UNIVERSITY OF WESTMINSTER

INFORMATICS INSTITUTE OF TECHNOLOGY

INFORMATIC INSTITUTE OF TECHNOLOGY.

4COSC006C Software Development I

Module Code & Module Name : 4COSC006C.2 Software Development I.

Module Leader : Mr. Guhanathan Poravi.

Issue Date : 30th March 2024

Student Details:

Student Name	IIT ID	UOW ID
S.R Walakuluarachchi	20230436	2083527

TABLE OF CONTENT

LIST OF FIGURES

ACKNOWLEDGMENT

I would like to express my sincere gratitude to Mr. Pooravi Guganathan, and our tutorial lecturer Mr. Lakshan Costa for their exceptional teaching during the programming module. Their dedication, clarity, and passion for the subject made the learning experience enjoyable and insightful.

Lectures guidance played a pivotal role in shaping my understanding of programming concepts and improving my coding skills. I appreciate their commitment to fostering a positive and conducive learning environment.

Thank you, Mr. Pooravi Guganathan, and Mr. Lakshan Costa for being an inspiring and supportive instructor throughout this module.

I would also like to thank my friends who helped me to complete this assignment

01.PROBLEM STATEMENT

The assignment is to build a Personal Finance Tracker in Python, emphasizing core programming concepts like lists, loops, functions, input/output operations, and input validation. Initially, the application will employ lists for managing financial transactions, enabling basic CRUD operations (Create, Read, Update, Delete) without using dictionaries. Later, it will transition to using dictionaries to represent expenses, enhancing data management and program design. This shift also introduces file I/O with JSON for data persistence and bulk data processing. Through this project, learners will deepen their understanding of Python programming, while gaining practical experience in data handling, program architecture, and testing within a real-world context.

02.PSEUDOCODE

```
BEGIN
Import json
From datetime import datetime

transactions = {}

Function load_transactions():
    Try:
        Open 'transactions.json' file for reading
        Load transactions from file into global variable 'transactions'
    Except FileNotFoundError:
        Set 'transactions' to an empty dictionary

Function save_transactions():
    Open 'transactions.json' file for writing
    Save transactions from global variable 'transactions' to file in JSON format with indentation

Function read_bulk_transactions(file_name):
    Try:
        Open the specified file for reading
        For each line in the file:
            Split the line into parts by comma
            If the number of parts is 3:
                Extract category, amount, and date from parts
                Capitalize category
                Convert amount to float
```

```
            Append a new transaction to 'transactions' under the specified category
        Else:
            Print "Invalid format in line"
    Print "Bulk transactions added successfully."
    Call save_transactions() to save transactions to file
  Except FileNotFoundError:
    Print "File not found."

Function add_transactions():
  While True:
    Get transaction category from user and capitalize it
    While True:
      Try:
        Get transaction amount from user as float
        Break loop
      Except ValueError:
        Print "Invalid amount. Please enter a valid number."
    While True:
      Try:
        Get transaction date from user in format YYYY-MM-DD
        Validate date format
        Break loop
      Except ValueError:
        Print "Invalid date format. Please enter as YYYY-MM-DD."
    Append the transaction to 'transactions' under the specified category
    Ask user if they want to add more transactions
    If user does not want to add more transactions:
      Break loop
```

```
    Print "Transaction added successfully!"
    Call save_transactions() to save transactions to file

Function view_transactions():
    If 'transactions' is empty:
        Print "No transactions found."
    Else:
        For each category and its transactions in 'transactions':
            Print category
            For each transaction in the category:
                Print transaction details

Function update_transactions():
    Call view_transactions() to display current transactions
    If 'transactions' is empty:
        Return
    While True:
        Get transaction category to update from user and capitalize it
        If category is in 'transactions':
            Print transactions under the category
            Break loop
        Else:
            Print "No transactions found for this category."
    While True:
        Get field to update from user (Amount/Date) and capitalize it
        If field is "Amount" or "Date":
            Get index of transaction to update from user
            If index is valid:
```

```
            Get new value for the field from user
            Update the transaction with the new value
            Print "Field updated successfully!"
            Break loop
        Else:
            Print "Invalid transaction index."
    Else:
        Print "Invalid field. Please enter 'Amount' or 'Date'."

Function delete_transactions():
    Call view_transactions() to display current transactions
    If 'transactions' is empty:
        Return
    While True:
        Get transaction category to delete from user and capitalize it
        If category is in 'transactions':
            Print transactions under the category
            Break loop
        Else:
            Print "No transactions found for this category."
    While True:
        Get index of transaction to delete from user
        If index is valid:
            Delete the transaction at the specified index
            Print "Transaction deleted successfully!"
            Break loop
        Else:
            Print "Invalid transaction index."
```

```
Function display_summary():
    Call view_transactions() to display current transactions
    Initialize max_amount to 0
    For each category and its transactions in 'transactions':
        For each transaction in the category:
            If transaction amount is greater than max_amount:
                Update max_amount with transaction amount
    Print "The total of the expenses are: " followed by max_amount

Function main_menu():
    Call load_transactions() to load transactions from file
    While True:
        Print menu options
        Get user choice
        If choice is '1':
            Call add_transactions() to add a transaction
        ElseIf choice is '2':
            Call view_transactions() to view transactions
        ElseIf choice is '3':
            Call update_transactions() to update a transaction
        ElseIf choice is '4':
            Call delete_transactions() to delete a transaction
        ElseIf choice is '5':
            Get file name from user
            Call read_bulk_transactions() to read bulk transactions from file
        ElseIf choice is '6':
            Call display_summary() to display a summary
```

```
    ElseIf choice is '7':
        Print "Exiting program."
        Break loop
    Else:
        Print "Invalid choice. Please try again."

If this script is run directly:
    Call main_menu() to start the main menu loop
END
```

