**Python Sets**

Set is an unordered and unindexed collection of items in Python. Unordered means when we display the elements of a set, it will come out in a random order. Unindexed means, we cannot access the elements of a set using the indexes like we can do in [list](https://beginnersbook.com/2018/02/python-list/) and [tuples](https://beginnersbook.com/2018/02/python-tuple/).

The elements of a set are defined inside square brackets and are separated by commas. For example –

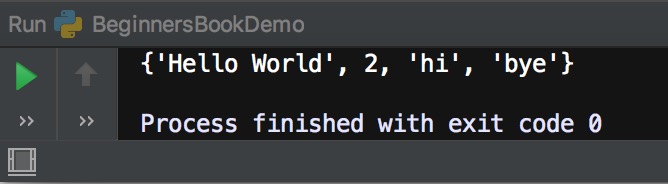
myset = [1, 2, 3, 4, "hello"]

**Python Set Example**

# Set Example

myset = {"hi", 2, "bye", "Hello World"}

print(myset)

**Output:**  


**Checking whether an item is in the set**

We can check whether an item exists in Set or not using “in” operator as shown in the following example. This returns the boolean value true or false. If the item is in the given set then it returns true, else it returns false.

Set Example

myset = {"hi", 2, "bye", "Hello World"}

# checking whether 2 is in myset

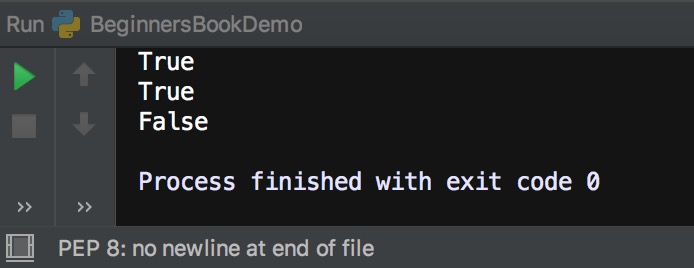
print(2 in myset)

# checking whether "hi" is in myset

print("hi" in myset)

# checking whether "BeginnersBook" is in myset

print("BeginnersBook" in myset)

**Output:**  


**Loop through the elements of a Set in Python**

We can loop through the elements of a set in Python as shown in the following elements. As you can see in the output that the elements will appear in random order each time you run the code.

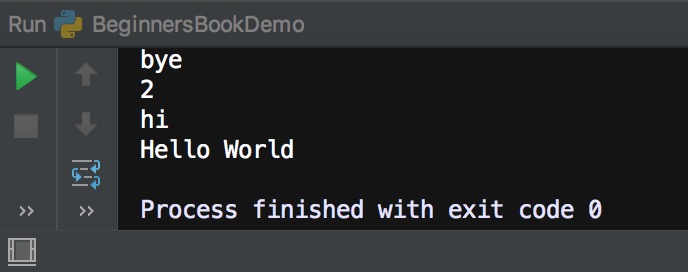
# Set Example

myset = {"hi", 2, "bye", "Hello World"}

# loop through the elements of myset

for a in myset:

print(a)

**Output:**  


**Python – Add or remove item from a Set**

We can add an item in a Set using add() function and we can remove an item from a set using remove() function as shown in the following example.

# Set Example

myset = {"hi", 2, "bye", "Hello World"}

print("Original Set:", myset)

# adding an item

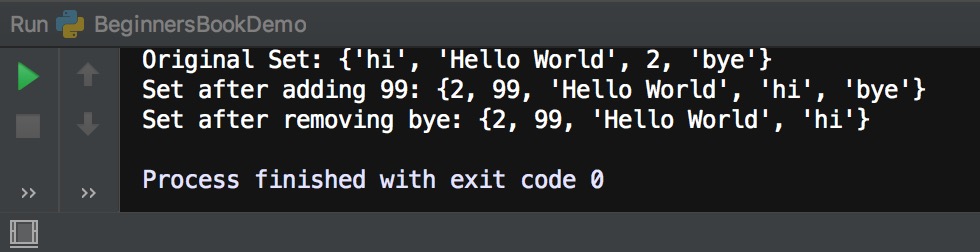
myset.add(99)

print("Set after adding 99:", myset)

# removing an item

myset.remove("bye")

print("Set after removing bye:", myset)

