Programming 1 Formative Project 1 - **Budget Tracker** [50 marks]

Submission date: Sunday 6th Dec 2025

This project is a simple command-line Budget Tracker written in Python.
It allows a user to record income and expenses, list all transactions, apply filters, and view a summary of their financial activity.

It runs entirely in the terminal and stores all data in memory only (no files), as required by the assignment.
The project also uses Object-Oriented Programming with inheritance:

- Transaction (base class)
- Income(Transaction)
- Expense(Transaction)
- And a BudgetTracker class to manage records.

Features

Add Transactions	**✔ List Transactions**	**Filter Transactions**	**✔ Summary View**	**✔ Robustness**
• Add **income** or **expense**	• Displays all transactions in a clean, table-like format	You can filter by: • **Type →** income/expense	• Total income	• Simple error handling for invalid menu inputs
• Input: date, amount, category, description		• **Category →** e.g., food, salary, transport	• Total expenses	• Prevents empty or invalid values
• Validates numeric amounts and non-empty fields		• **Month →** e.g., `2025-10`	• Balance (income and expenses)	

Reflection

Working on this Budget Tracker project taught me how to apply Object-Oriented Programming in practice. I learned how classes and objects help organize code, and how inheritance can reduce repetition by allowing Income and Expense to share attributes under the Transaction base class. I also improved by breaking the program into smaller functions that each perform a single, clear task, such as adding transactions, filtering, and validating user input.

One of the main challenges I faced was structuring the program in a clean, logical way. At first, keeping the menu separate from the logic was confusing, but splitting the code into multiple files (models.py, tracker.py, main.py) made it easier to manage. Another challenge was input validation, especially ensuring that the user enters correct numeric values and correct formats. I had to test the menu many times to make sure the program did not crash when unexpected values were entered.

If I had more time and was not limited by the project scopes and had the required knowledge, I would improve the project by adding file I/O so that the user's transactions persist across sessions. I would also add more advanced features like sorting, charts, undoing the last transaction, and maybe even turning this into a GUI application using Tkinter or a web application using Flask. I would also like to add automated test cases to ensure the program stays reliable as it grows.

Overall, this project helped me understand OOP, user input handling, and program structure much better, and it gave me confidence in building larger Python applications.

Logic flow

1. Program starts
2. Menu appears
3. User chooses an action
4. The program calls functions inside BudgetTracker
5. Transactions stored in memory
6. User sees results
7. Loop continues
8. Ends on user exit

What I Learned

- How to use **Object-Oriented Programming** in Python
- How subclasses work through **inheritance**
- How to design functions and menu systems
- How to validate user input
- How to work with lists and objects together
- Using Git for version control with frequent commits

Challenges Faced

- Understanding how inheritance connects subclasses to the parent class
- Formatting the printed output to make the menu more readable
- Getting filter logic correct (especially month filtering)
- Organizing the program into clean functions