Week 3

Ideas:

- **Online Banking System**: Students design a simplified online banking interface where users can check balances, transfer funds, and deposit/withdraw money. Through this, they must implement defensive measures against common issues like over-withdrawing, transferring negative amounts, or entering invalid account details. Errors should be handled gracefully with custom exceptions like "InsufficientFundsError" or "InvalidAmountError"
- **Library Management System**: Students create a system to manage book checkouts, returns, and reservations. They must guard against scenarios like checking out an already checked-out book, reserving a book that's not in the catalogue, or returning books that were never borrowed. Each of these should be managed with specific exception handling and tailored error messages.
- **E-commerce Checkout Process**: Using a mock e-commerce platform, students will develop a checkout process. This process must validate user input for shipping details, payment information, and product selection. Students must handle potential errors like expired credit cards, invalid shipping addresses, or attempting to purchase out-of-stock items using defensive programming techniques.

Defensive Programming: Online Banking System (SafeBank)

[Case Study Story] --> In the digital age, online banking has become the norm. SafeBank, a leading financial institution, is looking to revamp its online banking interface to ensure it's not only user-friendly but also robust against potential errors and misuse.

The new system should allow users to perform common banking operations like checking balances, transferring funds between accounts, and depositing or withdrawing money. However, with the convenience of online banking comes the responsibility of ensuring transactions are error-free and secure. For instance, a user shouldn't be able to transfer negative amounts or over-withdraw beyond their account balance.

Students are tasked with designing this new interface. They must implement defensive measures to handle common issues. If a user tries to withdraw more than their current balance, the system should raise an "InsufficientFundsError". Similarly, attempting to transfer a negative amount should trigger an

"InvalidAmountError". These custom exceptions, along with others, will ensure that errors are not only prevented but also communicated clearly to the user.

Topics to be consolidated

- Principles and importance of defensive programming in real-world applications.
- Recognizing and addressing various error types: user-induced, environmental, and logical.
- Implementing conditional statements for effective input validation.
- Exception handling in Python: managing unexpected errors gracefully.
- Designing custom exceptions for specific error scenarios.
- Debugging techniques and their applications in refining code.
- Professional development: crafting impactful LinkedIn profiles, CVs, and cover letters.