03.PYTHON CODE

```
import json
from datetime import datetime

# Global dictionary for storing transactions
transactions = {}

# Functions for file handling
def load_transactions():
    global transactions
    try:
        with open('transactions.json', 'r') as file:
            transactions = json.load(file)
    except FileNotFoundError:
        transactions = {}

def save_transactions():
    with open('transactions.json', 'w') as file:
        json.dump(transactions, file, indent=4)

def read_bulk_transactions(file_name):
    global transactions
    try:
        with open(file_name, 'r') as file:
            for line in file:
                parts = line.strip().split(',')
                if len(parts) == 3:
```

```
                category, amount, date = map(str.strip, parts)
                category = category.capitalize()
                amount = float(amount)
                transactions.setdefault(category, []).append({"Amount": amount, "Date": date})
            else:
                print("Invalid format in line:", line)
        print("Bulk transactions added successfully.")
        save_transactions()
    except FileNotFoundError:
        print(f"File '{file_name}' not found.")

# Function for adding a single transaction
def add_transactions():
    while True:
        category = input("\nEnter transaction category: ").capitalize()
        while True:
            try:
                amount = float(input("Enter the amount: "))
                break
            except ValueError:
                print("Invalid amount. Please enter a valid number.")
        while True:
            try:
                date = input(f"Enter the date for {category} (YYYY-MM-DD): ")
                datetime.strptime(date, "%Y-%m-%d")
                break
            except ValueError:
                print("Invalid date format. Please enter as YYYY-MM-DD.")
```

```
        transactions.setdefault(category, []).append({"Amount": amount, "Date": date})
        choice = input("Do you want to add more transactions? (Y/N): ").upper()
        if choice != "Y":
            break
    print("\nTransaction added successfully!")
    save_transactions()

# Function for viewing all transactions
def view_transactions():
    if not transactions:
        print("No transactions found.")
    else:
        for category, details in transactions.items():
            print(f"\nCategory: {category}")
            for idx, trans in enumerate(details, start=1):
                print(f"{idx}. Amount: {trans['Amount']}, Date: {trans['Date']}")

# Function for updating a transaction
def update_transactions():
    view_transactions()
    if not transactions:
        return
    while True:
        category = input("\nEnter the transaction category to update: ").capitalize()
        if category in transactions:
            print(f"{category} transactions: {transactions[category]}")
            break
        else:
```

```
            print("No transactions found for this category.")
    while True:
        field = input("\nEnter the field to update (Amount/Date): ").capitalize()
        if field in ["Amount", "Date"]:
            idx = int(input("Enter the index of the transaction to update: "))
            if 0 < idx <= len(transactions[category]):
                new_value = input(f"Enter the new {field}: ")
                transactions[category][idx - 1][field] = new_value
                print(f"{field} updated successfully!")
                break
            else:
                print("Invalid transaction index.")
        else:
            print("Invalid field. Please enter 'Amount' or 'Date'.")

# Function for deleting a transaction
def delete_transactions():
    view_transactions()
    if not transactions:
        return
    while True:
        category = input("Enter the transaction category to delete from: ").capitalize()
        if category in transactions:
            print(f"{category} transactions: {transactions[category]}")
