Strings & Joining Text Together

AiCE Software Engineering Foundations — Python Programming

Text is everywhere in programming — names, messages, addresses, descriptions, responses. Strings are how Python handles all of it. This lesson explores what strings really are and how to join them together into meaningful output.

1. What Is a String?

A string is a sequence of characters enclosed in quotation marks. Characters include letters, digits, spaces, punctuation, and symbols — anything you can type on a keyboard can be part of a string.

The word 'sequence' matters here. A string is not just a blob of text — it's an ordered series of individual characters, each with a position. The string "Lagos" is made up of five characters, L-a-g-o-s, in that exact order. This ordered nature is what makes strings powerful to work with.

Python strings can be written three ways:

```
# Double quotes
city = "Abuja"

# Single quotes
country = 'Ghana'

# Triple double quotes (for multi-line strings)
description = """AiCE is a platform dedicated to
restoring Africa's global leadership
through technology and education."""
```

All three create valid strings. Double quotes are most common in professional Python code; single quotes are equally valid and often used when a string itself contains a double quote; triple quotes handle multi-line text, covered later in this lesson.

Strings Are Immutable

Once a string is created, it cannot be changed in place. If you want a modified version, Python creates a brand new string with the changes applied. This is called immutability — a fundamental characteristic of Python strings you'll encounter again later.

2. Creating Strings — Practical Examples

```
# Strings using double quotes
name = "Ngozi Adeyemi"
city = "Port Harcourt"

# Strings using single quotes
country = 'Nigeria'

print(name)
print(city)
print(country)
```

Each variable holds a string value, and print() displays it exactly as stored — no quotation marks appear in the output. Python strips the quotes when displaying a string because the quotes are syntax, not content.

3. String Concatenation — Joining Strings with +

Concatenation is the process of joining two or more strings into a single string, using the + operator:

```
first_name = "Amara"
last_name = "Diallo"

full_name = first_name + last_name
print(full_name)
```

This produces AmaraDiallo — with no space, because concatenation joins strings exactly as they are. To add a space, include it explicitly:

```
full_name = first_name + " " + last_name
print(full_name)
```

Now the output is Amara Diallo. The " " in the middle is a string containing a single space character.

Concatenation with Descriptive Labels

```
name = "Kofi Mensah"
city = "Accra"

print("Name: " + name)
print("City: " + city)
```

This produces Name: Kofi Mensah and City: Accra. Each print statement joins a label string with a variable — this label + value pattern is fundamental to readable program output.

The TypeError Trap — Concatenating Strings and Numbers

Concatenation only works between strings. Trying to concatenate a string with an integer or float raises a TypeError:

```
name = "Adesina"
age = 26

# This will cause a TypeError
print("Name: " + name + ", Age: " + age)
```

This produces:

```
TypeError: can only concatenate str (not "int") to str
```

The fix is to convert the integer to a string first, using the str() function:

```
# Correct: convert age to string first
print("Name: " + name + ", Age: " + str(age))
```

This produces Name: Adesina, Age: 26.

> **str(), int(), float() — The Conversion Trio**
> *These three functions convert data between types. str() converts to string, int() converts to integer, and float() converts to decimal. You'll use all three regularly, especially once user input enters the picture later in this program.*

Key Takeaways

A string is an ordered, immutable sequence of characters, writable with single, double, or triple quotes. Concatenation with + joins strings exactly as written — with no automatic spacing — and only works between strings, which is why str() exists to bridge numbers into text form. The next lesson introduces f-strings, the modern, professional way Python developers embed variables directly inside text, along with multi-line strings and a few more essential string techniques.