EDUFY

print(), variables, f-strings and input(): your first program that talks back

Maximilian Bahrami · 11 November 2026

Show values with print(), keep and change them in variables, build sentences with f-strings, and ask the user questions with input(). Plus the mistake every beginner makes with input(), and a small receipt program to write yourself.

In lesson 1 you met Python's data types. Now you'll put them to work: show values with print(), keep them in **variables** and change them, build sentences with **f-strings**, and ask the user a question with input(). By the end you'll write a small program that talks to its user. Every example runs right here: press **Run**, then press **Edit** and change it.

Your first line: print()

print() shows whatever you put in its brackets. Text goes in quotes:

```
print("Hello, world!")
print("I'm learning Python.")
print(7 + 3)
```

Three things to notice:

- Text in quotes is a **string**. You can use "double" or 'single' quotes, as long as both ends match.
- Numbers have **no quotes**. 7 + 3 is maths, so Python prints 10.
- Each print() starts a new line.

You can give print() several values, separated by commas. It puts a space between them:

```
print("Apples:", 4, "Pears:", 2)
```

Variables: a name for a value

A **variable** keeps a value under a name, so you can use it later. You make one with =:

```
name = "Ada"
age = 36

print(name)
```

```
print(age)
print(name, "is", age, "years old")
```

Unlike JavaScript, there's no let, const or var. You just write the name, =, and the value. You can give a variable a new value whenever you like:

```
score = 0
print(score)

score = score + 10
print(score)

score += 5 # the short way to write: score = score + 5
print(score)
```

Everything after # is a **comment**: a note for people, which Python skips.

Name variables in snake_case

> Python names are written in lower case, with underscores between the words: first_name, total_price, is_logged_in. That's called **snake_case**. Names can't start with a number or contain spaces, and upper and lower case are different: Name and name are two different variables.

A quick reminder: strings and numbers

From lesson 1: "7" in quotes is a **str**, 7 without quotes is an **int**. That matters as soon as you use +:

```
print(7 + 7) # 14: adding numbers
print("7" + "7") # 77: joining two strings
print("ha" * 3) # hahaha: a string repeated
```

Keep this in mind: it's the reason for the most common input() mistake further down.

f-strings: values inside a sentence

Joining text with + gets messy fast. Python has a better way, the **f-string**: put f before the quotes, and write variables in {curly brackets}:

```
name = "Ada"
age = 36

print(f"My name is {name} and I am {age} years old.")
print(f"Next year I'll be {age + 1}.")
```

Anything inside {} is real Python, so you can even do sums there, like {age + 1}. This is the way most Python code builds text today.

input(): ask the user something

input() stops, waits for the user to type an answer, and gives it back to you. The text in the brackets is the question.

Here the answers come from the **Input** box under the code: one line for each input(). Type your name there, then press **Run**:

```
name = input("What's your name? ")
print(f"Nice to meet you, {name}!")
```

input() always gives you a string

This is the most common surprise for beginners. Even if the user types 12, input() gives you the **text** "12", not the number. To do maths with it, turn it into a number with int() (whole numbers) or float() (decimals).

Put two lines in the Input box, for example Ada and 36, and run it:

```
name = input("Name: ")
age = int(input("Age: "))

print(f"{name}, in ten years you'll be {age + 10}.")
```

int(input("Age: ")) reads from the inside out: first input() gets the text, then int() turns it into a number.

Common mistakes

Adding a string and a number

Wrong

```
age = input("Age: ")
print(age + 1)
```

Right

```
age = int(input("Age: "))
print(age + 1)
```

input() gives a string, and Python won't add a number to a string. It stops with TypeError: can only concatenate str (not "int") to str. Turn the answer into a number with int() first.

Forgetting the f

Wrong

```
name = "Ada"
print("Hello, {name}!")
```

Right

```
name = "Ada"
print(f"Hello, {name}!")
```

Without the f, the curly brackets are just text, and Python prints Hello, {name}! exactly as written.

A space at the start of a line

Wrong

```
name = "Ada"
print(name)
```

Right

```
name = "Ada"
print(name)
```

In Python, spaces at the start of a line mean something: they show which lines belong together (you'll use that in the lesson on if). A line that starts with a space for no reason gives IndentationError: unexpected indent. For now, start every line at the left edge.

Exercise

A shopping receipt

Write a program that asks for:

1. the name of a product,
2. its price (it can have decimals, like 2.50),
3. how many the user buys.

Then print one line like this: 3 x Coffee = 7.5 euros

Hint: you need float() for the price and int() for the amount. Put three lines in the Input box, for example Coffee, 2.50 and 3.

Answer

```
product = input("Product: ")
price = float(input("Price: "))
amount = int(input("How many: "))

total = price * amount
print(f"{amount} x {product} = {total} euros")
```

The price uses float() because it can have decimals, the amount uses int() because you can't buy half a coffee. Storing the sum in total keeps the f-string short and easy to read.

Want exactly two decimals, like 7.50? Write {total:.2f} inside the f-string.

Quick check

What does this print?

```
a = "5"
b = "5"
print(a + b)
```

- 10
- ✓ 55
- An error

a and b are strings, because of the quotes. + joins two strings, so you get "55". To get 10, write the numbers without quotes, or turn them into numbers with int().

The user types 20. What is the type of age here: age = input("Age: ")?

- ✓ str
- int
- float

input() always gives a string, whatever the user types. Use int(input(...)) to get a number.

Key points

- print() shows values; text goes in quotes, numbers don't.
- A variable is a name for a value: name = "Ada". No let or const in Python.
- Name variables in snake_case: first_name, total_price.
- Strings, whole numbers, decimals and True/False are different types: str, int, float, bool.
- f-strings put values in text: f"Hi {name}, next year you're {age + 1}".

- input() always gives a string; use int() or float() to do maths with it.