UNIVERSITY OF WESTMINSTER

INFORMATICS INSTITUTE OF TECHNOLOGY

University of Westminster

Software Development I 4COSC006C

Test Plan

Name:	**Nirukashika Sewwandi Wijesiri**
Module:	**4COSC006C Programming**
Type of Assignment:	**Coursework**
UOW ID:	**w2082004**
IIT ID:	**20232818**

1. Abstract

This Python code is a command-line-based personal money tracker. The application enables users to carry out CRUD (Create, Read, Update, Delete) operations to handle people's financial transactions. Data permanence is achieved using JSON serialization, and the transactions are kept in a dictionary .

Adding a transaction, viewing all transactions, changing a transaction, removing a transaction, computing the summary of revenue and expenses, saving transactions to a JSON file, and loading transactions from a JSON file are just a few of the functions that the code can perform.

2. Acknowledgement

I would like to extend my heartfelt appreciation to our lecturer Mr. Guhanathan Poravi and our tutorial lecturer whose unwavering support and guidance helped me complete this assignment with excellence.

I am also grateful for the prompt response to my queries and his guidance and support was invaluable and greatly appreciated. In addition, I would like to express my gratitude to my friends for their constant encouragement and support in completing this assignment

Table of Contents

1) Personal Finance Tracker (Using Dictionaries)

Overview

This assignment involves developing a Personal Finance Tracker using Python, focusing on fundamental programming principles such as dictionaries, loops, functions, input/output, and input validation. Unlike the original specification, which utilized lists, your application will now use dictionaries to manage financial transactions, with keys representing the type of expenses and values being lists of expenses themselves. This change aims to enhance your understanding of data handling, program design, and testing in a practical context, with an added focus on file I/O for bulk data processing.

Objectives

- Apply essential Python programming constructs.

- Manage data using dictionary manipulations for more efficient data retrieval.

- Implement CRUD operations on financial transactions, leveraging JSON serialization for data storage.

- Introduce file operations for bulk reading of transactions from a text file, enhancing the program's usability for larger data sets.

- Conduct thorough testing and validation to ensure program robustness.

2)Test Plan

Test Case	Input	Expected Output	Actual Output	Pass or Fail
1. Adding a Transaction	• Category: "salary" • Amount: 500000 • Date: "2024-04-14"	Message: "Transaction added successfully."	Transaction added successfully.	Pass
2. Viewing Transactions	(Assuming transactions are added before)	A formatted JSON representation of all transactions	[Output of JSON representation]	Pass
3.Updating a Transaction	• Category to update: "salary" • Transaction number: 1 • New amount: 6000 • New date: "2024-04-13"	Message: "Transaction updated successfully."	Transaction updated successfully.	Pass
4.Deleting a Transaction	• Category to delete: "salary" • Transaction number: 1	Message: "Transaction deleted successfully."	Transaction deleted successfully.	Pass
5.Displaying Summary	(Assuming transactions are added before)	Summary of each category including total amount and count	[Output of summary]	Pass

6.Reading Bulk Transactions from File	Filename of a JSON file containing transactions	Loading transactions from the file	Transactions loaded successfully.	Pass
7.Saving Transactions to File	(Assuming transactions are added before)	Transactions saved to the file	Transactions saved successfully.	Pass
8.Exiting the Program	Choosing the option to exit	Exiting message is displayed	Exiting...	Pass

Error Handling

1.File Not Found	Entering a non-existing file name during file loading or bulk transaction reading.	Error message: "File not found."	File not found.	Pass
2.JSON Decoding Error	Introducing a syntax error in the JSON file or providing a non-JSON file during loading.	Error message: "Error decoding JSON: [error message]"	Error decoding JSON: [error message]	Pass

