SET

- It is a data structure use to store multiple objects with single name.
- Set is **mutable**.
- It allows to store only **immutable elements**.
- It **doesn't allow duplicate** elements.
- Insertion **order is not preserved**.
- It allows heterogeneous elements.
- It **doesn't support indexing and slicing**.
- It is mainly used to perform **mathematical operations** such as union, intersection etc.
- The notation of set is **{ }**.

Syntax to create an empty set:

set_name = set() #valid
type(set_name) → <class 'set'>

s = {} #not valid
type(s) → <class 'dict'>

Syntax to create set:

set_name = {ele1,ele2,ele3,…….}
e.g s = {1,3.5,'abc',False}

Iterating over the set:

```
s = {10,20,30,40,50}
for i in s:
    print(i)
```

OUTPUT:

10
20
30
40
50

Modifying a set in Python:

Sets are mutable. However, since they are unordered, indexing has no meaning.We cannot access or change an element of a set using indexing or slicing. Set data type does not support it.

Frozenset:

The frozenset is the immutable form of the normal sets, i.e., the items of the frozenset cannot be changed and therefore it can be used as a key in the dictionary.

The elements of the frozenset cannot be changed after the creation. We cannot change or append the content of the frozenset by using the methods like add() or remove().

The frozenset() method is used to create the frozenset object. The iterable sequence is passed into this method which is converted into the frozen set as a return type of the method.

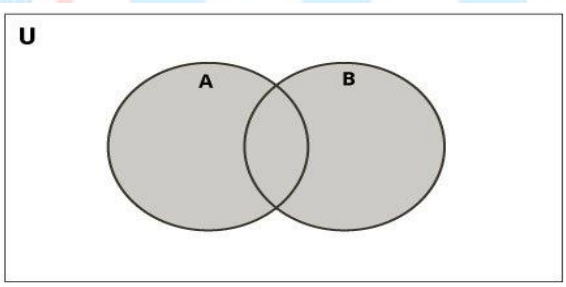
e.g. fs = frozenset([1,2,3,4])
type(fs) #<class 'frozenset'>

In the above example, list is converted into frozenset so that it became immutable.

Python Set Operations:

Sets can be used to carry out mathematical set operations like union, intersection, difference and symmetric difference. We can do this with operators or methods.

- **Set Union**



Set Union in Python

Union of A and B is a set of all elements from both sets.Union is performed using | operator. Same can be accomplished using the union() method.

```
# Set union method
# initialize A and B
A = {1, 2, 3, 4, 5}
B = {4, 5, 6, 7, 8}

# use | operator
print(A | B)
```

Output

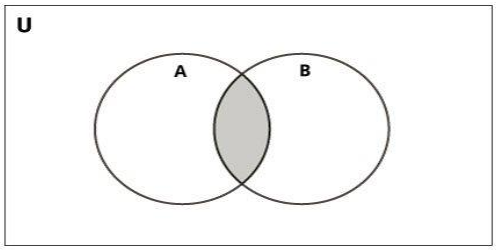
```
{1, 2, 3, 4, 5, 6, 7, 8}
```

Usage of method:

```
# use union function
>>> A.union(B)
{1, 2, 3, 4, 5, 6, 7, 8}

# use union function on B
>>> B.union(A)
{1, 2, 3, 4, 5, 6, 7, 8}
```

- **Set Intersection**



Set Intersection in Python

Intersection of A and B is a set of elements that are common in both the sets. Intersection is performed using & operator. Same can be accomplished using the intersection() method.

```
# Intersection of sets
# initialize A and B
A = {1, 2, 3, 4, 5}
B = {4, 5, 6, 7, 8}

# use & operator
print(A & B)
```

