Dictionaries

- Stores collection of various types of data
- Dictionaries are mutable.
- Dictionaries have pair of keys and values which is seperated with ':'.
- Keys are act as index of values in dictionary.
- Keys in dictionary are unique.
- It is represented as flower brackets.

  Syntax: {key:value}

In [1]:

```
dic = {'name':'xyz','age':20,'grade':'A','phno':1234}
dic
```

Out[1]:

```
{'name': 'xyz', 'age': 20, 'grade': 'A', 'phno': 1234}
```

In [2]:

```
dic['phno']
```

Out[2]:

```
1234
```

In [4]:

```
#mutable
dic['name'] = 'apssdc'
```

In [5]:

```
dic
```

Out[5]:

```
{'name': 'apssdc', 'age': 20, 'grade': 'A', 'phno': 1234}
```

In [6]:

```
print(dir(dict))
```

```
['__class__', '__contains__', '__delattr__', '__delitem__', '__dir__', '__do
c__', '__eq__', '__format__', '__ge__', '__getattribute__', '__getitem__',
'__gt__', '__hash__', '__init__', '__init_subclass__', '__iter__', '__le__',
'__len__', '__lt__', '__ne__', '__new__', '__reduce__', '__reduce_ex__', '__
repr__', '__reversed__', '__setattr__', '__setitem__', '__sizeof__', '__str_
_', '__subclasshook__', 'clear', 'copy', 'fromkeys', 'get', 'items', 'keys',
'pop', 'popitem', 'setdefault', 'update', 'values']
```

In [7]:

```
# items()
dic.items()
```

Out[7]:

```
dict_items([('name', 'apssdc'), ('age', 20), ('grade', 'A'), ('phno', 123
4)])
```

In [8]:

```
# keys()
dic.keys()
```

Out[8]:

```
dict_keys(['name', 'age', 'grade', 'phno'])
```

In [9]:

```
# values()
dic.values()
```

Out[9]:

```
dict_values(['apssdc', 20, 'A', 1234])
```

In [10]:

```
# update()
dic.update({'marks':85,'addr':'abc'})
```

In [11]:

```
dic
```

Out[11]:

```
{'name': 'apssdc',
 'age': 20,
 'grade': 'A',
 'phno': 1234,
 'marks': 85,
 'addr': 'abc'}
```

In [12]:

```
# pop()
dic.pop('grade')
```

Out[12]:

```
'A'
```

In [13]:

```
dic
```

Out[13]:

```
{'name': 'apssdc', 'age': 20, 'phno': 1234, 'marks': 85, 'addr': 'abc'}
```

In [14]:

```
# popitem()
dic.popitem()
```

Out[14]:

```
('addr', 'abc')
```

In [15]:

```
dic
```

Out[15]:

```
{'name': 'apssdc', 'age': 20, 'phno': 1234, 'marks': 85}
```

In [16]:

```
# setdefault()
dic.setdefault('D')
```

In [17]:

```
dic
```

Out[17]:

```
{'name': 'apssdc', 'age': 20, 'phno': 1234, 'marks': 85, 'D': None}
```

In [21]:

```
print(dic['D'])
```

```
None
```

In [22]:

```
dic['D'] = 'DELL'
```

In [23]:

```
dic
```

Out[23]:

```
{'name': 'apssdc', 'age': 20, 'phno': 1234, 'marks': 85, 'D': 'DELL'}
```

In [24]:

```
dic.setdefault('H','Hp')
```

Out[24]:

```
'Hp'
```

In [25]:

```
dic
```

Out[25]:

```
{'name': 'apssdc',
 'age': 20,
 'phno': 1234,
 'marks': 85,
 'D': 'DELL',
 'H': 'Hp'}
```

