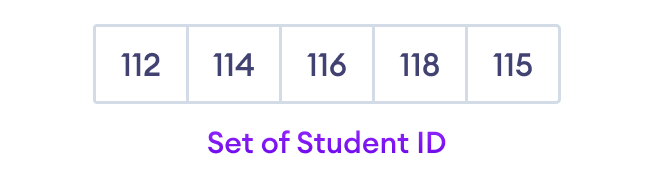
**Python Sets**

A set is a collection of unique data, meaning that elements within a set cannot be duplicated.

For instance, if we need to store information about student IDs, a set is suitable since student IDs cannot have duplicates.

Python Set Elements

**Create a Set in Python**

In Python, we create sets by placing all the elements inside curly braces {}, separated by commas.

# create a set of integer type

student\_id = {112, 114, 116, 118, 115}

print('Student ID:', student\_id)

# create a set of string type

vowel\_letters = {'a', 'e', 'i', 'o', 'u'}

print('Vowel Letters:', vowel\_letters)

# create a set of mixed data types

mixed\_set = {'Hello', 101, -2, 'Bye'}

print('Set of mixed data types:', mixed\_set)

**Output**

Student ID: {112, 114, 115, 116, 118}

Vowel Letters: {'u', 'a', 'e', 'i', 'o'}

Set of mixed data types: {'Hello', 'Bye', 101, -2}

**Note:** When you run this code, you might get output in a different order. This is because the set has no particular order.

**Create an Empty Set in Python**

Creating an empty set is a bit tricky. Empty curly braces {} will make an empty dictionary in Python.

To make a set without any elements, we use the set() function without any argument. For example,

# create an empty set

empty\_set = set()

# create an empty dictionary

empty\_dictionary = { }

**Output**

Data type of empty\_set: <class 'set'>

**Duplicate Items in a Set**

Let's see what will happen if we try to include duplicate items in a set.

numbers = {2, 4, 6, 6, 2, 8}

print(numbers) # {8, 2, 4, 6}

**Add and Update Set Items in Python**

Sets are mutable. However, since they are unordered, indexing has no meaning.

We cannot access or change an element of a set using indexing or slicing. The set data type does not support it.

**Add Items to a Set in Python**

In Python, we use the [add()](https://www.programiz.com/python-programming/methods/set/add) method to add an item to a set. For example,

numbers = {21, 34, 54, 12}

print('Initial Set:',numbers)

# using add() method

numbers.add(32)

print('Updated Set:', numbers)

**Output**

Initial Set: {34, 12, 21, 54}

Updated Set: {32, 34, 12, 21, 54}

numbers.add(32)

.

**Update Python Set**

The [update()](https://www.programiz.com/python-programming/methods/set/update) method is used to update the set with items other collection types (lists, tuples, sets, etc). For example,

companies = {'Lacoste', 'Ralph Lauren'}

tech\_companies = ['apple', 'google', 'apple']

# using update() method

companies.update(tech\_companies)

print(companies)

# Output: {'google', 'apple', 'Lacoste', 'Ralph Lauren'}

[e](https://www.programiz.com/python-programming/online-compiler)

**Remove an Element from a Set**

We use the [discard()](https://www.programiz.com/python-programming/methods/set/discard) method to remove the specified element from a set. For example,

languages = {'Swift', 'Java', 'Python'}

print('Initial Set:',languages)

# remove 'Java' from a set

removedValue = languages.discard('Java')

print('Set after remove():', languages)

[Run Code](https://www.programiz.com/python-programming/online-compiler)

**Output**

Initial Set: {'Python', 'Swift', 'Java'}

Set after remove(): {'Python', 'Swift'}

Here, we have used the discard() method to remove 'Java' from the languages set.

**Built-in Functions with Set**

Here are some of the popular built-in functions that allow us to perform different operations on a set.

|  |  |
| --- | --- |
| Function | Description |
| [len()](https://www.programiz.com/python-programming/methods/built-in/len) | Returns the length (the number of items) in the set. |
| [max()](https://www.programiz.com/python-programming/methods/built-in/max) | Returns the largest item in the set. |
| [min()](https://www.programiz.com/python-programming/methods/built-in/min) | Returns the smallest item in the set. |
| [sorted()](https://www.programiz.com/python-programming/methods/built-in/sorted) | Returns a new sorted list from elements in the set (does not sort the set itself). |
| [sum()](https://www.programiz.com/python-programming/methods/built-in/sum) | Returns the sum of all elements in the set. |
|  |  |
|  |  |
|  |  |

