Set

A set is an unordered collection of items. Every element is unique (no duplicates) and must be immutable (which cannot be changed). However, the set itself is mutable. We can add or remove items from it. Sets can be used to perform mathematical set operations like union, intersection, symmetric difference etc.

create a set

- A set is created by placing all the items (elements) inside curly braces {}, separated by comma or by using the built-in function set().
- It can have any number of items and they may be of different types (integer, float, tuple, string etc.). But a set cannot have a mutable element, like list, set or dictionary, as its element.

Example:

```
#set of integers

my_set ={1,2,3}

 print(my_set)

# set of mixed data types

my_set = {1.0, “hello” ,(1,2,3)}

print(my_set)
```

Using set()

Example:

```
my_set = set([1,2,3,2])

print(my_set)
```

Create an empty set

Empty curly braces {} will make an empty dictionary in Python. To make a set without any elements we use the set() function without any argument.

Example:

```
# initialize a with {}

a={}

# check type of a

print(type(a))

#output : <class 'dict' >

a=set()

# check type of a

print(type(a))

#output : <class 'set'>
```

Add new elements to a set

We cannot access or change an element of set using indexing or slicing. Set does not support it.

We can add single element using the add() method and multiple elements using the update() method. The update() method can take tuples, lists, strings or other sets as its argument. In all cases, duplicates are avoided.

Example:

```
sample_set = {1,2}

print(sample_set)

sample_set.add(2)

print(sample_set)

sample_set.update([1,2,7,8,9])

print(sample_set)

sample_set.update({10,1,12,3},[2,3,6])

print(sample_set)
```

Remove Elements from a set

A particular item can be removed from set using methods, discard() and remove().

The only difference between the two is that, while using discard() if the item does not exist in the set, it remains unchanged. But remove() will raise an error in such condition.

Example:

sample_set.remove(1)

sample_set.discard(2)

print(sample_set)

Similarly, we can remove and return an item using the pop() method. Set being unordered, there is no way of determining which item will be popped. It is completely arbitrary. We can also remove all items from a set using clear().

print("the removed elements are ",sample_set.pop())

Python Set Operations

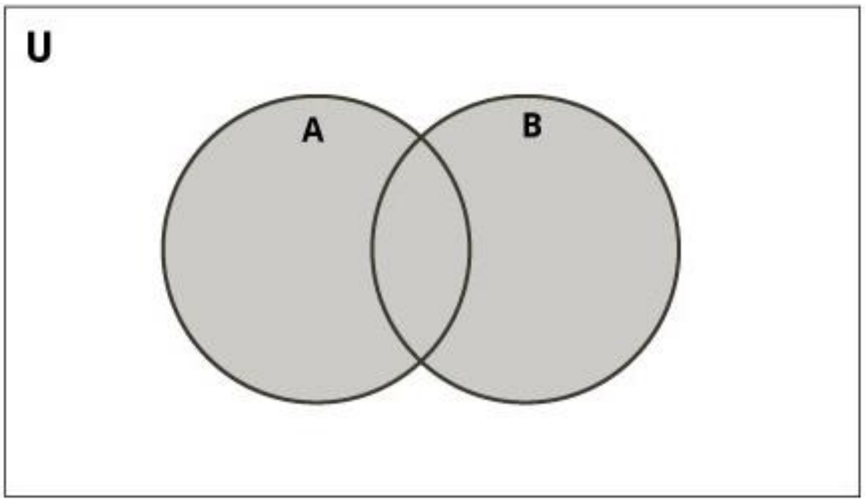
Sets can be used to carry out mathematical set operations like union, intersection, difference and symmetric difference. We can do this with operators or methods.

Let us consider the following two sets for the following operations.

A={1,2,3,4,5}

B={4,5,6,7,8}

Set Union

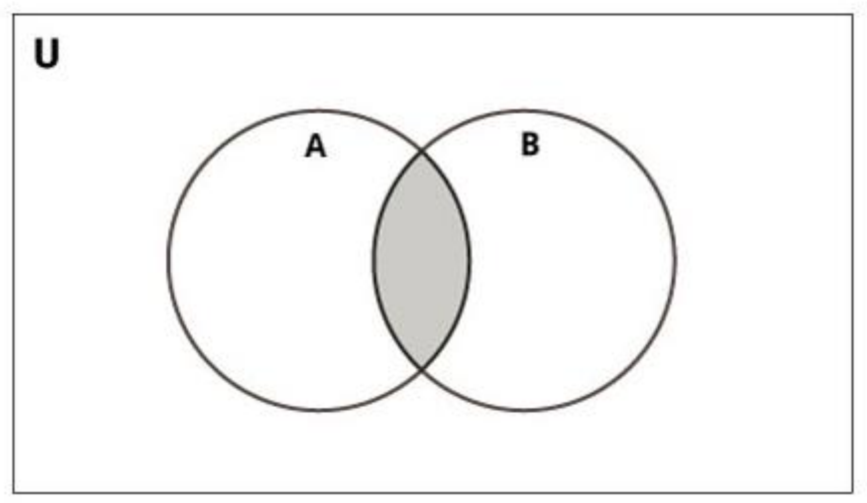


Union of A and B is a set of all elements from both sets. Union is performed using | operator. Same can be accomplished using the method union().

