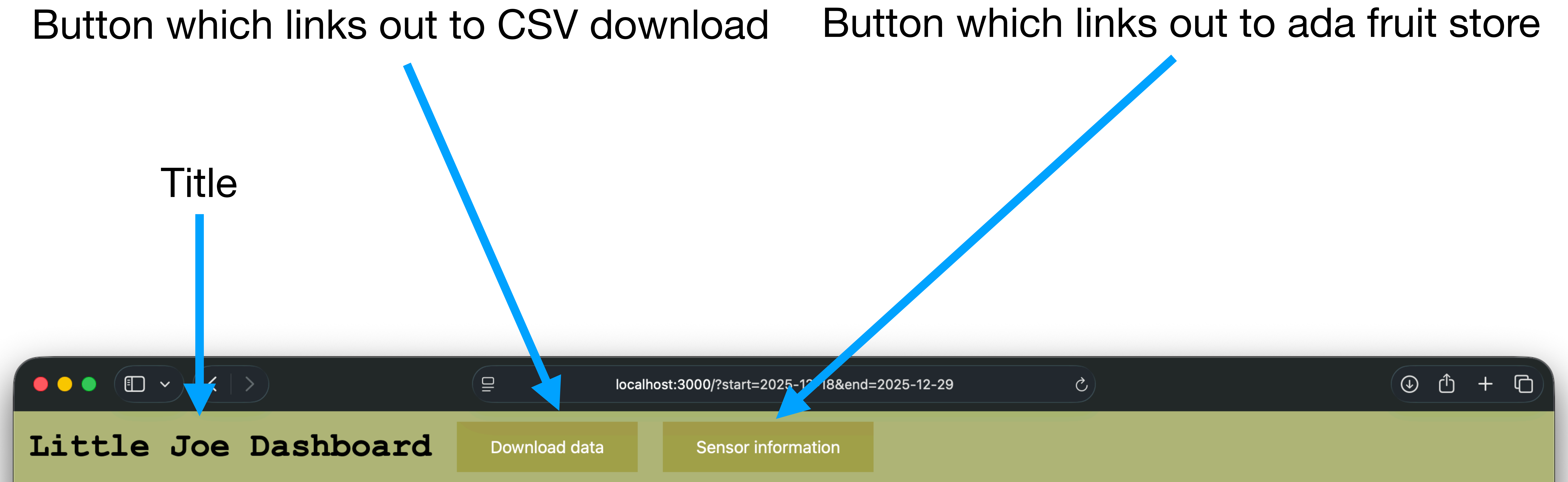
Little Joe web app

Overview

Sam Gardner

App layout overview

Part 1 - Head



Button which links out to CSV download

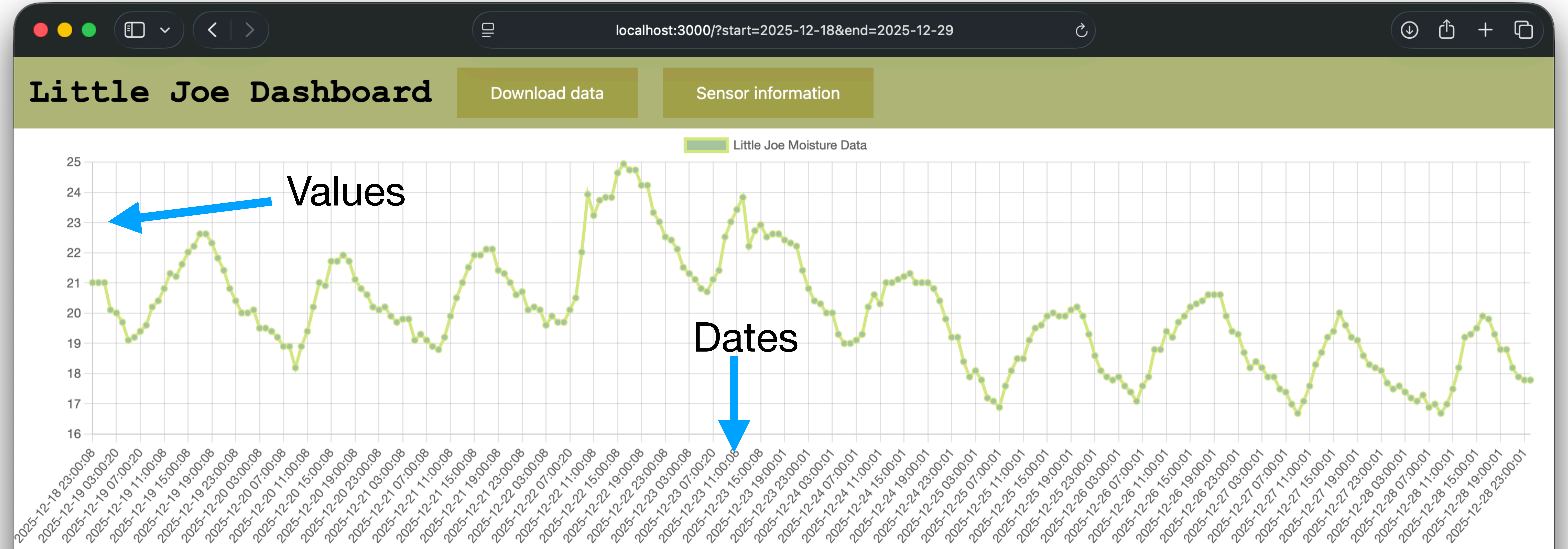
Button which links out to ada fruit store

