**Conditional Statements in Python**

Conditional statements allow you to control the flow of a program based on conditions (**True** / **False**).

**1️⃣ if Statement**

* Used to execute a block of code if a condition is true.

x = 10

if x > 5:

print("x is greater than 5")

**2️⃣ if-else Statement**

* Used when you need one block for true and another for false.

x = 3

if x > 5:

print("x is greater than 5")

else:

print("x is less than or equal to 5")

**3️⃣ if-elif-else Ladder**

* Used when you have multiple conditions.

x = 7

if x > 10:

print("x is greater than 10")

elif x == 7:

print("x is equal to 7")

else:

print("x is less than 10")

**4️⃣ Nested if**

* An if inside another if.

x = 20

if x > 10:

if x % 2 == 0:

print("x is greater than 10 and even")

**5️⃣ Conditional Expressions (Ternary Operator)**

* One-line shorthand for if-else.

x = 5

result = "Even" if x % 2 == 0 else "Odd"

print(result) # Output: Odd

**6️⃣ Logical Conditions**

You can combine multiple conditions using:

* **and**: All conditions must be True
* **or**: At least one condition is True
* **not**: Negates the condition

age = 18

has\_id = True

if age >= 18 and has\_id:

print("Allowed")

if age < 18 or not has\_id:

print("Not Allowed")

**7️⃣ Membership Conditions**

Check if a value exists in a collection using **in** or **not in**:

fruits = ["apple", "banana", "mango"]

if "apple" in fruits:

print("Apple is available")

if "grapes" not in fruits:

print("Grapes not available")

**8️⃣ Identity Conditions**

Check if two variables point to the same object using **is** or **is not**:

x = None

if x is None:

print("x is None")

if x is not None:

print("x is not None")

**9️⃣ Comparison Operators in Conditions**

* == → Equal
* != → Not equal
* < → Less than
* > → Greater than
* <= → Less than or equal
* >= → Greater than or equal

x = 10

if x != 5:

print("x is not equal to 5")

**🔹 Key Points**

1. Indentation (spaces/tabs) is **mandatory** in Python.
2. Boolean values **True/False** decide if a block runs.
3. Combine logical operators for complex conditions.
4. Use ternary operator for shorter inline conditions.