**Course-End Project: Personal Expense Tracker**

Steps to perform:

1. Add an expense:

• Create a function to prompt the user for expense details. Ensure you ask for:

* the date of the expense in the format YYYY-MM-DD
* the category of the expense, such as Food or Travel
* the amount spent
* A brief description of the expense
* Store the expense in a list as a dictionary, where each dictionary includes the date, category, amount, and description as key-value pairs

**Example: {'date': '2024-09-18', 'category': 'Food', 'amount': 15.50, 'description': 'Lunch with friends'}**

****

1. View expenses:

• Write a function to retrieve and display all stored expenses

* Ensure the function loops through the list of expenses and displays the date, category, amount, and description for each entry

• Validate the data before displaying it

* If any required details (date, category, amount, or description) are missing, skip the entry or notify the user that it’s incomplete 

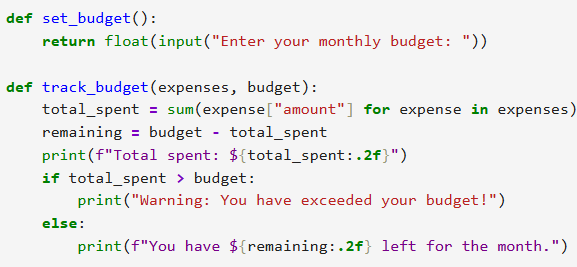
1. Set and track the budget:

• Create a function that allows the user to input a monthly budget. Prompt the user to:

* Enter the total amount they want to budget for the month

• Create another function that calculates the total expenses recorded so far

* Compare the total with the user’s monthly budget
* If the total expenses exceed the budget, display a warning (Example: You have exceeded your budget!)
* If the expenses are within the budget, display the remaining balance (Example: You have 150 left for the month)



1. Save and load expenses:

• Implement a function to save all expenses to a CSV file, with each row containing the date, category, amount, and description of each expense

• Create another function to load expenses from the CSV file. When the program starts, it should:

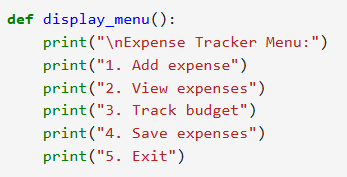
* Read the saved data from the file
* Load it back into the list of expenses so the user can see their previous expenses and continue from where they left off



1. Create an interactive menu:

• Build a function to display a menu with the following options:

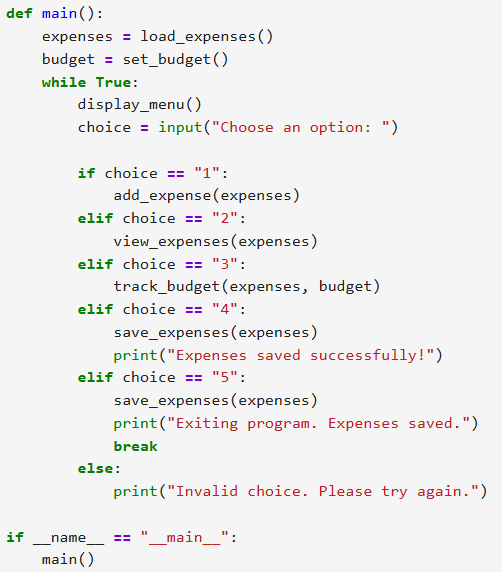
* Add expense
* View expenses
* Track budget
* Save expenses
* Exit



• Allow the user to enter a number to choose an option

• Implement the following conditions:

* If the user selects option 1, call the function to add an expense
* If the user selects option 2, call the function to view expenses
* If the user selects option 3, call the function to track the budget
* If the user selects option 4, call the function to save expenses to the file
* If the user selects option 5, save the expenses and exit the program



Output --:

