LESSON 56

Data Types in Backend

80 minutes (2 periods) | Python Web Backend | Outcome 2.16 | Outcome 1.22 | Outcome 1.23 | Outcome 2.7 | Outcome 3.3 | Outcome 2.19 | Outcome 2.18

WHAT THIS LESSON DOES

In this lesson, you'll learn about common data types in Python and how to apply them in Flask apps. Follow steps to set up a project, handle strings, integers, lists, and dictionaries, and test data processing in a backend context.

CURRICULUM OUTCOMES

Outcome	What the specification asks for
2.16	use data types that are common to procedural high-level languages
1.22	read, write, test, and modify computer programs
1.23	reflect and communicate on the design and development process
2.7	implement algorithms using a programming language to solve a range of problems
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
2.19	test solutions and decisions to determine their short-term and long-term outcomes
2.18	collect, store and sort both continuous and discrete data

WHAT STUDENTS SHOULD BE ABLE TO DO

- Understand the role of common data types in backend development.
- Apply strings, integers, lists, and dictionaries in a web application context.
- Develop skills to process user input using various data types.
- Build and test a Flask app to handle multiple data structures.
- Recognise how data types integrate with backend logic and user interactions.

BEFORE THE LESSON

- Run through the whole app.py yourself first: fresh folder, Python: Create Environment with Venv, then pip install flask, and confirm Flask installs behind the school filter before class.
- Check the terminal activates the .venv correctly on your room's machines, and confirm each student can save files to a folder they own (not a locked shared drive).
- Have the string route working and viewable in a browser so you can demonstrate a POST form submission before students build their own.

Data Types in Backend

HOW THE LESSON RUNS

Introduction
Create a Project Folder
Create the Environment
Create a New Terminal and Install Flask
Basic Flask App with String Data
Handling Integer Data
Working with Lists
Using Dictionaries
Challenge
Wrap Up

TEACHING MOVES

- 2. Create a Project Folder. Insist on the exact folder name and that VS Code has that folder open, not a parent. Half the later errors trace back to app.py sitting outside the opened folder.
- 3. Create the Environment. Watch for the .venv appearing and the interpreter switching. If the prompt does not show (.venv), the next pip install goes to the wrong Python. Fix this before anyone moves on.
- 5. Basic Flask App with String Data. Stress that form input always arrives as a string. Show it echoing back text unchanged so students see raw string data before any conversion is introduced.
- 6. Handling Integer Data. Make the point that request.form['age'] is still a string until int() converts it. Demonstrate the error when you skip int() and try to do arithmetic on '25'.
- 7. Working with Lists. Type comma-separated input live and show .split(',') producing the list. Point out extra spaces around commas become part of each item unless stripped.
- 8. Using Dictionaries. Contrast the dictionary with the list: access by key, not position. Build one from separate form fields so students see keys mapping to submitted values.
- 9. Challenge. Have students sketch which form field maps to which data type before coding. Circulate for the average-of-a-list idea, which forces int() conversion inside a loop.

WHAT TO WATCH FOR

- Students assume a number typed into a form arrives as an integer and can be added or averaged directly. Show that request.form always returns a string. Print type() of the value before and after int() so the conversion is visible, not assumed.
- Students think a page that loads without an obvious error means the route is correct. Have them submit deliberately awkward input (empty field, letters where a number is expected) and check the actual output, not just that the page rendered.
- Confusing list access by position with dictionary access by key. Put a list and a dictionary side by side on the board. Ask how you would fetch the age in each, drawing out index versus key.

DIFFERENTIATION

- Emerging: If the environment is fighting a student, pair them with a working machine and let them type into a running app rather than losing the lesson to venv setup. Give them just the string route to complete first, so they succeed with one data type before meeting four.
- Developing: Ask them to predict the output type at each step before running, then check against type(). This builds reasoning rather than copying.
- Proficient: Point them at the challenge's average-of-numbers route and ask them to handle invalid input gracefully, which is genuine Coursework Project evidence of testing and robustness. Ask them to explain when a dictionary is the better structure than a list, framing it as a Higher Level design decision.