In [26]:

```
# get()
dic.get('H')
```

Out[26]:

```
'Hp'
```

In [27]:

```
# fromkeys()
x = ('key1','key2','key3')
dict.fromkeys(x)
```

Out[27]:

```
{'key1': None, 'key2': None, 'key3': None}
```

In [30]:

```
y =0
dict2 = dict.fromkeys(x,y)
```

In [31]:

```
dict2['key2']
```

Out[31]:

```
0
```

In [32]:

```
dict2['key2'] = 'apple'
```

In [33]:

```
dict2
```

Out[33]:

```
{'key1': 0, 'key2': 'apple', 'key3': 0}
```

In [34]:

```
x = ('key1','key2','key3')
y = (1,2,3)
dict.fromkeys(x,y)
```

Out[34]:

```
{'key1': (1, 2, 3), 'key2': (1, 2, 3), 'key3': (1, 2, 3)}
```

In [35]:

```
# clear
dic.clear()
```

In [36]:

```
dic
```

Out[36]:

```
{}
```

In [37]:

```
print(len(dic))
```

```
0
```

In [43]:

```python
# Example for how to take dynamic dictionary in python
n = input("Enter a value: ")
m = input("Enter another value: ")
print("N=",n)
print("M=",m)
print("N type=",type(n))
print("M type=",type(m))
print(n+m)
```

```
Enter a value: Keerthi
Enter another value: Kollati
N= Keerthi
M= Kollati
N type= <class 'str'>
M type= <class 'str'>
KeerthiKollati
```

In [44]:

```python
n = int(input("Enter a value: "))
m = int(input("Enter another value: "))
print("N=",n)
print("M=",m)
print("N type=",type(n))
print("M type=",type(m))
print(n+m)
```

```
Enter a value: 4
Enter another value: 5
N= 4
M= 5
N type= <class 'int'>
M type= <class 'int'>
9
```

In [45]:

```python
f1 = float(input())
f2 = float(input())
print(type(f1))
print(type(f2))
print(f1+f2)
```

```
2.3
4.1
<class 'float'>
<class 'float'>
6.3999999999999995
```

In [46]:

```
# Example for how to take dynamic dictionary in python
n = int(input())
dic = {}
for i in range(n):
    k = input("Enter key:")
    v = int(input("Enter value:"))
    dic[k] = v
dic
```

```
2
Enter key:marks
Enter value:90
Enter key:addr
Enter value:123
```

Out[46]:

```
{'marks': 90, 'addr': 123}
```

In [47]:

```
# input --> lst = [1,3,2,1,1,2,3,3,3]
# output --> dic={1:3,3:4,2:2}

lst = [1,3,2,1,1,2,3,3,3]
dic = {}
for i in lst: #i=1,i=3, i=2, i=1, i=1
    dic[i] = lst.count(i) #{1:3, 3:4, 2:2}
dic
```

Out[47]:

```
{1: 3, 3: 4, 2: 2}
```

In [48]:

```
dic = {}
for i in lst:
    k = i
    v = lst.count(i)
    dic[k] = v
dic
```

Out[48]:

```
{1: 3, 3: 4, 2: 2}
```

Sets

- A set is unordered collection of data type that is iterable.
- A set is immutable(unchangeable).
- Python's set calss represents the mathematical notation of a set.

In [49]:

```
set1 = {6,2,9,10,4,2,3}
set1
```

Out[49]:

```
{2, 3, 4, 6, 9, 10}
```

In [50]:

```
len(set1)
```

Out[50]:

```
6
```

In [53]:

```
# immutable
set1[6] = 13
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-53-97fbbabc8211> in <module>
      1 # immutable
----> 2 set1[6] = 13

TypeError: 'set' object does not support item assignment
```

In [54]:

```
print(dir(set))
```

```
['__and__', '__class__', '__contains__', '__delattr__', '__dir__', '__doc_
_', '__eq__', '__format__', '__ge__', '__getattribute__', '__gt__', '__hash_
_', '__iand__', '__init__', '__init_subclass__', '__ior__', '__isub__', '__i
ter__', '__ixor__', '__le__', '__len__', '__lt__', '__ne__', '__new__', '__o
r__', '__rand__', '__reduce__', '__reduce_ex__', '__repr__', '__ror__', '__r
sub__', '__rxor__', '__setattr__', '__sizeof__', '__str__', '__sub__', '__su
bclasshook__', '__xor__', 'add', 'clear', 'copy', 'difference', 'difference_
update', 'discard', 'intersection', 'intersection_update', 'isdisjoint', 'is
subset', 'issuperset', 'pop', 'remove', 'symmetric_difference', 'symmetric_d
ifference_update', 'union', 'update']
```