**Output:**  


**Set Methods**

1. [add()](https://beginnersbook.com/2019/03/python-set-add-method/): This method adds an element to the Set.  
2. [remove()](https://beginnersbook.com/2019/03/python-set-remove-method/): This method removes a specified element from the Set  
3. [discard()](https://beginnersbook.com/2019/03/python-set-discard-method/): This method works same as remove() method, however it doesn’t raise an error when the specified element doesn’t exist.  
4. [clear()](https://beginnersbook.com/2019/03/python-set-clear-method/): Removes all the elements from the set.  
5. [copy()](https://beginnersbook.com/2019/03/python-set-copy-method/): Returns a shallow copy of the set.  
6. [difference()](https://beginnersbook.com/2019/03/python-set-difference-method/): This method returns a new set which is a difference between two given sets.  
7. [difference\_update()](https://beginnersbook.com/2019/03/python-set-difference_update-method/): Updates the calling set with the Set difference of two given sets.  
8. [intersection()](https://beginnersbook.com/2019/03/python-set-intersection-method/): Returns a new set which contains the elements that are common to all the sets.  
9. [intersection\_update()](https://beginnersbook.com/2019/03/python-set-intersection_update/): Updates the calling set with the Set intersection of two given sets.  
10. [isdisjoint()](https://beginnersbook.com/2019/04/python-set-isdisjoint-method/): Checks whether two sets are disjoint or not. Two sets are disjoint if they have no common elements.  
11. [issubset()](https://beginnersbook.com/2019/04/python-set-issubset-method/): Checks whether a set is a subset of another given set.  
12. [pop()](https://beginnersbook.com/2019/04/python-set-pop-method/): Removes and returns a random element from the set.  
13. [union()](https://beginnersbook.com/2019/04/python-set-union-method/): Returns a new set with the distinct elements of all the sets.  
14. [update()](https://beginnersbook.com/2019/04/python-set-update-method/): Adds elements to a set from other passed iterable.  
15. [symmetric\_difference()](https://beginnersbook.com/2019/04/python-set-symmetric_difference-method/): Returns a new set which is a symmetric difference of two given sets.  
16. [symmetric\_difference\_update()](https://beginnersbook.com/2019/04/python-set-symmetric_difference_update-method/): Updates the calling set with the symmetric difference of two given sets.

**Python Set add() Method with examples**

**Python Set add() method** is used to add an element to a Set. Since [Set](https://beginnersbook.com/2019/03/python-sets/) doesn’t allow duplicates, if you try to add an already existing element to a Set, it won’t do anything.

**Set add() Method Syntax**

set.add(item)

Here item is the element that we want to add to Set.

**Set add() method Example**

In the following example we have a Set myset and we are adding an element to the Set myset using add() method.

# Set add() method example

myset = {1, 2, "hello", 5, "bye"}

# displaying set before adding any element

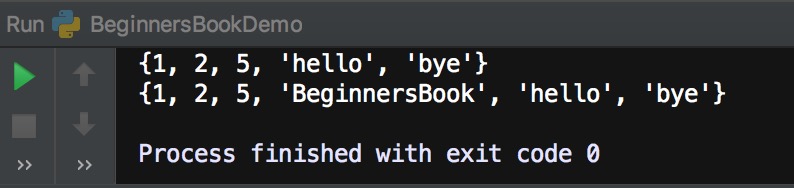
print(myset)

#adding an element to a the Set myset

myset.add("BeginnersBook")

# displaying set after adding an element

print(myset)

**Output:**  


**Example 2: Adding a tuple to a Set using add() method**

In the following example, we have a Set myset and a [tuple](https://beginnersbook.com/2018/02/python-tuple/) t1 and we are adding the tuple t1 to the Set myset. Similar to the elements, if we try to add an already existing tuple to a set then it won’t add it and it won’t raise any error.

# A Set

myset = {1, 2, "hi"}

# A tuple

t1 = (10, 20, "hello")

# displaying set before adding the tuple

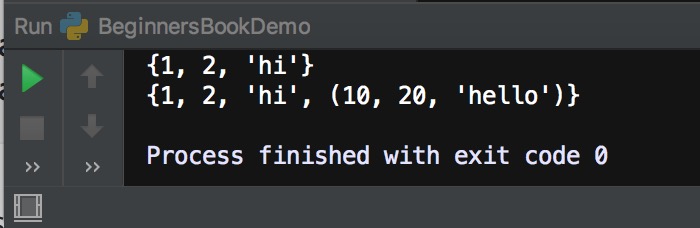
print(myset)

#adding the tuple t1 to the set myset

myset.add(t1)

# displaying set after adding the tuple t1

print(myset)

**Output:**  


**Python Set remove() Method with examples**

The remove() method in Python searches for the specified element in the given [Set](https://beginnersbook.com/2019/03/python-sets/) and removes it. If the specified element does not exist in the Set then an error is raised.

**Python Set remove() method syntax**

set.remove(item)

**Parameter**: Here item is the element which we want to remove the set.  
**Return value**: This method doesn’t return anything.

**Python Set remove() method example**

In the following example we have a Set of names and we are removing an element from this given set using the remove() method.

# A Set of names

names = {"Rick", "Negan", "Glenn"}

# Original Set

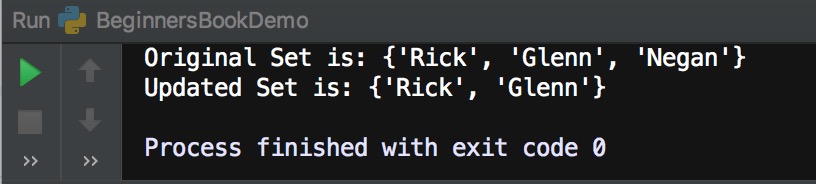
print("Original Set is:", names)

# Element "Negan" is removed from the Set

names.remove("Negan")

# Set after removing an element from it

print("Updated Set is:", names)

**Output:**  


**When element doesn’t exist in the Set**

If you try to remove an element from the Set which doesn’t exist then an error is raised. Lets take an example to see this behaviour of remove() method.

Here we are trying to remove a name “Chaitanya” from the given set of names but the element does not exist in the set names, thus an error is raised as shown in the output below.