Title

localhost:3000/?start=2025-12-18&end=2025-12-29

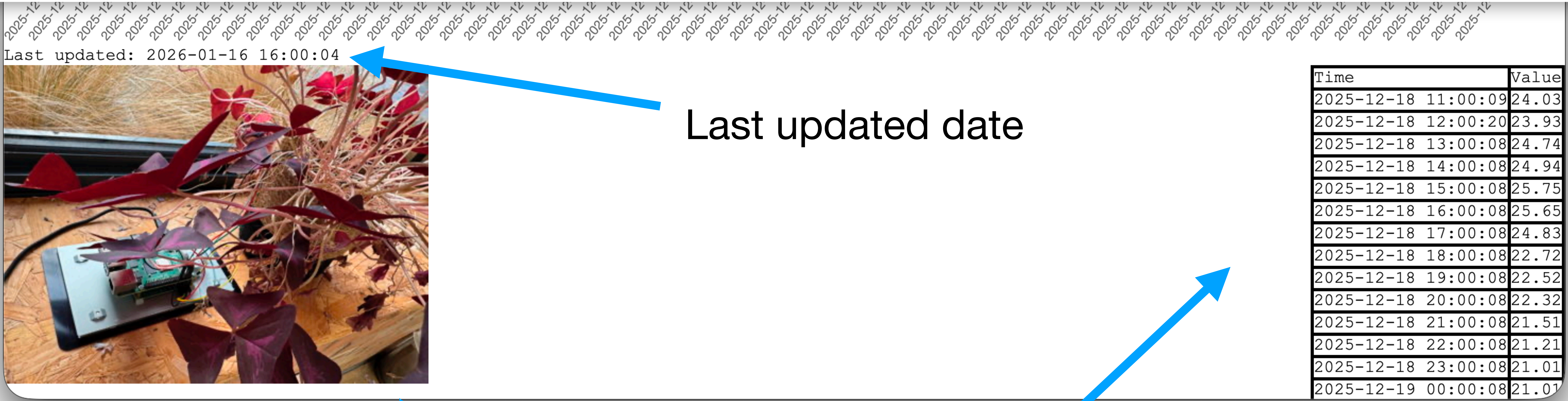
App layout overview

Part 1.1 - Body - graph



App layout overview

Part 1.1 - Body - graph



Last updated: 2026-01-16 16:00:04

Time	Value
2025-12-18 11:00:09	24.03
2025-12-18 12:00:20	23.93
2025-12-18 13:00:08	24.74
2025-12-18 14:00:08	24.94
2025-12-18 15:00:08	25.75
2025-12-18 16:00:08	25.65
2025-12-18 17:00:08	24.83
2025-12-18 18:00:08	22.72
2025-12-18 19:00:08	22.52
2025-12-18 20:00:08	22.32
2025-12-18 21:00:08	21.51
2025-12-18 22:00:08	21.21
2025-12-18 23:00:08	21.01
2025-12-19 00:00:08	21.01

