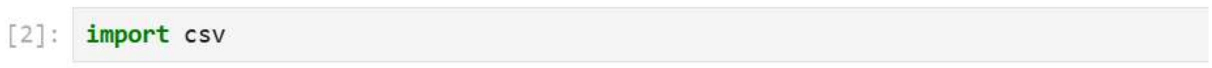
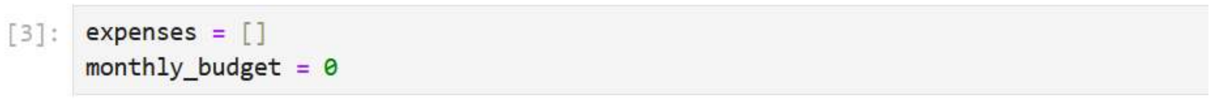
Problem Statement:

In today's fast-paced world, individuals need to track and manage their expenses effectively. You are tasked with building a Personal Expense Tracker that allows users to log daily expenses, categorize them, and track spending against a set monthly budget. The tracker should also have the ability to save and load expenses from a file for future reference.

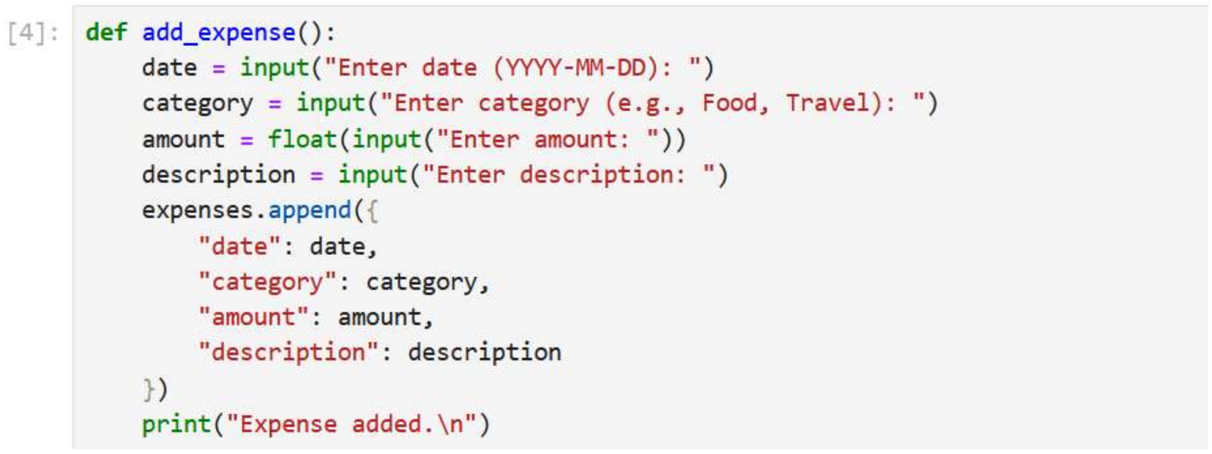
1. Add an Expenses: Design a function that allows users to add new expenses by entering the date, category (For example, Food, Travel), amount, and a brief description. These values are then stored in a list of dictionaries, where each dictionary represents one expense.



```
[2]: import csv
```