**Iterate Over a Set in Python**

x = {"Apple", "Peach", "Mango"}

# for loop to access each fruits

for x in fruits:

print(x)

**Output**

Mango

Peach

Apple

**Find Number of Set Elements**

We can use the [len()](https://www.programiz.com/python-programming/methods/built-in/len) method to find the number of elements present in a Set. For example,

even\_numbers = {2,4,6,8}

print('Set:',even\_numbers)

# find number of elements

print('Total Elements:', len(even\_numbers))

[Run Code](https://www.programiz.com/python-programming/online-compiler)

**Output**

Set: {8, 2, 4, 6}

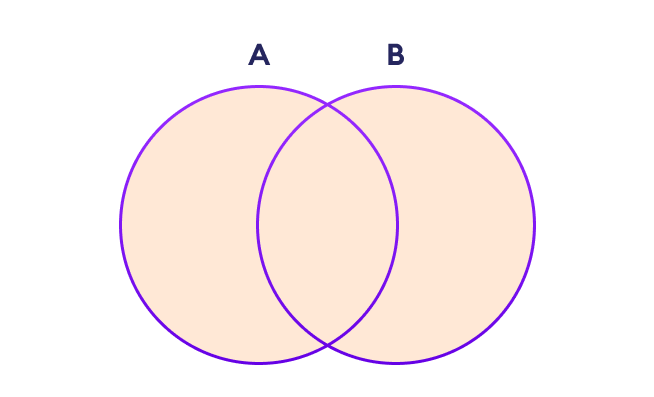
Total Elements: 4

**Python Set Operations**

Python Set provides different built-in methods to perform mathematical set operations like union, intersection, subtraction, and symmetric difference.

**Union of Two Sets**

The union of two sets A and B includes all the elements of sets A and B.

Set Union in Python

We use the | operator or the [union()](https://www.programiz.com/python-programming/methods/set/union) method to perform the set union operation. For example,

# first set

A = {1, 3, 5}

# second set

B = {0, 2, 4}

# perform union operation using |

print('Union using |:', A | B)

# perform union operation using union()

print('Union using union():', A.union(B))

[Run Code](https://www.programiz.com/python-programming/online-compiler)

**Output**

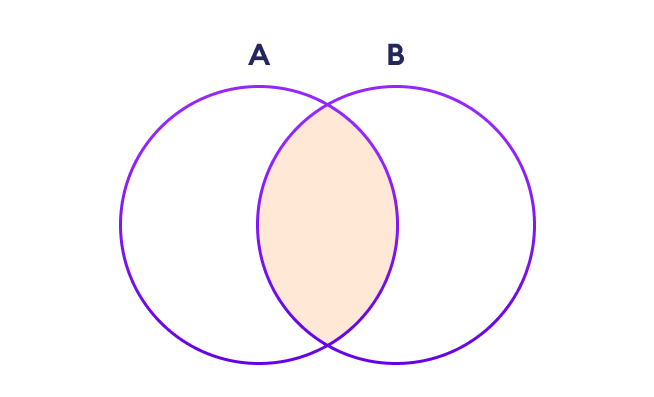
Union using |: {0, 1, 2, 3, 4, 5}

Union using union(): {0, 1, 2, 3, 4, 5}

**Note**: A|B and union () is equivalent to A ⋃ B set operation.

**Set Intersection**

The intersection of two sets A and B include the common elements between set A and B.



In Python, we use the & operator or the [intersection()](https://www.programiz.com/python-programming/methods/set/intersection) method to perform the set intersection operation. For example,

# first set

A = {1, 3, 5}

# second set

B = {1, 2, 3}

# perform intersection operation using &

print('Intersection using &:', A & B)

# perform intersection operation using intersection()

print('Intersection using intersection():', A.intersection(B))

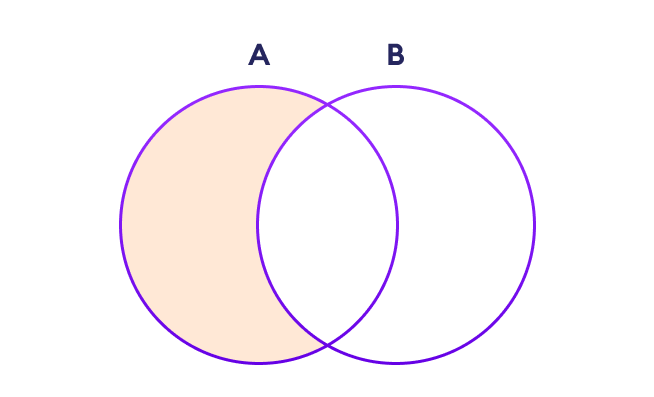
**Output**

Intersection using &: {1, 3}

Intersection using intersection(): {1, 3}

**Difference between Two Sets**

The difference between two sets A and B include elements of set A that are not present on set B.



