Strings: Text, Quotes, and Length

Python for Absolute Beginners · Lesson 6

What you'll learn

- What a string is
- Single quotes, double quotes, and when each is handy
- How to join strings together
- How to repeat a string
- How to count a string's length and read one character

What a string is

- A string is text: letters, digits, spaces, punctuation — wrapped in quotes
- The quotes tell Python “this is text, not a name and not a number”
- "42" is a string, not a number — that is why "5" + "3" gave "53" in Lesson 5
- If you need arithmetic, the value must be a number without quotes



A string in code

```
                    message = "Hello, world!"
print(message)

Output:
Hello, world!
```

The quotes mark the value as text.

Single quotes or double quotes?

- Both work, and they mean exactly the same thing
- Pick the kind that does not clash with the text itself
- Single quotes outside keep double quotes inside safe
- Double quotes outside keep an apostrophe, like in It’s, safe

Quotes in code

```
                    print('Sam said "Hi!"')
print("It's a sunny day.")

Output:
Sam said "Hi!"
It's a sunny day.
```

Choose whichever keeps your text intact; there is no other difference.



Joining and repeating strings

- + joins strings end to end — this is called concatenation
- Python adds nothing on its own: spaces, commas, and punctuation are your job
- * repeats a string — the same * as multiplication, but with text it means “repeat”

Joining strings in code

```
                        first = "Hello"
second = "Sam"
print(first + " " + second)

Output:
Hello Sam
```

Without the " " in the middle, you would get HelloSam.
A string and a number do not join: print("You are " + 9) gives TypeError: can only concatenate str (not "int") to str. Fix it with str(9), or use a comma: print("You are", 9).

Repeating a string in code

```
print("Ha! " * 3)
```

```
Output:
Ha! Ha! Ha!
```

With text, * means “repeat.”

Length, and reading one character

- len() counts how many characters a string has, including spaces
- Spaces count — the detail beginners most often miss
- You can read a single character by its position — and counting starts at 0, not 1
- Asking for a position past the end gives an error

Length in code

```
word = "hello world"
print(len(word))

Output:
11
```

h-e-l-l-o (5), one space (6), w-o-r-l-d (11).

Reading one character in code

```
word = "Sam"
print(word[0])
print(word[1])
print(word[2])
print(word[len(word) - 1])

Output:
S
a
m
m
```

word[0] is S, word[1] is a, word[2] is m. The last character is at position len(word) - 1, which prints m. Asking for word[3] gives IndexError: string index out of range.

Putting it together

```
name = "Mia"
print("Hello, " + name + "! You are learning Python.")

Output:
Hello, Mia! You are learning Python.
```

Strings, variables, and joining combine into the kind of line real programs use.

Hands-on exercise

- Store your first name in a variable and print a greeting built with +
- Print the length of your name with len() — count by hand and check they match
- Print the first and last characters of your name using positions (counting starts at 0 — and the last character is at position len(name) - 1)
- Try "Boo! " * 5 in the shell and explain the result in one sentence

Key takeaways

- A string is text in quotes; "42" is text, not a number
- Single and double quotes mean the same thing — pick the kind that does not clash with your text
- + joins strings; * repeats them; Python adds no spaces for you
- len() counts characters, including spaces
- Positions start at 0: word[0] is the first character

Take the quiz

Take the Lesson 6 quiz — 5 quick questions to check what you learned.

Next lesson

Talking to Your Program: input()