# Variables and Data Types

A variable is a named box that holds a value — learn the main data types, naming rules, and how to check a value's type.

## Introduction

Imagine you want to remember a number, a name, or a price while your program runs. You cannot keep shouting the value every time you need it. You store it in a **variable** — a name that holds a value. Later, you use the name, and Python brings back the value.

For example, `name = "Ali"` stores the word Ali under the name `name`. After that, whenever you write `name`, Python knows you mean Ali. If you change the value, the name stays the same but the value changes. This is why variables are so useful: one name, many uses.

Every value in Python has a **type**. A number is one type, a word is another type, and a true/false value is another type again. The type decides what you can do with the value — you can add numbers, but adding words works differently. In this lesson you will meet the five most common types.

*(Roman Urdu: Variable aik naam hai jo kisi value ko yaad rakhta hai — jaisay aik dabba jis par naam likha ho, aur andar value rakhi ho.)*

The hard part of this lesson is not typing `=`. It is choosing a name you will still understand tomorrow, and remembering that `"20"` and `20` are different things. One is text you can show on the screen; the other is a number you can add. Most early errors come from mixing those two up.

## Key points

- **A variable is created with `=`.** The name goes on the left, the value on the right: `age = 20`.
- **`int` means a whole number.** Examples: `20`, `0`, `-5`. No decimal point.
- **`float` means a number with a decimal point.** Examples: `3.5`, `99.99`, `2.0`.
- **`str` means a string, or text.** Text always goes inside quotes: `"Ali"`, `'Hello'`.
- **`bool` means true or false.** Only two values exist: `True` and `False` (capital first letter).
- **`None` means "no value yet".** It is useful as a starting value before the real value arrives.
- **Naming rules matter.** Names can contain letters, digits and underscores, but they cannot start with a digit, and they cannot contain spaces. Use clear names like `student_age`, not `x` or `a1b2`.
- **`type()` tells you the type.** `type(20)` gives `int`, and `type("Ali")` gives `str`.
- **Python is case-sensitive.** `Age` and `age` are two different names.
- **A variable holds one value at a time.** Giving it a new value replaces the old one — the old value is gone, not kept somewhere.

- **Converting changes the type, not just the look.** `int("250")` turns text into a number you can add; `"250"` on its own cannot be added to a number.

## Examples

### Example 1 — Five types in one program

```
age = 20
price = 99.50
name = "Ali"
is_student = True
middle_name = None

print(age)
print(price)
print(name)
print(is_student)
print(middle_name)
```

What you will get:

```
20
99.5
Ali
True
None
```

Notice that `99.50` prints as `99.5` — Python does not keep a useless zero at the end of a decimal number.

### Example 2 — Checking types with type()

```
print(type(20))
print(type(3.5))
print(type("Ali"))
print(type(True))
print(type(None))
```

What you will get:

```
<class 'int'>
<class 'float'>
<class 'str'>
<class 'bool'>
<class 'NoneType'>
```

The word `class` here just means "type". So `20` is of type `int`, and `"Ali"` is of type `str`.

### Example 3 — Changing a variable's value

```
score = 10
print(score)
score = 25
print(score)
score = score + 5
print(score)
```

What you will get:

```
10
25
30
```

The last line reads the old value of `score` (25), adds 5, and stores the new value (30) back into the same name.

### Example 4 — Swapping two values

To swap two variables you need a third, temporary name to hold one value while you move the other. Without it, the first value would be lost:

```
a = 5
b = 10
print(a, b)
temp = a
a = b
b = temp
print(a, b)
```

What you will get:

```
5 10
10 5
```

`print(a, b)` prints both values on one line, separated by a space. Trace this by hand: after `temp = a`, `temp` is 5; after `a = b`, `a` is 10; after `b = temp`, `b` is 5.

### Example 5 — Converting text to a number, and back again

Prices and ages often arrive as text — from a form, a file, or typed input. Convert before you do maths with them:

```
price_text = "250"
price = int(price_text)
total = price * 2
print(total)
print(type(total))
print("Total: " + str(total))
```

What you will get:

```
500
<class 'int'>
Total: 500
```

`int()` turns `"250"` into the number 250, so `250 * 2` is 500. Going the other way, `str(total)` turns 500 back into text so it can be joined with `"Total: "` using `+`.

### Example 6 — Real-world case: a student's fee record

A school office keeps one record per student: a name, a monthly fee, how many months, and whether it is paid. Four variables hold the whole record:

```
student_name = "Fatima"
monthly_fee = 2500
months = 3
paid = True
print(student_name, "fee for", months, "months:", monthly_fee * months)
print("Paid:", paid)
```

What you will get:

```
Fatima fee for 3 months: 7500
Paid: True
```

Real programs are mostly this: a few well-named variables, one small calculation, and a clear printed result.

## Quick recap

- A variable stores a value under a name, using `=`.
- Main types: `int`, `float`, `str`, `bool`, and `None`.
- Text goes in quotes; numbers do not.
- `type()` shows the type of any value.
- Names must start with a letter or underscore, with no spaces.
- Reassigning a variable replaces its old value.
- `int()` and `str()` convert values between text and numbers.

## Common mistakes

1. **Using `=` when you mean "is it equal?"** — `=` stores a value; comparing needs `==`, which you will learn in PY.6. Fix: for storing, use one `=`; for comparing, use two.
2. **Writing True and False in small letters** — `true` and `false` are not Python values. Fix: always write `True` and `False` with a capital first letter.
3. **Forgetting quotes around text** — `name = Ali` gives an error because Python thinks Ali is another variable name. Fix: write `name = "Ali"`.
4. **Starting a name with a digit** — `1st_name = "Ali"` is not allowed. Fix: write `first_name = "Ali"` instead.
5. **Using a vague name and forgetting what it holds** — after ten lines, `x` means nothing. Fix: use clear names like `total_price` or `student_name`.

1. **Using a Python keyword as a name** — `class = 10` fails because `class` already means something in Python. Fix: use `class_name` or `student_class` instead.
2. **Using a variable before creating it** — `print(age)` before any line says `age = ...` gives a NameError. Fix: always assign a variable before you use it.

## Practice

1. Create variables for your name (str), your age (int), your height in metres (float), and whether you are a student (bool). Print all four, then print the type of each one.
2. Create a variable `marks = 80`, print it, then change it to `90` and print it again.
3. Try `print(type(100))` and `print(type("100"))`. Write down why the two results are different, even though both look like 100.
4. Fix this broken code, then run it: `1name = Ali` and `is_student = true`. Write the correct version in your notebook.
5. Store the text `"50"` in a variable, convert it with `int()`, add 10, and print the result. Then try adding 10 to the text without converting, and read the error.

6. A book costs 350 rupees. Store the price and a quantity of 3 in variables, work out the total, and print it with a label.
7. Write a program with three variables — your name, your age next year (`age + 1`), and whether you like Python (`True` / `False`) — and print all three with their types.

## Practice quiz

**1. What is a variable?**

- A maths problem
- A named box that stores a value you can use later
- A Python command
- A type of error

**Answer:** A named box that stores a value you can use later

**2. What type is the value 3.5 in Python?**

- int
- str
- float
- bool

**Answer:** float

**3. Which variable name is NOT allowed in Python?**

- student_name
- marks2
- 2marks
- total_marks

**Answer:** 2marks

---

**Watch the video lesson:** https://youtu.be/uwTO6Jt3MG4

**Read online:** https://scopetreasure.com/notes/py-variables-data-types · More free tools and notes at https://scopetreasure.com