---: Set :---

If we want to represent a group of unique elements then we can go for sets. Set cannot store duplicate elements.

1. **Duplicates are not allowed.**
2. **Order is not preserved.**
3. **Objects are mutable.**
4. **Indexing is not allowed.**
5. **Slicing is not allowed.**
6. **Represented in { } with comma separated objects.**
7. **Homogeneous and Heterogeneous both objects are allowed.**

```
# Creating a set
s = {10,20,30,40}
print(s)
print(type(s))

O/P:--
{40, 10, 20, 30}
<class 'set'>

# Creating a set with different elements
s = {10,'20','Rahul', 234.56, True}
print(s)
print(type(s))

O/P:--
{'20', True, 234.56, 10, 'Rahul'}
<class 'set'>

# Creating a set using range function
s=set(range(5))
print(s)

O/P:--
{0, 1, 2, 3, 4}
```

```
# Duplicates not allowed
s = {10, 20, 30, 40, 10, 10}
print(s)
print(type(s))
O/P:--
{40, 10, 20, 30}
<class 'set'>

# Creating an empty set
s=set()
print(s)
print(type(s))

O/P:--
set()
<class 'set'>
```

 # Methods in set:----

1. **add(only_one_argument not iterable)**

```
s={10,20,30,50}
s.add(40)
print(s)

O/P:--
{40, 10, 50, 20, 30}
```

2. **update(iterable_obj1,iterable_obj2)**

```
s = {10,20,30}
l = [40,50,60,10]
s.update(l)
print(s)

O/P:--
{40, 10, 50, 20, 60, 30}


s = {10,20,30}
```

```
l = [40,50,60,10]
s.update(l, range(5))
print(s)

O/P:--
{0, 1, 2, 3, 4, 40, 10, 50, 20, 60, 30}
```

Difference between add() and update() methods in set:

1. We can use add() to add individual items to the set, whereas we can use update() method to add multiple items to the set.
2. The add() method can take one argument whereas the update() method can take any number of arguments but the only point is all of them should be iterable objects.

3. **copy() --Clone of set**

```
s={10,20,30}
s1=s.copy()
print(s1)

O/P:--
{10, 20, 30}
```

4. **pop()---** This method removes and returns some random element from the set.

```
s = {40,10,30,20}
print(s)
print(s.pop())
print(s)

O/P:--
{40, 10, 20, 30}
40
{10, 20, 30}
```

5. **remove(element) ---** This method removes specific elements from the set. If the specified element is not present in the set then we will get KeyError.

```
s={40,10,30,20}
s.remove(30)
```

```
print(s)

O/P:--
{40, 10, 20}

s={40,10,30,20}
s.remove(50)
print(s)

O/P:--
Traceback (most recent call last):
  File "E:\DataSciencePythonBatch\sets.py", line 65, in <module>
    s.remove(50)
KeyError: 50
```

6. **discard(element) ---** This method removes the specified element from the set. If the specified element is not present in the set, then we won't get any error.

```
s={10,20,30}
s.discard(10)
print(s)

O/P:--
{20, 30}

s={10,20,30}
s.discard(40)
print(s)

O/P:--
{10, 20, 30}
```

7. **clear() --- removes all elements from the set.**

```
s={10,20,30}
print(s)
s.clear()
print(s)
```

```
O/P:--
{10, 20, 30}
set()
```

MATHEMATICAL OPERATIONS ON SETS

1. **union() ---** This method return all elements present in both sets.

```
x={10,20,30,40}
y={30,40,50,60}
print(x.union(y))

O/P:--
{40, 10, 50, 20, 60, 30}
```

2. **intersection() ---** This method returns common elements present in both x and y.

```
x = {10,20,30,40}
y = {30,40,50,60}
print(x.intersection(y))
print(x&y)
print(y.intersection(x))
print(y&x)

O/P:--
{40, 30}
{40, 30}
{40, 30}
{40, 30}
```

3. **difference() ---** This method returns the elements present in x but not in y

```
x = {"apple", "banana", "cherry"}
y = {"google", "microsoft", "apple"}
z = x.difference(y)
print(z)
```

O/P:--- {‘banana’, ‘cherry’}

MATHEMATICAL OPERATIONS ON SETS

1. Union

A = {1,2,3,4,5,6,7}
B = {5,6,7,8,9,10}

print(A.union(B))
O/P:--
{1,2,3,4,5,6,7,8,9,10}

2 Intersection

A = {1,2,3,4,5,6,7}
B = {5,6,7,8,9,10}

print(A.intersection(B))

O/P:--
{5,6,7}

3. Difference

A = {1,2,3,4,5,6,7}
B = {5,6,7,8,9,10}

print(A.difference(B))

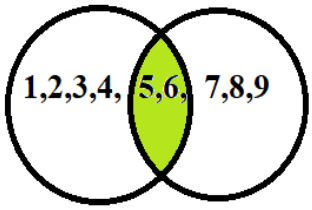
O/P:--
{1,2,3,4}

4. Symmetric_difference

A = {1,2,3,4,5,6,7}
B = {5,6,7,8,9,10}

print(A.symmetric_difference)
O/P:-- {1,2,3,4,8,9,10}

5. Intersection_update



A = {1,2,3,4,5,6}
B = {5,6,7,8,9}
A.intersection_update(B)
print(A)
O/P:-- {5,6}

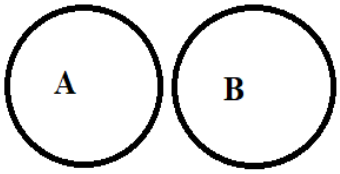
6. difference_update

A = {1,2,3,4,5,6}
B = {5,6,7,8,9}
A.difference_update(B)
print(A)
O/P:--
{1,2,3,4}

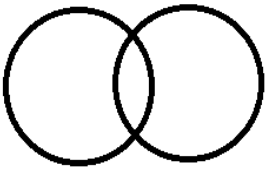
7. symmetric_difference_update

A = {1,2,3,4,5,6}
B = {5,6,7,8,9}
A.symmetric_difference_update(B)
print(A)
O/P:--
{1,2,3,4,7,8,9}

8. isdisjoint

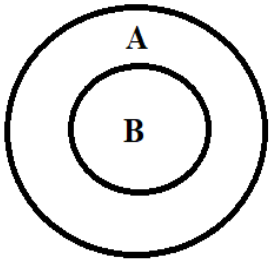


print(A.isdisjoint(B))
O/P:-
True

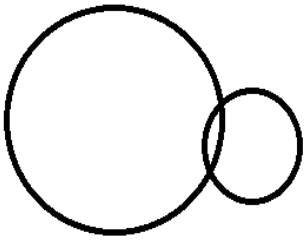


print(A.isdisjoint(B))
O/P:- False

9. issuperset

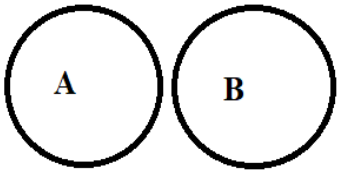


print(A.issuperset(B))
O/P:-- True

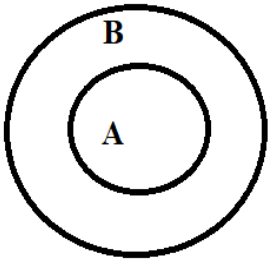


print(A.issuperset(B))
O/P:-- False

10. issubset



print(A.subset(B))
O/P:-- False



print(A.issubset(B))
O/P:-- True