**# Sets are used to store multiple items in a single variable.**

**# A set is a collection which is unordered, unchangeable*, and unindexed.**

**#  Note: Set items are unchangeable, but you can remove items and add new items.**

**# Set Items**

**# Set items are unordered, unchangeable, and do not allow duplicate values.**

**# Unchangeable**

**# Set items are unchangeable, meaning that we cannot change the items after the set has been created.**

**# Duplicate values will be ignored:**

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

print(thisset)
```

**#  The values True and 1 are considered the same value in sets, and are treated as duplicates:**

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

print(thisset)
```

**# The values False and 0 are considered the same value in sets, and are treated as duplicates:**

```
thisset = {"apple", "banana", "cherry", False, True, 0}

print(thisset)
```

**Get the number of items in a set:**

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

print(len(thisset))
```

**#  String, int and boolean data types:**

```
set1 = {"apple", "banana", "cherry"}

set2 = {1, 5, 7, 9, 3}

set3 = {True, False, False}


print(set1)

print(set2)

print(set3)
```

**From Python's perspective, sets are defined as objects with the data type 'set':**

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

print(type(myset))
```

**# The set() Constructor**

**# It is also possible to use the set() constructor to make a set.**

```
thisset = set(("apple", "banana", "cherry"))

print(thisset)
```

**# Access Items**

**# You cannot access items in a set by referring to an index or a key.**

**# But you can loop through the set items using a for loop, or ask if a specified value is present in a set, by using the in keyword.**

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

for x in thisset:

  print(x)
```

**# Check if "banana" is present in the set it will print True**

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

print("banana" in thisset)
```

**# Check if "banana" is NOT present in the set:**

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

print("banana" not in thisset)
```

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

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

thisset.add("orange")

print(thisset)
```

**# 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)
```

**# Add Any Iterable:**

**# The object in the update() method does not have to be a set, it can be any iterable object (tuples, lists, dictionaries etc.).**

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

mylist = ["kiwi", "orange"]

thisset.update(mylist)

print(thisset)
```

```
# Remove Item

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

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

thisset.remove("banana")

print(thisset)
```

```
# Remove "banana" by using the discard() method:

# Note: If the item to remove does not exist, discard() will NOT raise an error.

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

thisset.discard("banana")

print(thisset)
```

```
# Remove a random item by using the pop() method:

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

x = thisset.pop()

print(thisset)
```

**The clear() method empties the set:**

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

thisset.clear()

print(thisset)
```

```
# The del keyword will delete the set completely:

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

del thisset

print(thisset)
```

```python
# Loop through the set, and print the values:

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

for x in thisset:

  print(x)


# Python - Join Sets:


# 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)


# You can use the | operator instead of the union() method, and you will get the same result.

set1 = {"a", "b", "c"}

set2 = {1, 2, 3}

set3 = set1 | set2

print(set3)


# Join multiple sets with the union() method:

set1 = {"a", "b", "c"}

set2 = {1, 2, 3}

set3 = {"John", "Elena"}

set4 = {"apple", "bananas", "cherry"}

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

print(myset)
```

```
# Use | to join multiple  sets:
set1 = {"a", "b", "c"}
set2 = {1, 2, 3}
set3 = {"John", "Elena"}
set4 = {"apple", "bananas", "cherry"}
myset = set1 | set2 | set3 |set4
print(myset)
```

```
# Join a Set and a Tuple
# The union() method allows you to join a set with other data types, like lists or tuples.
x = {"a", "b", "c"}
y = (1, 2, 3)
z = x.union(y)
print(z)
```

```
# The update() method inserts the items in set2 into set1:
set1 = {"a", "b" , "c"}
set2 = {1, 2, 3}
set1.update(set2)
print(set1)
```

```
# Intersection
# Keep ONLY the duplicates
set1 = {"apple", "banana", "cherry"}
set2 = {"google", "microsoft", "apple"}
set3 = set1.intersection(set2)
print(set3)
```

**You can use the & operator instead of the intersection() method, and you will get the same result.**

set1 = {"apple", "banana", "cherry"}

set2 = {"google", "microsoft", "apple"}

set3 = set1 & set2

print(set3)

**# Note: The & operator only allows you to join sets with sets, and not with other data types like you can with the intersection() method.**

**# The intersection_update() method will also keep ONLY the duplicates**

set1 = {"apple", "banana", "cherry"}

set2 = {"google", "microsoft", "apple"}x

set1.intersection_update(set2)

print(set1)

**# The values True and 1 are considered the same value. The same goes for False and 0.**

**# Join sets that contains the values True, False, 1, and 0, and see what is considered as duplicates:**

set1 = {"apple", 1,  "banana", 0, "cherry"}

set2 = {False, "google", 1, "apple", 2, True}

set3 = set1.intersection(set2)

print(set3)

**# 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)

print(set3)


# You can use the - operator instead of the difference() method, and you will get the same result.

set1 = {"apple", "banana", "cherry"}

set2 = {"google", "microsoft", "apple"}

set3 = set1 - set.

print(set3)


# The difference_update() method will also keep the items from the first set that are not in the other set

set1 = {"apple", "banana", "cherry"}

set2 = {"google", "microsoft", "apple"}

set1.difference_update(set2)

print(set1)


# 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)

print(set3)
```

**# You can use the ^ operator instead of the symmetric_difference() method, and you will get the same result**.

```
set1 = {"apple", "banana", "cherry"}

set2 = {"google", "microsoft", "apple"}

set3 = set1 ^ set2

print(set3)
```

**# Note: The ^ operator only allows you to join sets with sets, and not with other data types like you can with the symmetric_difference() method.**

```
#The symmetric_difference_update() method will also keep all but the duplicates

set1 = {"apple", "banana", "cherry"}

set2 = {"google", "microsoft", "apple"}

set1.symmetric_difference_update(set2)

print(set1)
```