Set Data Structure

In [4]:
```
s={} # is not a empty set
type(s)
```
Out[4]:
```
dict
```

In [6]:
```
s=set() #Now this is type of set.
type(s)
```
Out[6]:
```
set
```

In [8]:
```
s1={2,3,23,58,43}
s1
```
Out[8]:
```
{2, 3, 23, 43, 58}
```

Add - adding a element in set.

In [11]:
```
s1.add(65)
```

In [13]:
```
s1.add(49)
s1
```
Out[13]:
```
{2, 3, 23, 43, 49, 58, 65}
```

Copy - copies the same set of elments to the other variable

In [16]:
```
s=s1.copy()
s
```
Out[16]:
```
{2, 3, 23, 43, 49, 58, 65}
```

Pop - pops random elements from the set.

In [19]:
```
s1.pop()
```
Out[19]:
```
65
```

In [21]:
```
s1.pop()
```
Out[21]:
```
2
```

In [23]:
```
s1
```
Out[23]:
```
{3, 23, 43, 49, 58}
```

remove - removes the element from the set

In [26]:
```
s1.remove(23)
s1
```
Out[26]:
```
{3, 43, 49, 58}
```

In [28]:
```
s1.remove(49)
```

In [30]:
```
s1
```
Out[30]:
```
{3, 43, 58}
```

In [32]:
```
s1.remove(9)  #Gives error if the element is not present in set.
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
Cell In[32], line 1
----> 1 s1.remove(9)

KeyError: 9
```

discard - discards the element like remove but if the element is not present in the set, then it does not give error.

In [35]:
```
s
```
Out[35]:
```
{2, 3, 23, 43, 49, 58, 65}
```

In [37]:
```
s.discard(23) # element present in the set.
```

```
s
```

Out[37]: {2, 3, 43, 49, 58, 65}

In [39]:
```
s.discard(199) # Element not present in the set.
s
```

Out[39]: {2, 3, 43, 49, 58, 65}

In [49]:
```
s={2,3,43,45,6,45,7}
s1
```

Out[49]: {2, 3, 6, 7, 43, 45}

In [51]:
```
s.clear()
```

In [53]:
```
s
```

Out[53]: set()

In [55]:
```
del s1
```

In [72]:
```
a={1:1,2:2,3:3}
b={3:3,2:0,1:0}
p={**a,**b}
print(p)
```

{1: 0, 2: 0, 3: 3}

Intersection

In [77]:
```
x = {"a", "b", "c"}
y = {"b", "c", "d"}
z = x.intersection(y)
print(z)
```

{'c', 'b'}

In [81]:
```
x&y
```

Out[81]: {'b', 'c'}

In [83]:
```
sorted(x)
```

Out[83]: ['a', 'b', 'c']

In [85]:
```
sorted(y)
```

Out[85]: ['b', 'c', 'd']

Difference

In [88]:
```
a={1,2,3,4,5,6,7}
b={5,6,7,8,9,10}
a.difference(b)
```

Out[88]: {1, 2, 3, 4}

In [90]:
```
a.difference_update(b)
```

In [98]:
```
a.add(5)
a.add(6)
a.add(7)
```

In [102…
```
b-a
```

Out[102… {8, 9, 10}

Symmetric Difference

In [108…
```
a.symmetric_difference(b)
```

Out[108… {1, 2, 3, 4, 8, 9, 10}

In [110…
```
b^a
```

Out[110… {1, 2, 3, 4, 8, 9, 10}

In [112…
```
a.symmetric_difference_update(b)
```

In [114…
```
a
```
Out[114…
```
{1, 2, 3, 4, 8, 9, 10}
```

Subset, Superset, Disjoint

In [117…
```
A = {1,2,3,4,5,6,7,8,9}
B = {3,4,5,6,7,8}
C = {10,20,30,40}
```
In [119…
```
A.issubset(B)  # A is part of B or not
```
Out[119…
```
False
```
In [121…
```
B.issubset(A)
```
Out[121…
```
True
```
In [129…
```
C.isdisjoint(A) # Neighbouring or not
```
Out[129…
```
True
```
In [131…
```
A.issuperset(B)  # A is father of B or not
```
Out[131…
```
True
```
In [127…
```
B.issuperset(C)
```
Out[127…
```
False
```