Set

Ques. What is Set?

- Sets are used to store multiple items in a single variable.
- Duplicate items are not allowed.
- A set is a collection which is both **unordered**, **unindexed** and **unchangeable**.
- Set items are **unchangeable**, but you can remove items and add new items.
- Sets are **unordered**, so we cannot be sure in which order the items will appear.
- Sets are written with **curly{}** brackets.
- Sets **do not allow duplicate** values.
- Example:-

```
thisset = {"apple", "banana", False, "cherry", 0, "apple", True, 1, 2}
print(thisset)

Output:- {True, 2, 'cherry', 'apple', 'banana'}
again hit output:- {False, True, 2, 'cherry', 'banana', 'apple'}
```

Note:-

- Sets are unordered, so you cannot be sure in which order the items will appear.

- **Unordered** means that the items in a set do not have a defined order.
- **Sets are unchangeable**, meaning that we cannot change the items after the set has been created. (Once a set is created, you cannot change its items, but you can add new items.)
- Sets **do not allow duplicate** values.

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

Output:- {'banana', 'apple', 'cherry'}
```

- Set items can be of any data type.(String, int and boolean)

```
set1 = {"abc", 34, True, 40, "male"}
print(set1)
output:- {True, 34, 40, 'male', 'abc'}
```

- 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)

Output:- {True, 2, 'banana', 'cherry', 'apple'}
```

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

```
thisset = {"apple", "banana", "cherry", False, 16, 0}
print(thisset)

Output:- {False, 16, 'banana', 'cherry', 'apple'}
```

Ques. Get the Length of a Set?

```
thisset = {"apple", "banana", "cherry"}
print(len(thisset))
Output:- 3
```

Ques. set() Constructor?

- It is also possible to use the set() constructor to make a set(the double round-brackets).

```
thisset = set(("apple", "banana", "cherry"))
print(thisset)
output:- {'apple', 'banana', 'cherry'}
```

Ques. Loop Sets?

```
# Loop Items
thisset = {"apple", "banana", "cherry"}
for x in thisset:
  print(x)

Output:-
banana
cherry
apple
```

Ques. Access Items of set?

- Loop through the set, and print the values.

```
thisset = {"apple", "banana", "cherry"}
for x in thisset:
  print(x)

output:-
apple
cherry
banana
```

- Check if "banana" is present in the set.

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

output:- True
```

- Check if "banana" is NOT present in the set:

```
thisset = {"apple", "banana", "cherry"}
print("banana" not in thisset)

Outpur:- False
```

Add Items of set

Ques. Add Items of set?

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

```
thisset = {"apple", "banana", "cherry"}
thisset.add("orange")
print(thisset)

Output:- {'cherry', 'orange', 'apple', 'banana'}
```

- Try to add an element that already exists.

```
thisset = {"apple", "banana", "cherry"}
thisset.add("apple")
print(thisset)

Output:- {'banana', 'apple', 'cherry'}
```

- **update() method:-**
  - To add items from another set into the current set, use the update() method.
  - **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"}
tropical = {"pineapple", "mango", "papaya"}
thisset.update(tropical)
print(thisset)

output:- {'apple', 'mango', 'cherry', 'pineapple', 'banana', 'papaya'}
```

```
thisset = {"apple", "banana", "cherry"}
mylist = ["kiwi", "orange"]
thisset.update(mylist)
print(thisset)

Output:- {'banana', 'cherry', 'apple', 'orange', 'kiwi'}
```

Remove Set

Ques. Remove Item of set?

- **remove() method:-** If the item to remove does not exist, remove() will raise an error.

```
thisset = {"apple", "banana", "cherry"}
thisset.remove("banana")
print(thisset)

Output:- {'apple', 'cherry'}
```

```
thisset = {"apple", "banana", "cherry"}
thisset.remove("")
print(thisset)

Output:-
throew error
```

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

```
thisset = {"apple", "banana", "cherry"}
thisset.discard("banana")
print(thisset)

Output:- {'apple', 'cherry'}
```

```
thisset = {"apple", "banana", "cherry"}
thisset.discard("")
print(thisset)

Output:- {"cherry", "apple", "banana"}
```

- **pop() method:-** you can also use the **pop()** method to remove an item, but this method will remove the last item. Remember that sets are unordered, so you will not know what item that gets removed.

```
thisset = {"apple", "banana", "cherry"}
x = thisset.pop()
print(x) #removed item
print(thisset) #the set after removal

Output:-
```

```
banana
{'cherry', 'apple'}
```

- The **clear()** method empties the set.

```
thisset = {"apple", "banana", "cherry"}
thisset.clear()
print(thisset)

output:- set()
```

- The **del** keyword will delete the set completely.

```
----------------------------------------------------------------------
# The del keyword will delete the set completely:
thisset = {"apple", "banana", "cherry"}
del thisset
print(thisset) #this will raise an error because the set no longer exists

output:- Error
```

Ques. What is difference between Discard() and Remove()?