Last updated date

Table of values

Image of little Joe

The inner workings of the web app

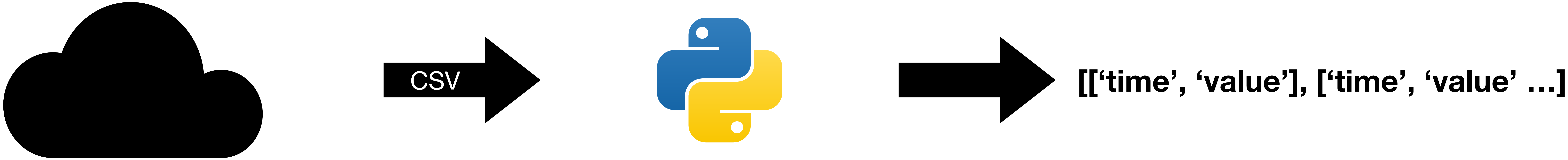
Part 2 - Data handling - collection

The first thing the script does is fetch the moisture data gathered by the moisture sensor by downloading it from a url as a CSV.

The inner workings of the web app

Part 2.1 - Data handling - formatting

Then the script converts the CSV file into a python list which contains a list for each line in the CSV file



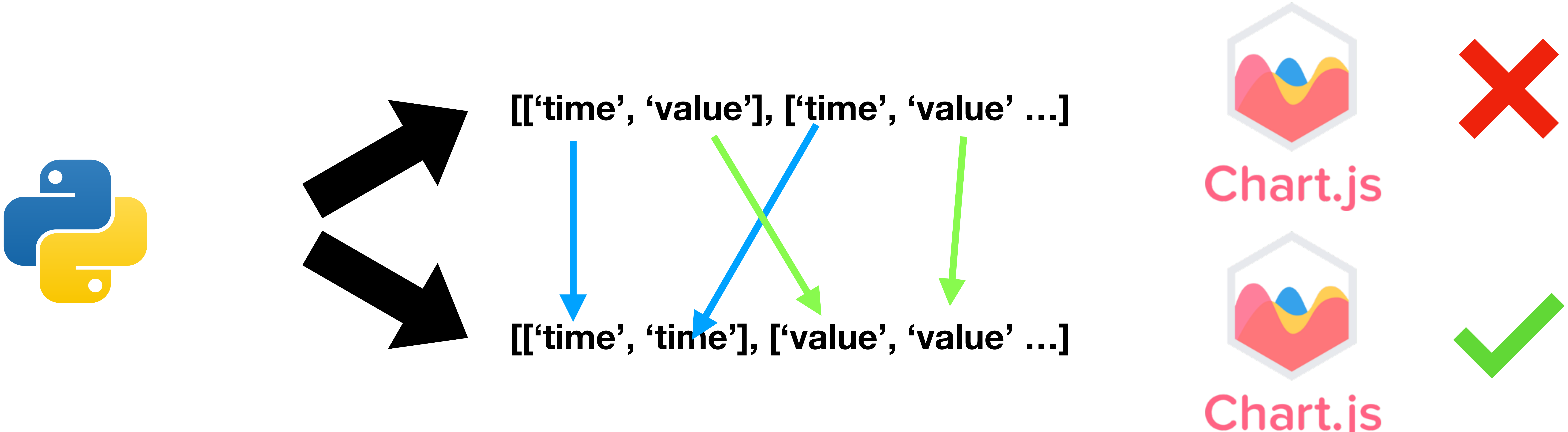
CSV

[[‘time’, ‘value’], [‘time’, ‘value’ …]

The inner workings of the web app

Part 2.2 - Data handling - formatting for chart.js

The data format for the table did not match the on required for chart.js so I needed to rearrange the lists.



