Personal Expense Tracker – Detailed Overview & Remarks

Introduction

Managing personal finances is essential for financial stability, and this **Personal Expense Tracker** provides a simple yet effective way to track and manage expenses. The script, written in Python, offers functionalities like **adding expenses, viewing recorded expenses, tracking budgets, and saving expense data in a CSV file**. It follows a structured approach to ensure ease of use, validation, and data persistence.

Code Breakdown and Explanation

1. Menu System

The script starts with the mainMenu() function, which prints a list of available options:

1. Add expense
2. View expenses
3. Track budget
4. Save expenses
5. Exit

The function menuConfirmation(value) confirms the user's choice before proceeding. If the user selects an option, a confirmation message appears, asking whether they want to continue or return to the main menu.

Key Features:

- Prevents accidental selection of options.
- Provides a structured navigation system.
- Ensures the user can confirm their action before proceeding.

---

2. Validating User Input

When adding expenses, the script verifies inputs using:

- **Date Validation (is_valid_date)**
    - Ensures the date is in YYYY-MM-DD format.
    - Uses regex to check that the date is structured correctly.
    - Prevents incorrect or invalid dates from being entered.
- **Expense Validation (validate_expense)**
    - Checks if the category is empty.
    - Ensures that the amount entered is **greater than zero**.
    - Prevents users from adding invalid data.

This approach ensures **data integrity** and **prevents errors** from occurring due to incorrect inputs.

---

3. Adding Expenses

The function add_expense() collects expense details from the user:

- **Date** (YYYY-MM-DD format)
- **Category** (type of expense)
- **Amount** (expense value, converted to float)
- **Note** (optional description)

If validation is successful, the expense is temporarily stored in the draft_expense list using the add_expense_to_list() function. Users are then prompted if they want to **add another expense** or return to the main menu.

Key Features:

- Prevents accidental or incorrect entries.
- Uses a **draft system** before saving expenses permanently.
- Allows users to add multiple expenses in a single session.

---

4. Viewing Expenses

The view_expense() function displays saved expenses in a **well-formatted tabular view**. It reads data from expense.csv, splits the contents into columns, and adjusts column widths using the append_remaining_space() function.

Table Format:

sql

CopyEdit

```
-------------------------------------------------------------------------------
   DATE    |      CATEGORY       |   AMOUNT   |         NOTE       |
---------------|-------------------------|--------------|-------------------------|
 2024-03-06   | FOOD                |  12.50     | Lunch at restaurant    |
 2024-03-07   | TRANSPORT           |   5.00     | Bus fare               |
-------------------------------------------------------------------------------
```

Key Features:

- Displays expense data in a structured format.

- Automatically aligns text for better readability.
- Prevents data misalignment while displaying different-length entries.

---

5. Tracking Budget

The function track_budget() allows users to **set a budget** and checks if their total expenses exceed the limit.

Steps:

1. User enters a budget amount.
2. The program reads expense.csv and **calculates total expenses**.
3. Compares total expenses with the **budget**.
4. Displays one of the following:
    - "You have exceeded your budget!" (if expenses are greater than budget)
    - "You have X amount left for the month" (if within budget)

Key Features:

- Ensures users remain financially aware.
- Provides **instant feedback** on spending habits.
- Helps users track expenses and maintain financial discipline.

---

6. Saving Expenses

The function save_expense() allows users to **review draft expenses before saving them permanently**. The user is prompted to confirm before saving.

Steps:

1. Displays draft expenses.
2. Asks for confirmation:
    - If **Yes (Y)** → Expenses are saved in expense.csv, and the draft list is cleared.
    - If **No (N)** → The draft list is discarded.

The function persist_expense() writes data from draft_expense into expense.csv and clears the draft.

Key Features:

- Prevents accidental data loss.
- Ensures users review expenses before saving them permanently.
- Allows users to discard unwanted entries.

---

7. File Handling and Persistence

To ensure expense data is **saved and available for future use**, the script uses file handling.

- **File Existence Check**

python

CopyEdit

```
try:
  open('expense.csv', "r")
except FileNotFoundError:
  expense_db = open('expense.csv', 'w')
  expense_db.close()
```

    - This checks if expense.csv exists.
    - If not, it creates an empty file to store future expense data.

- **Appending Data to File**

python

CopyEdit

```
expense_db = open('expense.csv', 'a')
for i in draft_expense:
  expense_db.write(i['DATE'] + ',' + i['CATEGORY'] + ',' + str(i['AMOUNT']) + ',' + i['NOTE'] + '\n')
expense_db.close()
```

    - Saves expenses **without overwriting existing data**.
    - Uses CSV format for easy data retrieval.

Key Features:

- **Persistent storage** ensures expenses are saved even after restarting the script.
- Avoids data loss by storing records permanently in a file.

Strengths and Advantages

1. **User-Friendly Navigation**
    - The menu-driven interface makes it **easy to use**, even for non-technical users.
2. **Validations for Data Integrity**
    - Prevents users from entering incorrect or incomplete data.
3. **Budget Tracking Feature**
    - Helps users **stay within their budget** and **avoid overspending**.
4. **Draft System for Expense Management**
    - Users can **review and confirm** before saving expenses permanently.
5. **File Handling for Data Persistence**
    - Ensures expense records are stored **even after the program exits**.
6. **Expandable & Customizable**
    - The code is **structured** to allow for future enhancements, such as:
        - Graphical user interface (GUI).
        - Database integration for better performance.
        - Expense categorization and analytics.

Potential Improvements

1. **Use of csv Module**
    - Instead of manually handling CSV data, Python's csv module could be used:

python

CopyEdit

import csv

with open('expense.csv', 'a', newline='') as file:

writer = csv.writer(file)

writer.writerow([date, category, amount, note])

- This improves **data security and handling**.

2. **Use of pathlib for File Handling**
    - pathlib provides a more modern approach to checking and handling files.
3. **Error Handling Improvements**
    - Currently, user input is assumed to be correct (e.g., integers for menu selection).
    - Input validation can be improved using **try-except blocks**.
4. **Graphical User Interface (GUI)**
    - A GUI-based version (using Tkinter, PyQt, or Flask) could make it even more user-friendly.

Conclusion

The **Personal Expense Tracker** is a well-structured Python script that helps users manage their expenses efficiently. By providing functionalities such as **adding expenses, viewing them, tracking budgets, and saving records**, this script serves as a simple yet effective tool for **financial management**.

With further enhancements like **database integration, a GUI, and better input validation**, it could become an even more powerful application. **For now, it remains a great command-line solution for tracking personal expenses and maintaining a budget.**