Joining Sets

Table of contents

Introduction

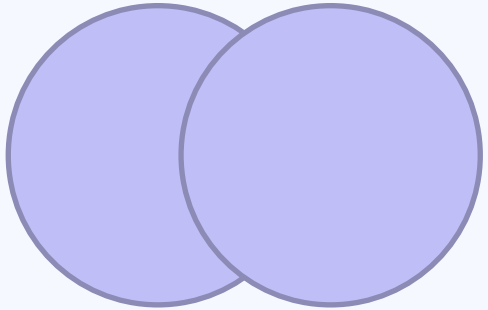
Python provides **multiple ways** to join sets using **set operations.**

They are
- Union (|)
- Intersection (&)
- Difference (-)
- Symmetric difference. (^)

These operations help in **combining** or **filtering elements** based on their presence in multiple sets.

Union (| or .union())

- The **union** operation combines all unique elements from two or more sets.
- Duplicate values are removed automatically.



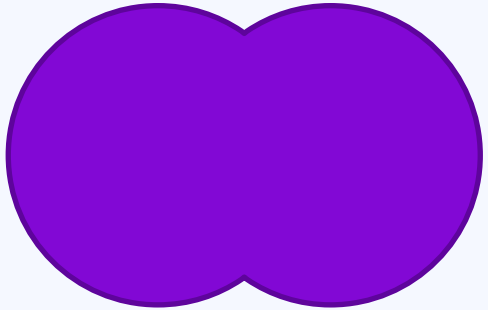
Syntax

```
set1 | set2
set1.union(set2, set3, ...)
```

- The **|** operator joins only **sets**, unlike **union()**, which works with other data types.

Union (| or .union())

- The **union** operation combines all unique elements from two or more sets.
- Duplicate values are removed automatically.



Syntax

```
set1 | set2
set1.union(set2, set3, ...)
```

- The **|** operator joins only **sets**, unlike **union()**, which works with other data types.

Union (| or .union())

- Example

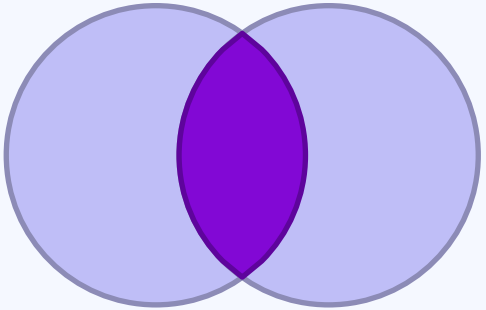
Python	Output

A = {1, 2, 3}
B = {3, 4, 5}
 	
# Using | operator 	
print(A | B)
 	
# Using union() method 	
print(A.union(B))
	{1, 2, 3, 4, 5}
	
	{1, 2, 3, 4, 5}

Intersection (& or .intersection())

The **intersection** operation returns only the elements that are present in **both** sets.

If there are no common elements, it returns an empty set.



Syntax

```
set1 & set2
set1.intersection(set2, set3, ...)
```

- The **&** operator joins only **sets**, unlike **intersection()**, which works with other data types.

Intersection (& or .intersection())

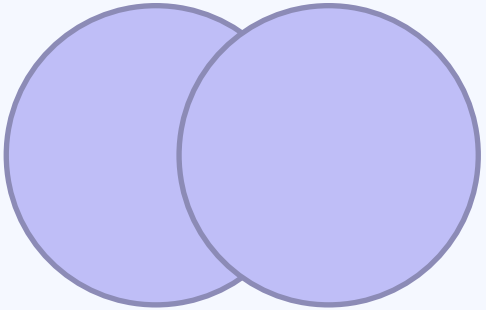
- Example

Python	Output

A = {1, 2, 3}
B = {3, 4, 5}
 	
# Using & operator 	
print(A & B)
 	
# Using intersection() method 	
print(A.intersection(B))
	{3}
	
	{3}

Difference (- or .difference())

- The **difference** operation returns elements that are in **one set but not in the other**.
- **Order matters**: A - B is not the same as B - A.
- Returns a new set with elements present in the first set but not in the second.



