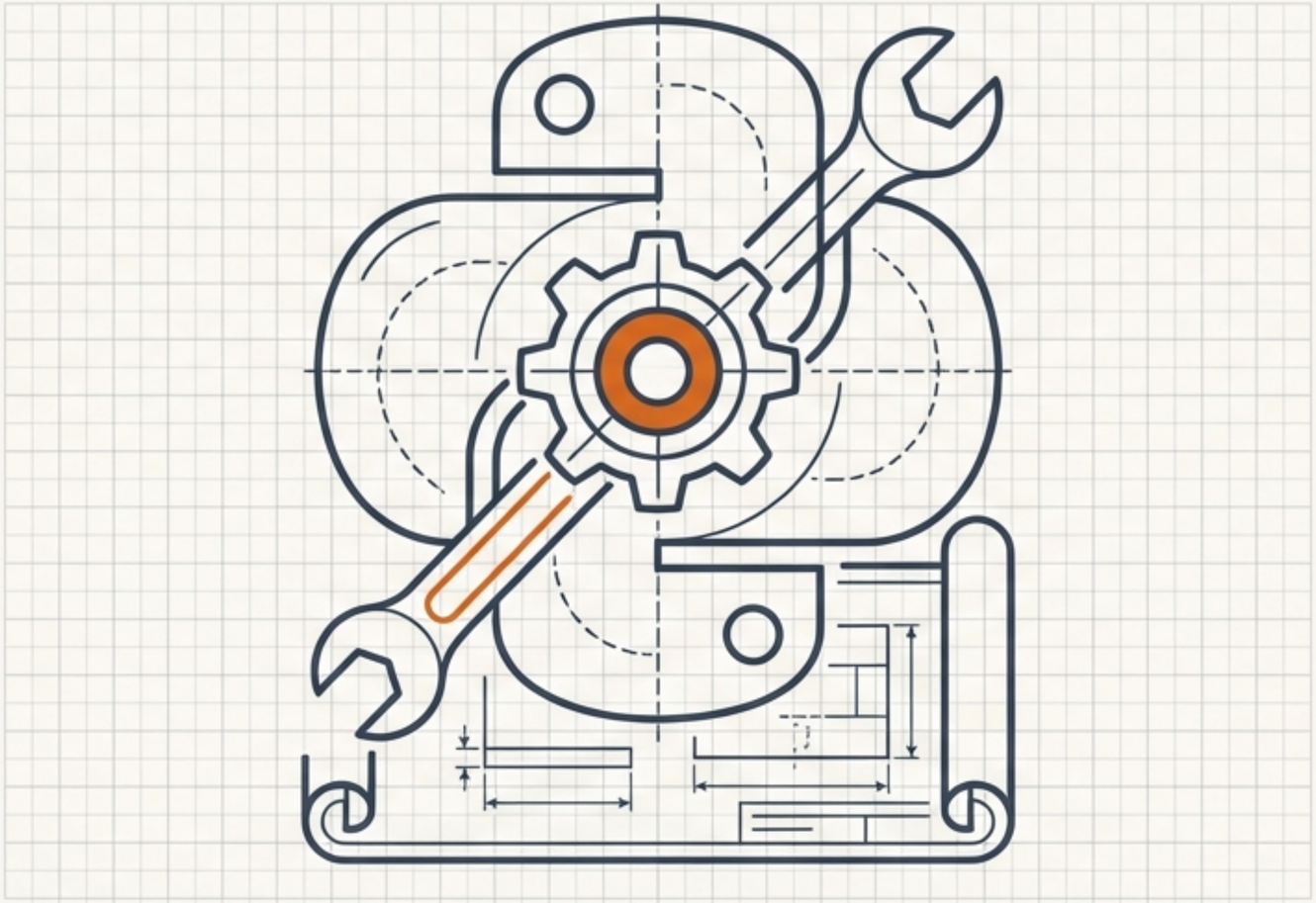
Build Your First Program in Python

A Beginner's Toolkit for Crafting Code



Here’s What We’re Building Today

```
# Get user information
print("=== User Registration ===")
name = input("Enter your name: ")
age = int(input("Enter your age: "))
height = float(input("Enter your height in feet: "))

# Process and display
is_adult = age >= 18
next_year_age = age + 1

# Display results
print("\n--- Your Information ---")
print(f"Name: {name}")
print(f"Age: {age} years old")
# ...
message = f"{name.upper()} is {status} and stands {height} feet tall."
print(f"\nSummary: {message}")
```

This might look complex now, but it’s built from a few simple parts. By the end of this guide, you'll understand every single line. Let’s open the toolkit.

