## 1. Class: ExpenseTracker

This is the heart of the application where all the logic resides.

## Attributes

total\_budget: Stores the total budget the user enters at the start.

expenses: A list to hold all the added expenses as dictionaries (each dictionary contains date, description, amount, and category).

remaining\_money: Tracks how much money is left after deducting all expenses.

Methods

add\_expense(date, description, amount, category="General")

Purpose: Adds a new expense.

Steps:

Checks if the amount is greater than zero and less than or equal to the remaining money.

If valid, adds the expense to the expenses list and deducts the amount from remaining\_money.

remove\_expense(index)

Purpose: Removes an expense by its position in the list.

Steps:

Validates if the index is within the valid range.

If valid, removes the expense and refunds its amount back to remaining\_money.

view\_expenses()

Purpose: Displays all expenses in a clear, table-like format.

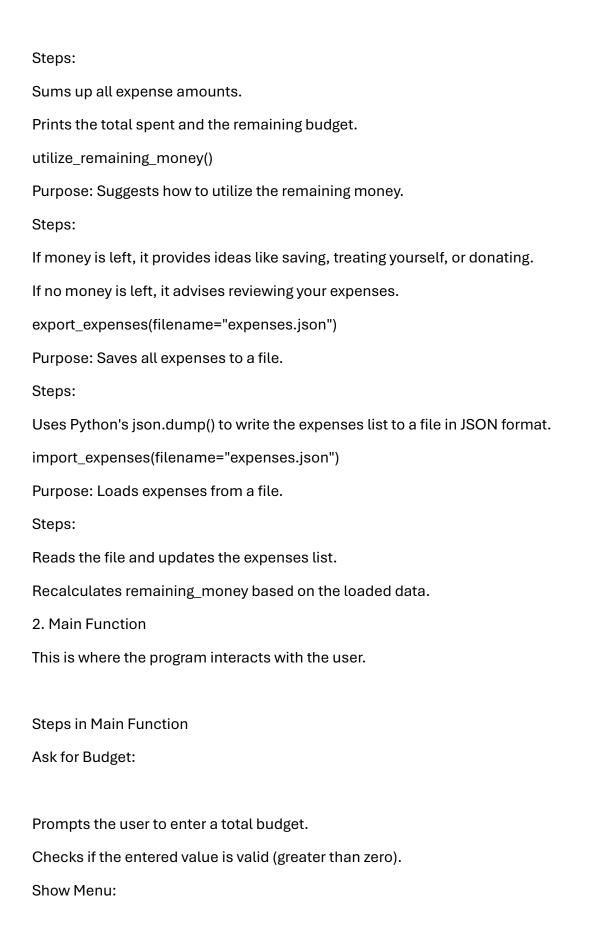
Steps:

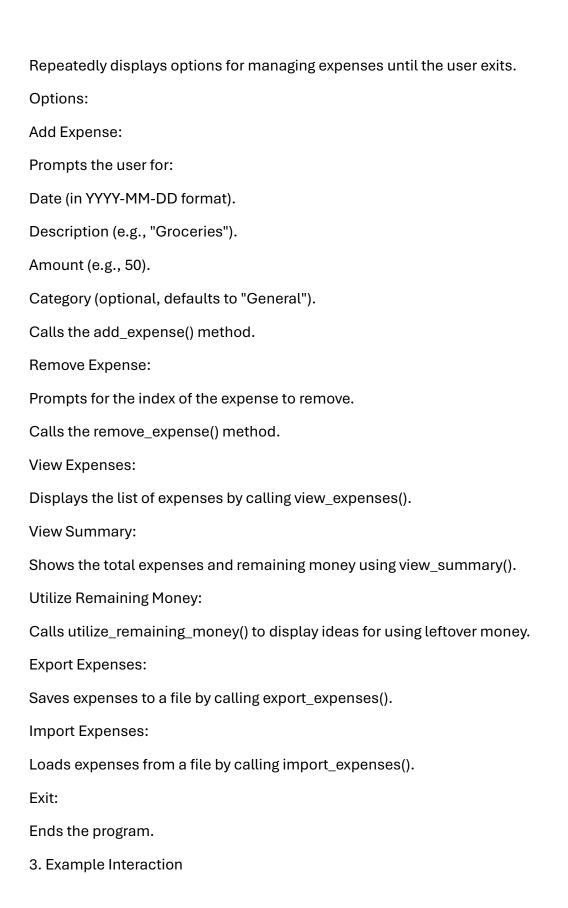
If there are no expenses, it prints a message.

Otherwise, it lists each expense with an index, date, description, category, and amount.

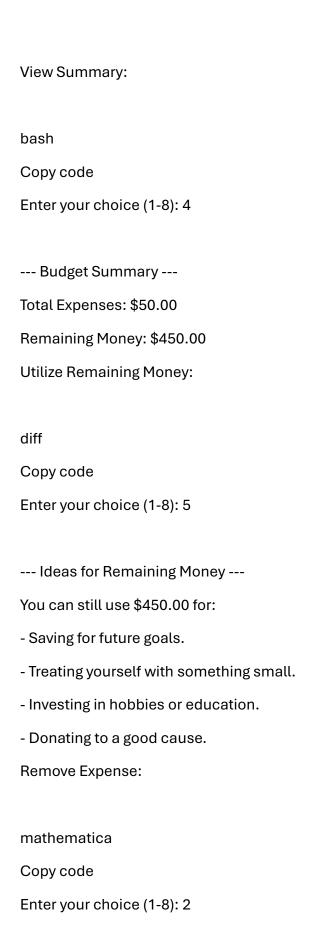
view\_summary()

Purpose: Shows the total amount spent and the remaining money.





Set Budget:
mathematica
Copy code
Enter your total budget: 500
Add Expense:
mathematica
Copy code
Menu:
1. Add Expense
Enter your choice (1-8): 1
Enter the date (YYYY-MM-DD): 2024-11-21
Enter the description: Groceries
Enter the amount: 50
Enter the category (default: General): Food
Expense added: Groceries (\$50.00).
View Expenses:
markdown
Copy code
Enter your choice (1-8): 3
Index Date Description Category Amount
1 2024-11-21 Groceries Food \$50.00



Enter the expense index to remove: 1
Removed expense: Groceries (\$50.00).
Exit:

Copy code

Enter your choice (1-8): 8

Goodbye!

Why It's Simpler?

The menu provides clear, straightforward options.

Each action is encapsulated in a function, making the code modular and easy to understand.

Error handling ensures invalid inputs (e.g., incorrect indices, invalid dates) are managed gracefully.

Suggestions for remaining money add practical value.

Let me know if you'd like further simplifications!