**Code Blocks and Colons in Python**

**1. Code Blocks in Python**

A **code block** is a group of statements that are meant to be executed together. In Python, **indentation** is used to define code blocks instead of braces {} like in languages such as C, C++, or Java.

**How Code Blocks Work**

* Python uses a colon (:) to indicate the start of a block.
* Code inside the block must be **indented** (typically 4 spaces per level).
* All lines within the same block must be indented at the same level.

**Example: Using Code Blocks with if Statements**

age = 20

if age >= 18: # Colon indicates the start of a block

print("You are an adult.") # This line is inside the block

print("You can vote.") # Also inside the block

**Output:**

You are an adult.

You can vote.

**Example: Code Blocks in Loops**

for i in range(3): # The loop starts here

print("Iteration:", i) # Inside the loop block

print("Hello") # Also inside the loop block

**Output:**

Iteration: 0

Hello

Iteration: 1

Hello

Iteration: 2

Hello

**Example: Incorrect Indentation (Error)**

if True:

print("This will cause an IndentationError") # Incorrect indentation

**Error:**

IndentationError: expected an indented block

**2. Boolean Logic in Python**

Boolean logic is used to perform comparisons and make decisions in programming. In Python, **Boolean values** can be True or False.

**1. Boolean Values**

print(True) # Output: True

print(False) # Output: False

**2. Boolean Expressions**

A Boolean expression is an expression that evaluates to either True or False.

**Comparison Operators (Used in Boolean Expressions)**

| **Operator** | **Description** | **Example** | **Result** |
| --- | --- | --- | --- |
| == | Equal to | 5 == 5 | True |
| != | Not equal to | 5 != 3 | True |
| > | Greater than | 7 > 3 | True |
| < | Less than | 2 < 5 | True |
| >= | Greater than or equal to | 10 >= 10 | True |
| <= | Less than or equal to | 8 <= 6 | False |

**Example: Using Comparison Operators**

x = 10

y = 20

print(x == y) # False

print(x < y) # True

print(x != y) # True

**3. Logical Operators**

Python provides logical operators to combine multiple conditions.

| **Operator** | **Description** | **Example** | **Result** |
| --- | --- | --- | --- |
| and | Returns True if **both** conditions are True | True and True | True |
| or | Returns True if **at least one** condition is True | True or False | True |
| not | Reverses the Boolean value | not True | False |

**Example: Using Logical Operators**

x = 5

y = 10

print(x > 2 and y < 15) # True (both conditions are True)

print(x > 2 or y > 20) # True (one condition is True)

print(not (x > 2)) # False (reverses True)

**Output:**

True

True

False

**4. Boolean Logic in if Statements**

temperature = 25

if temperature > 20 and temperature < 30:

print("The weather is nice today.")

**Output:**

The weather is nice today.

**5. Truthy and Falsy Values in Python**

Python treats some values as True (truthy) and others as False (falsy).

**Falsy Values**

The following values are considered False in Boolean contexts:

* 0 (zero)
* "" (empty string)
* None
* [], {}, () (empty list, dictionary, tuple)
* False

**Truthy Values**

All values that are **not falsy** are considered True.

**Example: Checking Truthy and Falsy Values**

print(bool(0)) # False

print(bool("")) # False

print(bool([])) # False

print(bool(5)) # True

print(bool("Hello")) # True

**Output:**

False

False

False

True

True

**Summary**

| **Concept** | **Explanation** |
| --- | --- |
| **Code Blocks** | Defined by indentation after a colon (:) |
| **Boolean Logic** | Evaluates conditions as True or False |
| **Comparison Operators** | Used to compare values (==, !=, >, <, >=, <=) |
| **Logical Operators** | and, or, not for combining conditions |
| **Truthy & Falsy Values** | Some values default to True or False in logical expressions |

By mastering **code blocks and Boolean logic**, you can write **efficient** and **logical** Python programs!