# A Set of names

names = {"Rick", "Negan", "Glenn"}

# Original Set

print("Original Set is:", names)

# Trying to remove element "Chaitanya"

names.remove("Chaitanya")

# Set after removing an element from it

print("Updated Set is:", names)

**Output:**

Original Set is: {'Glenn', 'Negan', 'Rick'}

Traceback (most recent call last):

File "/Users/chaitanyasingh/PycharmProjects/BeginnersBookProJ/venv1/BeginnersBookDemo.py",

line 8, in

names.remove("Chaitanya")

KeyError: 'Chaitanya'

Process finished with exit code 1

As you can see a KeyError is raised when we tried to remove an element from the set which was not present. If you do not want any error to be raised in such cases then use [discard() method](https://beginnersbook.com/2019/03/python-set-discard-method/) instead.

**Python Set discard() method with examples**

he **discard() method** in Python removes a specified element from the [Set](https://beginnersbook.com/2019/03/python-sets/). It works same as remove() method, however there is a **difference between remove() and discard() method**. The remove() method raises an error when the specified element doesn’t exist in the given set, however the discard() method doesn’t raise any error if the specified element is not present in the set and the set remains unchanged.

**Python Set discard() method syntax**

set.discard(item)

**Parameter**: Here item is the element which we want to remove the set.  
**Return value**: This method doesn’t return anything.

**Python Set discard() method example**

In the following example we have a set of numbers and we are removing few elements from the given set using discard() method.

# A Set of numbers

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

# displaying the set before removing anything

print("Original Set is:", numbers)

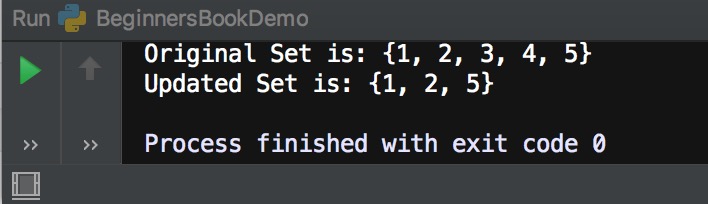
# Element 3 & 4 are removed from the Set

numbers.discard(3)

numbers.discard(4)

# displaying the set after discard() method

print("Updated Set is:", numbers)

**Output:**  


**If element doesn’t exist in the Set**

If the specified element does not exist in the given Set then this method doesn’t raise any error and the set remains unchanged. Lets take an example to see this in action.

# A Set of numbers

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

# displaying the set before removing anything

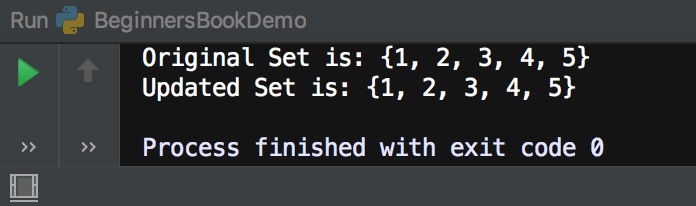
print("Original Set is:", numbers)

# trying to remove the element 99 from the set

numbers.discard(99)

# displaying the set after discard() method

print("Updated Set is:", numbers)

**Output:**  


As you can see in the output that no errors were raised and the set remained unchanged when we tried to remove an element from the set which was not present. If you want the error to be raised in such cases then use the [remove() method](https://beginnersbook.com/2019/03/python-set-remove-method/) instead.

**Python Set clear() Method with examples**

The clear() method in Python removes all the elements from a given [Set](https://beginnersbook.com/2019/03/python-sets/).

**Python Set clear() method Syntax**

set.clear()

**Parameter**: None. This method doesn’t take any parameters.  
**Return value**: This method doesn’t return any value, it just removes all the items from the set.

**Python Set clear() method example**

In the following example we have a set of numbers and we are calling clear() method to remove all the elements of this set.

# A Set of numbers

numbers = {5, 8, 10, 9, 1}

# Displaying set before executing clear()

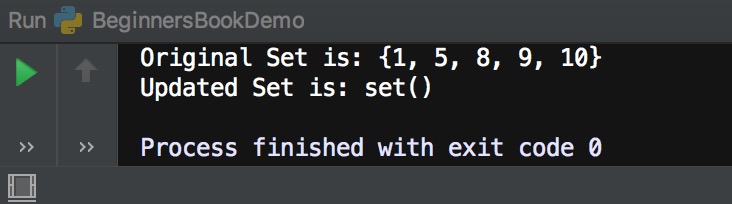
print("Original Set is:", numbers)

# removing all the elements from the set

numbers.clear()

# displaying set after calling clear()

print("Updated Set is:", numbers)

**Output:**  


As you can see in the output that there are no elements in the set after we have called the clear() method. If you want to remove a specific element from the Set then using [remove()](https://beginnersbook.com/2019/03/python-set-remove-method/) or [discard() method](https://beginnersbook.com/2019/03/python-set-discard-method/).

**Python Set copy() Method with examples**

The **copy() method** in Python returns a copy of the Set. We can copy a set to another set using the = operator, however copying a set using = operator means that when we change the new set the copied set will also be changed, if you do not want this behaviour then use the copy() method instead of = operator. We will discuss these with the help of examples.