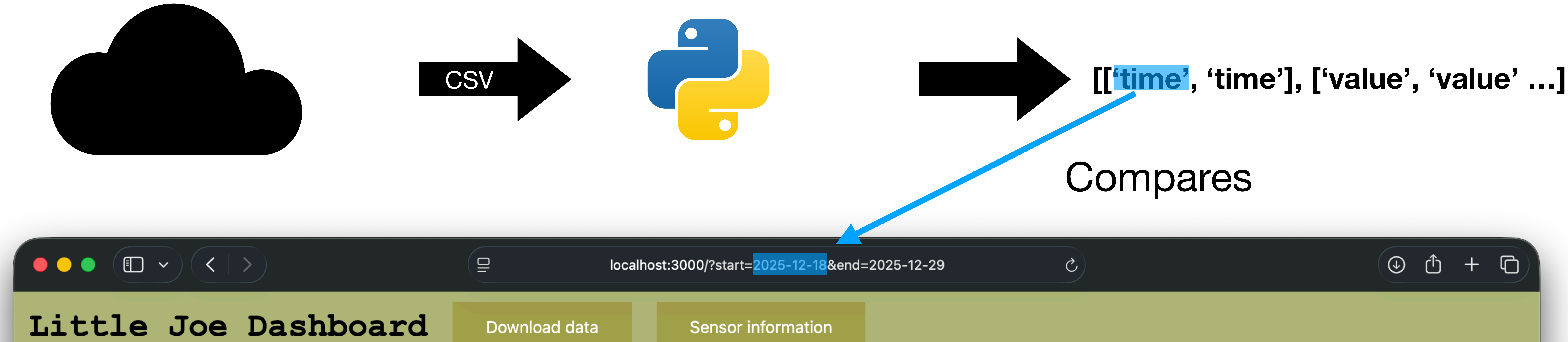
[['time', 'value'], ['time', 'value' …]

[['time', 'time'], ['value', 'value' …]

The inner workings of the web app

Part 2.4 - Data handling - setting the data range via URL queries

To set the data range of the graph I took the queries specified in the URL and compared them with the timestamps in the data list to find their index.



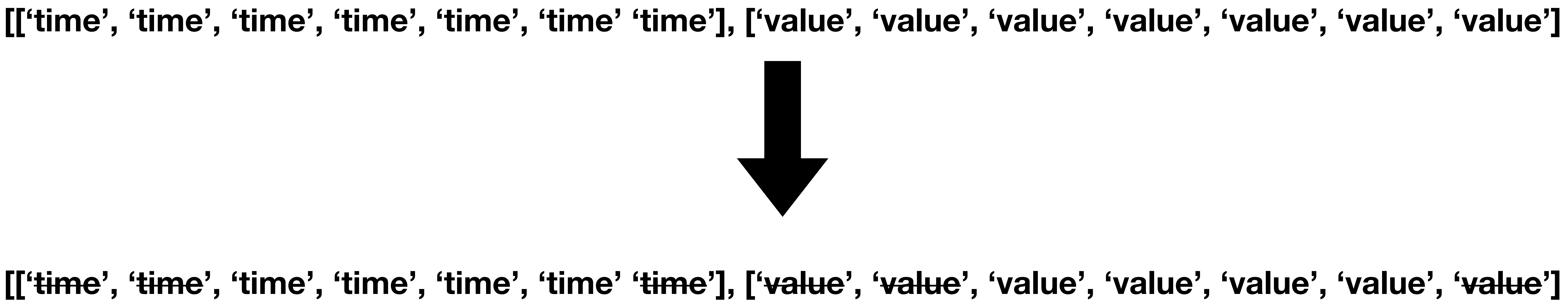
[['time', 'time'], ['value', 'value' …]

localhost:3000/?start=2025-12-18&end=2025-12-29

The inner workings of the web app

Part 2.4 - Data handling - slicing data

With the index of the timestamp specified in the URL I sliced the list to only show the values in the specified range



