Sets in Python

Dipesh Thapa

October 17, 2024

1 Introduction

A **set** in Python is an unordered collection of unique elements. It is a built-in data type that provides high-performance operations for membership testing, removing duplicates from sequences, and performing mathematical set operations such as union, intersection, difference, and symmetric difference. Sets are mutable, meaning their contents can be changed.

2 Creating a Set

Sets can be created using curly braces or the `set()` constructor.

2.1 Empty Set

To create an empty set, use the `set()` constructor. Using curly braces with nothing inside will create an empty dictionary instead of a set.

```
# Using curly braces
set_a = {1, 2, 3}

# Using the set constructor
set_b = set()  # This creates an empty set
```

3 Basic Operations on Sets

3.1 Adding Elements

To add a single element to a set, use the `add()` method.

```
set_a = {1, 2, 3}
set_a.add(4)  # set_a becomes {1, 2, 3, 4}
```

3.2 Removing Elements

You can remove an element using `remove()` or `discard()`.

```
set_a.remove(3)  # Raises KeyError if 3 is not present
set_a.discard(2)  # Does not raise an error if 2 is
   not present
```

3.3 Clearing a Set

To remove all elements from a set, use the `clear()` method.

```
set_a.clear()  # set_a is now empty
```

3.4 Checking Membership

To check if an element exists in a set, use the `in` operator.

```
if 1 in set_a:
    print("1␣is␣in␣the␣set")  # Output: 1 is in the
       set
```

3.5 Length of a Set

To find the number of elements in a set, use the `len()` function.

```
length = len(set_a)  # Returns the number of unique
   elements
```

4 Set Operations

4.1 Union

The union of two sets contains all unique elements from both sets. Use the `|` operator or the `union()` method.

```
set_a = {1, 2, 3}
set_b = {3, 4, 5}
union_set = set_a | set_b  # or set_a.union(set_b)
# union_set becomes {1, 2, 3, 4, 5}
```

4.2 Intersection

The intersection of two sets contains only the elements that are present in both sets. Use the operator or the `intersection()` method.

```
intersection_set = set_a & set_b  # or set_a.
    intersection(set_b)
# intersection_set becomes {3}
```

4.3 Difference

The difference between two sets contains elements that are in the first set but not in the second. Use the - operator or the `difference()` method.

```
difference_set = set_a - set_b  # or set_a.difference(
    set_b)
# difference_set becomes {1, 2}
```

4.4 Symmetric Difference

The symmetric difference contains elements that are in either of the sets but not in both. Use the *operatorortheв*`symmetric_difference()`*method.*

```
symmetric_difference_set = set_a ^ set_b  # or set_a.
    symmetric_difference(set_b)
# symmetric_difference_set becomes {1, 2, 4, 5}
```

4.5 Subset and Superset

`To check if one set is a subset or superset of another, use the issubset() and issuperset() methods.`

```
set_c = {1, 2}
is_subset = set_c.issubset(set_a)  # True
is_superset = set_a.issuperset(set_c)  # True
```

4.6 Disjoint Sets

`To check if two sets have no elements in common, use the isdisjoint() method.`

```
set_d = {5, 6}
are_disjoint = set_a.isdisjoint(set_d)  # True
```

4.7 Copying a Set

`To create a shallow copy of a set, use the copy() method.`

```
set_copy = set_a.copy()  # Creates a copy of set_a
```

4.8 Iterating Over a Set

You can iterate over the elements of a set using a for loop.

```
for element in set_a:
    print(element)  # Prints each element in set_a
```

5 Practical Use Cases for Sets

Sets are particularly useful in various scenarios, including:

- Removing Duplicates: Quickly remove duplicates from lists.

  ```
  my_list = [1, 2, 2, 3, 4, 4, 5]
  unique_list = list(set(my_list))  # unique_list
      becomes [1, 2, 3, 4, 5]
  ```

- Membership Testing: Efficiently check if an item exists in a collection.

  ```
  my_set = {1, 2, 3}
  exists = 2 in my_set  # True
  ```

- Set Operations: Perform mathematical operations like union and intersection.

  ```
  set_a = {1, 2, 3}
  set_b = {3, 4, 5}
  common_elements = set_a & set_b  # {3}
  ```

- Finding Unique Items: Quickly find unique items in multiple lists.