- The remove() method will raise an error if the specified item does not exist, and the discard() method will not.

Join set

Ques. Join Two Set?

- **NOTE:-** Note: Both union() and update() will exclude any duplicate items.
- The **union()** method combines elements from two or more sets into a new set, eliminating duplicates.

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

# Using the union() method
union_set_method = set1.union(set2)
print(union_set_method)  # Output:  {'c', 'a', 1, 2, 3, 'b'}

# Using the | operator
union_set_operator = set1 | set2
print(union_set_operator)  # Output: {'c', 'a', 1, 2, 3, 'b'}
```

- The **update()** method to add elements from another set or any other iterable (like lists, tuples, or dictionaries) into an existing set.

```
set1 = {"a", "b" , "c"}
set2 = {1, 2, 3}
set1.update(set2)
print(set1)

output:- {'b', 'c', 1, 'a', 2, 3}
```

- The **intersection()** intersection() method returns a **new set** with an element that is **common** to all set

```
# syntex
# Using the intersection() method
set.intersection(set1, set2 ... etc.)
OR
# Using the & operator
set & set1 & set2 ... etc.
```

```
set1 = {2, 4, 5, 6}
set2 = {4, 6, 7, 8}
set3 = {4, 6, 8}

# intersection of two sets
print(set1.intersection(set2))  # Output:- {4, 6}
print(set1 & set2)              # Output:- {4, 6}
```

```
# intersection of three sets
print(set1.intersection(set2, set3)) # Output:- {4, 6}
```

- The **intersection_update()** method will keep only the items that are present in both sets.

```
x = {"apple", "banana", "cherry"}
y = {"google", "microsoft", "apple"}
x.intersection_update(y)
print(x)

output:- {'apple'}
```

- The **symmetric_difference()** method will return a new set, that contains only the elements that are NOT present in both sets.

```
set1 = {1, 2, 3, 4, 5}
set2 = {4, 5, 6, 7, 8}

# Using the symmetric_difference() method
symmetric_difference_set = set1.symmetric_difference(set2)
print(symmetric_difference_set)  # Output: {1, 2, 3, 6, 7, 8}

# Using the ^ operator
symmetric_difference_set2 = set1 ^ set2
print(symmetric_difference_set2) # Output: {1, 2, 3, 6, 7, 8}
```

- The **symmetric_difference_update()** method will keep only the elements that are NOT present in both sets.

```
set1 = {2, 4, 5, 6}
set2 = {4, 6, 7, 8}

set1.symmetric_difference_update(set2)
print(set1)

output:- {2, 5, 7, 8}
```

Set Method

Ques. Set Methods?

Method	Description
add()	Adds an element to the set
copy()	Returns a copy of the set
difference()	Returns a set containing the difference between two or more sets
difference_update()	Removes the items in this set that are also included in another, specified set
isdisjoint()	Returns whether two sets have a intersection or not
issubset()	Returns whether another set contains this set or not
issuperset()	Returns whether this set contains another set or not
remove()	Removes the specified element
discard()	Remove the specified item
pop()	Removes an element from the set
clear()	Removes all the elements from the set
union()	Return a set containing the union of sets
update()	Update the set with the union of this set and others
intersection()	Returns a set, that is the intersection of two other sets
intersection_update()	Removes the items in this set that are not present in other, specified set(s)
symmetric_difference()	Returns a set with the symmetric differences of two sets
symmetric_difference_update()	inserts the symmetric differences from this set and another

- **add():-** Adds an element to the set.

```
thisset = {"apple", "banana", "cherry"}
thisset.add("orange")
print(thisset)

Output:- {'cherry', 'orange', 'apple', 'banana'}

# Example2:- Try to add an element that already exists:
thisset = {"apple", "banana", "cherry"}
thisset.add("apple")
print(thisset)
```

```
Output:- {'apple', 'banana', 'cherry'}
```

- **clear():-** Removes all the elements from the set.

```
thisset = {"apple", "banana", "cherry"}
thisset.clear()
print(thisset)

Output:- set()
```

- **copy():-** Returns a copy of the set.

