SETS

A set is a collection of unordered items. It is Mutable and Unindexed. The set does not allow duplicate elements. Set items can appear in a different order every time you use them and cannot be referred to by index or key.

A set was created by placing all the items inside curly braces "**{ }**" and separated by comma, or by using the built-in set () functions.

It can have any number of items and they may be of different types(integer, float, tuple, string etc.).
But a set cannot have mutable elements like lists, sets, or dictionaries as its element.

CREATING SET:

We can not give an empty curly brace to create a set because it represents the dictionary. So here to create an empty set we use set() without any argument.

```
D={ }                          S=set()                  S1={1,2,3}
print(D)                       print(S)                  print(S1)
 O/P: <Class Dict>             O/P:<Class Set>          O/P:<Class Set>
```

SET METHODS:

add(): Adds an element to the set

```
    Eg:  x={1,1,1,2,3,3,4,4,5,6}
         x.add(123)
         print(x)
         O/P: {1, 2, 3, 4, 5, 6, 123}
```

update(): Update the set with another set, or any other iterable.

```
    Eg:  x={1,1,1,2,3,3,4,4,5,6}
         x.update({88,90)
         print(x)
         O/P: {1,2,3,4,5,99,90}
```

remove(): Removes the Specified element from the set

```
    Eg:  x={1,1,1,2,3,3,4,4,5,6}
         x.remove(5)
         print(x)
         O/P: {1, 2, 3, 4, 6}
```

pop(): Removes an element from the set

```
    Eg:  x={1,1,1,2,3,3,4,4,5,6}
```

```
x.pop()
print(x)
O/P: {2, 3, 4, 5, 6, 123}
```

copy(): Returns a copy of the set

Eg:
```
x={1,2,3,4,5,6}
y=x.copy()
print(y)
O/P: {1, 2, 3, 4,5,6}
```

clear(): Removes all the elements from the set

Eg:
```
x={1,1,1,2,3,3,4,4,5,6}
x.clear()
print(x)
O/P: set()
```

<u>Set Operations:</u>

Union(): Return a set containing the union of sets

Eg:
```
s1={1,2,3,4,6}
s2={4,5,6,7,8,9}
print(s1.union(s2))
O/P: {1,2,3,4,5,6,7,8,9}
```

intersection(): Returns a set, that is the intersection of two or more sets

Eg:
```
s1={1,2,3,4,6}
s2={4,5,6,7,8,9}
print(s1.intersection(s2))
O/P: {4,5}
```

difference(): Returns a set containing the difference between two or more sets

Eg:
```
s1={1,2,3,4,6}
s2={4,5,6,7,8,9}
print(s1.difference(s2))
O/P: {1,2,3}
```

Symmetric_difference(): Returns a set with the symmetric difference of two sets.

Eg:
```
s1={1,2,3,4,6}
s2={4,5,6,7,8,9}
print(s1.symmetric_difference(s2))
O/P: {1, 2, 3,5, 7, 8,9}
```

isdisjoint(): Returns whether two sets have an intersection or not

Eg:
```
x={1,2,3,4,6}
y={5,7,8,9}
print(x.isdisjoint())
O/P: True
```

issubset(): Returns whether another set contains this set or not

Eg: g1={1,2,3,4,6}

```
          g2={1,2,3,4,5,6,7,8,9}
          print(g1.issubset(g2))
          O/P: True
```
issuperset(): Returns whether this set contains another set or not

Eg: g1={1,2,3,4,6}
```
          g2={1,2,3,4,5,6,7,8,9}
          print(g2.issuperset(g1))
          O/P: True
```

<u>For loop with set:</u>

```
for i in {1,2,3,4}:
        If i==2:
          print("Yes")
          break
        else:
          print("False")
O/P: Yes
     Yes
```

<u>frozenSet():</u>

The frozenset() function returns an immutable frozenset object initialized with elements from the given iterable.

frozenset is just an immutable version of a Python set object. While elements of a set can be modified at any time, elements of the frozen set remain the same after creation.

Due to this, frozenset can be used as keys in a dictionary or as elements of another set. But like sets, it is not ordered(the elements can be set at any index).

```
Eg: h=[1,22,33,99,77]
    print(type(h))
    d=frozenset(h)
    print(d)
    O/P: <class List>
         frozenset({1,22,33,99,77})
```