Set Theory Operations

Let's take a look at how we can perform set theory operations in Python.

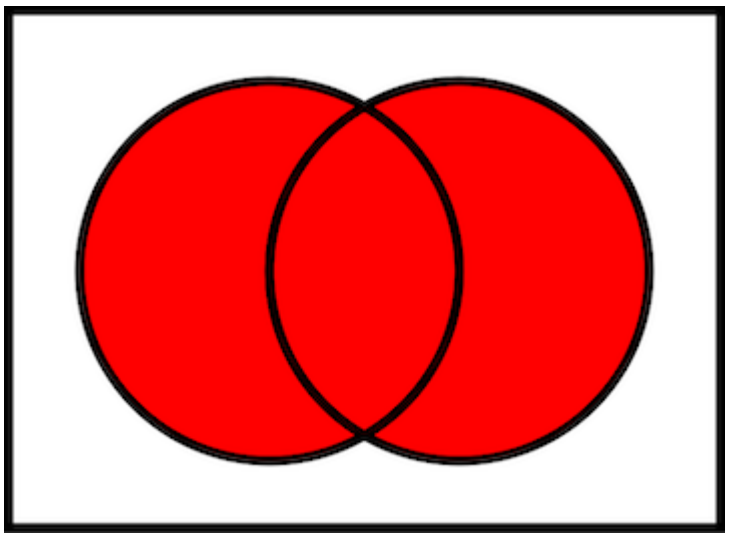
We'll cover the following

Folks familiar with mathematics will know that sets have three main operations, **union**, **intersection**, and **difference**.

The set data structure in Python supports all three.

Union

A union of two sets is the collection of all unique elements from both sets.



In Python, union can be performed using either the pipe operator, `|`, or the `union()` method:

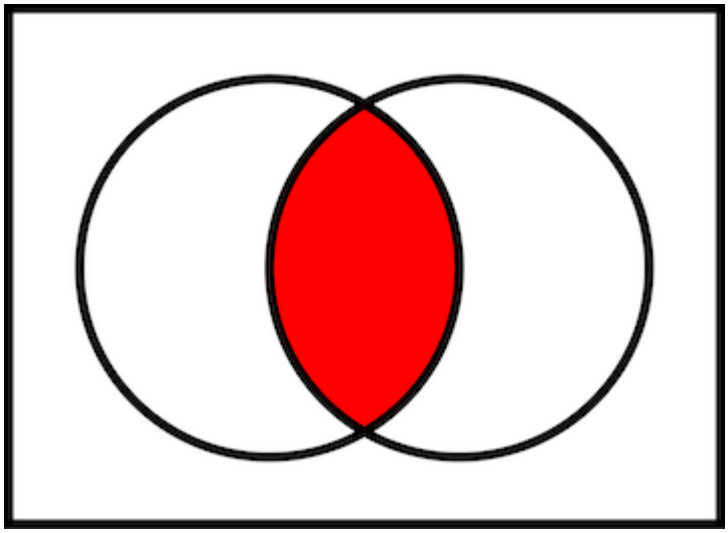
```
set_A = {1, 2, 3, 4}
set_B = {'a', 'b', 'c', 'd'}

print(set_A | set_B)
print(set_A.union(set_B))
```

```
print(set_B.union(set_A))
```

Since sets are unordered, the order of contents in the three outputs above does not matter.

Intersection

The intersection of two sets is the collection of unique elements which are common between them.

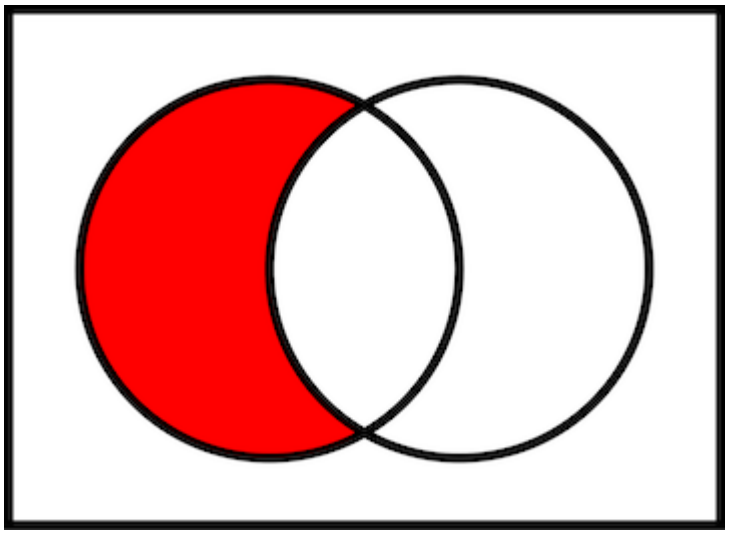


In Python, intersection can be performed using either the `&` operator or the `intersection()` method:

```
set_A = {1, 2, 3, 4}
set_B = {2, 8, 4, 16}

print(set_A & set_B)
print(set_A.intersection(set_B))
print(set_B.intersection(set_A))
```

Difference

The difference between two sets is the collection of all unique elements present in the first set but not in the second.



In Python, the difference between two sets can be found using either the `-` operator or the `difference()` method.

Do keep in mind that the difference operation is not *associative*, i.e., the positions of the sets matter.

`set_A - set_B` returns the elements which are only present in `set_A`.

`set_B - set_A` would do the opposite.

```
set_A = {1, 2, 3, 4}
set_B = {2, 8, 4, 16}


print(set_A - set_B)
print(set_A.difference(set_B))

print(set_B - set_A)
print(set_B.difference(set_A))
```

And with that, we come to the end of our tour through the data structures in Python.

In the next lesson, we'll learn how to convert one data structure to another.