**Synopsis**

| * Introduction * Methodology used * Software requirements * Program code * Output screen shot * Future outcomes * Conclusion |
| --- |

* **Introduction**

Expense tracking is an essential aspect of personal finance management, allowing individuals to monitor their spending habits, budget effectively, and save money. This Python program provides a command-line interface to manage expenses by adding, viewing, and removing expense records. It also saves the recorded expenses to a CSV file for later reference.

* **Methodology used**

The methodology for developing the Expenses Tracker involves several key steps:

1. **Data Modeling:**

The program uses the @dataclass decorator to define an Expense class that encapsulates details about each expense, such as amount, category, and description.

1. **User Interaction:**

Command-line inputs facilitate interaction with the user for adding, removing, and viewing expenses.

1. **Persistence:**

Expenses are saved in a CSV file to ensure data is stored even after the program ends.

1. **Error Handling:**

Basic input validation ensures robustness, such as handling invalid indices or non-numeric input for amounts.

1. **Iterative Actions:**

A loop enables users to perform multiple actions (add/remove/view/exit) until they choose to exit the program.

* **Software requirements**

To run the Expenses Tracker application, the following software requirements must be met:

* **Operating System** :Windows, macOS, or Linux.

Python Version:

Python 3.6 or higher.

* **Libraries Used**

- dataclasses: For defining structured Expense objects.

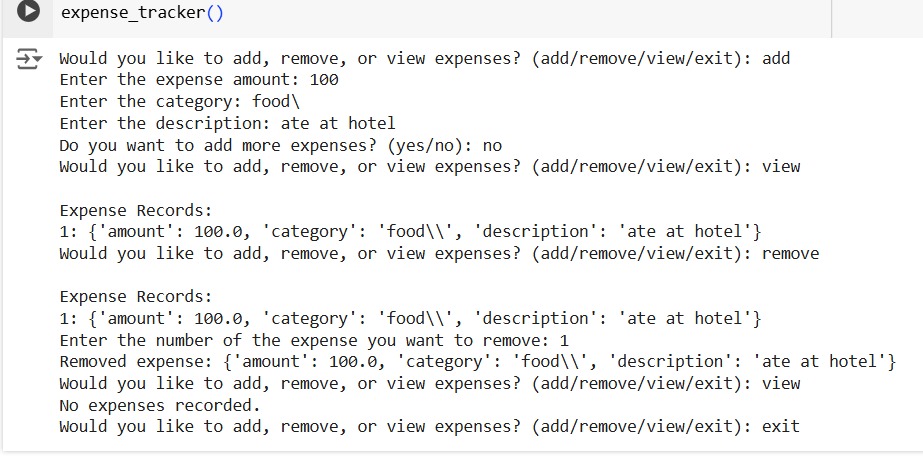
- csv: For writing and saving expense records to a file.

- typing: For specifying type annotations, e.g., List.

* **Program code**

| **from dataclasses import dataclass, asdict**  **import csv**  **from typing import List**  **@dataclass**  **class Expense:**  **amount: float**  **category: str**  **description: str**  **def save\_expenses(expenses: List[Expense], filename: str = "expenses.csv"):**  **with open(filename, mode='w', newline='') as file:**  **writer = csv.writer(file)**  **writer.writerow(['Amount', 'Category', 'Description']) # Header**  **for expense in expenses:**  **writer.writerow([expense.amount, expense.category, expense.description])**  **def display\_expenses(expenses: List[Expense]):**  **print("\nExpense Records:")**  **for index, record in enumerate(expenses):**  **print(f"{index + 1}: {asdict(record)}") # Print as dictionary for better readability**  **def remove\_expense(expenses: List[Expense]):**  **display\_expenses(expenses)**  **try:**  **index = int(input("Enter the number of the expense you want to remove: ")) - 1**  **if 0 <= index < len(expenses):**  **removed\_expense = expenses.pop(index)**  **print(f"Removed expense: {asdict(removed\_expense)}")**  **else:**  **print("Invalid index. No expense removed.")**  **except ValueError:**  **print("Please enter a valid number.")**  **def expense\_tracker():**  **expenses: List[Expense] = []**  **while True:**  **action = input("Would you like to add, remove, or view expenses? (add/remove/view/exit): ").lower()**    **if action == "add":**  **while True:**  **try:**  **amount = float(input("Enter the expense amount: ")) # Convert to float for validation**  **except ValueError:**  **print("Please enter a valid number for the amount.")**  **continue # Skip to the next iteration if input is invalid**  **category = input("Enter the category: ")**  **description = input("Enter the description: ")**  **expenses.append(Expense(amount, category, description))**    **add\_more = input("Do you want to add more expenses? (yes/no): ").lower()**  **if add\_more == "no":**  **break**  **elif action == "remove":**  **if expenses:**  **remove\_expense(expenses)**  **else:**  **print("No expenses to remove.")**  **elif action == "view":**  **if expenses:**  **display\_expenses(expenses)**  **else:**  **print("No expenses recorded.")**  **elif action == "exit":**  **break**  **else:**  **print("Invalid action. Please choose add, remove, view, or exit.")**  **save\_expenses(expenses) # Save expenses to CSV before exiting**  **expense\_tracker()** |
| --- |

* **Output screen shot**



* **Future Outcomes**

The Expenses Tracker project has the potential for several future enhancements and outcomes:

* **Graphical User Interface:**

Develop a GUI version using frameworks like Tkinter or PyQt for better usability.

* **Database Integration:**

Store expenses in a database for more extensive and secure data handling.

* **Analytics Features:**

Introduce graphs and reports to analyze spending trends over time

* **Mobile App Version:**

Extend the functionality to mobile platforms for ease of access.

* **Conclusion**

This expense tracker serves as a foundational tool for managing personal finances. With its simple design and CSV-based data storage, it enables users to record, review, and remove expenses effectively. Future enhancements could significantly improve usability, scalability, and user experience.