We use the - operator or the [difference()](https://www.programiz.com/python-programming/methods/set/difference) method to perform the difference between two sets. For example,

# first set

A = {2, 3, 5}

# second set

B = {1, 2, 6}

# perform difference operation using &

print('Difference using &:', A - B)

# perform difference operation using difference()

print('Difference using difference():', A.difference(B)

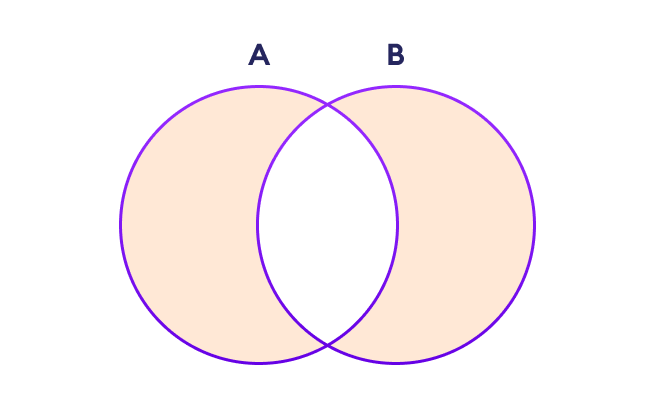
**Output**

Difference using &: {3, 5}

Difference using difference(): {3, 5}

**Set Symmetric Difference**

The symmetric difference between two sets A and B includes all elements of A and B without the common elements.



In Python, we use the ^ operator or the [symmetric\_difference()](https://www.programiz.com/python-programming/methods/set/symmetric_difference) method to perform symmetric differences between two sets. For example,

# first set

A = {2, 3, 5}

# second set

B = {1, 2, 6}

# perform difference operation using &

print('using ^:', A ^ B)

# using symmetric\_difference()

print('using symmetric\_difference():', A.symmetric\_difference(B))

**Output**

using ^: {1, 3, 5, 6}

using symmetric\_difference(): {1, 3, 5, 6}

**Check if two sets are equal**

We can use the == operator to check whether two sets are equal or not. For example,

# first set

A = {1, 3, 5}

# second set

B = {3, 5, 1}

# perform difference operation using &

if A == B:

print('Set A and Set B are equal')

else:

print('Set A and Set B are not equal')

**Output**

Set A and Set B are equal

if A == B

**Other Python Set Methods**

There are many set methods, some of which we have already used above. Here is a list of all the methods that are available with the set objects:

|  |  |
| --- | --- |
| Method | Description |
| [add()](https://www.programiz.com/python-programming/methods/set/add) | Adds an element to the set |
| [clear()](https://www.programiz.com/python-programming/methods/set/clear) | Removes all elements from the set |
| [copy()](https://www.programiz.com/python-programming/methods/set/copy) | Returns a copy of the set |
| [difference()](https://www.programiz.com/python-programming/methods/set/difference) | Returns the difference of two or more sets as a new set |
| [difference\_update()](https://www.programiz.com/python-programming/methods/set/difference_update) | Removes all elements of another set from this set |
| [discard()](https://www.programiz.com/python-programming/methods/set/discard) | Removes an element from the set if it is a member. (Do nothing if the element is not in set) |
| [intersection()](https://www.programiz.com/python-programming/methods/set/intersection) | Returns the intersection of two sets as a new set |
| [intersection\_update()](https://www.programiz.com/python-programming/methods/set/intersection_update) | Updates the set with the intersection of itself and another |
| [isdisjoint()](https://www.programiz.com/python-programming/methods/set/isdisjoint) | Returns True if two sets have a null intersection |
| [issubset()](https://www.programiz.com/python-programming/methods/set/issubset) | Returns True if another set contains this set |
| [issuperset()](https://www.programiz.com/python-programming/methods/set/issuperset) | Returns True if this set contains another set |
| [pop()](https://www.programiz.com/python-programming/methods/set/pop) | Removes and returns an arbitrary set element. Raises KeyError if the set is empty |
| [remove()](https://www.programiz.com/python-programming/methods/set/remove) | Removes an element from the set. If the element is not a member, raises a KeyError |
| [symmetric\_difference()](https://www.programiz.com/python-programming/methods/set/symmetric_difference) | Returns the symmetric difference of two sets as a new set |
| [symmetric\_difference\_update()](https://www.programiz.com/python-programming/methods/set/symmetric_difference_update) | Updates a set with the symmetric difference of itself and another |
| [union()](https://www.programiz.com/python-programming/methods/set/union) | Returns the union of sets in a new set |
| [update()](https://www.programiz.com/python-programming/methods/set/update) | Updates the set with the union of itself and others |

The add() method adds a given element to a [set](https://www.programiz.com/python-programming/set). If the element is already present, it doesn't add any element.