```
fruits = {"apple", "banana", "cherry"}
x = fruits.copy()
print(x)

Output:- {'banana', 'apple', 'cherry'}
```

- **difference():-** Returns a set containing the difference between two or more sets.

```
x = {"apple", "banana", "cherry"}
y = {"google", "microsoft", "apple"}

z = x.difference(y)

print(z)

Output:- {'cherry', 'banana'}

# Example2:- Reverse the first example. Return a set that contains the items that
only exist in set y, and not in set x:
x = {"apple", "banana", "cherry"}
y = {"google", "microsoft", "apple"}

z = y.difference(x)

print(z)

Output:- {'microsoft', 'google'}
```

difference_update():-

- Removes the items in this set that are also included in another, specified set.
- The difference_update() method removes the items that exist in both sets.

- The difference_update() method is different from the difference() method, because the difference() method returns a new set, without the unwanted items, and the difference_update() method removes the unwanted items from the original set.

```
x = {"apple", "banana", "cherry"}
y = {"google", "microsoft", "apple"}

x.difference_update(y)

print(x)

Output:- {'banana', 'cherry'}
```

- **discard():-**
- Remove the specified item.
- This method is different from the remove() method, because the remove() method will raise an error if the specified item does not exist, and the discard() method will not.

```
thisset = {"apple", "banana", "cherry"}
thisset.discard("banana")

print(thisset)

Output:- {'cherry', 'apple'}
```

- **intersection():-** Returns a set, that is the intersection of two other sets

```
x = {"a", "b", "c"}
y = {"c", "d", "e"}
z = {"f", "g", "c"}

result = x.intersection(y, z)

print(result)

Output:- {'c'}
```

- **intersection_update():-** Removes the items in this set that are not present in other, specified set(s).
- The intersection_update() method is different from the intersection() method, because the intersection() method returns a new set, without the unwanted items, and the intersection_update() method removes the unwanted items from the original set.

```
x = {"a", "b", "c"}
y = {"c", "d", "e"}
z = {"f", "g", "c"}
```

```
x.intersection_update(y, z)

print(x)

Output:- {'c'}
```

- **isdisjoint():-** Returns whether two sets have a intersection or not.
- Return True if no items in set x is present in set y.

```
x = {"apple", "banana", "cherry"}
y = {"google", "microsoft", "facebook"}

z = x.isdisjoint(y)

print(z)

Output:- True
```

- **issubset():-** Returns whether another set contains this set or not
- Return True if all items in set x are present in set y.

```
x = {"a", "b", "c"}
y = {"f", "e", "d", "c", "b", "a"}

z = x.issubset(y)

print(z)

Output:- True
```

- **issuperset():-** Returns whether this set contains another set or not
- Return True if all items set y are present in set x:

```
x = {"f", "e", "d", "c", "b", "a"}
y = {"a", "b", "c"}

z = x.issuperset(y)

print(z)

Output:- True
```

- **pop():-** The pop() method removes a random item from the set.

```
fruits = {"apple", "banana", "cherry"}
fruits.pop()
print(fruits)

Output:- {'apple', 'banana'}
```

- **remove():-** The remove() method removes the specified element from the set.
- This method is different from the discard() method, because the remove() method will raise an error if the specified item does not exist, and the discard() method will not.

```
fruits = {"apple", "banana", "cherry"}
fruits.remove("banana")
print(fruits)

Output:- {'cherry', 'apple'}
```

- **symmetric_difference():-** Return a set that contains all items from both sets, except items that are present in both sets.

```
x = {"apple", "banana", "cherry"}
y = {"google", "microsoft", "apple"}

z = x.symmetric_difference(y)

print(z)

Output:- {'google', 'microsoft', 'banana', 'cherry'}
```

- **symmetric_difference_update():-** Remove the items that are present in both sets, AND insert the items that is not present in both sets.

```
x = {"apple", "banana", "cherry"}
y = {"google", "microsoft", "apple"}

x.symmetric_difference_update(y)

print(x)

Output:- {'google', 'banana', 'microsoft', 'cherry'}
```

- **union():-** Return a set containing the union of sets

```
x = {"a", "b", "c"}
y = {"f", "d", "a"}
z = {"c", "d", "e"}

result = x.union(y, z)

print(result)

Output:- {'b', 'e', 'f', 'd', 'c', 'a'}
```

- **update():-** The update() method updates the current set, by adding items from another set (or any other iterable).

```
x = {"apple", "banana", "cherry"}
y = {"google", "microsoft", "apple"}

x.update(y)
print(x)

Output:- {'microsoft', 'banana', 'cherry', 'google', 'apple'}
```