In [55]:

```
set1
```

Out[55]:

```
{2, 3, 4, 6, 9, 10}
```

In [56]:

```
# add()
set1.add(8)
```

In [57]:

```
set1
```

Out[57]:

```
{2, 3, 4, 6, 8, 9, 10}
```

In [58]:

```
# update()
set1.update([12,11,14])
set1

```

Out[58]:

```
{2, 3, 4, 6, 8, 9, 10, 11, 12, 14}
```

In [59]:

```
# discard()
set1.discard(12)
```

In [60]:

```
set1
```

Out[60]:

```
{2, 3, 4, 6, 8, 9, 10, 11, 14}
```

In [66]:

```
s = {1, 245, 4, 47, 5, 54, 8,'cd','b',9.8,2.5}
s
```

Out[66]:

```
{1, 2.5, 245, 4, 47, 5, 54, 8, 9.8, 'b', 'cd'}
```

In [63]:

```
s2 = {'b','r','d','a'}
s2
```

Out[63]:

```
{'a', 'b', 'd', 'r'}
```

In [67]:

```
set1
```

Out[67]:

```
{2, 3, 4, 6, 8, 9, 10, 11, 14}
```

In [68]:

```
# remove()
set1.remove(8)
```

In [69]:

```
set1
```

Out[69]:

```
{2, 3, 4, 6, 9, 10, 11, 14}
```

In [70]:

```
set1.remove(13)
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
<ipython-input-70-59d1afcec698> in <module>
----> 1 set1.remove(13)

KeyError: 13
```

In [72]:

```
set1.discard(13)
set1
```

Out[72]:

```
{2, 3, 4, 6, 9, 10, 11, 14}
```

In [73]:

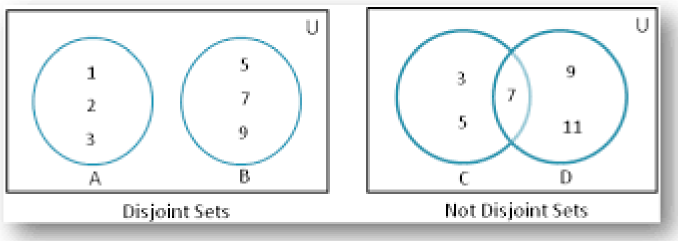
```
# pop()
set1.pop()
set1
```

Out[73]:

```
{3, 4, 6, 9, 10, 11, 14}
```

disjoint()

- Two sets are said to be disjoint if they do not have any common elements.



In [75]:

```
# isdisjoint()
A = {1,2,3}
B = {5,7,9}
print(A.isdisjoint(B))
print(B.isdisjoint(A))
```

```
True
True
```

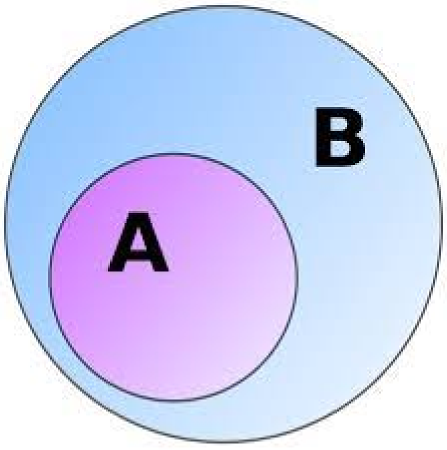
In [76]:

```
A = {1,3,7}
B = {5,7,9}
print(A.isdisjoint(B))
```

```
False
```

Superset

- Set B is said to be superset of set A if all elements of set A are in B.



