Variables: Storing Things

Python for Absolute Beginners · Lesson 4

What you'll learn

- What a variable is, and why programs need one
- How to store text and numbers in variables
- How to change a variable's value
- The rules for naming variables
- The difference between a name and a string

A variable is a labeled box

- A variable is a name that holds a value — think of it as a labeled box
- You put something inside, and whenever you use the name later, Python looks inside the box and uses what it finds there
- You create a variable by assigning a value to a name with the = sign
- Read = here as “store”, not as “equals” in the math sense

Storing text and a number

```
name = "Mia"
age = 9
print(name)
print(age)

Output:
Mia
9
```

name holds the text "Mia", and age holds the number 9. print(name) prints the value stored under that name — not the word "name". The same name works as many times as you like — print("Hello,", name) prints Hello, Mia again.

The quotes rule, again

- name = "Mia" stores text, because of the quotes
- age = 9 stores a number, because there are no quotes
- If you wrote age = "9", Python would store the text "9", not the number
- As you will see in Lesson 5, text and numbers behave very differently in arithmetic

Changing a value

- A variable's value can change while the program runs
- Each new assignment replaces what was there before



Updating a score

```
score = 0
print(score)
score = 10
print(score)
score = score + 5
print(score)

Output:
0
10
15
```

Read the last line as: take the current value of score (10), add 5, and store the result (15) back in score.

Naming rules

- Use letters, numbers, and underscores only: my_name, score2
- A name cannot start with a number: 2nd_score is not allowed, but score2 is fine
- No spaces: use an underscore instead, my_name not my name
- Do not use a word Python already uses, like if or for — Python stops with a SyntaxError right away
- Do not store a value in a built-in name like print or list — Python lets you do it, but print stops working afterward
- Names are case-sensitive: Score and score are two different variables, and Python will not warn you — pick one style and stay consistent. Python programmers use lowercase words joined by underscores, like my_score

Good names, bad names

- Good names describe what is inside: name, age, score, price
- Bad names describe nothing: x, a, thing
- When you come back to your own code a week later, good names are what make it readable

Name versus string

```
                              name = "Mia"
print(name)
print("name")

Output:
Mia
name
```

print(name) prints the value stored under the name. print("name") prints the word itself, because quotes mean “this is text.”

Hands-on exercise

- Create two variables, one holding your name (text) and one holding your age (a number), and print both
- Create a variable points set to 0, print it, then add 20 to it the way the lesson shows (points = points + 20), and print it again
- Try printing a variable you never created, read the error message, and explain in one sentence what it means

Key takeaways

- A variable is a labeled box: a name holding a value
- = means “store”, and a new assignment replaces the old value
- score = score + 5 updates a value from its old self
- Names use letters, numbers, and underscores; no spaces, no leading number, and case matters
- Quoted text is text; an unquoted name means “look up the stored value”

Take the quiz

Take the Lesson 4 quiz — 5 quick questions to check what you learned.

Next lesson

Numbers and Arithmetic