**Python Set copy() method Syntax**

set.copy()

**Parameters**: None. This method doesn’t take any parameter.  
**Return value**: Returns the copy of the given set

**Copying a Set using = operator**

In the following example we are copying a set to another set using = operator. The problem with this method is that if any of the set (old or new) is modified, the changes will reflect in both the sets. Here we have a set names and we copied this set to a new set names2.

After the copy using = operator, we have made changes in both the sets (added an element in new set and removed an element from old set), both the changes reflect in both the sets as shown in the following example. If you do not want this kind of behaviour and only want the set where you are making changes to be changed then use copy() method, which is discussed in the next section.

# A Set of names

names = {"Steve", "Rick", "Negan"}

names2 = names

# adding a new element in the new set

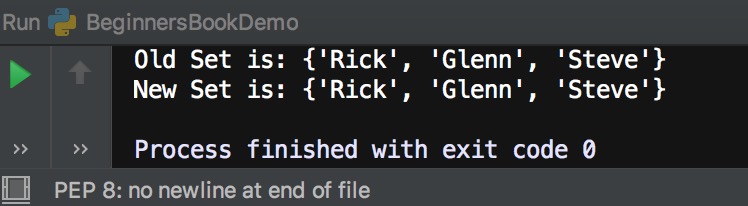
names2.add("Glenn")

# removing an element from the old set

names.remove("Negan")

print("Old Set is:", names)

print("New Set is:", names2)

**Output:**  


**Python Set copy() method example**

Lets take the same example that we have seen above, however here we are copying the set using the copy() method. As you can see in the output that only the change that we have made to the old and new set reflect and both doesn’t share the same copy of the set as shown in the above example.

# A Set of names

names = {"Steve", "Rick", "Negan"}

# copying using the copy() method

names2 = names.copy()

# adding "Glenn" to the new set

names2.add("Glenn")

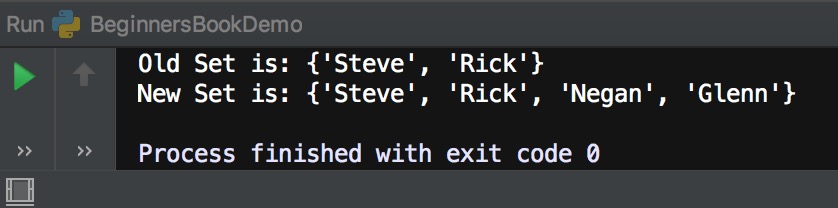
# removing "Negan" from the old set

names.remove("Negan")

# displaying both the sets

print("Old Set is:", names)

print("New Set is:", names2)

**Output:**  


**Python Set difference() method with examples**

The difference() method in Python returns the difference between two given [sets](https://beginnersbook.com/2019/03/python-sets/). Lets say we have two sets A and B, the difference between A and B is denoted by A-B and it contains the elements that are in Set A but in Set B.

**How the difference is calculated between two sets?**

Lets say we have two Sets X and Y as follows:  
X = {“hello”, 2, 5}  
Y = {2, 9, “bye”}

Difference between two sets is denoted by – sign.

Elements that are in Set X but not in Set Y  
X-Y = {“hello”, 5}

Elements that are in Set Y but not in Set X  
Y-X = {9, “bye”}

**Python Set difference() method Syntax**

X.difference(Y)

This is equivalent to X-Y.

**Parameters:** This method takes a Set as parameter.  
**Return Value:** It returns a Set that is a difference between two sets. For example X.difference(Y) would return a Set that contains the elements that are in Set X but not in Set Y.

**Python Set difference() method Example**

In the following example we have two sets X and Y. We are finding the difference between X & Y and Y & X using the difference() method.

# Set X

X = {"hello", 9, 10, "hi"}

# Set Y

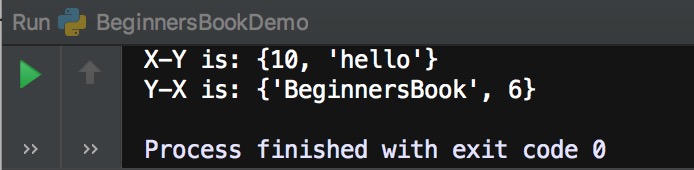
Y = {9, "hi", 6, "BeginnersBook"}

# Equivalent to X - Y

print("X-Y is:",X.difference(Y))

# Equivalent to Y - X

print("Y-X is:",Y.difference(X))

**Output:**  


**Difference between Sets using – Operator**

We can use the – operator on Sets. This works same as difference() method. Lets take the same example that we have seen above using – operator.

# Set X

X = {"hello", 9, 10, "hi"}

# Set Y

Y = {9, "hi", 6, "BeginnersBook"}

# Equivalent to X.difference(Y)

print("X-Y is:",X-Y)

# Equivalent to Y.difference(X)

print("Y-X is:",Y-X)

**Output:**

X-Y is: {10, 'hello'}

Y-X is: {'BeginnersBook', 6}

As you can see we got the same output.

**Python Set difference\_update() method with examples**

In the last tutorial, we discussed [Set difference() method](https://beginnersbook.com/2019/03/python-set-difference-method/), which returns the difference between two given [Sets](https://beginnersbook.com/2019/03/python-sets/). In this guide, we will see **difference\_update() method** which doesn’t return anything but it updates the first Set with the Set difference. For example, calling method like this A.difference\_update(B) would update the Set A with the A-B (elements that are in Set A but not in Set B).

