Set Methods #1

- Methods are the member function of a class
- Some of the built in methods of sets are **add( ) , copy( ) ,update ( ) , pop( ) , discard( ) , remove( ) , clear( ).**
- In a set if you want to add element you can do this with add( ) method.
- As sets are immutable we can add more values to a set.It will modify the same set , add( ) only add one value not multiple value at a time.
- You can add any type of values in a set (like str, int etc)

- The copy( ) will give the same copy of a given set . It is also called cloning of a set.

- If you want to add multiple values to a set then use update(iterable) method.
- In update( ) it 1st checks the if value is available or not in the set if yes it doesn't add that value/ no duplicate and vice versa.

```
>>>
>>> s = {1,2,3,4}
>>> s.add(5)
>>> s
{1, 2, 3, 4, 5}
>>> s.add(6,7)
Traceback (most recent call last):
  File "<pyshell#5>", line 1, in <module>
    s.add(6,7)
TypeError: set.add() takes exactly one argument (2 given)
>>>
>>> s.copy()
{1, 2, 3, 4, 5}
>>>
>>>
>>> s.update({6,7})
>>> s
{1, 2, 3, 4, 5, 6, 7}
>>>
```

#adding an element to a set

#you can only 1 element at a time

#making clone of the original set

#when you want to add more than one element use update

- **Pop( )** will remove 1 element in a set, which element we don't know but it'll remove.

```
>>>
>>> s1 = {10,20,30,40,50,60}
>>> s1.pop()
50
>>> s1.pop()
20
>>>
>>>
>>> s1.pop(40)
Traceback (most recent call last):
  File "<pyshell#7>", line 1, in <module>
    s1.pop(40)
TypeError: set.pop() takes no arguments (1 given)
>>>
```

#pop( ) will not take argument , it will given an error if argument given.

- In **discard (x)** you have to mention the element which you want to remove.
- There is no index available in set so you have to mention the element to be discarded.
- The **remove(x)** method , removes the element from the set, work same as discard(x) the only difference is discussed below in the example.

Example :

```
>>>
>>> s1 = {10,20,30,40,50,60}
>>> s1.discard(70)
>>> s1
{50, 20, 40, 10, 60, 30}
>>>
>>> s1 = {10,20,30,40,50,60}
>>> s1.remove(70)
Traceback (most recent call last):
  File "<pyshell#6>", line 1, in <module>
    s1.remove(70)
KeyError: 70
```

discard ignore if no element is found to be discarded .

#Remove gives an error when no element is found to be removed.

- The **clear( )** method , will clear all contents of set, it will make it as an empty set.

```
>>>
>>> s1 = {10,20,30,40,50,60}
>>> s1.clear()
>>> s1
set()
>>>
```