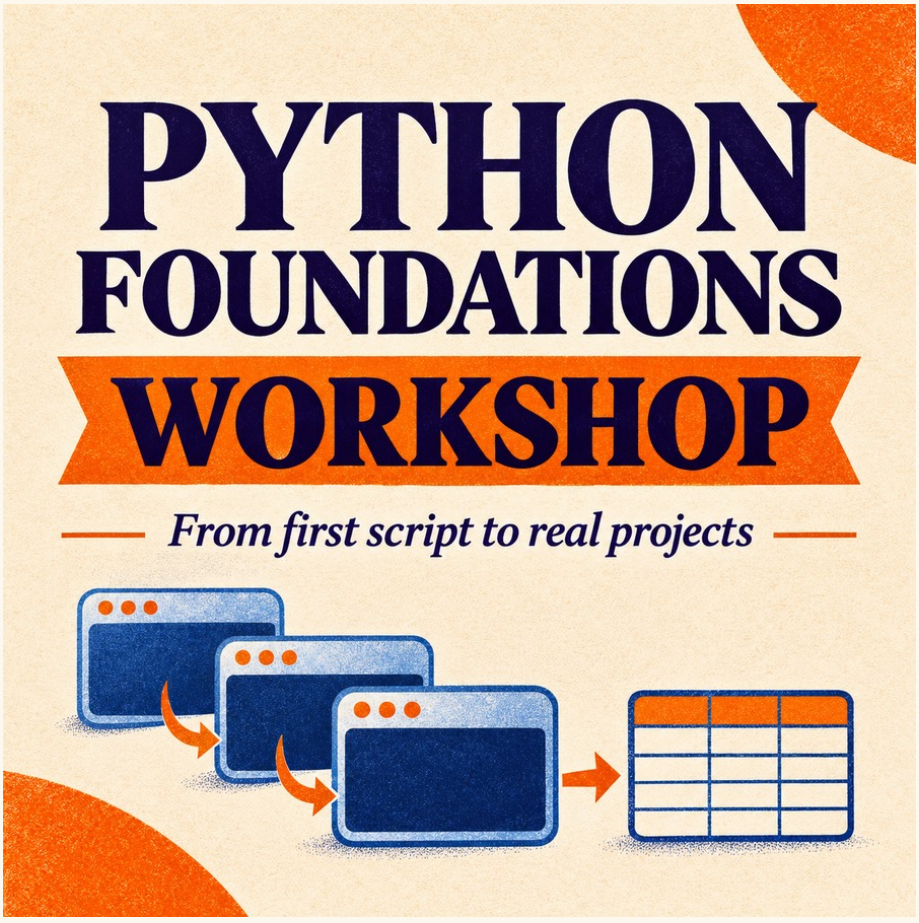
Python Foundations Workshop

From First Script to Real Projects



310 runnable demos, 65 practice problems, 58 workshops, and two Expense Tracker projects.

Companion website

Get Python running on Windows

Install Python 3.11 or newer from Python.org and install Visual Studio Code with Microsoft's Python extension. Open a new PowerShell terminal after installation so it can see updates to available commands.

```
py --version
py -c "import sys; print(sys.executable)"
```

If py is unavailable and python resolves to your installed interpreter, use python in the same commands. Check the version and executable path before continuing. Create a practice folder you own, save hello.py in it, and open the folder in VS Code.

```
print("Hello, Python!")
```

```
py hello.py
```

Run the command from the folder containing the file. A file-not-found error often means the terminal is looking in a different directory.

Official Python downloads

Get Python running on macOS or Linux

Use a supported Python 3.11 or newer. On macOS you can use the official Python.org installer. On Linux follow your distribution's supported installation method. Verify the command before editing system files or installing packages.

```
python3 --version
python3 -c 'import sys; print(sys.executable)'
```

Create and open a practice folder. Save a file named hello.py with the print statement from the previous page. Open a terminal in the same folder and run the file.

```
pwd
python3 hello.py
```

The rest of the book uses python as the interpreter command. Substitute python3 or use your virtual environment's interpreter if that is the command that resolves correctly on your machine.

Official Python tutorial

Run and debug in VS Code

Open your practice folder and install Microsoft's Python extension. In the Command Palette choose Python: Select Interpreter and pick the path you verified. Save the file before running it. Run Python File starts a terminal command using the selected interpreter.

```
price_cents = 125
quantity = 3
total = price_cents * quantity
print(total)
```

Set a breakpoint on the print line. Start Python debugging using the current file configuration. When execution pauses, inspect price_cents, quantity and total. Step over the print statement and compare the terminal output.

Try removing a closing parenthesis. Read the final traceback line and restore the syntax. If the wrong interpreter runs, print sys.executable in the script and compare with the status bar.

