# Measuring & Transforming Strings

*AiCE Software Engineering Foundations — Python Programming*

Python comes with a rich set of built-in tools for working with strings — measuring, transforming, searching, cleaning, and splitting text with minimal code. This lesson opens that toolkit, starting with measuring a string's length and changing its case.

## 1. What Are String Methods?

A method is a function that belongs to a specific type of data. String methods are functions built directly into every string object, called using dot notation:

```
variable_name.method_name()
```

You write the variable name, a dot, the method name, then parentheses — the parentheses are always required, even when you pass nothing inside them.

> **A Method Is Something a String Can Do to Itself**
> *A string can tell you its length. A string can convert itself to uppercase. A string can replace part of its content. All of these are actions the string performs when you call the appropriate method on it.*

## 2. len() — Measuring the Length of a String

len() returns the number of characters in a string — every letter, space, digit, punctuation mark, and symbol counts.

> **len() Is Technically Not a String Method**
> *len() is a built-in Python function, not a string method — you pass the string to it, rather than calling it on the string with dot notation: len(name), not name.len(). It's covered here alongside the methods because it's so fundamental to working with strings.*

```
# Measuring string lengths
name = "Adesina Olagunju"
city = "Lagos"

print(len(name))
print(len(city))

# Using f-strings for readable output
print(f"The name '{name}' has {len(name)} characters.")
```

This produces:

```
16
5
The name 'Adesina Olagunju' has 16 characters.
```

Notice that spaces are counted as characters — "Adesina Olagunju" has 16 characters: 7 for Adesina, 1 for the space, and 8 for Olagunju.

> **len() Works on More Than Strings**
> *len() also works on lists, tuples, and other data structures — anything with a countable number of elements. On a string, it counts characters; on a list, it counts items. You'll see this again once lists are introduced.*

## 3. .upper() and .lower() — Changing Case

.upper() converts every letter in a string to uppercase. .lower() converts every letter to lowercase. Non-letter characters — spaces, numbers, punctuation — are unaffected:

```
name = "Adesina Olagunju"

print(name.upper())
print(name.lower())
```

This produces:

```
ADESINA OLAGUNJU
adesina olagunju
```

> **These Methods Don't Change the Original**
> *name.upper() returns a new string; it doesn't modify name itself. After calling it, name still holds "Adesina Olagunju". To keep the transformed version, assign it to a variable: name_upper = name.upper().*

One of the most common uses of .lower() is making comparisons case-insensitive — handling 'Nigeria', 'nigeria', and 'NIGERIA' identically:

```
user_country = "NIGERIA"

if user_country.lower() == "nigeria":
    print("Welcome, Nigerian developer!")
```

This pattern will feel completely natural once conditional statements are covered later in this program — understand the concept now, and it'll click quickly then.

## 4. .capitalize() and .title() — Formatting Names

Two more case methods, especially useful for names and titles:

| Method | What It Does |
|---|---|
| .capitalize() | Makes the very first character uppercase, the rest lowercase |
| .title() | Makes the first character of every word uppercase, the rest lowercase |

```
name = "amara diallo"

print(name.capitalize())
print(name.title())
```

This produces:

```
Amara diallo
Amara Diallo
```

.capitalize() only fixes the first letter of the entire string, leaving the rest lowercase. .title() fixes the first letter of every word — for formatting names and headings, .title() is almost always the more useful choice.

> **A Known Quirk of .title()**
> *"africa's technology leaders".title() produces "Africa'S Technology Leaders" — .title() treats the character right after an apostrophe as the start of a new 'word' and capitalizes it. This is a known edge case; .title() is perfectly adequate for the level of work in this course, but it isn't flawless.*

## Key Takeaways

String methods are called with dot notation — variable.method() — and none of the methods covered here modify the original string; they all return a new one. len() counts characters (including spaces), .upper()/.lower() convert case entirely, and .capitalize()/.title() handle names and headings with slightly different rules. The next lesson continues with methods for substituting and cleaning text — .replace(), .strip(), and .split() — tools you'll use constantly once real, messy data enters your programs.