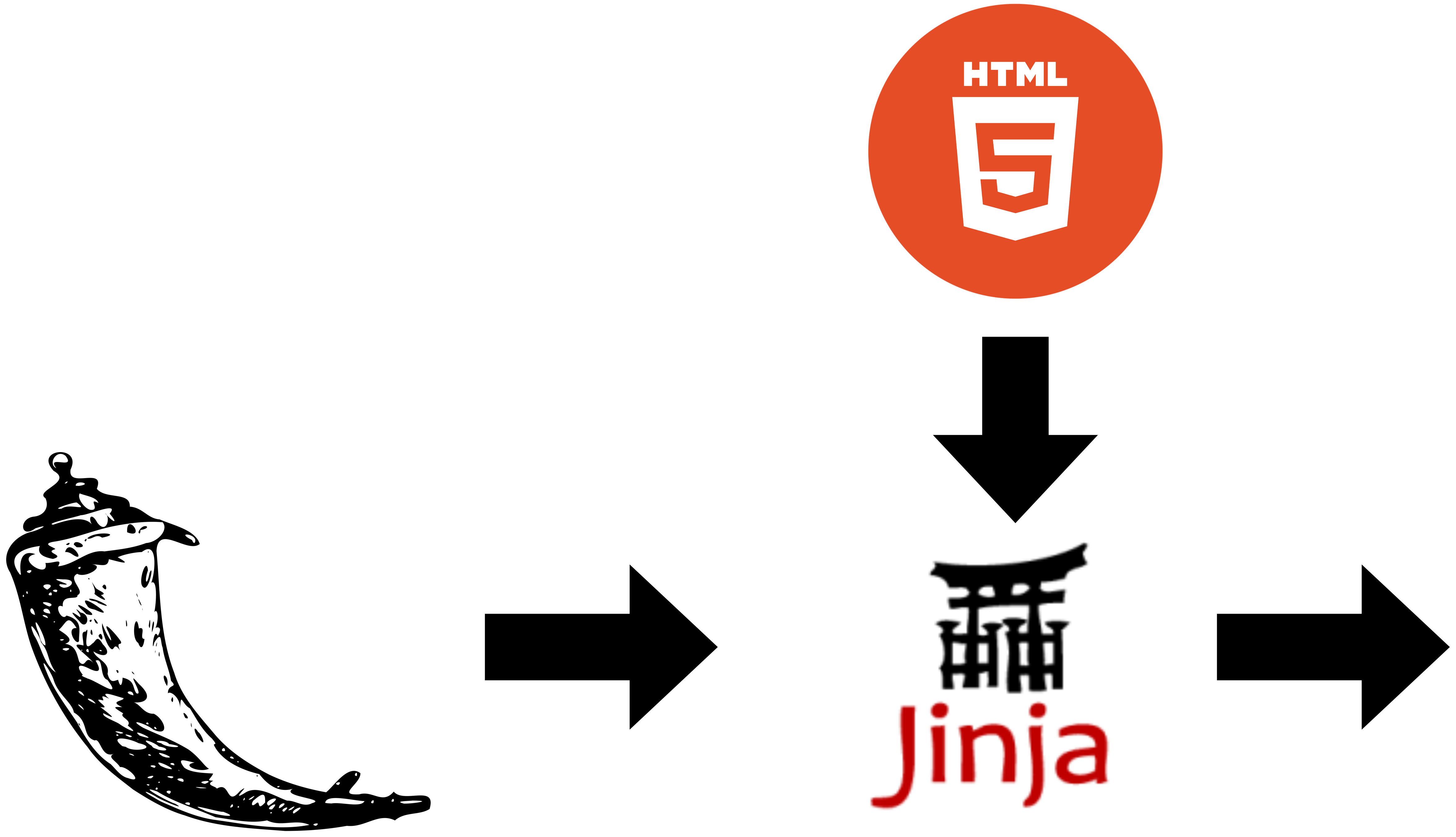
[[‘time’, ‘time’, ‘time’, ‘time’, ‘time’, ‘time’ ‘time’], [‘value’, ‘value’, ‘value’, ‘value’, ‘value’, ‘value’, ‘value’]