3)Test Summary

Test Summary for Personal Finance Tracker

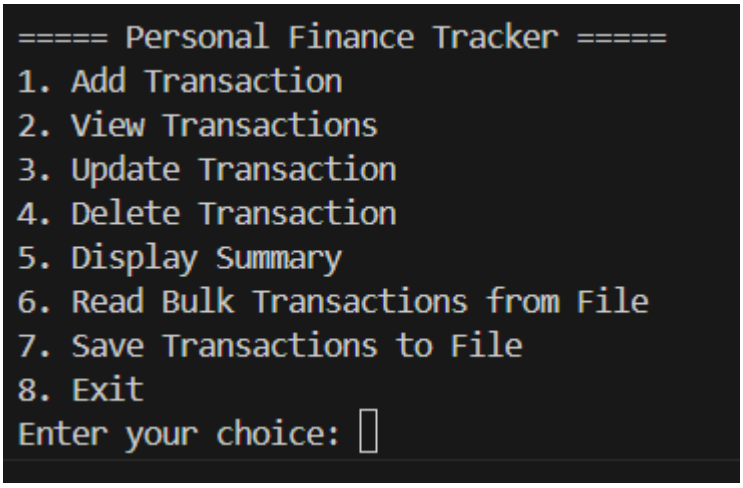
- Total Test Cases: 10
- Pass: 10
- Fail: 0
- Pass Rate: 100%

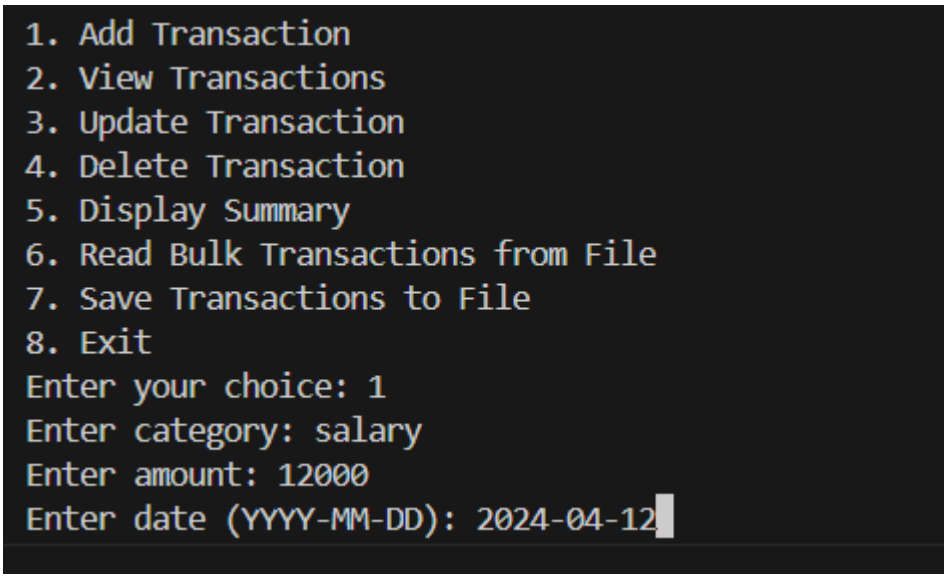
Summary:

All test cases passed successfully without any failures. The Personal Finance Tracker program demonstrated correct behavior in handling transactions, displaying summaries, saving and loading transactions from files, and gracefully handling potential errors such as file not found and JSON decoding errors. Therefore, the program is considered to be functioning as expected and ready for deployment.

4) Screenshots of Test Cases

1.Adding a Transaction





```
===== Personal Finance Tracker =====
1. Add Transaction
2. View Transactions
3. Update Transaction
4. Delete Transaction
5. Display Summary
6. Read Bulk Transactions from File
7. Save Transactions to File
8. Exit
Enter your choice: 1
Enter category: salary
Enter amount: 12000
Enter date (YYYY-MM-DD): 2024-04-12
Transaction added successfully.
```

2.view Transaction

```
===== Personal Finance Tracker =====
1. Add Transaction
2. View Transactions
3. Update Transaction
4. Delete Transaction
5. Display Summary
6. Read Bulk Transactions from File
7. Save Transactions to File
8. Exit
Enter your choice: 2
```

```
{
  "salary": [
    {
      "amount": 69230.0,
      "date": "2024-04-17"
    },
    {
      "amount": 70000.0,
      "date": "2024-04-12"
    },
    {
      "amount": 8000.0,
      "date": "2024-04-10"
    },
    {
      "amount": 12000.0,
      "date": "2024-04-12"
    }
  ],
  "groceries": [
    {
      "amount": 4500.0,
      "date": "2024-01-13"
    }
  ],
```

