Python Foundations Workshop IV

Build Web Applications



96 runnable cases across 48 operating patterns, with three connected projects that take a workshop booking app from a catalog to sign-in and accessible reservation workflows.

Companion website

Build on Parts I–III

You should be able to run a Python module, validate input, write an assertion and follow a SQLite transaction. Part II served JSON and Part III ran jobs. This volume builds an interface people use in a browser. It covers HTML, forms, session state, permissions and accessibility.

LOCAL SETUP

```
python --version
python -m venv .venv
```

Use Python 3.12 or newer for this edition. Use the virtual environment interpreter directly if your terminal does not activate it. On Windows it is .venv\Scripts\python.exe. On macOS and Linux it is .venv/bin/python. The bundled examples and projects use the standard library.

HOW THE LABS WORK

Work through pairs of cases. They share an implementation and test different conditions. Predict the output and compare the trace. Run the assertion, reproduce a failure in a scratch copy, then make the change request.

ONE SKILL TO REFRESH BEFORE CHAPTER 1

Run the projects locally

Extract the complete ZIP, enter its projects folder, and run the following commands with the same interpreter. Use a fresh scratch database and the two synthetic accounts. The listener binds to loopback.

```
python -m unittest -v
python run_desk.py --database practice.db
# Open the printed URL. Press Ctrl+C to stop.
```

Sign in as jo with classroom-jo-demo or mi with classroom-mi-demo. The reference server is for local learning. A public service needs a maintained production stack and HTTPS.

Python WSGI reference

W3C accessible forms tutorial

OWASP CSRF prevention

OWASP password storage

OWASP session management

Route a browser request

Route requests by method and path. Decode dynamic IDs and keep URL construction separate from HTML interpolation.

FOUR PATTERNS AND EIGHT CASES

01.01 Open a workshop detail page / 6

01.02 Reject an extra path segment / 15

01.03 Match a catalog read / 24

01.04 Reject a mutation via GET / 33

01.05 Separate route and search query / 42

01.06 Reject an external authority / 51

01.07 Encode a route link with search / 60

01.08 Omit an empty query / 69

Explicit route tables make small apps easy to inspect. A maintained framework can supply path converters, method routing and URL generation as requirements grow.

Open a workshop detail page

Build brief

ASSIGNMENT

Open a workshop detail page. Predict the result and identify the input boundary that controls this case.

```
decide("/workshops/14")
```

CHAPTER CONTEXT

Route requests by method and path. Decode dynamic IDs and keep URL construction separate from HTML interpolation.

INPUTS, RETURN SHAPE, AND FIRST EXPECTED RESULT

Open a workshop detail page

Reference implementation

```
import re
def decide(path):
    m = re.fullmatch(r"/workshops/([1-9][0-9]{0,5})",path)
    return ("detail",int(m.group(1))) if m else ("missing",None)
result = decide("/workshops/14")
print(repr(result))
```

Run the bundled file with Python before trying a variation.

`labs/ch01_01.py`

Open a workshop detail page

Observed behavior and trace

RECORDED OUTPUT

```
('detail', 14)
```

EXECUTED LINE NUMBERS AND COUNTS

```
01 x 1    02 x 1    03 x 1    04 x 1    05 x 1    06 x 1
```

Trace counts follow the current Python thread. Some callbacks or worker bodies can run in another thread. Compare the output with the contract even if some worker lines are absent from this trace.

FIND THE LINE RESPONSIBLE FOR A VISIBLE OUTPUT

Open a workshop detail page

Verification

Run this check after the reference implementation. The bundled script includes the check, and a successful run means the assertion passed as well as producing the recorded output.

```
expected = ("detail",14)
assert type(result) is type(expected)
assert result == expected
```

DESIGN A BOUNDARY CHECK

A prefix match can route more paths than the page actually handles.

SMALLEST INPUT, EXPECTED RESULT, AND WHAT THE ASSERTION PROVES

Open a workshop detail page

Diagnose a failure

DEFECT TO INVESTIGATE

A prefix match can route more paths than the page actually handles.

REPRODUCE IN A SCRATCH COPY

Write a counterexample based on the defect. Run it first against the reference, then make one deliberately flawed edit that introduces the defect. Record the changed output or exception before restoring the reference.

ORIGINAL LINE AND CHANGED LINE

OBSERVED DIFFERENCE

Open a workshop detail page

Compare design choices

ENGINEERING CHOICE

Explicit route tables make small apps easy to inspect. A maintained framework can supply path converters, method routing and URL generation as requirements grow.

PAIRED CASE

Reject an extra path segment

```
decide("/workshops/14/edit")
# Expected: ("missing",None)
```

Compare the paired case with the original. Name the condition that changed and explain which behavior should remain invariant.

ALTERNATIVE AND DECISION CRITERION

Open a workshop detail page

Implement the change request

NEW REQUIREMENT

Give the application a separate edit route and test an unknown suffix.

PRESERVE A PASSING BEHAVIOR

```
expected = ("detail",14)
assert type(result) is type(expected)
assert result == expected
```

Decide whether this original check still describes the supported interface. Add a new check for the changed behavior and keep it alongside the original when both contracts apply.

CHANGE AND NEW VERIFICATION

Open a workshop detail page

Integrate it with a caller

CALLER CONTRACT

Open a workshop detail page. Predict the result and identify the input boundary that controls this case.

Sketch a second caller that supplies data from a terminal command, an import file, or a fixture. Keep the calculation independent from that input boundary. State which errors the caller can handle and which should stop the workflow.

REVIEW THE HANDOFF

Give the application a separate edit route and test an unknown suffix.

INPUTS, OUTPUT, AND A FAILURE THE CALLER MUST HANDLE

Open a workshop detail page

Acceptance record

Submit the original run, one counterexample, and one version that handles the change request. Keep the script and invocation together so another reader can reproduce your result.

EXPECTED BEHAVIOR

Open a workshop detail page. Predict the result and identify the input boundary that controls this case.

REGRESSION TO CATCH

A prefix match can route more paths than the page actually handles.

COMMAND, OBSERVED RESULT, AND REMAINING LIMIT

Reject an extra path segment

Build brief

ASSIGNMENT

Reject an extra path segment. Predict the result and identify the input boundary that controls this case.

```
decide("/workshops/14/edit")
```

CHAPTER CONTEXT

Route requests by method and path. Decode dynamic IDs and keep URL construction separate from HTML interpolation.

INPUTS, RETURN SHAPE, AND FIRST EXPECTED RESULT

Reject an extra path segment

Reference implementation

```
import re
def decide(path):
    m = re.fullmatch(r"/workshops/([1-9][0-9]{0,5})",path)
    return ("detail",int(m.group(1))) if m else ("missing",None)
result = decide("/workshops/14/edit")
print(repr(result))
```

Run the bundled file with Python before trying a variation.

`labs/ch01_02.py`

Reject an extra path segment

Observed behavior and trace

RECORDED OUTPUT

```
('missing', None)
```

EXECUTED LINE NUMBERS AND COUNTS

```
01 x 1    02 x 1    03 x 1    04 x 1    05 x 1    06 x 1
```

Trace counts follow the current Python thread. Some callbacks or worker bodies can run in another thread. Compare the output with the contract even if some worker lines are absent from this trace.

FIND THE LINE RESPONSIBLE FOR A VISIBLE OUTPUT