Syntax

```
set1 - set2
set1.difference(set2)
```

Difference (- or .difference())

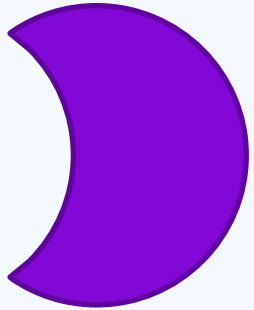
- The **difference** operation returns elements that are in **one set but not in the other**.
- **Order matters**: A - B is not the same as B - A.
- Returns a new set with elements present in the first set but not in the second.

Syntax

```
set1 - set2
set1.difference(set2)
```

Difference (- or .difference())

- The **difference** operation returns elements that are in **one set but not in the other**.
- **Order matters**: A - B is not the same as B - A.
- Returns a new set with elements present in the first set but not in the second.



Syntax

```
set2 - set1
set2.difference(set1)
```

Difference (- or .difference())

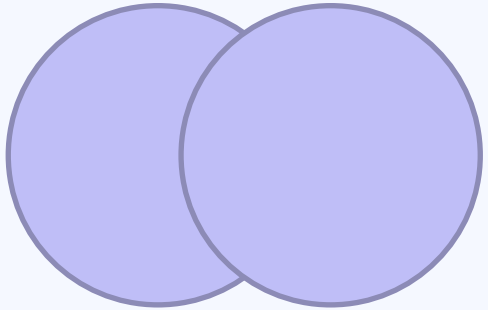
- Example

Python	Output

A = {1, 2, 3}
B = {3, 4, 5}
 	
# Using - operator 	
print(A - B)
 	
# Using difference() method 	
print(A.difference(B))
 	
# Reversing the order 	
print(B - A) # Output:
	{1, 2} (Elements in A but not in B)
	
	{1, 2}
	
	{4, 5} (Elements in B but not in A)

Symmetric Difference (^ or .symmetric_difference())

- The **symmetric difference** operation returns elements that are in either **one of the sets but not in both**.
- Similar to union - intersection: (A | B) - (A & B).

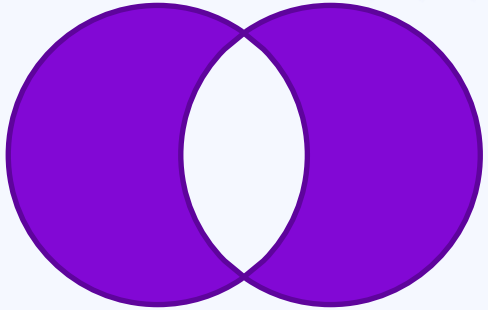


Syntax

```
set1 ^ set2
set1.symmetric_difference(set2)
```

Symmetric Difference (^ or .symmetric_difference())

- The **symmetric difference** operation returns elements that are in either **one of the sets but not in both**.
- Similar to union - intersection: (A | B) - (A & B).



Syntax

```
set1 ^ set2
set1.symmetric_difference(set2)
```

Symmetric Difference (^ or .symmetric_difference())

- Example

Python	Output

A = {1, 2, 3}
B = {3, 4, 5}
 	
# Using ^ operator 	
print(A ^ B)
 	
# Using symmetric_difference() method 	
print(A.symmetric_difference(B)
	{1, 2, 4, 5}
	
	{1, 2, 4, 5}

Updating Sets In-Place

- Python provides in-place versions of these operations that modify the original set.

Operation	Method	In-Place Version
Union	`set1.union(set2)`	`set1.update(set2)`
Intersection	`set1.intersection(set2)`	`set1.intersection_update(set2)`
Difference	`set1.difference(set2)`	`set1.difference_update(set2)`
Symmetric Difference	`set1.symmetric_difference(set2)`	`set1.symmetric_difference_update(set2)`

Practice Set - 3

Download Link in **Description** and **Pinned Comment**

WATCH

Level up your coding with each episode in this focused Python series.

Next Video!

Practice Set - 3
Solution