source_code for personal expense tracker

```
import csv
from datetime import datetime

# Global list to store expenses
expenses = []

# Load expenses from CSV file
def load_expenses_from_file(filename="expenses.csv"):
    try:
        with open(filename, mode='r') as file:
            reader = csv.DictReader(file)
            for row in reader:
                # Validate and load only complete and correct rows
                if validate_expense(row):
                    row['amount'] = float(row['amount'])  # Convert amount to float
                    expenses.append(row)
                else:
                    print(f"Skipping incomplete or invalid row: {row}")
    except FileNotFoundError:
        print(f"No existing file found. Starting fresh!")

# Save expenses to CSV file
def save_expenses_to_file(filename="expenses.csv"):
    with open(filename, mode='w', newline='') as file:
        fieldnames = ['date', 'category', 'amount', 'description']
        writer = csv.DictWriter(file, fieldnames=fieldnames)
        writer.writeheader()
        writer.writerows(expenses)
    print(f"Expenses saved to {filename}.")

# Validate expense data (ensure date format, non-empty fields, etc.)
def validate_expense(expense):
    try:
        # Validate date format
        datetime.strptime(expense['date'], "%Y-%m-%d")
        # Ensure category, amount, and description are non-empty
        if not expense['category'] or not expense['description']:
            return False
        if float(expense['amount']) < 0:
            return False
        return True
    except (ValueError, KeyError):
```

```
        return False

# Add an expense
def add_expense():
    date = input("Enter the date (YYYY-MM-DD): ")
    try:
        # Validate the date format
        datetime.strptime(date, "%Y-%m-%d")
    except ValueError:
        print("Invalid date format. Please use YYYY-MM-DD.")
        return

    category = input("Enter the category: ")
    try:
        amount = float(input("Enter the amount: "))
    except ValueError:
        print("Invalid amount. Please enter a numeric value.")
        return
    description = input("Enter a brief description: ")

    # Create an expense dictionary
    expense = {
        'date': date,
        'category': category,
        'amount': amount,
        'description': description
    }

    # Validate the new expense before adding it
    if validate_expense(expense):
        expenses.append(expense)
        print("Expense added successfully!")
    else:
        print("Failed to add expense due to invalid data.")

# View all expenses with validation
def view_expenses():
    if not expenses:
        print("No expenses to show.")
        return

    for i, expense in enumerate(expenses, start=1):
        print(f"{i}. {expense['date']} - {expense['category']} - ${expense['amount']:.2f} -
{expense['description']}")
```

```
# Set and track monthly budget
def track_budget():
    try:
        budget = float(input("Enter your monthly budget: "))
    except ValueError:
        print("Invalid budget. Please enter a numeric value.")
        return

    total_expenses = sum(expense['amount'] for expense in expenses)

    if total_expenses > budget:
        print(f"You have exceeded your budget! Total expenses: ${total_expenses:.2f}")
    else:
        remaining_budget = budget - total_expenses
        print(f"You have ${remaining_budget:.2f} left for the month.")

# Main interactive menu
def display_menu():
    while True:
        print("\n--- Personal Expense Tracker Menu ---")
        print("1. Add Expense")
        print("2. View Expenses")
        print("3. Track Budget")
        print("4. Save Expenses")
        print("5. Exit")

        choice = input("Enter your choice (1-5): ")

        if choice == '1':
            add_expense()
        elif choice == '2':
            view_expenses()
        elif choice == '3':
            track_budget()
        elif choice == '4':
            save_expenses_to_file()
        elif choice == '5':
            save_expenses_to_file()  # Automatically save on exit
            print("Exiting program.")
            break
        else:
            print("Invalid choice. Please choose between 1-5.")
```

```
# Load expenses from file when program starts
load_expenses_from_file()

# Display the menu to start interaction
display_menu()
```