Operation	HashSet	TreeSet
Insertion	O(1)	O(log n)
Removal	O(1)	O(log n)
Contains	O(1)	O(1)

Definition

A set is a collection of distinct elements that are unordered, allows efficient membership testing and eliminating duplicate elements.

Adding Elements

You can add elements to a set using the `add` operation. If the element is already present, it will not be added again.

Removing Elements

Sets provide the `remove` operation to remove an element from the set. If the element is not present, an error may be raised, depending on the programming language or implementation.

Membership Testing

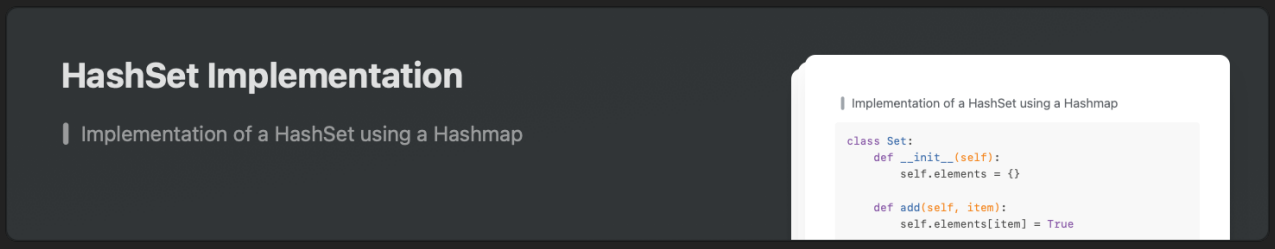
Sets provide a fast way to test if an element belongs to the set. This is achieved through a hash-based lookup mechanism, which allows constant-time membership testing on average.

No Indexing

Unlike arrays or lists, sets do not support indexing. You cannot access elements in a set by their position. Instead, sets are typically used for membership testing and checking if an element exist

Use Cases

- Eliminating duplicates from a list
- Checking if two sets have any elements in common
- Implementing mathematical operations like union, intersection, and difference.



↑ Sets

HashSet Implementation

> Implementation of a HashSet using a Hashmap

```
class Set:
    def __init__(self):
        self.elements = {}

    def add(self, item):
        self.elements[item] = True

    def remove(self, item):
        if item in self.elements:
            del self.elements[item]
        else:
            raise KeyError("Item not found in the set")

    def contains(self, item):
        return item in self.elements

    def size(self):
        return len(self.elements)
```