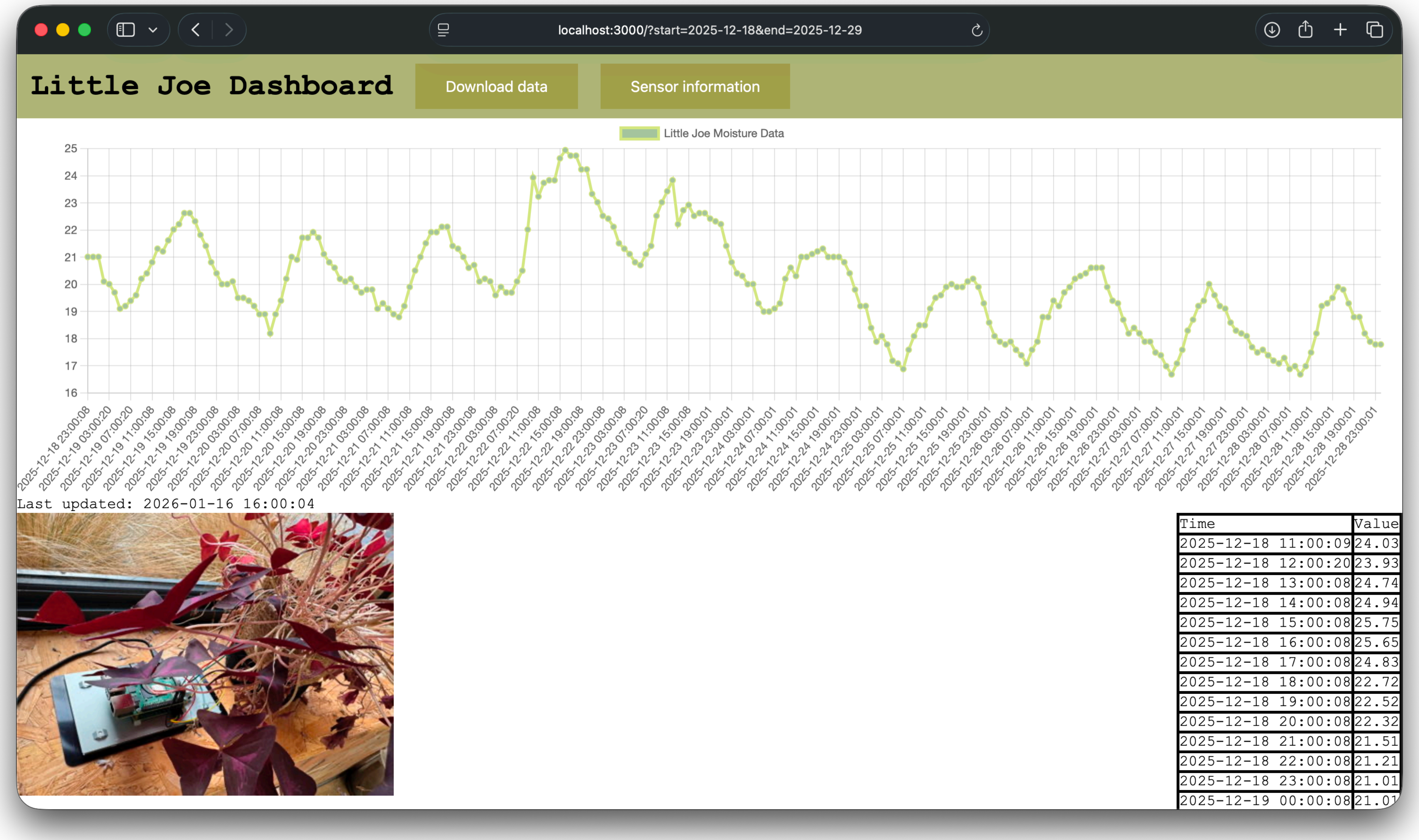
[[‘~~time~~’, ‘~~time~~’, ‘time’, ‘time’, ‘time’, ‘time’ ‘~~time~~’], [‘~~value~~’, ‘~~value~~’, ‘value’, ‘value’, ‘value’, ‘value’, ‘~~value~~’]

The inner workings of the web app

Part 3 - Rendering template

Flask can now use this data to render the page with jinja using a HTML template.





