Python Sets

- **Introduction to Sets**:
    - Sets in Python are used to store unique values.
    - Declared using curly braces `{}`.
        - Example: `set_a = {1, 2, 3, 4, 5}`.
- **Key Features of Sets**:
    - **No Duplicates**: If duplicates are added, they are automatically removed.
        - Example: Adding another `5` to `set_a` still results in `{1, 2, 3, 4, 5}`.
    - **Unordered**: Elements in a set do not follow a specific order.
    - **Not Subscriptable**: Sets do not support indexing, so attempting `set_a[0]` raises a `TypeError`.
- **Basic Set Operations**:
    - **Add Items**: Use `.add(value)` to add a single element to a set.
        - Example: `set_a.add(6)` results in `{1, 2, 3, 4, 5, 6}`.
    - **Remove Items**:
        - `.remove(value)`: Removes the specified element; raises an error if the element doesn't exist.
        - `.discard(value)`: Similar to `.remove()`, but does not raise an error for non-existent elements.
        - Example: `set_a.remove(2)` or `set_a.discard(2)` results in `{1, 3, 4, 5}`.
- **Mathematical Operations with Sets**:
    - **Union**:
        - Combines elements from two sets, excluding duplicates.
        - Methods: `.union(set_b)` or `set_a | set_b`.
        - Example: For `set_a = {1, 2, 3, 4, 5}` and `set_b = {4, 5, 6, 7, 8}`, the result is `{1, 2, 3, 4, 5, 6, 7, 8}`.
    - **Intersection**:
        - Finds common elements between two sets.
        - Methods: `.intersection(set_b)` or `set_a & set_b`.
        - Example: Result for `set_a` and `set_b` is `{4, 5}`.
    - **Difference**:
        - Returns elements in one set that are not in the other.
        - Methods: `.difference(set_b)` or `set_a - set_b`.
        - Example: Result for `set_a` and `set_b` is `{1, 2, 3}`.

- **Symmetric Difference**:
    - Returns elements in either set, but not in both.
    - Methods: `.symmetric_difference(set_b)` or `set_a ^ set_b`.
    - Example: Result for `set_a` and `set_b` is `{1, 2, 3, 6, 7, 8}`.

- **Additional Notes**:

  - Sets do not maintain order, so elements may appear in different orders in the output.
  - Unlike lists, sets cannot be accessed by index (e.g., `set_a[0]` raises a `TypeError`).
  - Sets are suitable for operations requiring unique elements and mathematical operations like union or intersection.

- **Conclusion**:

  - Sets are useful for working with unique and unordered data.
  - They provide several methods for adding, removing, and performing mathematical operations efficiently.