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.

A set can have any number of items and they may be of different types (integer, float, tuple, string, etc.). But a set cannot have mutable elements like lists, sets or dictionaries as its elements.

Let's see an example,

```
# 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}
```

In the above example, we have created different types of sets by placing all the elements inside the curly braces `{}`.

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 = { }

# check data type of empty_set
print('Data type of empty_set:', type(empty_set))

# check data type of dictionary_set
print('Data type of empty_dictionary:', type(empty_dictionary))
```

Output

```
Data type of empty_set: <class 'set'>
Data type of empty_dictionary: <class 'dict'>
```

Here,

- *empty_set* - an empty set created using `set()`
- *empty_dictionary* - an empty dictionary created using `{}`

Finally, we have used the `type()` function to know which class *empty_set* and *empty_dictionary* belong to.

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}
```

Here, we can see there are no duplicate items in the set as a set cannot contain duplicates.

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() 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}
```

In the above example, we have created a set named *numbers*. Notice the line,

```
numbers.add(32)
```

Here, `add()` adds **32** to our set.

Update Python Set

The 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'}
```

Here, all the unique elements of `tech_companies` are added to the `companies` set.

Remove an Element from a Set

We use the 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)
```

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
all()	Returns `True` if all elements of the set are true (or if the set is empty).
any()	Returns `True` if any element of the set is true. If the set is empty, returns `False`.
enumerate()	Returns an enumerate object. It contains the index and value for all the items of the set as a pair.
len()	Returns the length (the number of items) in the set.
max()	Returns the largest item in the set.
min()	Returns the smallest item in the set.
sorted()	Returns a new sorted list from elements in the set(does not sort the set itself).
sum()	Returns the sum of all elements in the set.

Iterate Over a Set in Python

```
fruits = {"Apple", "Peach", "Mango"}

# for loop to access each fruits
for fruit in fruits:
    print(fruit)
```

Output

```
Mango
Peach
Apple
```

Here, we have used for loop to iterate over a set in Python.

Find Number of Set Elements

We can use the 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))
```

Output

```
Set: {8, 2, 4, 6}
Total Elements: 4
```

Here, we have used the `len()` method to find the number of elements present in a Set.

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

Set Union in Python

We use the `|` operator or the 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))
```

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 â‹ƒ B` set operation.

Set Intersection

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

Set Intersection in Python

Set Intersection in Python

In Python, we use the `&` operator or the 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}
```

Note: `A&B` and `intersection()` is equivalent to `A â‹, B` set operation.

Difference between Two Sets

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

Set Difference in Python

Set Difference in Python

We use the `-` operator or the 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}
```

Note: `A - B` and `A.difference(B)` is equivalent to `A - B` set operation.

Set Symmetric Difference

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

Set Symmetric Difference in Python

Set Symmetric Difference in Python

In Python, we use the ^ operator or the 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
```

In the above example, *A* and *B* have the same elements, so the condition

```
if A == B
```

evaluates to `True`. Hence, the statement `print('Set A and Set B are equal')` inside the `if` is executed.

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()	Adds an element to the set
clear()	Removes all elements from the set
copy()	Returns a copy of the set
difference()	Returns the difference of two or more sets as a new set

difference_update()	Removes all elements of another set from this set
discard()	Removes an element from the set if it is a member. (Do nothing if the element is not in set)
intersection()	Returns the intersection of two sets as a new set
intersection_update()	Updates the set with the intersection of itself and another
isdisjoint()	Returns `True` if two sets have a null intersection
issubset()	Returns `True` if another set contains this set
issuperset()	Returns `True` if this set contains another set
pop()	Removes and returns an arbitrary set element. Raises `KeyError` if the set is empty
remove()	Removes an element from the set. If the element is not a member, raises a `KeyError`
symmetric_difference()	Returns the symmetric difference of two sets as a new set
symmetric_difference_update()	Updates a set with the symmetric difference of itself and another
union()	Returns the union of sets in a new set
update()	Updates the set with the union of itself and others

Also Read:

- Python set()
- Python Set Methods

Table of Contents