Example:

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

Set Intersection



Intersection of A and B is a set of elements that are common in both sets.

Intersection is performed using & operator. Same can be accomplished using the method intersection().

Example:

# initialize A and B

A={1,2,3,4,5}

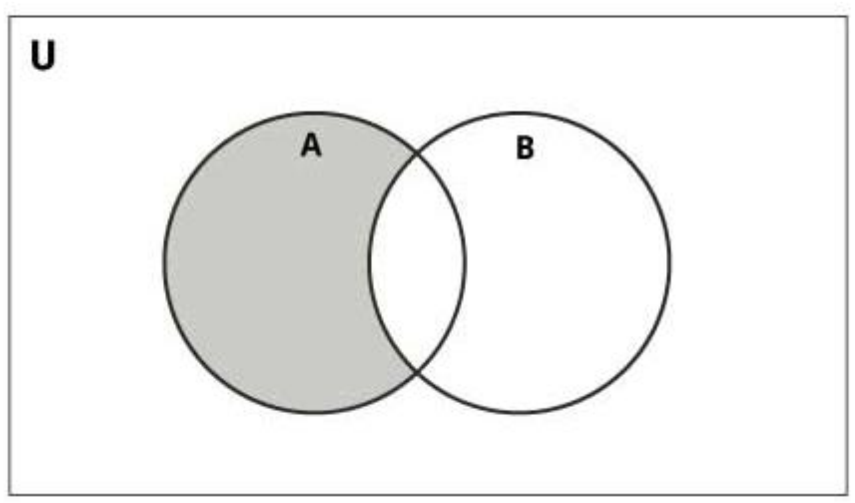
B={4,5,6,7,8}

# use & operator

Print(A & B)

# output : {4,5}

Set Difference



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

Example:

# use difference function on A

>>>A.difference(B)

{1,2,3}   #output

#use - operator on B

>>>B – A

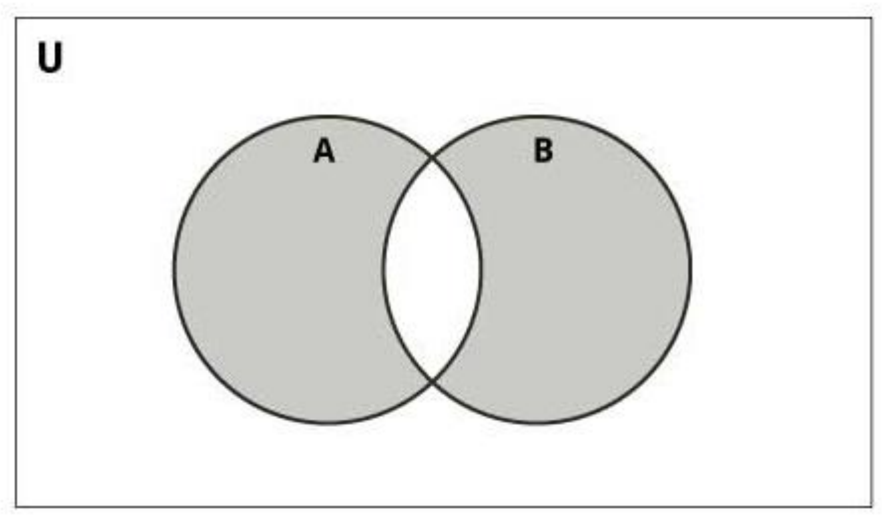
{8,6,7}   # output

# use difference function on B

>>>B.difference(A)

{8,6,7}   # output

Set Symmetric Difference



Symmetric Difference of A and B is a set of elements in both A and B except those that are common in both. Symmetric difference is performed using ^ operator. Same can be accomplished using the method symmetric_difference().

Example:

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

Python Frozenset

Frozenset is a new class that has the characteristics of a set, but its elements cannot be changed once assigned. While tuples are immutable lists, frozensets are immutable sets. Frozensets can be created using the function frozenset().

Example:

```
>>>A.isdisjoint(B)

False

>>>A.diffrerence(B)

frozenset({1,2})   #output

>>>A | B

frozenset({1,2,3,4,5,6})   #output

>>>A.add(3)

…AttributeError: ‘frozenset’  object has no attribute  ‘add’
```