# PYTHON TUTORIAL FOR BEGINNERS

Source: www.youtube.com/@RishabhMishraOfficial

**Chapter - 18**

## Set in Python

- What is a Set
- Create Sets
- Set Operations
- Set Methods
- Set Iteration
- Set Comprehensions

### Set in Python

A **set** is a collection of **unique items** in Python. Sets **do not** allow **duplicate** items and do not maintain any particular order so it **can't be indexed**.

**Characteristics of Sets:**

- **Unordered:** Elements have no defined order. You cannot access elements by index.
- **Unique Elements:** No duplicates allowed. Each element must be distinct.
- **Mutable:** You can add or remove elements after creation.
- **Immutable Elements:** individual elements inside a set cannot be modified/replaced

Example:

```
vowels = {'a', 'e', 'i', 'o', 'u'}
```

### Create Set in Python

There are two primary ways to create a set in Python:

```
1. Using Curly Braces {}
my_set = {1, 2, 3, 4, 5}
print(my_set)  # Output: {1, 2, 3, 4, 5}

2. Using the set() Constructor
my_set = set([1, 2, 3, 4, 5])
print(my_set)  # Output: {1, 2, 3, 4, 5}
```

**Note:** An empty set cannot be created using {} as it creates an empty dictionary. Use set() instead.

```
empty_set = set()
print(empty_set)  # Output: set()
```

## Set Operations

**1. Adding Elements :** Use the add() method to add a single element to a set.

```
fruits = {'apple', 'banana'}
fruits.add('cherry')
print(fruits)  # Output: {'apple', 'banana', 'cherry'}
```

**2. Removing Elements:** Use the remove() or discard() methods to remove elements.

- **remove()** raises an error if the element is not found.
- **discard()** does not raise an error if the element is missing.

```
fruits = {'apple', 'banana', 'cherry'}

# Using remove()
fruits.remove('banana')
print(fruits)  # Output: {'apple', 'cherry'}

# Using discard()
fruits.discard('orange')  # No error even if 'orange' is not in the set
print(fruits)  # Output: {'apple', 'cherry'}
```

## Set Methods

**1. Union:** Combines elements from two sets, removing duplicates.

```
set_a = {1, 2, 3}
set_b = {3, 4, 5}
union_set = set_a.union(set_b)
print(union_set)  # Output: {1, 2, 3, 4, 5}
```

**Alternative Syntax:** `union_set = set_a | set_b`

**2. Intersection:** Includes only elements present in both sets.

```
set_a = {1, 2, 3}
set_b = {2, 3, 4}
intersection_set = set_a.intersection(set_b)
print(intersection_set)  # Output: {2, 3}
```

**Alternative Syntax:** `intersection_set = set_a & set_b`

**3. Difference:** Elements present in the first set but not in the second.

```
set_a = {1, 2, 3, 4}
set_b = {3, 4, 5}
difference_set = set_a.difference(set_b)
print(difference_set)  # Output: {1, 2}
```

**Alternative Syntax:** `difference_set = set_a - set_b`

**4. Symmetric Difference:** Elements in either set, but not in both.

```
set_a = {1, 2, 3}
set_b = {3, 4, 5}
sym_diff_set = set_a.symmetric_difference(set_b)
print(sym_diff_set)  # Output: {1, 2, 4, 5}
```

**Alternative Syntax:** `sym_diff_set = set_a ^ set_b`

## Set Iterations – Loop

You can use a **for loop** to go through each element in a set.

```
# Using for loop - Printing each number from a set
numbers = {1, 2, 3, 4, 5}
for number in numbers:
    print(number)
```

```
# Using while loop - first convert set to a list then use while loop because sets do not support indexing.
```

## Set Comprehension

Set comprehensions allow concise and readable creation of sets. Similar to list comprehensions but for sets.

```
# Syntax:
new_set = {expression for item in iterable if condition}

# Example:
squares = {x**2 for x in range(1, 6)}
print(squares)  # Output: {1, 4, 9, 16, 25}
```

## Set Common Use Cases

- **Removing Duplicates:** Easily eliminate duplicate entries from data.
- **Membership Testing:** Quickly check if an item exists in a collection.
- **Set Operations:** Perform mathematical operations like union, intersection, and difference.
- **Data Analysis:** Useful in scenarios requiring unique items, such as tags, categories, or unique identifiers.

```
# Example:  Removing Duplicates from a List
numbers = [1, 2, 2, 3, 4, 4, 5]
unique_numbers = set(numbers)
print(unique_numbers)  # Output: {1, 2, 3, 4, 5}
```

Python Tutorial Playlist: **Click Here**

https://www.youtube.com/playlist?list=PLdOKnrf8EcP384Ilxra4UlK9BDJGwawg9