LESSON 54

Templates in Flask

80 minutes (2 periods) | Python Web Backend | Outcome 1.22 | Outcome 1.23 | Outcome 2.6 | Outcome 2.7 | Outcome 2.16 | Outcome 2.20 | Outcome 3.3

WHAT THIS LESSON DOES

In this lesson, you'll explore how to use templates in Flask to build dynamic web pages. Follow step-by-step instructions to set up a project, create a Flask app, and render HTML templates with variables and loops for dynamic content.

CURRICULUM OUTCOMES

Outcome	What the specification asks for
1.22	read, write, test, and modify computer programs
1.23	reflect and communicate on the design and development process
2.6	construct algorithms using appropriate sequences, selections/conditionals, loops and operators to solve a range of problems, to fulfil a specific requirement
2.7	implement algorithms using a programming language to solve a range of problems
2.16	use data types that are common to procedural high-level languages
2.20	identify and fix/debug warnings and errors in computer code and modify as required
3.3	use appropriate programming languages to develop an interactive website that can display information from a database that meets a set of users' needs

WHAT STUDENTS SHOULD BE ABLE TO DO

- Understand the purpose and benefits of using templates in web development.
- Learn to set up and render HTML templates in a Flask application.
- Master passing data from Python to HTML templates for dynamic content.
- Apply Jinja2 features like variables and loops to create interactive web pages.
- Develop skills in organising Flask projects with separate template folders.

BEFORE THE LESSON

- Run through the whole build yourself first: fresh folder, venv, pip install flask, then the render_template steps. The templates folder must be named exactly 'templates' and sit at the project root or Flask will not find index.html.
- Confirm pip install flask works behind the school filter on a student machine, not just yours.
- Have the finished app.py and index.html saved somewhere reachable so a stuck student can compare, but do not hand it out before the challenge.

Templates in Flask

HOW THE LESSON RUNS

- Introduction
- Create a Project Folder
- Create the Environment
- Create a New Terminal and Install Flask
- Creating a Basic Flask App
- Setting Up Templates
- Rendering a Template
- Passing Variables to Templates
- Using Loops in Templates
- Challenge
- Wrap Up

TEACHING MOVES

- 3. Create the Environment. Make students create a fresh venv even though they made one before. The point is a repeatable habit. Check the '.venv' folder appears and the terminal prompt shows the environment is active before anyone installs Flask.
- 5. Creating a Basic Flask App. Have everyone reach the plain text 'Welcome' message in the browser before touching templates. This proves the app runs, so any later error is about templates specifically, not setup.
- 6. Setting Up Templates. Stress the folder must be named 'templates', lowercase, at project root. Wrong name or location gives a TemplateNotFound error that looks like a code bug but is not. Show them the error deliberately if time allows.
- 7. Rendering a Template. Point out render_template must be imported and the filename is passed as a string, not the folder path. The HTML lives in the template now, not in the return statement.
- 8. Passing Variables to Templates. Line up the two files side by side. The keyword argument name=name in Python feeds the {{ name }} in the HTML. Have them change 'David' to their own name and reload to prove the link is live.
- 9. Using Loops in Templates. Draw the distinction between {{ }} for output and {% %} for logic like the for loop. Ask why {% endfor %} is needed here when Python uses indentation instead.
- 10. Challenge. Resist showing the answer. Nudge with questions: how do you reach a dictionary value in a template, and what does a Jinja2 if look like given the for loop they just used? Let them search the pattern.

WHAT TO WATCH FOR

- Students think a page that displays correctly means their code is correct, when the browser is actually showing a cached earlier version after a change. Show them a hard refresh, and get them into the habit of confirming the terminal reloaded the server after each save before judging the result.
- Confusing {{ }} and {% %}, using the output braces to try to write a loop or an if. State the rule plainly: double braces print a value, brace-percent runs logic and must be closed. Have them label each pair in their own file.
- Believing the HTML still lives in app.py, so they edit the return string expecting the page to change. Once render_template is in use, the visible content comes from index.html. Have them edit the template and reload to see which file controls the page.

DIFFERENTIATION

- Emerging: If TemplateNotFound appears, check the folder name and location with them before looking at any code. It is almost always the folder. Pair them with a student who has a working render step so they can copy the folder structure by eye.
- Developing: Ask them to explain aloud how name=name in Python connects to {{ name }} in the HTML before they move to the loop. Have them add a second variable and place it in the page themselves, without new instructions.
- Proficient: Push them fully into the challenge: pass a dictionary and display key-value pairs, then add a Jinja2 conditional for an empty list. Ask them to research template inheritance with a base layout, a Higher Level extension of the separation idea this lesson introduces.