We all know that: Set is a collection and immutable means sets cannot be modified.

In [1]:
```
#lets create sets:
a={1,2,34,44,7}
b={2,9,4,5,8,7,66,44,2}
```

In [2]:
```
#sets accepts duplicate value or not? lets check from b
print(b)
```
```
{2, 66, 4, 5, 7, 8, 9, 44}
```

In [6]:
```
#Does tuple or list accept duplicate value?
l=[1,2,2,3]
t=(9,9,8,7)
print(l)    #yes no error means they contain.
```
```
[1, 2, 2, 3]
```

In [3]:
```
#no, sets does not contain duplicate value. Similarly, we know that dictionary also does not contain duplicate
type(a)
```
Out[3]:
```
set
```

In [4]:
```
len(a)  #length fuction syntax is similar for all- tuple, string, list, dictionary
```
Out[4]:
```
5
```

Add item in set: Once we created the set, we cannot change item inside set, but, we can add and delete

In [7]:
```
#we have two set a and b,lets add b to a
a.update(b)
```

In [8]:
```
print(a)
```
```
{1, 2, 34, 66, 4, 5, 7, 8, 9, 44}
```

In [9]:
```
#we can add list to sets too
print(l)
```
```
[1, 2, 2, 3]
```

In [10]:
```
print(a)
```
```
{1, 2, 34, 66, 4, 5, 7, 8, 9, 44}
```

In [11]:
```
a.update(l)
```

In [12]:
```
print(a)
```
```
{1, 2, 34, 66, 4, 5, 7, 8, 9, 3, 44}
```

Remove set item - remove, pop, clear, del

In [16]:
```
a.remove(34)
```

In [17]:
```
print(a)
```
```
{1, 2, 66, 4, 5, 7, 8, 9, 3, 44}
```

In [18]:
```
a.remove(66,5)
```
```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[18], line 1
----> 1 a.remove(66,5)

TypeError: set.remove() takes exactly one argument (2 given)
```

In [19]:
```
#we cannot remove 2 item at a once.
```

In [21]:
```
a.pop()   #in set, pop() removes random element
```
Out[21]:
```
2
```

In [22]:
```
print(a)
```
```
{66, 4, 5, 7, 8, 9, 3, 44}
```

In [23]:
```
a.clear()
```

In [24]:
```
print(a)
```
```
set()
```

```
In [25]: #now set is empty, after we used clear method, lets delete this set
         del a
```

```
In [26]: print(a)
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[26], line 1
----> 1 print(a)

NameError: name 'a' is not defined
```

Join sets- union, update,intersection_update,intersection,symmetric_difference, symmetric_difference_update,difference,difference_update,

```
In [38]: #lets create three sets again to see all of example
         a={1,2,3,4,5,6,7,8,9,10}
         b={1,2,3,4,5}
         c={2,6,7,8,9}
```

```
In [30]: #Union is similar to update
         d={4,5,6}
         e={6,7,8,9}
         d.union(e)

Out[30]: {4, 5, 6, 7, 8, 9}
```

```
In [31]: #intersection and intersection_update
         b.intersection(c)

Out[31]: {2}
```

```
In [32]: #what was common in b and c 2 is common. But if we print b, will it give updated sets?
         print(b)

         {1, 2, 3, 4, 5}
```

```
In [36]: #no so in this type of case, if we need new sets we use update version
         b.intersection_update(c)
```

```
In [37]: print(b)

         {2}
```

```
In [39]: #now its get updated. So, lets create agin previous sets otherwise we cannot practice because b is laready upda
```

```
In [40]: a={1,2,3,4,5,6,7,8,9,10}
         b={1,2,3,4,5}
         c={2,6,7,8,9}
```

```
In [41]: #symmetric_difference
         b.symmetric_difference(c)

Out[41]: {1, 3, 4, 5, 6, 7, 8, 9}
```

Symmetric difference provide output of both sets, which are not similar in both sets.

```
In [43]: b.issubset(a)

Out[43]: True
```

```
In [44]: #b is sub set of a because, a contain all values of b and more than that too.
```

```
In [45]: a.difference(b)

Out[45]: {6, 7, 8, 9, 10}
```

```
In [ ]: #difference subtract the similar elements.
```

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js