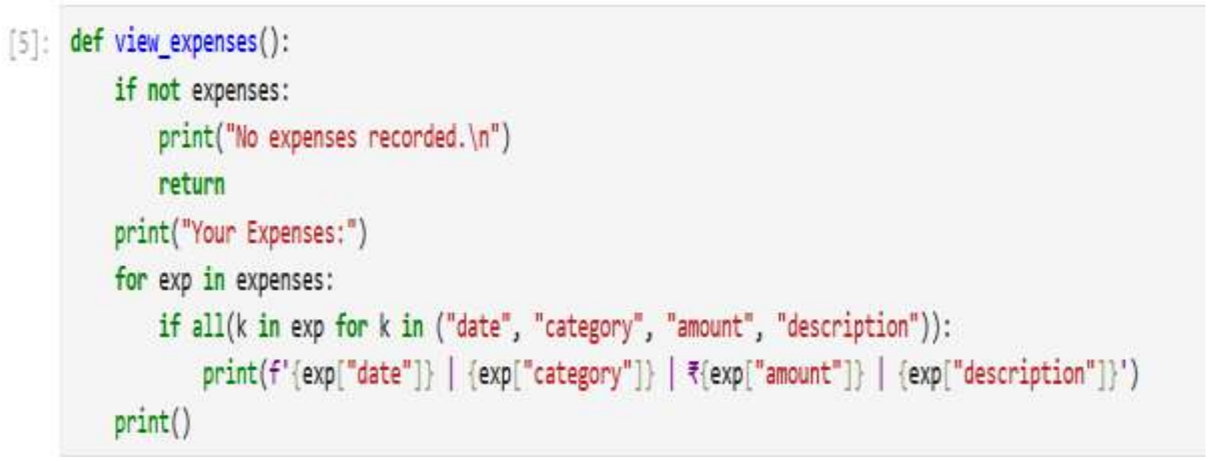


```
[3]: expenses = []
     monthly_budget = 0
```



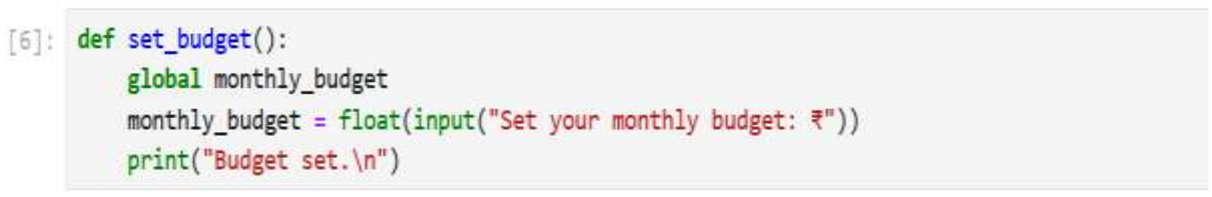
```
[4]: def add_expense():
         date = input("Enter date (YYYY-MM-DD): ")
         category = input("Enter category (e.g., Food, Travel): ")
         amount = float(input("Enter amount: "))
         description = input("Enter description: ")
         expenses.append({
             "date": date,
             "category": category,
             "amount": amount,
             "description": description
         })
         print("Expense added.\n")
```

2. View Expenses: Implement a function that displays all recorded expenses. The function ensures that all required fields (date, category, amount, description) are present before displaying the expense. If no expenses are recorded, it informs the user.

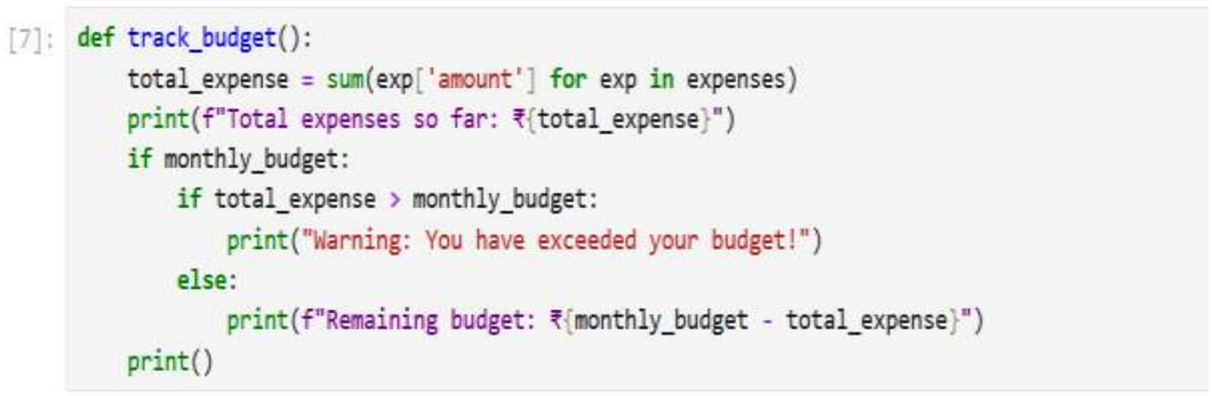


```
[5]: def view_expenses():
         if not expenses:
             print("No expenses recorded.\n")
             return
         print("Your Expenses:")
         for exp in expenses:
             if all(k in exp for k in ("date", "category", "amount", "description")):
                 print(f'{exp["date"]} | {exp["category"]} | ₹{exp["amount"]} | {exp["description"]}')
         print()
```