### Example

prime\_numbers = {2, 3, 5, 7}

# add 11 to prime\_numbers

prime\_numbers.add(11)

print(prime\_numbers)

# Output: {2, 3, 5, 7, 11}

The remove() method removes the specified element from the [set](https://www.programiz.com/python-programming/set).

### Example

languages = {'Python', 'Java', 'English'}

# remove English from the set

languages.remove('English')

print(languages)

# Output: {'Python', 'Java'}

# Python Set copy()

The copy() method returns a copy of the [set](https://www.programiz.com/python-programming/set).

### Example

numbers = {1, 2, 3, 4}

# copies the items of numbers to new\_numbers

new\_numbers = numbers.copy()

print(new\_numbers)

# Output: {1, 2, 3, 4}

## Example 2: Copy Set using = operator

We can also copy the set by simply using the = [operator](https://www.programiz.com/python-programming/operators).

names = {"John", "Charlie", "Marie"}

# copy set using = operator

new\_names = names

print('Original Names: ', names)

print('Copied Names: ', new\_names)

**Output**

Original Names: {'John', 'Marie', 'Charlie'}

Copied Names: {'John', 'Marie', 'Charlie'}

# Python Set difference\_update()

The difference\_update() method computes the difference between two [sets](https://www.programiz.com/python-programming/set) (A - B) and updates set A with the resulting set.

### Example

# sets of numbers

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

B = {2, 3, 5, 7, 11}

# computes A - B and updates A with the resulting set

A.difference\_update(B)

print('A = ', A)

# Output: A = {1, 9}

# Python Set discard()

The discard() method removes the specified item from the [set](https://www.programiz.com/python-programming/set).

### Example

numbers = {2, 3, 4, 5}

# removes 3 and returns the remaining set

numbers.discard(3)

print(numbers)

# Output: numbers = {2, 4, 5}

# Python Set isdisjoint()

The isdisjoint() method returns True if two [sets](https://www.programiz.com/python-programming/set) don't have any common items between them, i.e. they are disjoint. Else the returns False.

### Example

A = {1, 2, 3, }

B = {4, 5, 6}

# checks if set A and set B are disjoint

print(A.isdisjoint(B))

# Output: True

## Example 1: Python Set disjoint()

A = {1, 2, 3}

B = {4, 5, 6}

C = {6, 7, 8}

print('A and B are disjoint:', A.isdisjoint(B))

print('B and C are disjoint:', B.isdisjoint(C))

**Output**

A and B are disjoint: True

B and C are disjoint: False

# Python Set issubset()

The issubset() method returns True if [set](https://www.programiz.com/python-programming/set) A is the subset of B, i.e. if all the elements of set A are present in set B . Else, it returns False.

### Example

A = {1, 2, 3}

B = {1, 2, 3, 4, 5}

# all items of A are present in B

print(A.issubset(B))

# Output: True

# Python Set pop()

The pop() method removes an item from a [set](https://www.programiz.com/python-programming/set) and returns the removed item.

### Example

A = {'a', 'b', 'c', 'd'}

removed\_item = A.pop()

print(removed\_item)

# Output: c

# Python Set symmetric\_difference()

The symmetric\_difference() method returns all the items present in given [sets](https://www.programiz.com/python-programming/set), except the items in their intersections.

### Example

A = {'a', 'b', 'c', 'd'}

B = {'c', 'd', 'e' }

# returns all items to result variable except the items on intersection

result = A.symmetric\_difference(B)

print(result)

# Output: {'a', 'b', 'e'}

# Example tuple

t = (1, 2, 3, 4)

# Convert tuple to list

a = list(t)

print(a)

**Output**

[1, 2, 3, 4]

# Example tuple

t = (1, 2, 3, 4)

# Initialize an empty list

a = []

# Convert tuple to list using a for loop

for item in t:

a.append(item)

print(a)