**Python Set difference\_update() Syntax**

X.difference\_update(Y)

This would update the Set X with the X-Y  
X-Y: Elements that are only in Set X and not in Set Y

Parameters: This method accepts a Set as a parameter.  
Return Value: None. This method doesn’t return anything.

**Python Set difference\_update() Example**

In the following example we have two Sets X and Y. We are calling X.difference\_update(Y) which updates the Set X with the Set difference X – Y.

# Set X

X = {"hello", 9, 10, "hi"}

# Set Y

Y = {9, "hi", 6, "BeginnersBook"}

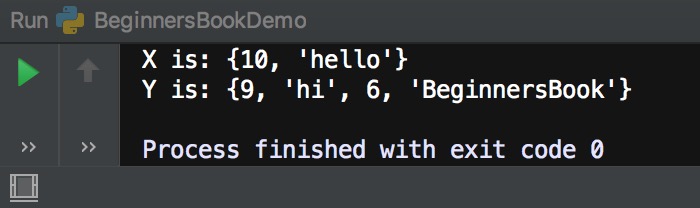
# calling difference\_update() method

X.difference\_update(Y)

# Displaying Set X and Y

print("X is:", X)

print("Y is:", Y)

**Output:**  


**Python Set intersection() method with examples**

Set intersection is denoted by **∩** symbol. **Python Set intersection() method** returns a new set with elements that are common to all the [sets](https://beginnersbook.com/2019/03/python-sets/).

**Python Set intersection() method Syntax**

X.intersection(Y) is equivalent to X ∩ Y.

X ∩ Y = Y ∩ X = The Set with the elements that are common to Set X and Y.

**Parameter**: This method accepts a Set as a parameter.  
**Return value**: This method returns a new set with the elements that are common to all the sets.

**Python Set intersection() method Example**

In the following example, we have three sets X, Y and Z. We are demonstrating the use of intersection() method with the help of few examples. In the third print statement we are finding the intersection between all the three sets.

# Set X

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

# Set Y

Y = {4, 5, 6, 7}

# Set Z

Z = {5, 6, 7, 8, 9}

# X ∩ Y

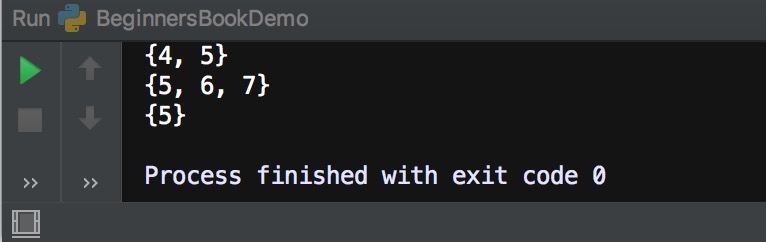
print(X.intersection(Y))

# Y ∩ Z

print(Y.intersection(Z))

# X ∩ Y ∩ Z

print(X.intersection(Y, Z))

**Output:**  


**Python Set Intersection Using & operator**

We can also use & operator to find the intersection between sets. This works similar to the intersection() method. Lets take an example to understand the use of & operator. We are taking the same example that we have seen above but here we will use the & operator instead of intersection() method.

# Set X

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

# Set Y

Y = {4, 5, 6, 7}

# Set Z

Z = {5, 6, 7, 8, 9}

# X ∩ Y

print(X&Y)

# Y ∩ Z

print(Y&Z)

# X ∩ Y ∩ Z

print(X&Y&Z)

**Output:**

{4, 5}

{5, 6, 7}

{5}

As you can see we got the same output that we have got using the intersection() method.

**Python Set intersection\_update() method wi** we will discuss the intersection\_update() method that does not return anything, however it updates the calling Set with the intersection Set. For example calling this method like this X.intersection\_update(Y) would update the Set X with the X ∩ Y.

**Set intersection\_update() method Syntax**

X.intersection\_update(Y)

This will update the Set X with X ∩ Y (elements that are common to both the Sets X and Y).

**Parameter**: This method accepts Sets as parameters.  
**Return value**: It doesn’t return anything, it just updates the calling Set.

**Python Set intersection\_update() method example**

In the following example we have two sets X and Y. Here we are calling the method like this: X.intersection\_update(Y), this will update the Set X with the intersection values of X and Y.

# Set X

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

# Set Y

Y = {4, 5, 6, 7}

# X will have the elements of X ∩ Y

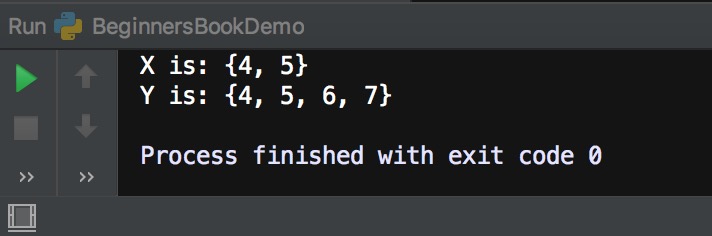
X.intersection\_update(Y)

# display X

print("X is:", X)

# display Y

print("Y is:", Y)

**Output:**  


**Set intersection\_update() with more than one Parameters**

In the following example we have three Sets X, Y and Z. Here we are passing more than one parameters to the intersection\_update() method. To find the intersection between more than two sets we can pass the additional sets in the intersection\_update() method as shown in the following example.

