Sophie Ashmore 25358401

Reflection A2 Food Expiry Tracker

Creating my food expiry tracker was both an invigorating and challenging experience, but ultimately rewarding. This project allowed me to move from a very surface level understanding of code to a beginner-intermediate level of confidence. Developing my food expiry tracker in Visual Studio code was my first experience coding a working prototype from scratch. The goal was to reduce unnecessary food waste by allowing users to record grocery items and their expiry dates.

Throughout this process, I experienced both breakthroughs and setbacks that reshaped my understanding of programming and data. I have structured this reflection around Moon's (2006) *"What? So what? Now what?"* framework, using each 'moment' as evidence of how I moved from confusion to confidence through applied learning.

Moment 1: Discovering Data Doesn't Save Automatically

What?

My first week was about creating a simple structure of my prototype, with my goal for that week being to let users enter input food names as well as expiry dates. The first version worked, until I closed and reopened the program. All my data had vanished, and each time I re-ran the code, the list was empty again. My immediate reaction was confusion and mild panic, as I assumed Python would "remember" what had been entered. I also tested the wrong date format (29/10/2025) and got a ValueError. At first, this was frustrating, as I didn't know what "format" meant in this context.

So what?

I was quick to realise with some simple research that when you close a python project, everything stored in variables is erased from memory.
After research, I learned that when a Python program closes, all variables are erased from memory. I also received a ValueError when entering dates incorrectly (e.g. 29/10/2025). This taught me that syntax and data formats must be consistent. Using VS Code's terminal made it easy to re-run and test changes quickly, which helped me understand the value of debugging and iteration. Using VS Code's terminal helped me fix these problems easily because I could instantly re-run and see results.

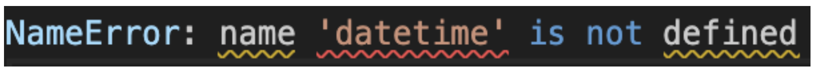
Now what?

This stage gave me foundational confidence, as I learned that a central part of programming is learning to understand how the computer thinks (QUOTE HERE). I also realised that errors were not failures but signposts for understanding system logic. This lesson shaped how I approached every later problem.

Moment 2: Using Datetime for Expiry Calculations

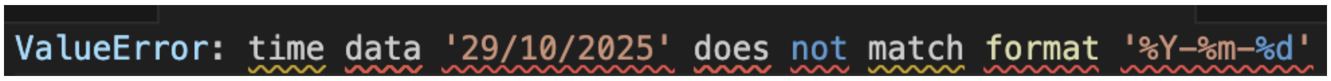
What?

By Week 8, I aimed to calculate the number of days left before each item expired. I introduced Python's datetime library, but my early attempts produced confusing errors such as:



```
NameError: name 'datetime' is not defined
```

Figure 1: Screen shot of datetime library not being imported properly



```
ValueError: time data '29/10/2025' does not match format '%Y-%m-%d'
```

Figure 2: Error message showcasing incorrect format python doesn't understand

These errors remained discouraging to me as it was only the first couple weeks of my project, and I was "already feeling overwhelmed with learning to use VS code" (Project Journal, 12/09). However, after following the Python Datetime Tutorial on W3Schools (2025) and watching the Python for Students tutorial (Madecraft, 2024)

After attempts to correct my syntax, the result finally returned as a single integer displaying "Milk expires in 3 days", and I felt confident with the progress I had made thus far within my project.

So what?

This was the first moment I felt my understanding of programming deepen. Charles Petzold (2023) describes in 'Code: The Hidden Language' that programming transforms human meaning into machine readable logic, and I understood this directly when seeing "expiry days" calculated.

Now what?

By understanding datetime, I saw how Python can transform human ideas (like "expiry date") into something you can measure.

Moment 3: Data persistence through Json

What?

Having established accurate expiry calculations, I wanted to make my project save data between each run. Initially, I kept overwriting files or saving empty lists. Utilizing the json library, I successfully stored food items in a file named fridge_data.json. Initial errors, such as overwriting data with empty lists or unintentional duplication, revealed weaknesses in my approach to state management. “I sat there and started at a blank file, I erased everything” (Journal Entry, 22/09). As Dr. Charles Severance wisely advises in Python for Informatics, 'If something seems particularly hard, there is usually no value in staying up all night and staring at it, come back to it with fresh eyes.' This philosophy proved essential during the debugging phase of my prototype, where I was feeling overwhelmed and confused, I took a break and came back to it with fresh eyes.

So what?

This experience showed how persistence separates a “script” from a functioning program. As Sweigart (2025, p. 112) writes, “Data persistence is what turns a test into a tool.” I learned to back up files and use validation logic to prevent duplicate entries.

Now what?

