# Theory: Set operations



21 minutes 10 / 10 problems solved

Start practicing

One of the main features of sets is that they allow you to perform **mathematical set operations**, such as intersection and union. That is, if you have two or more sets, you can use special methods to see which objects are contained within each of these sets, or objects present in one of them but not in the another, or get the total set of objects contained in every set at hand. Why not learn more about this useful set functionality?



## §1. Union

Before we start, it must be said that you can perform each set operation in two ways: by using an operator or by calling a method. So, first, let's see how it works with the simplest operation — union. When you perform a union on two sets, you get a new set that comprises all the elements that were within the united sets. The set method for this is (quite obviously) referred to as `union`, and it should be called as a method of one of your sets that accepts another set as an argument: `A.union(B)`. Here is an example:

```
democrats = {'Kennedy', 'Obama'}
republicans = {'Trump', 'Lincoln'}
presidents = democrats.union(republicans)
print(presidents)
# The output is {'Kennedy', 'Obama', 'Trump', 'Lincoln'}
```

You can do the same by means of the operator `|`, the syntax for which is more straightforward. You simply put the operator between your sets, just like that: `A | B`.

```
democrats = {'Kennedy', 'Obama'}
republicans = {'Trump', 'Lincoln'}
# operator
presidents = democrats | republicans
# method
also_presidents = democrats.union(republicans)
# let's compare
print(presidents == also_presidents)
# The output is True
```

You can also unite sets without creating a whole new set — just by adding all the elements from one set to another one. The operator for that is `|=` (`A |= B`) and the method is called `update`. Check this out:

```
ghostbusters = {'Peter', 'Raymond', 'Egon'}
soldiers = {'Winston'}
secretaries = {'Janine'}

ghostbusters |= soldiers
ghostbusters.update(secretaries)
print(ghostbusters)
# The output is {'Peter', 'Raymond', 'Egon', 'Winston', 'Janine'}
```

Now, what other set operations do we have?

## §2. Intersection

The intersection allows you to get only the objects that are present in each set. So, by calling the method `intersection` or by using the operator `&` in the same manner as for union, you can see what your sets have in common.

```
light_side = {'Obi-Wan', 'Anakin'}
dark_side = {'Palpatine', 'Anakin'}
both_sides = light_side.intersection(dark_side)
print(both_sides)
# The output is {'Anakin'}
print(light_side & dark_side)
# The output is {'Anakin'}
```

To delete from the first set all the elements that are absent in the second set, and leave only the elements that both sets contain, you can use the operator `&=` or the method `intersection_update`.

```
creatures = {'human', 'rabbit', 'cat'}
pets = {'rabbit', 'cat'}
creatures.intersection_update(pets)
print(creatures)
# The output is {'rabbit', 'cat'}
beasts = {'crocodile', 'cat'}
creatures &= beasts
print(creatures)
# The output is {'cat'}
```

You can probably guess the next operation we're going to tackle!

## §3. Difference

Difference operation is equal to the simple subtraction of sets: as a result, you'll get the set containing all the unique elements of the first set. The name of the method, `difference`, is as predictable as the operator `-`.

```
painters = {'Klimt', 'Michelangelo', 'Picasso'}
ninja_turtles = {'Michelangelo', 'Leonardo'}
print(painters.difference(ninja_turtles))
# The output is {'Klimt', 'Picasso'}
print(painters - ninja_turtles)
# The output is {'Klimt', 'Picasso'}
```

Similarly to previous operations, to remove from your set all the elements present in the second set without creating a new collection, you can address the `difference_update` method or the operator `-=`.

```
criminals = {'Al Capone', 'Blackbeard', 'Bonnie and Clyde'}
gangsters = {'Al Capone'}
pirates = {'Blackbeard'}

criminals.difference_update(gangsters)
criminals -= pirates
print(criminals)
# The output is {'Bonnie and Clyde'}
```

So far so good! Though, we still haven't mentioned a couple of details you may find important.

## §4. Methods and operators: what's the difference?

Mind that syntax is not the only difference between using a set operation method and an operator. More importantly, a set operator requires both arguments to be sets, while the method only demands this from the first one — and the second argument can be any iterable object, for example, a list or a string. In this case, the method will create a set out of the second argument implicitly (that is, by itself, without your interference).

```
santa_claus_sound = set('ho ho ho')
pirate_sound = 'yo ho ho'
ho_sound = santa_claus_sound.intersection(pirate_sound)
print(ho_sound)
# The output is {'h', 'o', ' '}
print(santa_claus_sound & pirate_sound)
# Causes TypeError
```

Also, sometimes it happens that you don't have specific variables for each of your sets, for example, you store them all in some container. How do you quickly find an intersection or a union of all these nameless sets? With the help of the operation set method and the asterisk (`*`) operator (it is used to "unpack" containers; let's not go into details of this operator now, though): `set.method(*list_of_sets)`. Watch this:

```
# sets are within a container

languages = [{'c', 'c++', 'python'}, {'python', 'javascript'}, {'python', 'java'}]
the_best = set.intersection(*languages)
print(the_best)
# The output is {'python'}
```

So, now when you know all this, set operations will faithfully serve you!

> Do not forget about frozenset — this data type also supports all of the above operations.

## §5. Conclusions

Let's sum it up:

- you can perform union, intersection and difference operations on sets,
- each operation has two versions: one of them returns a new set and another updates the existing one,
- there are two ways of calling each operation: by method and by operator.

Report a typo

**132** users liked this theory. **2** didn't like it. **What about you?**

Start practicing

Comments (4) Hints (0) Useful links (0) Show discussion