VS Code Python tutorial

Environment and first programs

Install an interpreter, choose it in VS Code, and run a saved script from an editor and a terminal. Compare the interpreter and working directory when the two routes behave differently.

IN THIS CHAPTER

5 concept demos · 5 practice problems · 4 workshops

CONCEPT SEQUENCE

01 Print a first message

02 See the active interpreter

03 Check the Python version

04 Find the working directory

05 Run top to bottom

WHAT I EXPECT TO BUILD

Print a first message

Concept demo · Read and predict

The interpreter evaluates the argument and print writes it followed by a newline.

RUNNABLE CODE

```
01  print("Hello, workshop!")
```

Bundle file: demos/ch01_01.py

PREDICT THE OUTPUT BEFORE RUNNING

Print a first message

Concept demo · Run and trace

RECORDED OUTPUT

```
Hello, workshop!
```

EXECUTED LINES AND VISIT COUNTS

```
01 x 1
```

Match these line numbers to the preceding page. Function bodies run when called. Branches and loop bodies can show different counts when the input changes.

TEST ONE VARIATION

Change a sample value or add a second call. Predict which output and visit counts could change. Re-run the file and explain the difference using the rule on the previous page.

CHANGED INPUT, RESULT AND EXPLANATION

See the active interpreter

Concept demo · Read and predict

The path helps you confirm which interpreter runs a file. Compare it with the interpreter selected in VS Code.

RUNNABLE CODE

```
01  import sys
02  print(sys.executable)
```

Bundle file: demos/ch01_02.py

PREDICT THE OUTPUT BEFORE RUNNING

See the active interpreter

Concept demo · Run and trace

RECORDED OUTPUT

```
<active Python executable>
```

EXECUTED LINES AND VISIT COUNTS

```
01 x 1    02 x 1
```

Match these line numbers to the preceding page. Function bodies run when called. Branches and loop bodies can show different counts when the input changes.

TEST ONE VARIATION

Change a sample value or add a second call. Predict which output and visit counts could change. Re-run the file and explain the difference using the rule on the previous page.

CHANGED INPUT, RESULT AND EXPLANATION

Check the Python version

Concept demo · Read and predict

This reports the major and minor version from inside the running program.

RUNNABLE CODE

```
01  import sys
02  print(sys.version_info.major, sys.version_info.minor)
```

Bundle file: demos/ch01_03.py

PREDICT THE OUTPUT BEFORE RUNNING

Check the Python version

Concept demo · Run and trace

RECORDED OUTPUT

```
3 12
```

EXECUTED LINES AND VISIT COUNTS

```
01 x 1    02 x 1
```

Match these line numbers to the preceding page. Function bodies run when called. Branches and loop bodies can show different counts when the input changes.

TEST ONE VARIATION

Change a sample value or add a second call. Predict which output and visit counts could change. Re-run the file and explain the difference using the rule on the previous page.

CHANGED INPUT, RESULT AND EXPLANATION

Find the working directory

Concept demo · Read and predict

Relative file paths start from the working directory, which is often the folder opened in the terminal.

RUNNABLE CODE

```
01  from pathlib import Path
02  print(Path.cwd().name)
```

Bundle file: demos/ch01_04.py

PREDICT THE OUTPUT BEFORE RUNNING

Find the working directory

Concept demo · Run and trace

RECORDED OUTPUT

```
<current folder>
```

EXECUTED LINES AND VISIT COUNTS

```
01 x 1    02 x 1
```

Match these line numbers to the preceding page. Function bodies run when called. Branches and loop bodies can show different counts when the input changes.

TEST ONE VARIATION

Change a sample value or add a second call. Predict which output and visit counts could change. Re-run the file and explain the difference using the rule on the previous page.

CHANGED INPUT, RESULT AND EXPLANATION

Run top to bottom

Concept demo · Read and predict

Two separate calls give two observable checkpoints in execution order.

RUNNABLE CODE

```
01  print("before")
02  # A comment explains intent without producing output.
03  print("after")
```

Bundle file: demos/ch01_05.py

PREDICT THE OUTPUT BEFORE RUNNING

Run top to bottom

Concept demo · Run and trace

RECORDED OUTPUT

```
before
after
```

EXECUTED LINES AND VISIT COUNTS

```
01 x 1    03 x 1
```

Match these line numbers to the preceding page. Function bodies run when called. Branches and loop bodies can show different counts when the input changes.

TEST ONE VARIATION

Change a sample value or add a second call. Predict which output and visit counts could change. Re-run the file and explain the difference using the rule on the previous page.

CHANGED INPUT, RESULT AND EXPLANATION

Three-line welcome

Practice · Assignment

BUILD FROM THIS BRIEF

Print WELCOME, then three equals signs, then a line inviting learners to run a file.

APPROACH

Identify the inputs and their types. Write a small first version in a new file. Compare the result with your own prediction, then inspect the reference solution.

PLAN AND BOUNDARY CASE