3)update Transaction

```
===== Personal Finance Tracker =====
1. Add Transaction
2. View Transactions
3. Update Transaction
4. Delete Transaction
5. Display Summary
6. Read Bulk Transactions from File
7. Save Transactions to File
8. Exit
Enter your choice: 3
Enter category to update: salary
Current transactions under salary:
1. Amount: 69230.0, Date: 2024-04-17
2. Amount: 70000.0, Date: 2024-04-12
3. Amount: 8000.0, Date: 2024-04-10
4. Amount: 12000.0, Date: 2024-04-12
Enter transaction number to update: 1
Enter new amount: 1200
Enter new date (YYYY-MM-DD): 2024-01-12
Transaction updated successfully.
```

4)Delete Transaction

```
===== Personal Finance Tracker =====
1. Add Transaction
2. View Transactions
3. Update Transaction
4. Delete Transaction
5. Display Summary
6. Read Bulk Transactions from File
7. Save Transactions to File
8. Exit
Enter your choice: 4
Enter category to delete: salary
Current transactions under salary:
1. Amount: 1200.0, Date: 2024-01-12
2. Amount: 70000.0, Date: 2024-04-12
3. Amount: 8000.0, Date: 2024-04-10
4. Amount: 12000.0, Date: 2024-04-12
Enter transaction number to delete: 4
Transaction deleted successfully.
```

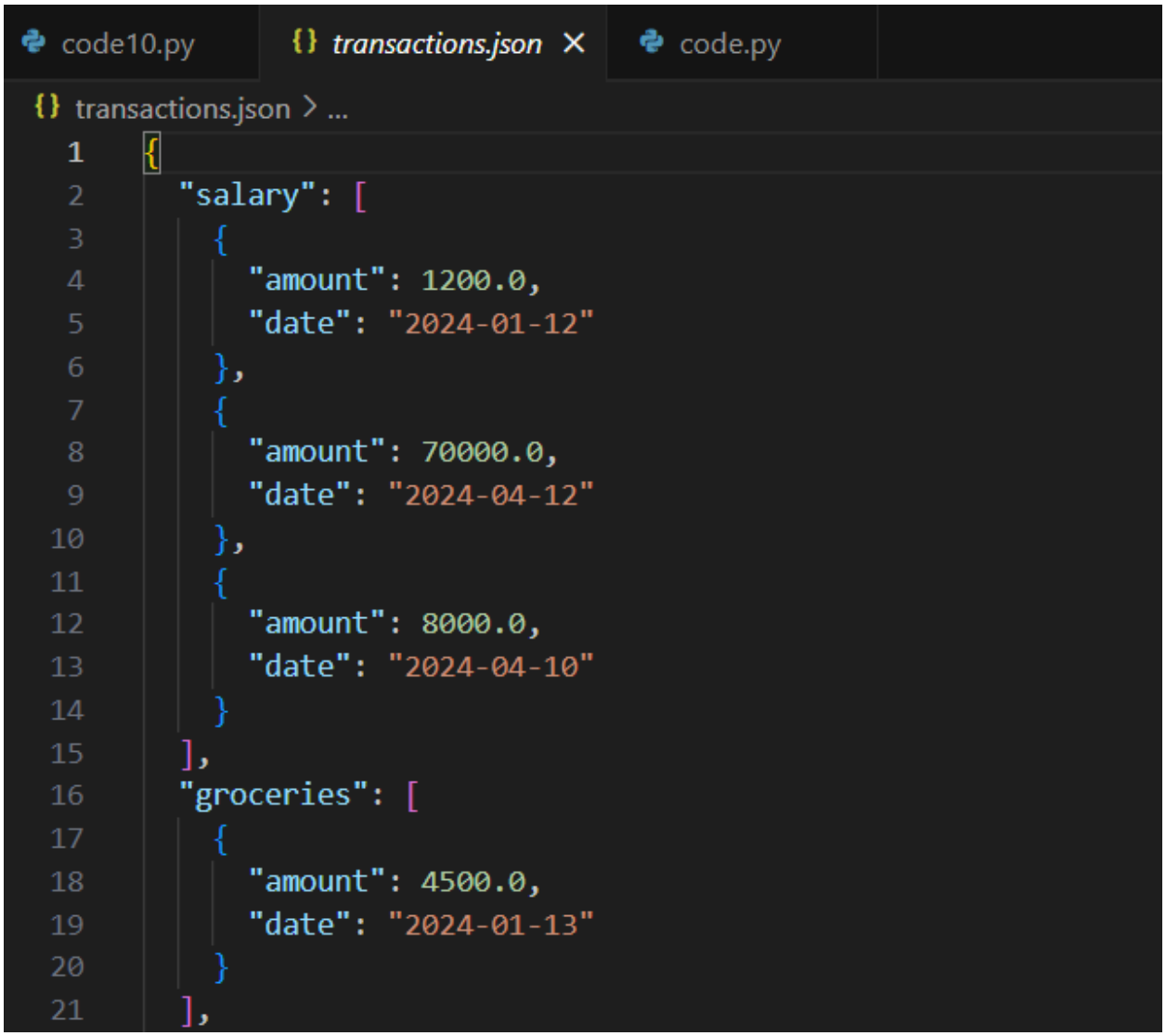
5)Display Summary

```
===== Personal Finance Tracker =====
1. Add Transaction
2. View Transactions
3. Update Transaction
4. Delete Transaction
5. Display Summary
6. Read Bulk Transactions from File
7. Save Transactions to File
8. Exit
Enter your choice: 5
salary: Total Amount - 79200.0, Count - 3
groceries: Total Amount - 4500.0, Count - 1
grocerie: Total Amount - 4500.0, Count - 1
rent: Total Amount - 7800.0, Count - 3
```

