🔹 1. Boolean Data Type (`bool`)

Definition:

A **Boolean** is a special data type that can only have **two values**:
`True` or `False`.

Explanation:

- Boolean values are used when we want to **check conditions** or make decisions.
- These are the **result** of comparison or logical operations.
- `True` and `False` must be written with **capital letters**.

Examples:

```
is_raining = True
print(is_raining)         # True

is_hungry = False
print(type(is_hungry))    # <class 'bool'>
```

✅ Booleans from Comparisons:

```
x = 10
y = 5

print(x > y)   # True
print(x == y)  # False
```

📝 Practice Questions:

1. Create a variable `is_sunny = True` and print it.
2. Try: `print(8 < 10)` and `print(15 == 20)`
3. Check `type(True)` in Python.

🔹 2. Comparison Operators

Operator	Meaning	Example	Result
`==`	Equal to	`5 == 5`	`True`
`!=`	Not equal to	`5 != 3`	`True`
`>`	Greater than	`10 > 5`	`True`
`<`	Less than	`3 < 7`	`True`
`>=`	Greater than or equal to	`10 >= 10`	`True`
`<=`	Less than or equal to	`8 <= 6`	`False`

🧪 Examples:

```
a = 6
b = 9

print(a > b)     # False
print(a != b)    # True
```

Practice Questions:

1. Check if `25` is greater than `20`.
2. Take 2 inputs and check if they are equal.
3. Try: `10 <= 10` and `10 >= 11`.

🔹 3. Logical Operators

Definition:

Logical operators combine **multiple conditions** and return a Boolean (`True` or `False`).

Operator	Meaning	Example	Result
`and`	True if **both** are True	`5 > 2 and 10 > 3`	True
`or`	True if **one** is True	`5 > 10 or 10 > 3`	True
`not`	Reverses the result	`not (5 > 2)`	False

Examples:

```
x = 5
y = 10

print(x > 3 and y < 20)     # True
print(x > 10 or y < 20)     # True
print(not x > 10)           # True
```

📝 Practice Questions:

1. Take two numbers and check if both are greater than 10.

2. Check if either of them is even using `or`.

3. Use `not` to reverse a condition like `x > 100`.

🔹 1. What are Conditional Statements?

Definition:

Conditional statements are used to let a program **make decisions** based on conditions.
They help a program decide **what to do next**, depending on **True or False** outcomes.

Real-Life Examples:

- **If it rains**, we take an umbrella.
- **If you're hungry**, eat food.
- **If you get 90+ marks**, you get an A grade.

Just like we take actions based on situations, Python does the same.

Why do we need conditional statements in programming?

- To **control what code should run** and when
- To **avoid running all code blindly**
- To write **smart, dynamic programs**

Python Provides:

Keyword	Purpose
`if`	Check a condition
`elif`	Check another condition (optional)
`else`	Run code if no conditions are True

We'll start with just the **`if`** statement.

🔹 2. The `if` Statement (Basic Decision Maker)

Definition:

The `if` statement is used to **check a condition**.
If the condition is **True**, the code under it runs.

Syntax:

```
if condition:
    # code to run if condition is True
```

Don't forget the **colon `:`**
Always use **indentation** for code under `if`

Example 1: Basic Number Check

```
number = 10

if number > 5:
```

```
    print("Number is greater than 5")
```

Since `number > 5` is True, it prints the message.

Example 2: Nothing happens if False

```
age = 14

if age >= 18:
    print("You can vote.")
```

Output: Nothing happens because the condition is False.

Example 3: String check

```
city = "Mumbai"

if city == "Mumbai":
    print("Welcome to the financial capital!")
```

Example 4: User input

```
marks = int(input("Enter marks: "))

if marks >= 90:
    print("Excellent! You got an A.")
```

Practice Questions:

Try writing these with only `if`:

1. Check if a number is **positive**.
2. Ask for user's name. If it's `"admin"`, print `"Access granted"`
3. Take input for temperature. If it's above 40, print `"It's too hot!"`
4. Check if an entered number is divisible by 2.
5. If user enters a number >1000, print `"That's a big number!"`
6. Check if entered age is above 60 and print `"You are a senior citizen"`
7. Ask for marks. If they are 100, print `"Perfect score!"`