Programming 1 — Formative Project (Week 7)

Weighting: 50 % of module grade (formative)
Mode: Individual Submission
Language: Python 3

Project Overview

In this project, you'll build a **Budget Tracker** (runs on the command line) that allows users to record income and expenses, view and filter transactions, and display summaries within a single terminal session. This project specification will be available in the repository shared by the facilitator, and students will be required to clone this repository on their local machine.

Project Requirements

Your program must:

1. **Add Transactions**
   - Accept date, amount, category, description, and transaction type (income or expense).
   - Store the transactions in a list or dictionary.
2. **List Transactions**
   - Display all transactions in a clean, readable format.
3. **Filter Transactions**
   - Filter by type (income/expense), category, or month (e.g., `2025-10`).
4. **Summarize Budget**
   - Display total income, total expenses, balance, and per-category totals.
5. **Validate Input**
   - Handle invalid menu choices and amounts gracefully.
6. **Session Only**
   - All data remains in memory during execution. No saving or loading to files.

Technical Requirements

- **Strings & Conditionals:** For menu control, validation, and flow.
- **Loops:** Main program loop and transaction iteration.
- **Functions:** Break down features logically (e.g., *`add_income()`*, *`filter_transactions()`*, *`show_summary()`*).
- **Collections:** Use lists or dictionaries for transactions and summaries.

- **OOP:**
    - `Transaction` class (attributes: *`date`*, *`amount`*, *`category`*, *`description`*, *`type`*).
    - `BudgetTracker` class (methods for add/list/filter/summary).
- **Inheritance:** Implement `Income` and `Expense` as subclasses of `Transaction`.
- **Robustness:** Input validation (e.g., numeric amount, menu selection), graceful handling of empty datasets.

Sample Menu

```
1) Add income
2) Add expense
3) List transactions
4) Filter (by category / type / month)
5) Show summary
0) Exit
```

Deliverables

- **GitHub repository** with regular commits
- **Python source code** (`.py` files)
- **README.md** (max 2 pages) with:
    - Project overview & features
    - Instructions to run the program
    - Menu structure
    - Sample interactions
- **Screenshots** showing *add*, *`list`*, *`filter`*, *and* *`summary`*
- **Short reflection** (max 1 page):
    - What did you learn?
    - Challenges you faced?
    - How do you intend to improve it, given more time?

Suggested Class Skeleton

e.g class Transaction

```
class Transaction:
    def __init__(self, date, amount, category, description, ttype):
        self.date = date
        self.amount = float(amount)
        self.category = category.lower().strip()
        self.description = description
        self.type = ttype  # 'income' or 'expense'
```

Expectations

- Clean, user-friendly menu
- Program runs without errors or crashes
- Logical structure with functions and OOP
- Inheritance applied purposefully
- Proper Git history — no single final commit dump

Optional Features

- Budget threshold warnings
- Top spending categories
- Undo last transaction
- Basic test assertions

Assessment Breakdown (50 %)

Section	Description	Marks
1. Environment & Setup	Git environment ready, repo created & cloned, baseline commit, print test passed	8
2. Strings & Conditionals	Clean CLI, branching logic, input validation	8
3. Functions & Modularity	Logical code structure, reusable functions	8
4. Loops & Collections	Menu loop, correct storage, and iteration	8
5. Classes (OOP)	Transaction & BudgetTracker implemented correctly	9
6. Inheritance & Correctness	Inheritance is used meaningfully, program runs without crashing	9
TOTAL		**50**