3. Set and Track Budget: Allow the user to set a monthly budget, and then calculate total expenses to compare against the budget. The track budget function alerts the user if the total expenses exceed the budget, and informs the remaining budget if still within the limit.

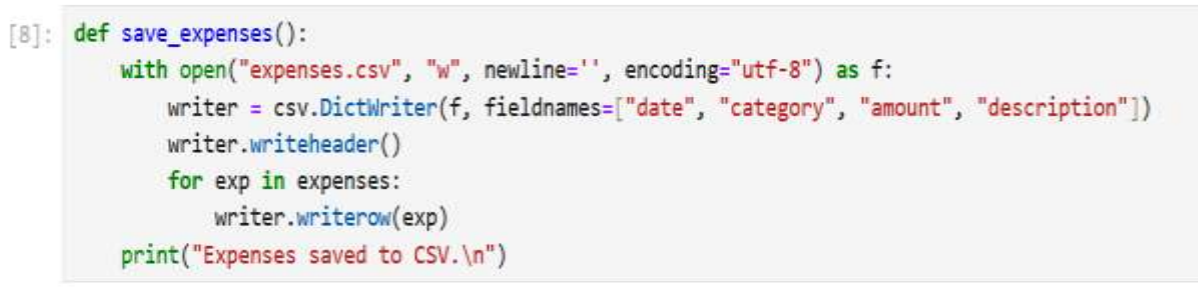


```
[6]: def set_budget():
         global monthly_budget
         monthly_budget = float(input("Set your monthly budget: ₹"))
         print("Budget set.\n")
```

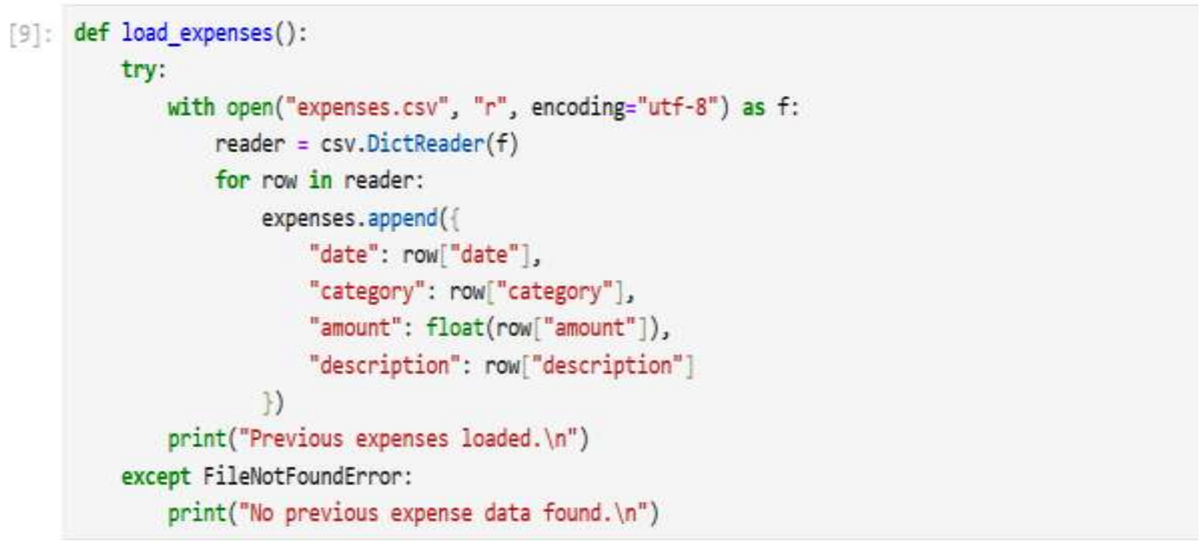


```
[7]: def track_budget():
         total_expense = sum(exp['amount'] for exp in expenses)
         print(f"Total expenses so far: ₹{total_expense}")
         if monthly_budget:
             if total_expense > monthly_budget:
                 print("Warning: You have exceeded your budget!")
             else:
                 print(f"Remaining budget: ₹{monthly_budget - total_expense}")
         print()
```

