Sets

What is Sets?

- Sets are used to store multiple items in a single variable.
- A set is a collection which is *unordered*, *unchangeable**, *unindexed and do not allow duplicate values*.
- Sets are written with curly brackets. `{}`
- To create empty set, we cant use just `{}` , instead we use `x = set{}`
- Set doesn't use Indexing, Instead they use Hashing to store variables.

```
thisset = {"apple", "banana", "cherry"}
print(thisset) # {'banana', 'apple', 'cherry'}

# Duplicate not allowed
thisset = {"apple", "banana", "cherry", "apple"}

print(thisset) # {'cherry', 'apple', 'banana'}
```

💡 Note: the set list is unordered, meaning: the items will appear in a random order.

💡 **Note:**

- The values `True` and `1` are considered the same value in sets, and are treated as duplicates
- The values `False` and `0` are considered the same value in sets, and are treated as duplicates:

```
thisset = {"apple", "banana", "cherry", True, 1, 2}

print(thisset) # {True, 2, 'banana', 'cherry', 'apple'}
```

- Once a set is created, you cannot change its items, but you can remove items and add new items.

Add Items

💡 Once a set is created, you cannot change its items, but you can add new items.

To add one item to a set use the `add()` method.

```
thisset = {"apple", "banana", "cherry"}

thisset.add("orange")

print(thisset) # {'cherry', 'apple', 'orange', 'banana'}
```

Add Sets

To add items from another set into the current set, use the `update()` method.

```
thisset = {"apple", "banana", "cherry"}
tropical = {"pineapple", "mango", "papaya"}

thisset.update(tropical)

print(thisset) # {'cherry', 'pineapple', 'apple', 'banana', 'papaya', 'mango'}
```

Remove Item

To remove an item in a set, use the `remove()`, or the `discard()` method.

```
thisset1 = {"apple", "banana", "cherry"}

thisset.remove("banana")

thisset2 = {"apple", "banana", "cherry"}

thisset.discard("banana")

print(thisset1) # {'cherry', 'apple'}

print(thisset2) # {'cherry', 'apple'}
```

💡

- If the item to remove does not exist, `remove()` will raise an error.
- If the item to remove does not exist, `discard()` will **NOT** raise an error.

```
To clear the Content of Set
```

```
thisset = {"apple", "banana", "cherry"}

thisset.clear()

print(thisset) # set()
```

```
To delete whole Set

thisset = {"apple", "banana", "cherry"}

del thisset

print(thisset) # NameError: name 'thisset' is not defined
```

Join Sets

Union

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) # set1 | set2

print(set3) # {'c', 1, 2, 'b', 'a', 3}
```

```
Multiple Join

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

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

# {2, apple, 'c', 3, 'a', Elena, John, banana, 1, cherry, 'b'}
```

```
Joining Tuple and Set

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

z = x.union(y)
print(z) # {'c', 'a', 1, 'b', 3, 2}
```

💡 The `|` operator only allows you to join sets with sets, and not with other data types like you can with the `union()` method.

Update

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) # {2, 1, 'a', 'b', 3, 'c'}
```

💡 Both `union()` and `update()` will exclude any duplicate items.

Intersection

Keep ONLY the duplicates

The `intersection()` method will return a new set, that only contains the items that are present in both sets.

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

set3 = set1.intersection(set2) # set1 & set2

print(set3) # {'apple'}
```

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"}

set1.intersection_update(set2)

print(set1) # {'apple'}
```

Difference

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.

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

```
set3 = set1.difference(set2) # set1 - set2

print(set3) # {'banana', 'cherry'}
```

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"}

set1.difference_update(set2)

print(set1) # {'banana', 'cherry'}
```

Symmetric Differences

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

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

set3 = set1.symmetric_difference(set2) # set1 ^ set2

print(set3) # {'google', 'banana', 'microsoft', 'cherry'}
```