In [78]:

```
# issuperset()
B = {1,3,5,7,9}
A = {1,5,7}
print(B.issuperset(A))
print(A.issubset(B))
```

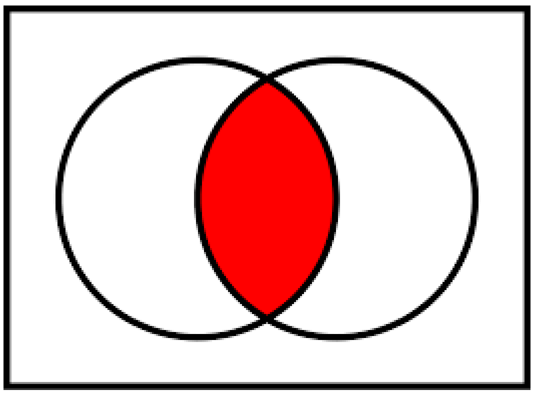
```
True
True
```

In [79]:

```
# union()
print(B.union(A))
```

```
{1, 3, 5, 7, 9}
```

intersection()



In [83]:

```
A = {1,3,5,7,9}
B = {1,5,8,2}
res = A.intersection(B)
print(res)
print(A)
print(B)
```

```
{1, 5}
{1, 3, 5, 7, 9}
{8, 1, 2, 5}
```

In [86]:

```
# intersection_update()
A = {1,3,5,7,9}
B = {1,5,8,2}
print(A.intersection_update(B))
print(A)
print(B.intersection_update(A))
print(B)
```

```
None
{1, 5}
None
{1, 5}
```

In [93]:

```
# difference
A =  {'a','b','c','d'}
B = {'c','g','f'}
print(A-B)
```

```
{'a', 'd', 'b'}
```

In [88]:

```
print(B-A)
```

```
{'f', 'g'}
```

In [91]:

```
print(A.difference(B))
print(B.difference(A))
print(A)
print(B)
```

```
{'a', 'd', 'b'}
{'f', 'g'}
{'a', 'c', 'd', 'b'}
{'f', 'g', 'c'}
```

In [95]:

```
A =  {'a','b','c','d'} # A-B ={a,b,d}
B = {'c','g','f'}
```

In [98]:

```
A.difference_update(B)
# B.difference_update(A)
print(A)
# print(B)
```

```
{'a', 'd', 'b'}
```

In [99]:

```
print(B)
```

```
{'f', 'g', 'c'}
```

In [100]:

```
B.difference_update(A)
```

In [101]:

```
print(B)
```

```
{'f', 'g', 'c'}
```

In [102]:

```
A = {'a','s','d','f'}
B = {'c','d'}
B.difference_update(A)
```

In [103]:

```
print(B)
```

```
{'c'}
```

In [104]:

```
A.difference_update(B)
```

In [105]:

```
print(A)
```

```
{'a', 'f', 's', 'd'}
```

In [106]:

```
# symmetric_difference
A =  {'a','b','c','d'}
B = {'c','g','f'}
print(A.symmetric_difference(B))
```

```
{'b', 'f', 'd', 'g', 'a'}
```

In [107]:

```
print(B.symmetric_difference(A))
```

```
{'b', 'f', 'd', 'g', 'a'}
```

In [108]:

```
print(A)
print(B)
```

```
{'a', 'c', 'd', 'b'}
{'f', 'g', 'c'}
```

In [110]:

```
A.symmetric_difference_update(B)
```

In [111]:

```
print(A)
```

```
{'b', 'f', 'd', 'g', 'a'}
```

In [112]:

```
print(B)
```

```
{'f', 'g', 'c'}
```

In [113]:

```
B.symmetric_difference_update(A)
```

In [114]:

```
print(B)
```

```
{'b', 'd', 'a', 'c'}
```

In [129]:

```
num1=13
num2=7
print("%d / %d =%1.2f"%(num1,num2,num1/num2))
```

```
13 / 7 =1.86
```

In [121]:

```
import math
```

In [126]:

```
print(math.ceil(1.86))
```

```
2
```

In [ ]:

```

```