4. Save and Load Expenses → Save: Save all the recorded expenses into a CSV file using the **save_expenses** function. The file includes the date, category, amount, and description of each expense. → Load: Load expenses from the CSV file when the program starts using the **load_expenses** function. If no file is found, the user is informed that no prior data exists.

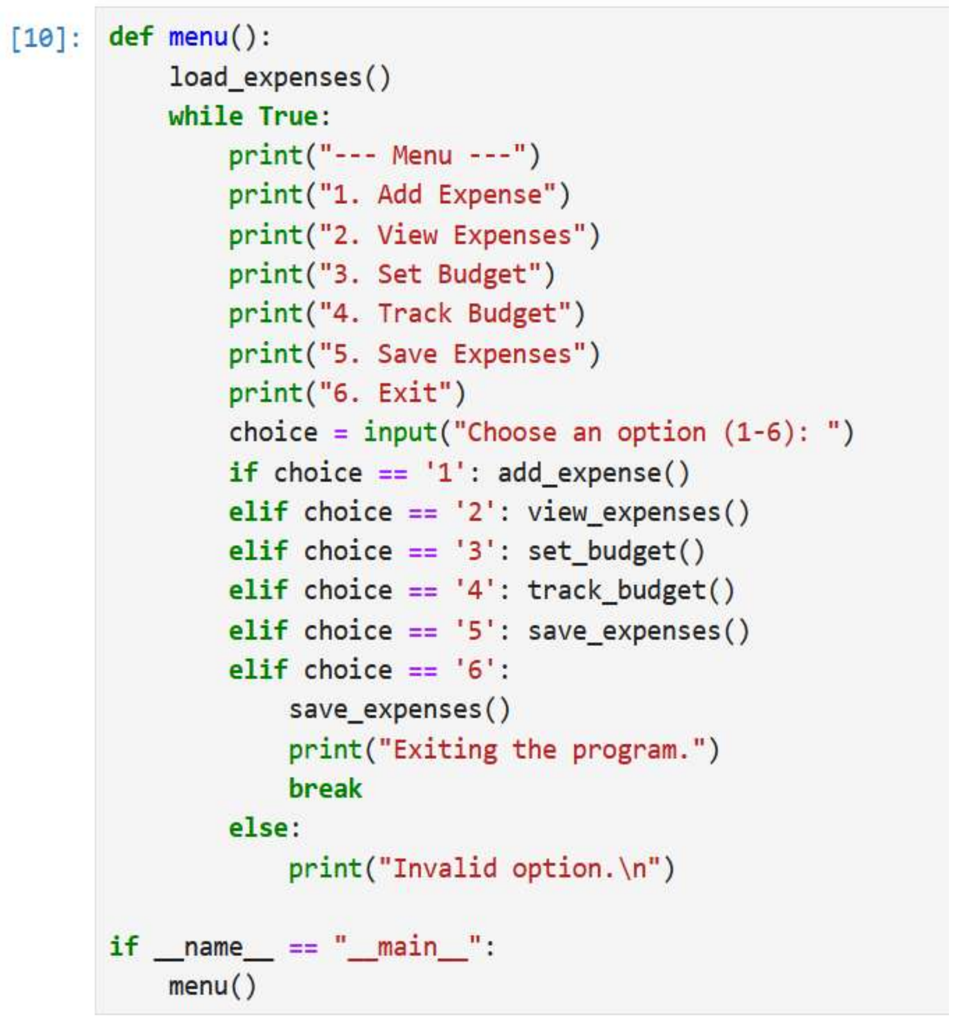


```
[8]: def save_expenses():
         with open("expenses.csv", "w", newline='', encoding="utf-8") as f:
             writer = csv.DictWriter(f, fieldnames=["date", "category", "amount", "description"])
             writer.writeheader()
             for exp in expenses:
                 writer.writerow(exp)
         print("Expenses saved to CSV.\n")
```



```
[9]: def load_expenses():
         try:
             with open("expenses.csv", "r", encoding="utf-8") as f:
                 reader = csv.DictReader(f)
                 for row in reader:
                     expenses.append({
                         "date": row["date"],
                         "category": row["category"],
                         "amount": float(row["amount"]),
                         "description": row["description"]
                     })
             print("Previous expenses loaded.\n")
         except FileNotFoundError:
             print("No previous expense data found.\n")
```