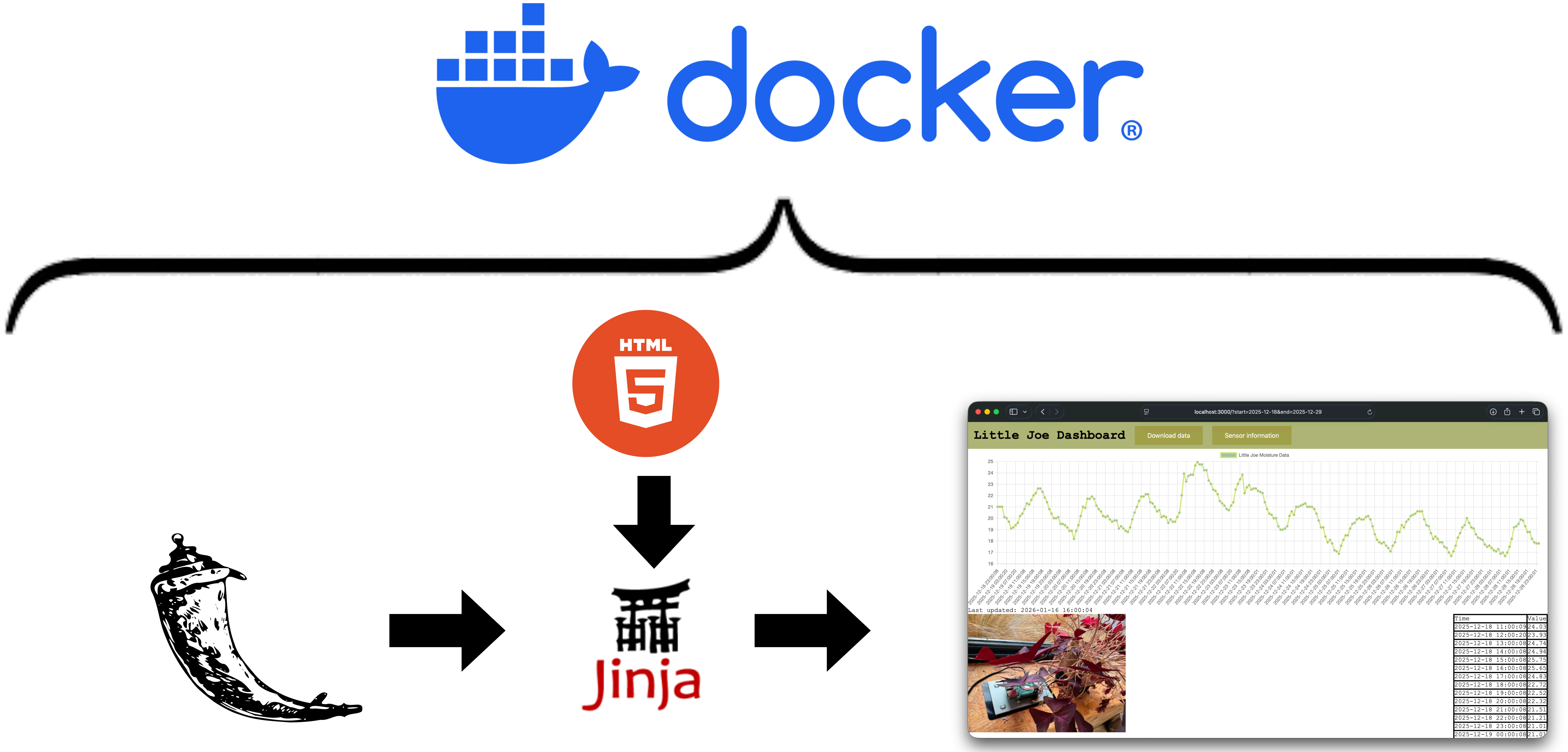
Last updated: 2026-01-16 16:00:04

Time	Value
2025-12-18 11:00:09	24.03
2025-12-18 12:00:20	23.93
2025-12-18 13:00:08	24.74
2025-12-18 14:00:08	24.94
2025-12-18 15:00:08	25.75
2025-12-18 16:00:08	25.65
2025-12-18 17:00:08	24.83
2025-12-18 18:00:08	22.72
2025-12-18 19:00:08	22.52
2025-12-18 20:00:08	22.32
2025-12-18 21:00:08	21.51
2025-12-18 22:00:08	21.21
2025-12-18 23:00:08	21.01
2025-12-19 00:00:08	21.01

The inner workings of the web app

Part 3.1 - Running the whole process in docker

We can go one step further and make this process highly portable by compiling it into a docker image.



Last updated: 2026-01-16 16:00:04

Time	Value
2025-12-18 11:00:09	24.03
2025-12-18 12:00:20	23.93
2025-12-18 13:00:08	24.74
2025-12-18 14:00:08	24.94
2025-12-18 15:00:08	25.75
2025-12-18 16:00:08	25.65
2025-12-18 17:00:08	24.83
2025-12-18 18:00:08	22.72
2025-12-18 19:00:08	22.52
2025-12-18 20:00:08	22.32
2025-12-18 21:00:08	21.51
2025-12-18 22:00:08	21.21
2025-12-18 23:00:08	21.01
2025-12-19 00:00:08	21.01

The inner workings of the web app

Part 3.1 - Running the whole process in docker

To do this I added a:

- .dockerignore file

- Dockerfile file

- Requirements.txt file