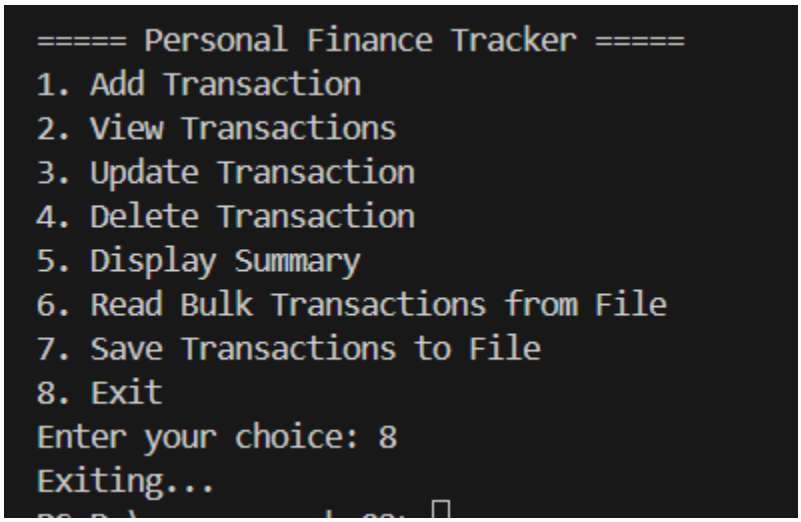
6)Read Bulk Transaction from File

```
===== Personal Finance Tracker =====
1. Add Transaction
2. View Transactions
3. Update Transaction
4. Delete Transaction
5. Display Summary
6. Read Bulk Transactions from File
7. Save Transactions to File
8. Exit
Enter your choice: 6
Enter filename to read transactions from: transactions.json
```

7)Save Transaction to file



8)Exit


===== Personal Finance Tracker =====
1. Add Transaction
2. View Transactions
3. Update Transaction
4. Delete Transaction
5. Display Summary
6. Read Bulk Transactions from File
7. Save Transactions to File
8. Exit
Enter your choice: 8
Exiting...