Day 19: Sets Method

Join Sets

There are several ways to join two or more sets in Python:

- The `union()` and `update()` methods joins all items from both sets.
- The `intersection()` method keeps ONLY the duplicates.
- The `difference()` method keeps the items from the first set that are not in the other set(s).
- The `symmetric_difference()` method keeps all items EXCEPT the duplicates.

Union (|) method

The `union()/ (|)` method returns a new set with all items from both sets.

```
set1 = {"a", "b", "c"}
set2 = {1, 2, 3}

set3 = set1.union(set2)
print(set3)

# Join multiple sets
set1 = {"a", "b", "c"}
set2 = {1, 2, 3}
set3 = {"John", "Elena"}
set4 = {"apple", "bananas", "cherry"}

myset = set1.union(set2, set3, set4)
print(myset)

myset = set1 | set2 | set3 |set4
print(myset)

# Join a set & tuple
```

```
x = {"a", "b", "c"}
y = (1, 2, 3)

z = x.union(y)
print(z)
```

Update() method

- The `update()` method inserts all items from one set into another.
- The `update()` changes the original set, and does not return a new set.

```
set1 = {"a", "b" , "c"}
set2 = {1, 2, 3}

set1.update(set2)
print(set1)
```

- Both `union()` and `update()` will exclude any duplicates.

Intersection(&) method

- The `intersection()/ (&)` method will return a new set, that only contains the items that are present in both sets.
- The `intersection_update()` method will also keep ONLY the duplicates, but it will change the original set instead of returning a new set.

```
set1 = {"apple", "banana", "cherry"}
set2 = {"google", "microsoft", "apple"}

set3 = set1.intersection(set2)
print(set3)

set3 = set1 & set2
print(set3)
```

```
# intersection_update()
set1 = {"apple", "banana", "cherry"}
set2 = {"google", "microsoft", "apple"}

set1.intersection_update(set2)
print(set1)
```

- Only sets to sets allowed, not sets to tuples.

Difference(-) method

- The `difference()/ (-)` method will return a new set that will contain only the items from the first set that are not present in the other set.
- The `difference_update()` method will also keep the items from the first set that are not in the other set, but it will change the original set instead of returning a new set.

```
set1 = {"apple", "banana", "cherry"}
set2 = {"google", "microsoft", "apple"}

set3 = set1.difference(set2)
print(set3)

# difference_update()
set1 = {"apple", "banana", "cherry"}
set2 = {"google", "microsoft", "apple"}

set1.difference_update(set2)
print(set1)
```

symmetric_difference(^) method

- The `symmetric_difference()/ (^)` method will keep only the elements that are NOT present in both sets.

- The `symmetric_difference_update()` method will also keep all but the duplicates, but it will change the original set instead of returning a new set.

```
set1 = {"apple", "banana", "cherry"}
set2 = {"google", "microsoft", "apple"}

set3 = set1.symmetric_difference(set2)
print(set3)

set3 = set1 ^ set2
print(set3)

# symmetric_difference_update()
set1 = {"apple", "banana", "cherry"}
set2 = {"google", "microsoft", "apple"}

set1.symmetric_difference_update(set2)
print(set1)
```