Booleans, Comparisons & the if Statement

AiCE Software Engineering Foundations — Python Programming

Decision-making is what separates a thinking program from a simple calculator. This lesson opens the topic of conditional logic — the boolean values conditions depend on, the comparison operators that build them, and the if statement that acts on them.

1. What Is a Conditional Statement?

A conditional statement tells Python: if a certain condition is true, do this — otherwise, do something else. It's the programming equivalent of everyday decision-making.

Think about how a mobile money agent processes a withdrawal. Before approving it, the system checks: does the account have sufficient balance? If yes, the transaction proceeds. If no, it's declined and the user is notified. That check is a conditional statement.

In Python, conditional statements are written with three keywords: if, elif, and else — together, they let a program evaluate conditions and follow different paths depending on the result.

2. Boolean Values — True and False

Before writing conditionals, you need the data type they depend on: booleans. A boolean is a value that is either True or False — nothing in between. In Python, the boolean type is called bool.

```
is_enrolled = True
has_paid = False

print(is_enrolled)
print(type(is_enrolled))
```

This produces:

```
True
<class 'bool'>
```

True and False Must Be Capitalized

True and False in Python are always capitalized, with an uppercase first letter. true and false (lowercase) are not recognized as boolean values and will cause a NameError. Every conditional statement evaluates to either True or False, and Python uses that result to decide which code path to follow.

3. Comparison Operators

Comparison operators compare two values and return a boolean result — the tools you use to build conditions:

Operator	Meaning	Example	Result
==	Equal to	5 == 5	True
!=	Not equal to	5 != 3	True
>	Greater than	10 > 7	True
<	Less than	3 < 8	True
>=	Greater than or equal to	5 >= 5	True
<=	Less than or equal to	4 <= 3	False

```
score = 75

print(score == 50)      # False
print(score > 50)       # True
print(score <= 70)      # False

# String comparisons work too
country = "Nigeria"
print(country == "Nigeria")   # True
print(country == "nigeria")   # False — case-sensitive!
```

== vs. =

A very common beginner mistake is using a single equals sign (=) instead of double (==) when comparing values. A single = is assignment — it stores a value. Double == is comparison — it checks whether two values are equal. Writing if score = 75 instead of if score == 75 is a SyntaxError. Always use == for comparison inside a conditional.

4. The if Statement

The simplest conditional structure is the if statement. It evaluates a condition and, if it's True, executes the indented block beneath it. If False, the block is skipped entirely:

```
if condition:
    # code to run if condition is True
    # (must be indented — 4 spaces)
```

The colon at the end of the if line is mandatory, and the block beneath it must be indented — Python uses indentation to define which code belongs to the if block.

```
score = 78

if score >= 50:
    print("Congratulations! You passed the exam.")
    print(f"Your score: {score}/100")

print("Thank you for taking the AiCE assessment.")
```

Because score (78) is greater than or equal to 50, the condition is True and both indented lines execute; the final print runs regardless, since it has no indentation and isn't part of the if block. Changing score to 35 would skip the entire indented block and go straight to the unindented line at the bottom.

Indentation Is Not Optional in Python

In most programming languages, indentation is a style choice. In Python, it's part of the syntax. If a code block isn't properly indented beneath an if statement, Python raises an IndentationError or executes code in the wrong order. Always use 4 spaces consistently, and never mix tabs and spaces.

Key Takeaways

A boolean is either True or False — always capitalized — and comparison operators like ==, !=, >, and < are how you build the conditions that produce those booleans. The if statement runs its indented block only when its condition evaluates to True, and Python's strict indentation rules mean 4 consistent spaces matter as much as the logic itself. The next lesson adds else and elif, giving your programs the ability to choose between several different paths, and introduces the logical operators that combine multiple conditions into one.