A set is an unordered collection of unique and immutable elements. It is useful when you want to avoid duplicates or perform set operations like union, intersection, and difference.

**Step 1: Create a Set**

You can create a set using curly braces {} or the set() function.

**Example:**

# Creating a set

my\_set = {1, 2, 3, 4}

print("Set:", my\_set)

# Using set() function

another\_set = set([4, 5, 6, 6, 7]) # Duplicate '6' will be removed

print("Another Set:", another\_set)

**Output:**

Set: {1, 2, 3, 4}

Another Set: {4, 5, 6, 7}

**Explanation:**

* Sets do not allow duplicate values.
* Sets are unordered, so elements may not appear in the order you added them.

**Step 2: Add and Remove Elements**

You can add elements using add() and remove them using remove() or discard().

**Example:**

my\_set = {1, 2, 3}

# Adding an element

my\_set.add(4)

print("After Adding:", my\_set)

# Removing an element

my\_set.remove(2) # Raises an error if the element is not present

print("After Removing:", my\_set)

# Discarding an element

my\_set.discard(5) # Does not raise an error if the element is not present

print("After Discarding:", my\_set)

**Output:**

After Adding: {1, 2, 3, 4}

After Removing: {1, 3, 4}

After Discarding: {1, 3, 4}

**Explanation:**

* add() adds an element to the set.
* remove() raises an error if the element does not exist.
* discard() does not raise an error if the element does not exist.

**Step 3: Set Operations**

provides built-in methods to perform operations like union, intersection, difference, etc.

**Example 1: Union**

Combines all unique elements from two sets.

set1 = {1, 2, 3}

set2 = {3, 4, 5}

# Union

result = set1.union(set2)

print("Union:", result)

**Output:**

Union: {1, 2, 3, 4, 5}

**Example 2: Intersection**

Finds common elements between two sets.

# Intersection

result = set1.intersection(set2)

print("Intersection:", result)

**Output:**

Intersection: {3}

**Example 3: Difference**

Finds elements in one set but not in the other.

# Difference

result = set1.difference(set2)

print("Difference:", result)

**Output:**

Difference: {1, 2}

**Example 4: Symmetric Difference**

Finds elements in either set but not in both.

# Symmetric Difference

result = set1.symmetric\_difference(set2)

print("Symmetric Difference:", result)

**Output:**

Symmetric Difference: {1, 2, 4, 5}

**Step 4: Check Membership**

You can check if an element exists in a set using the in keyword.

**Example:**

my\_set = {1, 2, 3, 4}

# Check membership

print(3 in my\_set) # True

print(5 in my\_set) # False

**Output:**

graphql

True

False

**Step 5: Iterate Over a Set**

You can iterate over a set using a for loop.

**Example:**

my\_set = {10, 20, 30, 40}

# Iterating through the set

for num in my\_set:

print(num)

**Output:**

10

20

30

40

**Real-World Example: Removing Duplicates**

# List with duplicates

data = [1, 2, 2, 3, 4, 4, 5]

# Remove duplicates using a set

unique\_data = list(set(data))

print("Unique Data:", unique\_data)

**Output:**

less

Unique Data: [1, 2, 3, 4, 5]

**Key Points to Remember:**

1. **Unique Elements**: Sets automatically remove duplicates.
2. **Unordered**: The order of elements in a set is not guaranteed.
3. **Mutable**: You can add or remove elements from a set.
4. **Set Operations**: Useful for mathematical operations like union, intersection, and difference.