The Foundation: Storage Bins and Raw Materials

Tool #1: The Storage Bins (Variables)

A variable is a named container for storing a value. Think of it as a labeled bin in your workshop.


```
name = "Alice"
age = 25
is_student = True
```

The Raw Materials (Data Types)

These are the fundamental materials you'll store and work with.

Integer (`int`): For whole numbers.
`count = 10`

Float (`float`): For numbers with decimals.
`price = 19.99`

String (`str`): For text.
`greeting = "Hello"`

Boolean (`bool`): For True/False logic.
`is_active = True`

In Python, you don't need to declare a variable's type beforehand. It's determined automatically when you assign a value.

Workshop Prep: Checking and Reshaping Your Materials

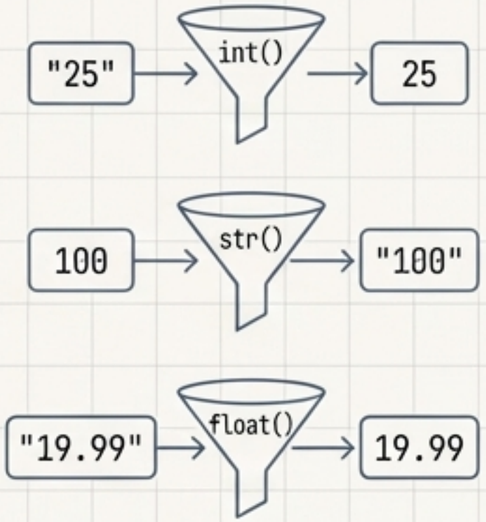
Identifying Materials with `type()`

Use the `type()` function to inspect a variable and see what kind of material you're working with.


```
JetBrains Mono
name = "Alice"
age = 25
print(type(name))  # Output: <class 'str'>
print(type(age))   # Output: <class 'int'>
```

Converting Between Types

Sometimes you need to convert one material into another, like turning a number stored as text into a real number for calculations.




```
# String to integer
age_str = "25"
age_int = int(age_str) # Result: 25

# Integer to string
count = 100
count_str = str(count) # Result: "100"

# String to float
price_str = "19.99"
price_float = float(price_str) # Result: 1
```

⚠ CRITICAL

The input() function (which we'll see later) always gives you a string. You'll almost always need to convert it!

The Power Tools: Arithmetic Operators

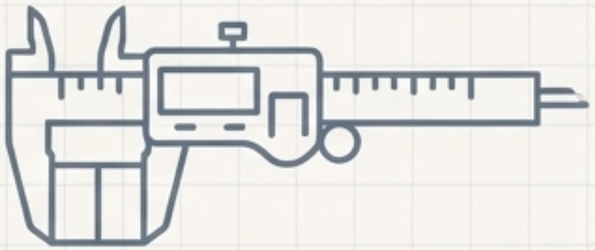
Your primary tools for performing calculations.

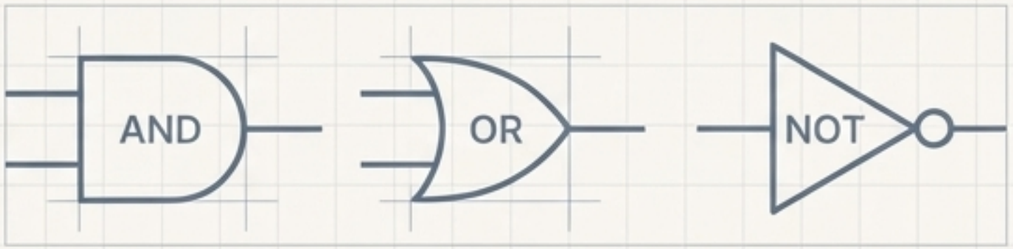
Operator	Description	Example
+	Addition	`5 + 3` → `8`
-	Subtraction	`10 - 4` → `6`
*	Multiplication	`7 * 3` → `21`
**	Exponentiation	`2 ** 3` → `8`
/	Division (always float)	`10 / 3` → `3.333...`
//	Floor Division (integer)	`10 // 3` → `3`
%	Modulus (remainder)	`10 % 3` → `1`

Code in Action

```
a = 10
b = 3
print(f"a / b = {a / b}")     # 3.333...
print(f"a // b = {a // b}")   # 3
print(f"a % b = {a % b}")     # 1
```

Precision Tools for Making Decisions





Comparison Operators (The Calipers)

These tools compare two values and return a Boolean (`True` or `False`). They are the foundation of decision-making in your code.

`==`	Equal to	`<`	Less than
`!=`	Not equal to	`>=`	Greater than or equal to
`>`	Greater than	`<=`	Less than or equal to

```
print(10 > 5) → True
```

Logical Operators (The Circuitry)

Use these to combine multiple `True/False` conditions.