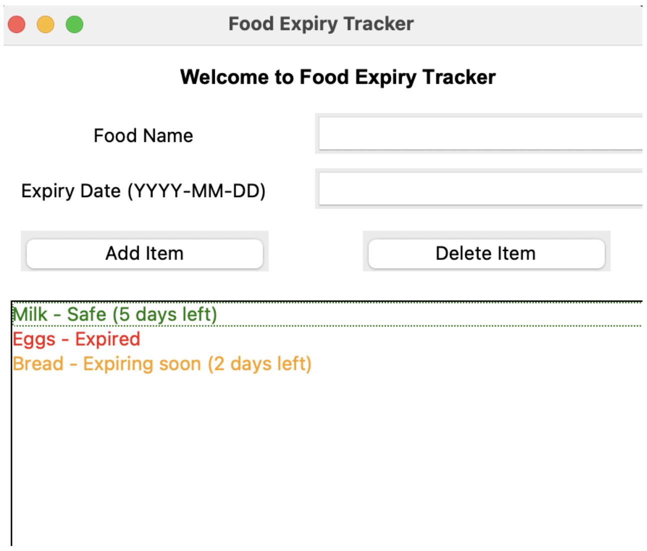
I began testing smaller code sections and saving duplicates before major edits. These habits strengthened my problem-solving and attention to reliability.

<u>Moment 4: User Interface</u>

What?

By week 10, I had introduced a basic but effective GUI interface following the Bros Code (2024) Python Gui Tutorial. The window included input boxes for item name and expiry date, and a colour coded display list coded by urgency (green safe, orange for soon, red expired).

When demonstrating the prototype to my peers, they provided valuable feedback: the colour coding worked, but the layout was unclear, saying that my buttons were too close together and labels lacked spacing. This was particularly useful as they were peer reviewing from an actual users perspective, bringing things to my attention I had not considered on my own. (quote) I also found that entering text incorrectly caused the interface to crash. I wanted to change this to make sure instead of crashing, the interface would just display a clear error message to display invalid.



Welcome to Food Expiry Tracker

Food Name

Expiry Date (YYYY-MM-DD)

```
Milk - Safe (5 days left)
Eggs - Expired
Bread - Expiring soon (2 days left)
```

Figure 3: Food expiry tracker GUI showcasing items and their expiry dates

So what?

This feedback taught me that effective programming and GUI involves designing for users. Moon (2006) describes how reflection involves not only technical understanding but emotional and contextual awareness. I also learned to use VS Code's error highlights and terminal logs to track GUI issues effectively.

Now what?

I revised my GUI layout to improve its readability and overall clarity, teaching me to learn to balance being precise whilst considering user experience. While the current Food Expiry Tracker GUI is intentionally basic, its simplicity reflects its functionality takes precedence over aesthetics, prioritising user flow and visual communication , focused on the backend logic.

<u>Moment 5: Ambitions beyond skill level</u>

What?

In the final weeks, I ambitiously attempted to add notifications, recommended food items and recipe suggestions to my prototype, but continuously received "ModuleNotFoundError" and syntax errors. "I feel like everything I attempted or tried to fix created another problem" (21/10) I wrote in my journal entry that day, leaving me frustrated with my project.

So what?

I quickly learnt that my ambitions for the project did not match my understanding, leading me to be overly optimistic in my goals. Sweigart (2025, p. 44) explains that “adding a new feature ripples through the entire system.” I also connected this to Petzold’s (2023, p. 194) observation that “programming is not just mechanical but linguistic, every idea must be defined with precision.”

Now what?

Although I was unable to figure out how to fully implement these stretch goals from my A1, the process had reframed my perspective. I aim to explore API’s and notification systems in the future completing similar projects using frameworks like “schedule”. I learnt to understand not just why my code works, but the majority of the time completing my project, why it had failed.

Reference list:

Bros Code. (2024). *Python GUI Tutorial: Tkinter for Beginners* [YouTube Video]. YouTube.https://www.youtube.com/watch?v=lyoyTlltFVU&list=PLZPZq0r_RZOOeQBaP5SeMjl2nwDcJaV0T

Madecraft. (2024). *Python for Students: Learn Datetime* [Video Tutorial]. LinkedIn Learning.https://www.linkedin.com/learning/learning-python-2021/the-date-time-and-datetime-classes

Moon, J. (2006). *Learning Journals: A Handbook for Reflective Practice and Professional Development* (2nd ed.). Routledge.

Petzold, C. (2023). *Code: The Hidden Language of Computer Hardware and Software* (2nd ed.). Microsoft Press.

Sweigart, A. (2019). *Automate the Boring Stuff with Python, 2nd Edition: Practical Programming for Total Beginners* (2nd edition). No Starch Press, Incorporated.

Severance, C. (2013, 9 September). *Python for informatics: Exploring information* (Versi 0.0.7, Bab 1). Dr. Chuck. Diambil dari https://www.dr-chuck.com/py4inf/html-007/cfbook.html