5. Create an Interactive Menu The interactive menu allows users to navigate through the options of adding an expense, viewing expenses, tracking their budget, saving expenses, or exiting the program. When exiting, it ensures that any newly added expenses are saved to the file.



```
[10]: def menu():
          load_expenses()
          while True:
              print("--- Menu ---")
              print("1. Add Expense")
              print("2. View Expenses")
              print("3. Set Budget")
              print("4. Track Budget")
              print("5. Save Expenses")
              print("6. Exit")
              choice = input("Choose an option (1-6): ")
              if choice == '1': add_expense()
              elif choice == '2': view_expenses()
              elif choice == '3': set_budget()
              elif choice == '4': track_budget()
              elif choice == '5': save_expenses()
              elif choice == '6':
                  save_expenses()
                  print("Exiting the program.")
                  break
              else:
                  print("Invalid option.\n")

      if __name__ == "__main__":
          menu()
```

Output

```
Previous expenses loaded.

--- Menu ---
1. Add Expense
2. View Expenses
3. Set Budget
4. Track Budget
5. Save Expenses
6. Exit
Choose an option (1-6):  1
Enter date (YYYY-MM-DD):  2025-09-29
Enter category (e.g., Food, Travel):  entertainment
Enter amount:  350
Enter description:  navaratri pass
Expense added.

--- Menu ---
```

```
1. Add Expense
2. View Expenses
3. Set Budget
4. Track Budget
5. Save Expenses
6. Exit
Choose an option (1-6):  2
Your Expenses:
2025-09-29 | entertainment | ₹350.0 | navaratri pass

--- Menu ---
1. Add Expense
2. View Expenses
3. Set Budget
4. Track Budget
5. Save Expenses
6. Exit
Choose an option (1-6):  3
Set your monthly budget: ₹ 26000
Budget set.

--- Menu ---
1. Add Expense
2. View Expenses
3. Set Budget
4. Track Budget
5. Save Expenses
6. Exit
Choose an option (1-6):  4
Total expenses so far: ₹350.0
Remaining budget: ₹25650.0


--- Menu ---
1. Add Expense
2. View Expenses
3. Set Budget
4. Track Budget
5. Save Expenses
6. Exit
Choose an option (1-6):  5
Expenses saved to CSV.

--- Menu ---
1. Add Expense
2. View Expenses
3. Set Budget
4. Track Budget
5. Save Expenses
6. Exit
Choose an option (1-6):  6
```

```
Expenses saved to CSV.

Exiting the program.
```