# Set X

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

# Set Y

Y = {4, 5, 6, 7}

# Set Z

Z = {5, 6, 7, 8, 9}

# X will have the elements of X ∩ Y ∩ Z

X.intersection\_update(Y, Z)

# display X

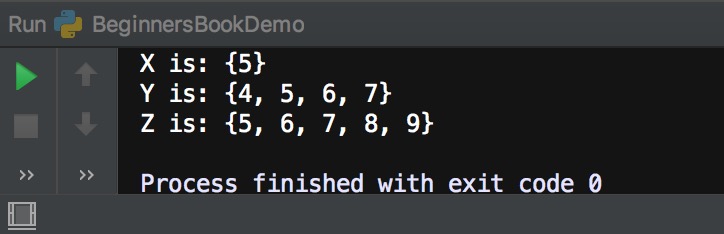
print("X is:", X)

# display Y

print("Y is:", Y)

# display Z

print("Z is:", Z)

**Output:**  


**Python Set isdisjoint() method with examples**

**Python Set isdisjoint() method** checks whether the two [sets](https://beginnersbook.com/2019/03/python-sets/) are disjoint sets or not. If the sets are disjoint, this method returns true else it returns false. Two sets are said to be disjoint if they do not have any common elements. For example Set {1, 2, 3} and Set {4, 5, 6} are disjoint sets because they do not have any common elements.

**Set isdisjoint() method Syntax**

set.isdisjoint(iterable)

**Parameter**: It accepts any iterable such as Set, List, tuple, dictionary etc. as a parameter and converts it into a Set and then checks whether the Sets are disjoint or not.  
**Return Value**: It returns a boolean value true or false. True if the sets are disjoint else it returns false.

**Python Set isdisjoint() method Example**

In the following example, we have three Sets X, Y and Z and we are checking whether set X & Y and X & Z are disjoint sets or not using isdisjoint() method. Since Set X and Y have common elements the isdisjoint() method returns false and because X and Z have no common elements, this method returns true for them.

# Set X

X = {4, 5}

# Set Y

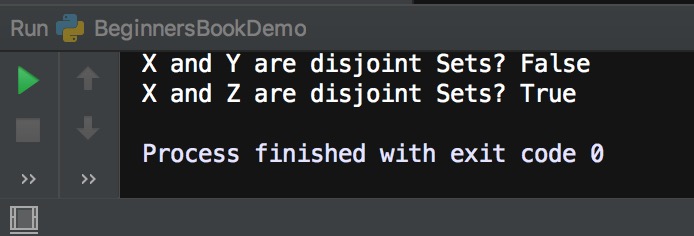
Y = {4, 5, 6}

# Set Z

Z = {10, 20, 30}

print("X and Y are disjoint Sets?", X.isdisjoint(Y))

print("X and Z are disjoint Sets?", X.isdisjoint(Z))

**Output:**  


**Set isdisjoint() method with other iterable such as List, Tuple & Dictionary**

When we pass the other iterable such as [List](https://beginnersbook.com/2018/02/python-list/), [tuple](https://beginnersbook.com/2018/02/python-tuple/) and [dictionary](https://beginnersbook.com/2019/03/python-dictionary/) in isdisjoint() method, it converts them into a Set and then checks whether the sets are disjoint or not.

When converting a **dictionary to a Set**, the keys are considered as the elements of the converted set.

# Set A

A = {4, 5}

# List lis

lis = [10, 20, 30]

# Tuple t

t = (10, "hello")

# Dictionary dict, Set is formed on Keys

dict = {4 : 'Four', 5: 'Five'}

# Dictionary dict2

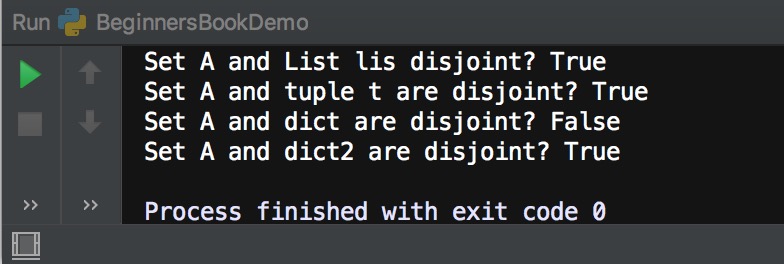
dict2 = {'Four': 4, 'Five': 5}

print("Set A and List lis disjoint?", A.isdisjoint(lis))

print("Set A and tuple t are disjoint?", A.isdisjoint(t))

print("Set A and dict are disjoint?", A.isdisjoint(dict))

print("Set A and dict2 are disjoint?", A.isdisjoint(dict2))

**Output:**  


**Python Set issubset() method with examples**

The **issubset() method** in Python checks whether a given Set is a subset of another specified Set. If all the elements of a given Set is present in another Set then the given set is called the subset of another Set.

For example Lets say we have two sets A: {1, 2, 3} & B: {1, 2, 3, 4}, If we want to check whether A is a subset of B then we call this method like this – A.issubset(B), this method would return true because all the elements of set A are present in B which means A is a subset of B.