- `and`: Returns `True` only if **both** statements are true
- `or`: Returns `True` if **at least one** statement is true
- `not`: Reverses the result (`True` becomes `False`)

```
age = 25
has_license = True
# Both must be true
can_drive = age >= 18 and has_license  # True
# Reverses the boolean
is_not_licensed = not has_license      # False
```

A Special Material: Working with Text (Strings)

Strings are sequences of characters—the primary way you'll handle text.

Creating Strings

You can use single, double, or triple quotes. Triple quotes are perfect for text that spans multiple lines.

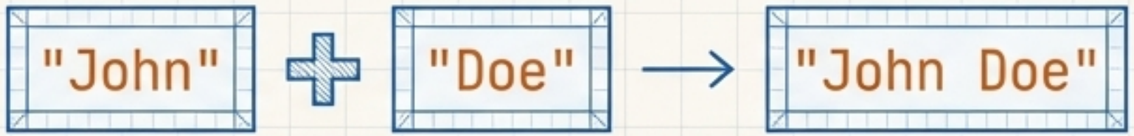
```
# Single quotes
name = 'Alice'
```

```
# Double quotes (useful for strings with apostrophes)
message = "It's Python"
```

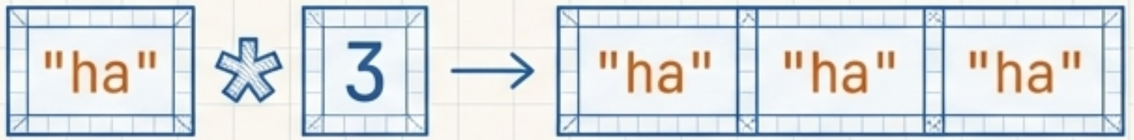
```
# Triple quotes for multi-line strings
description = """Python is a
               high-level programming
    language."""
```

Combining and Repeating Strings

Use `+` to join strings (concatenation) and `*` to repeat them.




```
first_name = "John"
last_name = "Doe"
full_name = first_name + " " + last_name # "John Doe"
```

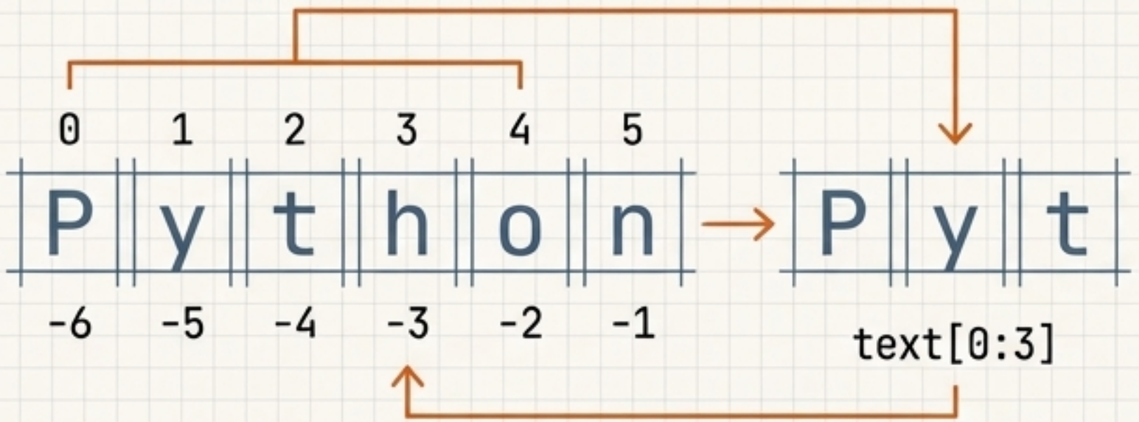



```
laugh = "ha" * 3 # "hahaha"
```

Advanced String Craftsmanship

Indexing and Slicing (Precision Cuts)

Access individual characters or create substrings.
Indexing starts at 0. Negative indices count from the end.




```
text = "Python"
first_char = text[0]   # "P"
last_char = text[-1]   # "n"
substring = text[0:3]  # "Pyt"
substring = text[2:]   # "thon"
```

Common Methods (Finishing Tools)

Strings come with powerful built-in functions called methods.

`.upper()` / `.lower()`: Change case.
`"Py".lower()` → `"py"`

`.strip()` Remove leading/trailing whitespace.
`" Hello ".strip()` → `"Hello"`

`.replace()`: Find and replace text.
`"Hello".replace("e", "a")` → `"Hallo"`

`.split()`: Break a string into a list.
`"a,b,c".split(",")` → `['a', 'b', 'c']`

`in`: Check for presence.
`"Py" in "Python"` → `True`

The Pro-Grade Label Maker: Formatting with F-Strings

F-strings (formatted string literals) are the modern, readable, and recommended way to embed expressions inside strings.

