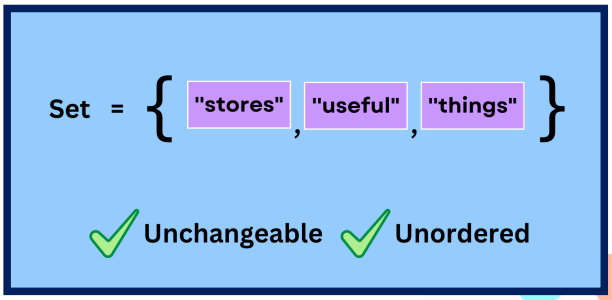
# Sets in Python

In Python, a set is an unordered collection of unique elements. Sets are mutable, but the elements inside them must be immutable (e.g., numbers, strings, tuples).




## Key Uses of Sets in Python

- Remove duplicates from lists or other collections.
- Perform set operations like union, intersection, difference, etc.
- Fast membership testing (much faster than lists).
- Compare datasets to find common or unique elements.
- Filter data based on allowed or restricted values.
- Find unique characters or words in strings or documents.
- Efficient data lookup without duplicates or ordering.
- Set comprehension for creating sets using expressions.
- Count unique values in a collection quickly.

## Common Set Operations

| Operation | Syntax / Method | Example |
|---|---|---|
| Add element | set.add(x) | s.add(5) |
| Remove element | set.remove(x) | s.remove(3) |
| Discard element | set.discard(x) | s.discard(3) |
| Clear set | set.clear() | s.clear() |
| Union | `set1 | set2` |
| Intersection | set1 & set2 | a & b |
| Difference | set1 - set2 | a - b |
| Symmetric Difference | set1 ^ set2 | a ^ b |
| Check subset | a <= b | a.issubset(b) |
| Check superset | a >= b | a.issuperset(b) |

# Sets in Python

**1. Add Element – set.add(x)**

- Adds a single element to the set.
- If the element already exists, it won't be added again.

**Example**:

```
s = {1, 2, 3}
s.add(4)
print(s)
```

**Output**:

```
{1, 2, 3, 4}
```

**2. Remove Element – set.remove(x)**

- Removes a specific element from the set.
- If the element doesn't exist, Python gives an error.

**Example:**

```
s = {1, 2, 3}
s.remove(2)
print(s)
```

**Output:**

```
{1, 3}
```

Example of Error:

```
s.remove(5)   # 5 not in set → KeyError
```

**3. Discard Element – set.discard(x)**

- Works like remove(), but it does not show an error if the element isn't present.

**Example:**

```
s = {1, 2, 3}
s.discard(2)   # removes 2
s.discard(5)   # does nothing, no error
print(s)
```

**Output:**

```
{1, 3}
```

**4. Clear Set – set.clear()**

- Removes all elements from the set, leaving it empty.

**Example:**

```
s = {1, 2, 3}
s.clear()
print(s)
```

**Output:**

```
set()
```

**5. Union — set1 | set2 or set1.union(set2)**

• Combines elements from both sets (no duplicates).

**Example:**

```
a = {1, 2, 3}
b = {3, 4, 5}
print(a | b)
```

**Output:**

```
{1, 2, 3, 4, 5}
```

**6. Intersection — set1 & set2 or set1.intersection(set2)**

• Returns elements common to both sets.

**Example:**

```
a = {1, 2, 3, 4}
b = {3, 4, 5, 6}
print(a & b)
```

**Output:**

```
{3, 4}
```

**7. Difference — set1 - set2 or set1.difference(set2)**

• Returns elements that are in the first set but not in the second.

**Example:**

```
a = {1, 2, 3, 4}
b = {3, 4, 5}
print(a - b)
```

**Output:**

```
{1, 2}
```

**8. Symmetric Difference — set1 ^ set2 or set1.symmetric_difference(set2)**

• Returns elements that are in either of the sets, but not in both.

## Sets in Python

**Example:**
```
a = {1, 2, 3}
b = {3, 4, 5}
print(a ^ b)
```

**Output:**
{1, 2, 4, 5}

**9. Check Subset — a <= b or a.issubset(b)**

• Checks if all elements of set a are present in set b.

**Example:**
```
a = {1, 2}
b = {1, 2, 3, 4}
print(a <= b)
```

**Output:**
True

**10. Check Superset — a >= b or a.issuperset(b)**

• Checks if set a contains all elements of set b.

**Example:**
```
a = {1, 2, 3, 4}
b = {2, 3}
print(a >= b)
```

**Output:**
True

**Add Multiple Elements in Sets**

Use .update() to add more than one element.

```
fruits = {"apple", "banana", "cherry"}
fruits.update(["grape", "orange"])
print(fruits)
```

**Output:**

{'apple', 'banana', 'mango', 'grape', 'orange'}

# Sets in Python

**No Duplicates Allowed**

If you try to add the same element twice, Python automatically removes duplicates.

**Example:**

```
numbers = {1, 2, 2, 3, 4, 4}
print(numbers)
```

**Output:**

```
{1, 2, 3, 4}
```

**Solved Example: Write a Python program to:**

1. Store the names of students in **Class A** and **Class B** using sets.
2. Find the students who are **common in both classes**.
3. Find the students who are **only in Class A** (not in Class B).
4. Display the results using print statements.

**Solution:**

```
students_classA = {"Ravi", "Sita", "Amit", "Rahul"}
students_classB = {"Amit", "Rina", "Rahul", "Neha"}

# Find students common in both classes
common_students = students_classA & students_classB

# Find students unique to Class A
only_A = students_classA - students_classB

print("Common Students:", common_students)
print("Only in Class A:", only_A)
```

**Explanation:**

1. **Creating Sets:**

```
students_classA = {"Ravi", "Sita", "Amit", "Rahul"}
students_classB = {"Amit", "Rina", "Rahul", "Neha"}
```

- Here, we created two sets.
- Each set stores the names of students in that class.
- Sets automatically remove duplicates (if any).

2. **Finding Common Students (Intersection):**

```
common_students = students_classA & students_classB
```

- The & operator finds the **intersection** of two sets.
- It gives the elements that are **present in both sets**.
- In this case: "Amit" and "Rahul" are common in both classes.

## Sets in Python

3. **Finding Students Only in Class A (Difference):**
only_A = students_classA - students_classB
- The - operator finds the **difference** between two sets.
- It gives elements that are **in Class A but not in Class B.**
- Here, "Ravi" and "Sita" are only in Class A.

### Assignment

**Ques 1:** Create two sets:
A = {1, 2, 3, 4, 5}
B = {4, 5, 6, 7, 8}

Perform and print the following:
1. Union of both sets
2. Intersection of both sets
3. Difference of A and B
4. Symmetric difference of A and B

**Expected Output:**
Union: {1, 2, 3, 4, 5, 6, 7, 8}
Intersection: {4, 5}
Difference (A - B): {1, 2, 3}
Symmetric Difference: {1, 2, 3, 6, 7, 8}

**Ques 2:** Write a Python program to:
- Create a set named fruits containing "apple", "banana", "mango".
- Add "orange" to the set.
- Remove "banana" from the set.
- Display the final set.

**Expected Output:**
{'apple', 'orange', 'mango'}