

Variables, types & operators

Python Reference

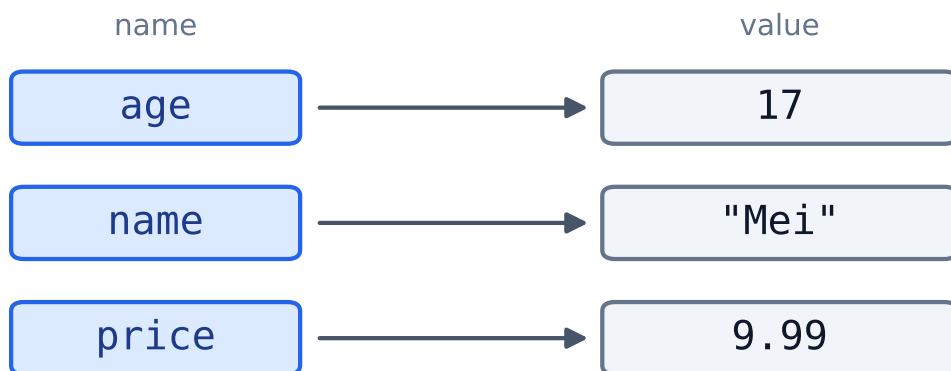
Variables & assignment

A variable is a name for a value . You make one with =, which is called assignment . The name goes on the left; the value goes on the right.

```
age = 17
name = "Mei"
price = 9.99
print(age, name, price)
```

Now `age` holds 17. Use the name anywhere you need the value, and change it later:

```
age = 17
age = age + 1    # age is now 18
print(age)
```



variable: name → value in memory

Each variable name points to a value in memory

- The = sign does not mean "equal". It means "store the right side under the left name".
- To test if two values are equal, use == (see below).

Numbers: int & float

Python has two main number types. An integer (`int`) is a whole number like 17. A float (`float`) has a decimal point like 9.99.

These operators work on numbers:

Operator	Meaning	Example	Result
+	add	3 + 2	5
-	subtract	3 - 2	1
*	multiply	3 * 2	6
/	divide (always float)	7 / 2	3.5
//	integer divide	7 // 2	3
%	remainder (modulo)	7 % 2	1
**	power	2 ** 3	8

- / always gives a float, so 4 / 2 is 2.0.
- // and % go together: 17 // 5 is 3, and 17 % 5 is 2.

Expressions & type conversion

An expression is anything that has a value, like 3 + 4 * 2. Python uses normal maths order (* and / before + and -); add brackets to make the order clear.

`input()` gives a string, so convert it before doing maths. Changing a value from one type to another is type conversion :

```
age = int("17")           # text "17" -> number 17
price = float("9.99")     # text -> 9.99
label = str(17)           # number -> text "17"
print(age, price, label)
```

- `int("abc")` fails, so only convert text that looks like a number.
- Mixing types fails too: `"age: " + 17` is an error; write `"age: " + str(17)`.

Booleans & comparison

A Boolean is one of just two values: `True` or `False`. A comparison gives back a Boolean.

Operator	Meaning
<code>==</code>	equal to
<code>!=</code>	not equal to
<code>< ></code>	less than / greater than
<code><= >=</code>	less / greater than or equal to

```
print(7 > 2)          # True
print(3 == 3.0)       # True
age = 20
print(age >= 18)      # True
```

Join comparisons with `and`, `or`, `not`:

```
age = 20
print(age >= 13 and age <= 19)  # True only for a teenager
```

Common mistakes

- `/` always gives a float, even `4 / 2` is `2.0`. Use `//` when you want a whole number.
- Using `=` where you mean `==`: `=` assigns a value, `==` tests whether two values are equal.
- Joining a string and a number: `"age: " + 5` raises a `TypeError`. Convert first with `str(5)`.
- Floats are not exact, so `0.1 + 0.2` is not exactly `0.3` — avoid `==` on float results.