Output

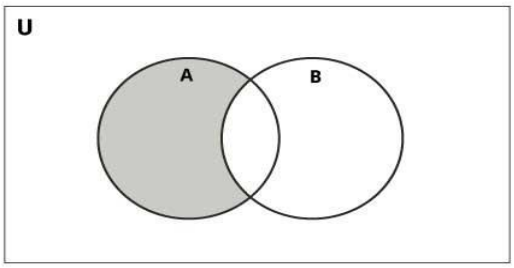
```
{4, 5}
```

Usage of method:

```
# use intersection function on A
>>> A.intersection(B)
{4, 5}

# use intersection function on B
>>> B.intersection(A)
{4, 5}
```

- **Set Difference**



Set Difference in Python

Difference of the set B from set A(A - B) is a set of elements that are only in A but not in B. Similarly, B - A is a set of elements in B but not in A. Difference is performed using - operator. Same can be accomplished using the difference() method.

```
# Difference of two sets
# initialize A and B
A = {1, 2, 3, 4, 5}
B = {4, 5, 6, 7, 8}

# use - operator on A
print(A - B)
```

Output

```
{1, 2, 3}
```

Usage of method:

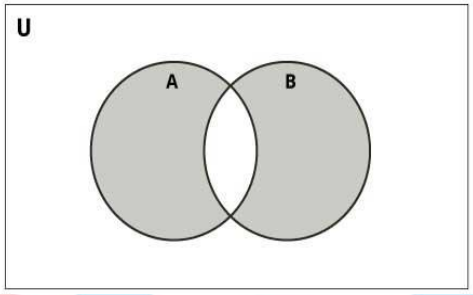
```
# use difference function on A
>>> A.difference(B)
```

```
{1, 2, 3}

# use - operator on B
>>> B - A
{8, 6, 7}

# use difference function on B
>>> B.difference(A)
{8, 6, 7}
```

- **Set Symmetric Difference**



Set Symmetric Difference in Python

Symmetric Difference of A and B is a set of elements in A and B but not in both (excluding the intersection). Symmetric difference is performed using ^ operator. Same can be accomplished using the method symmetric_difference().

```
# Symmetric difference of two sets
# initialize A and B
A = {1, 2, 3, 4, 5}
B = {4, 5, 6, 7, 8}

# use ^ operator
print(A ^ B)
```

Output

```
{1, 2, 3, 6, 7, 8}
```

Usage of method:

```
# use symmetric_difference function on A
>>> A.symmetric_difference(B)
{1, 2, 3, 6, 7, 8}
```

```
# use symmetric_difference function on B
>>> B.symmetric_difference(A)
{1, 2, 3, 6, 7, 8}
```

Other Python Set Methods:

Method	Description
add()	Adds an element to the set
clear()	Removes all elements from the set
copy()	Returns a copy of the set
difference()	Returns the difference of two or more sets as a new set
difference_update()	Removes all elements of another set from this set
discard()	Removes an element from the set if it is a member. (Do nothing if the element is not in set)
intersection()	Returns the intersection of two sets as a new set
intersection_update()	Updates the set with the intersection of itself and another
isdisjoint()	Returns True if two sets have a null intersection
issubset()	Returns True if another set contains this set
issuperset()	Returns True if this set contains another set
pop()	Removes and returns an arbitrary set element. Raises KeyError if the set is empty
remove()	Removes an element from the set. If the element is not a member, raises a KeyError

symmetric_difference()	Returns the symmetric difference of two sets as a new set
symmetric_difference_update()	Updates a set with the symmetric difference of itself and another
union()	Returns the union of sets in a new set
update()	Updates the set with the union of itself and others

Built-in Functions with Set:

Function	Description
all()	Returns True if all elements of the set are true (or if the set is empty).
any()	Returns True if any element of the set is true. If the set is empty, returns False.
len()	Returns the length (the number of items) in the set.
max()	Returns the largest item in the set.
min()	Returns the smallest item in the set.
sorted()	Returns a new sorted list from elements in the set(does not sort the set itself).
sum()	Returns the sum of all elements in the set.