sequentially and randomly	In application development set is used to group individual objects where duplicates are not allowed, perform membership testing, removing duplicates from sequences and mathematical operations	In application development dictionary is used to organize data as key and value pair.

	(union,intersections,..)	
List created using []	Set is created using {}	Dictionary is created using {}
List class represents list object	Set class represents set object	dict class represents dictionary object

Packing and Unpacking