Case Study 1 – Steps 6-7

Step 6 – Test

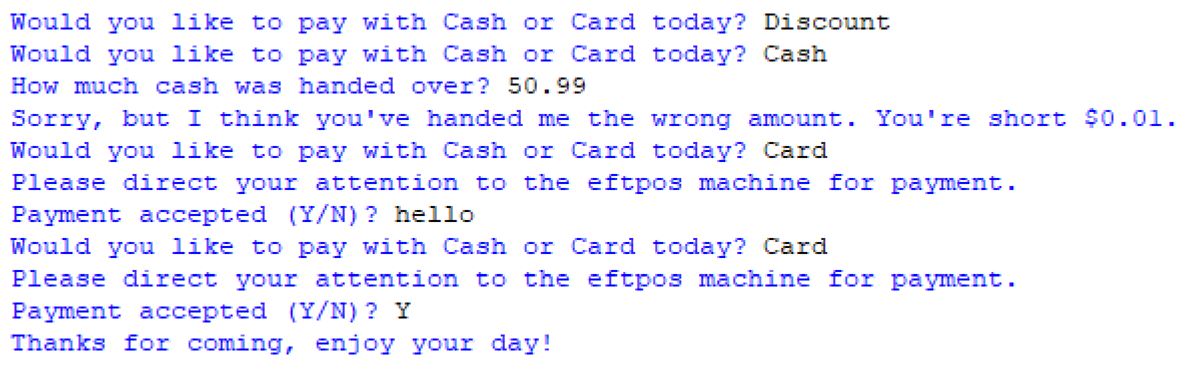
Tested cash payment system:

- Initially allowed you to submit any cash amount and would continue
- Added error handling so if there is less than the expected amount of cash, it will tell you how much money short you are and return to payment.

Tested receipt output:

- Wasn't giving the single item value as you would expect on a receipt when ordering multiple items
- Updated the receipt so it shows item quantity, item, single price, and then shows the total price of that item on a new line.

QA tester (asked my partner to use it and see if they could cause errors) showed that during the payment section at multiple stages you can input random words that aren't a valid selection and it returns to the start of the payment function without asking you to try a valid input.



```
Would you like to pay with Cash or Card today? Discount
Would you like to pay with Cash or Card today? Cash
How much cash was handed over? 50.99
Sorry, but I think you've handed me the wrong amount. You're short $0.01.
Would you like to pay with Cash or Card today? Card
Please direct your attention to the eftpos machine for payment.
Payment accepted (Y/N)? hello
Would you like to pay with Cash or Card today? Card
Please direct your attention to the eftpos machine for payment.
Payment accepted (Y/N)? Y
Thanks for coming, enjoy your day!
```

- Inserted error handling that will tell you the input is not valid and re-ask the question advising the input was not valid.

Step 7 – Refine with GenAI (Co-Pilot)

Prompt:

I have created a POS system in Python. The assumption is that this will be used by a staff member rather than the self-serve. Do you have any recommendations to the process flow or logic?

##All code copied and pasted underneath##

Output:

When asking Co-Pilot how it could improve my code, it initially suggested using classes instead of global variables. The reasoning it gave was to make the code more scalable. After further research, this isn't a bad idea in terms of reliability of the code, however at the time of writing I wasn't across the use of classes and the negatives of using global variables. If I was improving this code, I would take this feedback on.

The second suggestion provided was to number the menu items rather than relying on them being typed in, due to human error. This is something I already considered, however due to time constraints was not something I wanted to invest time into during the development due to my inexperience with Python.
In the same vain, quantity support was also suggested – Allowing people to order 2x of an item in one go – however, due to my inexperience I didn't want to utilise my time figuring this out.

Other suggestions were around cleaning up my logic to be more robust and allow for more clear exit conditions. These are fantastic suggestions from Co-Pilot and should have been something I thought about when designing the POS system.
Co-Pilot also provided further ideas for future proofing, saving orders externally (e.g. Keeping a database), and an order confirmation step in case any mistakes were made, or if customers had changed their minds.

Response generated using Microsoft Co-Pilot to Stephen James, 09 September 2025, https://copilot.microsoft.com/shares/ebfV6Y6XNHYTWnrxAnttq