# 😊 Sets and Booleans

### Sets in Python

A **set** is an unordered collection of unique elements. It is useful for storing distinct values and performing mathematical set operations like union, intersection, and difference.

### Characteristics of Sets

1. **Unordered** – Elements do not maintain a fixed order.
2. **Mutable** – Elements can be added or removed.
3. **Unique Elements** – Duplicate values are automatically removed.
4. **Heterogeneous** – Can contain different data types, but only immutable types (e.g., numbers, strings, tuples).

### Basic Set Operations

- **Creating a Set**: Defined using curly braces {} or the set() function.
- **Adding Elements**: Use add().
- **Removing Elements**: Use remove(), discard(), or pop().
- **Set Operations**: Supports union(), intersection(), difference(), and symmetric_difference().
- **Checking Membership**: Use in to check if an element exists in the set.

### Common Set Methods

- add(value) – Adds an element.
- remove(value) – Removes an element, raises an error if not found.
- discard(value) – Removes an element, does nothing if not found.
- union(set2) – Combines two sets.
- intersection(set2) – Finds common elements.

### Booleans in Python

A **Boolean** (bool) represents one of two values: **True** or **False**. It is often used for logical operations and control flow in conditional statements.

### Boolean Values

- True – Represents a true condition.
- False – Represents a false condition.

### Boolean Operations

- **Comparison Operators** (==, !=, <, >, <=, >=) return True or False.

- **Logical Operators** (and, or, not) combine Boolean values.

**Truthy and Falsy Values**

In Python, some values evaluate to False in a Boolean context:

- 0, None, False, '' (empty string), [] (empty list), {} (empty dictionary). All other values evaluate to True.

Booleans are widely used in decision-making and logical computations.

## 😄 To begin with the Lab

1. Sets are an unordered collection of *unique* elements. We can construct them by using the set () function.

```
[1]: x = set()

[2]: # We add to sets with the add() method
     x.add(1)

[3]: #Show
     x

[3]: {1}
```

2. Here you can see that we added another number to the set. Notice how it won't place another 1 there. That's because a set is only concerned with unique elements.

```
[4]: # Add a different element
     x.add(2)

[5]: #Show
     x

[5]: {1, 2}

[6]: # Try to add the same element
     x.add(1)

[7]: #Show
     x

[7]: {1, 2}
```

3. Here we created a list with repeated items and we called it we get the unique values.

[8]:
```
# Create a list with repeats
list1 = [1,1,2,2,3,4,5,6,1,1]
```

[9]:
```
# Cast as set to get unique values
set(list1)
```

[9]:
```
{1, 2, 3, 4, 5, 6}
```

4. Python comes with Booleans (with predefined True and False displays that are basically just the integers 1 and 0). It also has a placeholder object called None.

[10]:
```
# Set object to be a boolean
a = True
```

[11]:
```
#Show
a
```

[11]:
```
True
```

5. We can also use comparison operators to create Booleans.

[12]:
```
# Output is boolean
1 > 2
```

[12]:
```
False
```

6. We can use None as a placeholder for an object that we don't want to reassign yet.

[13]:
```
# None placeholder
b = None
```

• • •

[14]:
```
# Show
print(b)
```

```
None
```