Python Set

set is the collection of the unordered items.

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

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

Set items are unchangeable, but you can remove items and add new items.

Sets cannot have two items with the same value.

Unlike other collections in Python, there is no index attached to the elements of the set, i.e., we cannot directly access any element of the set by the index.

```
days = {"Monday", "Tuesday", "Wednesday", "Thursday", "Friday", "Saturday",
"Sunday"}
print(days)
print(type(days))

Days = set(["Monday", "Tuesday", "Wednesday", "Thursday", "Friday", "Saturday",
"Sunday"])
set3 = {}
print(type(set3))
```

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

Days = set(["Monday", "Tuesday", "Wednesday", "Thursday", "Friday", "Saturday", "Sunday"])

Days .add("Jan")

Print(Days)

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

```
s={12,13,14}
s1={21,22,23}
s.update(s1)
print(s)
```

iterable in list to set

```
a1 = {12,13,14}
a2 = [21,22]
a1.update(a2)
print(a1)
```

Remove Item

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

```
a1 = {12,13,14}
a1.remove(12)
print(a1)
```

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

```
a1 = {12,13,14}
a1.discard(12)
print(a1)
```

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

Pop()

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.

The return value of the pop() method is the removed item.

```
a1 = {12,13,14}
a1.pop()
print(a1)
```

Note: Sets are *unordered*, so when using the pop() method, you do not know which item that gets removed.

The clear() method empties the set:

```
a1 = {12,13,14}
a1.clear()
print(a1)
```

The `del` keyword will delete the set completely:

```
a1 = {12,13,14}
del a1
print(a1)
```

Python Set Operations

Union of two Sets

The union of two sets is calculated by using the pipe (|) operator. The union of the two sets contains all the items that are present in both the sets.

```
d1 = {"Monday",  "Friday", "Saturday", "Sunday","Sunday"}
d2={"Monday", "Tuesday", "Wednesday"}

print(d1|d2)
```

Using union() method

```
d1 = {"Monday",  "Friday", "Saturday", "Sunday","Sunday"}
d2={"Monday", "Tuesday", "Wednesday"}

print(d1.union(d2))
```

You can use the `union()` method that returns a new set containing all items from both sets, or the `update()` method that inserts all the items from one set into another:

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

Intersection of two sets

The intersection of two sets can be performed by the **and &** operator or the **intersection() function**.

```
d1 = {"Monday",  "Friday", "Saturday", "Sunday","Sunday"}
d2={"Monday", "Tuesday", "Wednesday"}
print(d1&d2)
```

Using intersection() method

```
d1 = {"Monday",  "Friday", "Saturday", "Sunday","Sunday"}
d2={"Monday", "Tuesday", "Wednesday"}
```

```
print(d1.intersection(d2))
```

The intersection_update() method

The **intersection_update()** method removes the items from the original set that are not present in both the sets (all the sets if more than one are specified).

The **intersection_update()** method is different from the intersection() method since it modifies the original set by removing the unwanted items, on the other hand, the intersection() method returns a new set.

```
d1 = {"gautam",  "creative", "design"}
d2={"dharmik","mohit","gautam"}
d1.intersection_update(d2)
print(d1)
```

Difference between the two sets

The difference of two sets can be calculated by using the subtraction (-) operator or **difference()** method.

```
d1 = {"gautam",  "creative", "design"}
d2={"dharmik","mohit","gautam"}
print(d1-d2)
```

Using Difference method

```
d1 = {"gautam",  "creative", "design"}
d2={"dharmik","mohit","gautam"}
print(d1.difference(d2))
```

Symmetric Difference of two sets

The symmetric difference of two sets is calculated by ^ operator or **symmetric_difference()** method. Symmetric difference of sets, it removes that element which is present in both sets.

```
set1 = {"a", "b" , "c"}
set2 = {"a", 2, 3}
set3=set1.symmetric_difference(set2)
print(set3)
```

symmetric_difference_update()

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

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

Set Methods

Python has a set of built-in methods that you can use on sets.

Method	Description
add()	Adds an element to the set
clear()	Removes all the elements from 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
discard()	Remove the specified item
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)
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
pop()	Removes an element from the set
remove()	Removes the specified element
symmetric_difference()	Returns a set with the symmetric differences of two sets
symmetric_difference_update()	inserts the symmetric differences from this set and another
union()	Return a set containing the union of sets
update()	Update the set with the union of this set and others