**Python Set issubset() method Syntax**

set.issubset(set)

**Parameter:** This method accepts a Set as a parameter.  
**Return Value:** Boolean value. This method returns true or false based on the comparison, if the calling set is a subset of the set passed as a parameter then this method returns true else it returns false.

**Python Set issubset() method Example**

In the following example we have three set numbers and we are checking whether they are subset of each other using **issubset() method**. X is a subset of Y because all the elements of Set X are present in set Y, similarly Z is a subset of Y because all the elements of Set Z are present in Set Y.

# Set X

X = {1, 2, 3}

# Set Y

Y = {1, 2, 3, 4}

# Set Z

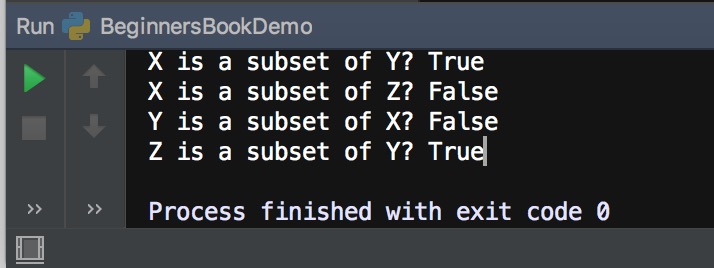
Z = {3, 4}

print("X is a subset of Y?", X.issubset(Y))

print("X is a subset of Z?", X.issubset(Z))

print("Y is a subset of X?", Y.issubset(X))

print("Z is a subset of Y?", Z.issubset(Y))

**Output:**  


**Python Set pop() method with examples**

The **pop() method** in Python removes a **random** element from a given set and returns the removed element.

**Set pop() method Syntax**

set.pop()

**Parameter:** This method doesn’t take any parameter.  
**Return Value:** It returns the removed element of the Set.

If you call this method on an empty set then an error is raised.

**Python Set pop() method Example**

In the following example we have a Set X and we are calling pop() method on this set. The pop() method removes a random element from the set and returns the removed element.

# My Set

X ={"hello", 2, 10, 99, "hi"}

# Displaying Set before calling pop()

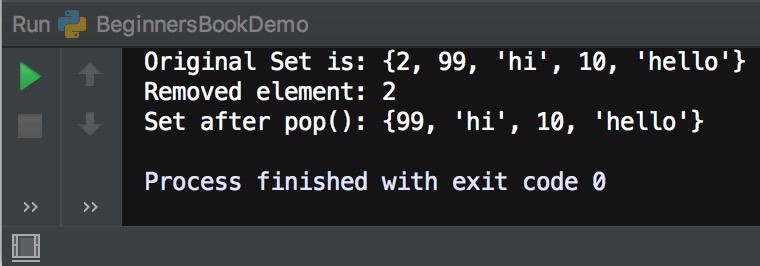
print("Original Set is:",X)

# Calling pop() method

print("Removed element:",X.pop())

# Displaying Set after calling pop()

print("Set after pop():", X)

**Output:**  


**Python Set union() method with examples**

The **Set union() method** returns a new set with the distinct elements from all the given [Sets](https://beginnersbook.com/2019/03/python-sets/). The union is denoted by ∪ symbol. Lets say we have a Set A: {1, 2, 3} and Set B: {2, 3, 4} then to find the union of these sets we can call this method either like this A.union(B) or B.union(A) and the returned set would contain the elements {1, 2, 3, 4}.

**Set union() method Syntax**

set.union(set1, set2,...)

**Parameters:** This method accepts sets as parameters.  
**Return value:** This method returns a set with the distinct elements of all the sets.

**Python Set union() method example**

In the following example we have three sets and we are finding union of sets X & Y, X & Z and X, Y & Z using union() method. We can also use | operator to find the union of sets.

# Set X

X = {1, 2, 3}

# Set Y

Y = {2, 3, 4}

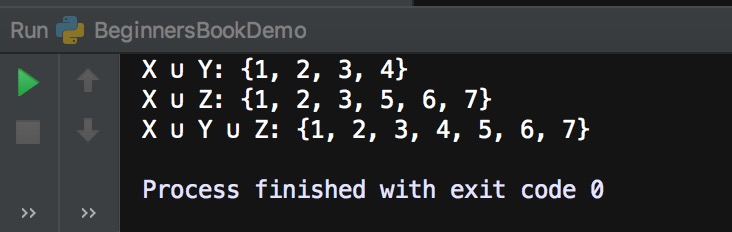
# Set Z

Z = {5, 6, 7}

print("X ∪ Y:", X.union(Y))

print("X ∪ Z:", X.union(Z))

print("X ∪ Y ∪ Z:", X.union(Y, Z))

**Output:**  


**Finding union of sets using | operator**

Lets take the same example that we have seen above, however here we will use the | operator instead of union() method to find the union of sets.