Placeholder: The variable or expression to insert.

Literal Text: The static part of your string.

Prefix: Tells Python this is an f-string.

```
name = "Alice"
age = 25
message = f"My name is {name} and I am {age} years old."
print(message)
# Output: My name is Alice and I am 25 years old.
```

Powerful Features

Expressions inside

You can do math or call functions directly inside the braces.

```
print(f"{name} will be {age + 1} next year.")
```

Number Formatting

Control decimal places easily.

```
height = 5.6
print(f"Height: {height:.1f} feet") # "Height: 5.6 feet"
```

You might see older `.format()` or % methods, but f-strings are the standard for new Python code.

The Workbench: Interacting with Your Program

Displaying Your Work with `print()`

The `print()` function is the primary way to display output to the console.


```
# Basic printing
print("Hello, World!")

# Multiple items (adds spaces automatically)
print("Alice", "is", 25) # Alice is 25

# Custom separator
print("A", "B", "C", sep="-") # A-B-C
```

Getting User Materials with `input()`

The `input()` function pauses the program and waits for the user to type something and press Enter.


```
name = input("Enter your name: ")
print(f"Hello, {name}!")
```

`input()` ALWAYS returns a string!

If you need a number, you must convert the result from `input()`.

```
age = int(input("Enter your age: "))
```

Grand Assembly: Putting It All Together

Get raw text (`str`) from the user.

```
# Get user information
print("=== User Registration ===")
name = input("Enter your name: ")
age = int(input("Enter your age: "))
height = float(input("Enter your height in feet: "))

# Process and display
is_adult = age >= 18
next_year_age = age + 1

# Display results
print("\n--- Your Information ---")
print(f"Name: {name}")
# ... more code ...
message = f"{name.upper()} is {status} and stands {height} feet tall."
print(f"\nSummary: {message}")
```

Get text, then convert it to a number (`int`).

Use a comparison tool to get a `True`/`False` value.

Perform calculations.

Use f-strings for clean, professional output.

Use built-in tools to modify materials on the fly.

Your Toolkit Reference Sheet

Data & Types

- **int**

```
age = 25
```

- `float: price = 19.99`
- `str: name = "Alice"`
- `bool:`
  `is_active = True`

**Conversion

```
int(), float(), str()
```

Operators

**Arithmetic

```
+, -, *, /, //, %, **
```

**Comparison

```
==, !=, >, <, >=, <=
```

**Logical

```
and, or, not
```

**Assignment

```
x = 10, x += 5
```

String Essentials

**Creation

```
'...', "...", """..."""
```

**Joining

```
+, *
```

**Slicing

```
text[0], text[-1], text[1:5]
```

**Formatting

```
f"Value: {my_var}"
```

**Key Methods

```
.upper(), .strip(),
.split(), .replace()
```

Input/Output

```
print("Message", var)

variable = input("Prompt: ")
```

The Troubleshooting Guide: Common Questions & Pitfalls

Problem	Solution
Trying to add a string and a number (`"Age: " + 25`) results in a `TypeError`.	You can't mix types with `+`. Convert the number to a string with `str(25)` or, even better, use an f-string: `f"Age: {25}"`.
Confusing assignment `=` with comparison `==`.	`=` stores a value (`x = 5`). `==` checks if two values are equal (`x == 5`).
`input()` gives you `"5"` instead of `5`.	`input()` always returns text. If you need a number for math, you must convert it: `age = int(input("Age: "))`.
`"2" + "3"` results in `"23"`.	For strings, `+` means concatenation (joining). For math, convert them to numbers first: `int("2") + int("3")` gives `5`.
Division by zero causes a `ZeroDivisionError`.	Your program will crash. Always check if a divisor is not zero (`if divisor != 0:`) before performing division.

Blueprints for Your Next Projects

You have the tools. The best way to become a skilled builder is to start building. Try tackling these projects to practice your new skills.

Personal Intro

Create variables for your name, age, and favorite color. Print them in a single, formatted sentence using an f-string.

Simple Calculator

Ask the user for two numbers. Display their sum, difference, product, and quotient.

Age Calculator

Ask the user for their birth year and calculate their current age (assume the current year is 2025).

Name Manipulator

Create a string with your full name. Use string methods to display it in all uppercase, print only the first name using slicing, and show how many characters it contains.

Temp Converter

Ask for a temperature in Celsius and convert it to Fahrenheit. The formula is

`F = (C * 9/5) + 32`.