Nested Conditions, String Comparisons & Ternary Expressions

AiCE Software Engineering Foundations — Python Programming

This closing lesson on conditionals covers three remaining techniques — nesting one condition inside another, comparing strings directly in conditions, and a compact one-line shorthand for simple if-else logic — then brings everything from this unit together in one complete decision-making program.

1. Nested Conditional Statements

A nested conditional is an if statement placed inside another if statement — it lets you check a secondary condition only after the first has already been confirmed True:

```
age    = int(input("Age: "))
income = float(input("Monthly income (NGN): "))
score  = int(input("Credit score (300-850): "))

if age >= 18:
    if income >= 150000:
        if score >= 650:
            print("Loan APPROVED – All criteria met.")
        else:
            print("Loan DECLINED – Credit score below 650.")
    else:
        print("Loan DECLINED – Monthly income below 150,000
NGN.")
else:
    print("Loan DECLINED – Applicant must be at least 18 years
old.")
```

The program checks three conditions in sequence, and each inner check only runs if the outer check has already passed — exactly how real loan eligibility systems work: age first, then income, then credit score.

Nesting Depth

While nesting is powerful, avoid going more than three levels deep — deep nesting makes code hard to read and maintain. Combining conditions with and, covered in the previous lesson, can often flatten a deeply nested structure into something cleaner.

2. Comparing Strings in Conditions

Conditional statements work just as well with strings as with numbers, using == and != for comparison:

```
country = input("Enter your country: ").strip().lower()

if country == "nigeria":
    print("Welcome! The Nigerian AiCE hub is Lagos.")
elif country == "ghana":
    print("Welcome! The Ghanaian AiCE hub is Accra.")
else:
    print(f"Welcome from {country.title()}! Online access
available.")
```

Notice that the input is converted to lowercase before comparison — this means a user can type 'NIGERIA', 'nigeria', or 'Nigeria' and the program handles all three identically, a pattern first introduced back in the string methods lesson.

3. Conditional Expressions — The Ternary Operator

Python supports a compact one-line version of if-else, called a conditional expression or ternary operator, useful for assigning one of two values based on a condition:

```
# Standard if-else
if score >= 50:
    result = "Pass"
else:
    result = "Fail"
```

```
# Equivalent conditional expression (one line)
result = "Pass" if score >= 50 else "Fail"
```

The syntax reads naturally: assign "Pass" if score >= 50, otherwise assign "Fail". Both versions produce identical results:

```
age = int(input("Age: "))
paid = input("Has fee been paid? (yes/no): ").strip().lower()

status     = "Adult" if age >= 18 else "Minor"
fee_status = "Cleared" if paid == "yes" else "Pending"

print(f"Age Status : {status}")
print(f"Fee Status : {fee_status}")
```

> **When to Use Which Form**
> *Use the one-line conditional expression when the logic is genuinely simple and the shorter form reads more clearly. Use the full if-else block when the logic is more complex, or when each branch needs to run more than a single statement.*

4. Bringing It All Together — The AiCE Smart Admission Portal

A complete program combining if, elif, else, logical operators, string comparisons, and user input into one full admission decision system:

```
name = input("Full name: ").strip().title()
age  = int(input("Age: "))
gpa  = float(input("Previous GPA (0.0-4.0): "))
has_laptop = input("Do you own a laptop? (yes/no):
").strip().lower()

age_eligible = age >= 16
gpa_eligible = gpa >= 2.0
has_device   = has_laptop == "yes"
```

```
if not age_eligible:
    decision = "DECLINED"
    reason   = "Applicant must be at least 16 years old."
elif not gpa_eligible:
    decision = "DECLINED"
    reason   = "Minimum GPA of 2.0 required."
elif gpa >= 3.5 and age_eligible:
    decision = "ADMITTED — SCHOLARSHIP TRACK"
    reason   = "Exceptional academic record."
elif gpa >= 2.5 and has_device:
    decision = "ADMITTED — STANDARD TRACK"
    reason   = "Meets all standard entry requirements."
else:
    decision = "WAITLISTED"
    reason   = "Further review required."

print(f"DECISION : {decision}")
print(f"REASON   : {reason}")
```

This program demonstrates the full power of conditional logic — nesting, elif chains, logical operators, and string comparisons all working together. Every real admission system, loan portal, and eligibility checker you'll ever build is a more sophisticated version of exactly this structure.

Key Takeaways

Nested conditionals check a secondary condition only after an outer one has passed, but should rarely go deeper than three levels. Strings compare with == and != exactly like numbers do, and converting input to lowercase first makes string comparisons forgiving of how a user actually types something. The ternary operator — value_if_true if condition else value_if_false — compresses simple if-else logic into a single line. With conditional logic fully covered, the next lesson introduces loops — the mechanism that lets Python repeat an action automatically, instead of writing the same code over and over.