# Set X

X = {1, 2, 3}

# Set Y

Y = {2, 3, 4}

# Set Z

Z = {5, 6, 7}

print("X ∪ Y:", X|Y)

print("X ∪ Z:", X|Z)

print("X ∪ Y ∪ Z:", X|Y|Z)

**Output:**

X ∪ Y: {1, 2, 3, 4}

X ∪ Z: {1, 2, 3, 5, 6, 7}

X ∪ Y ∪ Z: {1, 2, 3, 4, 5, 6, 7}

**Python Set update() method with examples**

The **Set update() method** accepts another iterable such as [Set](https://beginnersbook.com/2019/03/python-sets/), [List](https://beginnersbook.com/2018/02/python-list/), [String](https://beginnersbook.com/2018/02/python-strings/), [Dictionary](https://beginnersbook.com/2019/03/python-dictionary/) as a parameter and adds the elements of these iterable to the calling set. This method converts the passed iterable to the set before adding their elements to the calling Set. For example – Lets say we have a Set A: {1, 2, 3} and a List lis: [2, “hello”] then calling A.update(lis) would update the set A and the elements of set A after update would be {1, 2, 3, “hello”}.

**Set update() method Syntax**

set.update(iterable)

**Parameters:** This method accepts iterable (list, tuple, dictionary etc.) as parameters.  
**Return Value:** It does not return anything, it just updates the calling Set.

**Python Set update() method example**

In the following example we have two Sets of numbers X & Y and we are calling X.update(Y) to add the elements of set Y to the Set X.

# Set X

X = {1, 2, 3}

# Set Y

Y = {2, 3, 4}

# Displaying X & Y before update()

print("X is:",X)

print("Y is:",Y)

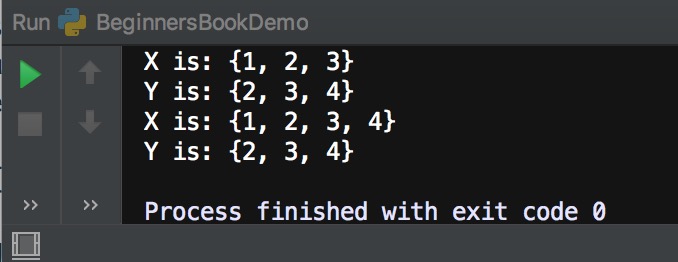
# Calling update() method

X.update(Y)

# Displaying X & Y after update()

print("X is:",X)

print("Y is:",Y)

**Output:**  


**Set update() method example with List, Tuple, String, Dictionary**

In the following example we are adding the elements of a list, tuple, string & dictionary to the calling set X using the update() method.

# Set X

X = {1, 2, 3}

# List lis

lis = [3, 5]

# tuple

t = (77, 99)

# String

s = "abc"

# Dictionary dict

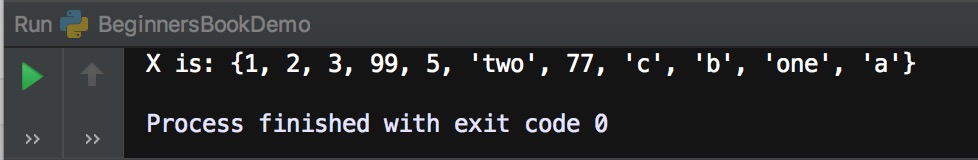
dict = {"one": 1, "two": 2}

# Calling update() method

X.update(lis, t, s, dict)

# Displaying X after update()

print("X is:",X)

**Output:**  


**Python Set symmetric\_difference() method with examples**

**Set symmetric\_difference()** method returns a symmetric difference of two given [sets](https://beginnersbook.com/2019/03/python-sets/). A symmetric difference of two sets X and Y contains the elements that are in either set X or Set Y but not in both. For example – symmetric difference of set {1, 2, 3} and {2, 3, 4} would be {1, 4} because elements 2 and 3 are present in both the sets.

**Set symmetric\_difference() syntax**

set.symmetric\_difference(another\_set)

**Parameter**: It takes a set as a parameter  
**Return Value**: It returns a new set which is a symmetric difference of the two given sets.

**Python Set symmetric\_difference() Example**

In the following example we have three sets X, Y and Z. Sets Y and Z are same. When we find the symmetric difference between same sets it returns nothing, as shown in the output of the following example. We can also find the symmetric difference using ^ operator, which is discussed in the next section of this same article.

# Set X

X = {1, 2, 3}

# Set Y

Y = {2, 3, 4}

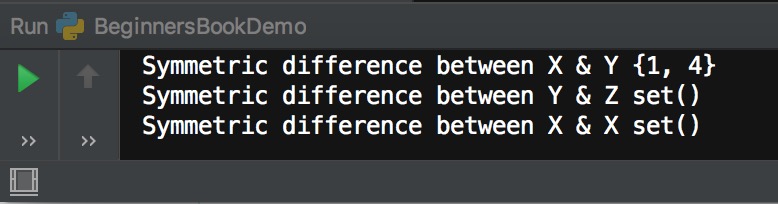
# Set Z

Z = {2, 3, 4}

print("Symmetric difference between X & Y", X.symmetric\_difference(Y))

print("Symmetric difference between Y & Z", Y.symmetric\_difference(Z))

print("Symmetric difference between X & X", X.symmetric\_difference(X))

**Output:**  


**Finding the symmetric difference between two sets using ^ operator**

We can use the ^ operator instead of symmetric\_difference() method to find the symmetric difference between two sets as shown in the following example.

# Set X

X = {1, 2, 3}

# Set Y

Y = {2, 3, 4}

# Set Z

Z = {2, 3, 4}

print(X^Y)

print(X^Z)

print(Y^Z)

# symmetric difference with self

print(X^X)

print(Y^Y)

**Output:**

{1, 4}

{1, 4}

set()

set()

set()