            break
        else:
            print("No transactions found for this category.")
    while True:
```

```
        idx = int(input("Enter the index of the transaction to delete: "))
        if 0 < idx <= len(transactions[category]):
            transactions[category].pop(idx - 1)
            print("Transaction deleted successfully!")
            break
        else:
            print("Invalid transaction index.")

# Function for displaying a summary
def display_summary():
    view_transactions()
    max_amount = 0
    for category, details in transactions.items():
        for trans in details:
            if trans["Amount"] > max_amount:
                max_amount = trans["Amount"]
    print("\nThe total of the expenses are: ", max_amount)

# Main menu function
def main_menu():
    load_transactions()
    while True:
        print("\nPersonal Finance Tracker")
        print("1. Add Transaction")
        print("2. View Transactions")
        print("3. Update Transactions")
        print("4. Delete Transactions")
        print("5. Read Bulk Transactions")
```

```
        print("6. Display Summary")
        print("7. Exit")
        choice = input("\nEnter your choice: ")
        if choice == '1':
            add_transactions()
        elif choice == '2':
            view_transactions()
        elif choice == '3':
            update_transactions()
        elif choice == '4':
            delete_transactions()
        elif choice == '5':
            file_name = input("Enter the file name to read bulk transactions from: ")
            read_bulk_transactions(file_name)
        elif choice == '6':
            display_summary()
        elif choice == '7':
            print("Exiting program.")
            break
        else:
            print("Invalid choice. Please try again.")

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

04. TEST CASES

Test Component	Test No	Test Input	Expected Result	Actual Result	Pass / Fail
Main Menu	1	None	Displaying the main menu with options and asking choice.	Displaying the main menu with options and asking choice.	Pass
Add Transactions	2.0	**Valid Input:** Category: Salary Amount: 1000 Date: 2024-03-17	Display “Transaction Added Successfully”	Display “Transaction Added Successfully”	Pass
	2.1	**Invalid Input:** Amount : abc	Display “Invalid amount, Please enter a valid amount”	Display “Invalid amount, Please enter a valid amount”	Pass
View Transactions	3.0	**View transactions when there are no transactions:** Transactions[]	Display “No Transactions Found”	Display “No Transactions Found”	Pass
	3.1	**View transactions when there are existing transactions:** Category: Salary Amount: 1000 Date: 2024-03-17	Category: Salary 1. Amount: 1000.0, Date: 2023-03-17	Category: Salary 1. Amount: 1000.0, Date: 2023-03-17	Pass
Update Transactions	4.0	**Valid Input:** Update an existing Transaction	Category: Salary 1. Amount: 1000.0, Date: 2023-03-17 Enter the transaction category to update: salary	Category: Salary 1. Amount: 1000.0, Date: 2023-03-17 Enter the transaction category to update: salary	Pass

			Salary transactions: [{'Amount': 1000.0, 'Date': '2023-03-17'}] Enter the field to update (Amount/Date): amount Enter the index of the transaction to update: 1 Enter the new Amount: 2500 Amount updated successfully!	Salary transactions: [{'Amount': 1000.0, 'Date': '2023-03-17'}] Enter the field to update (Amount/Date): amount Enter the index of the transaction to update: 1 Enter the new Amount: 2500 Amount updated successfully!!	
	4.1	**Invalid Input:** Index of transaction to update: 5	Displaying “Invalid transaction index”	Displaying “Invalid transaction index”	Pass
Delete Transaction.	5.0	**Valid input: Delete an existing transaction**. Index of transaction to delete: 1	Delete the selected transaction and Display “Transaction delete successfully.”	Delete the selected transaction and Display “Transaction delete successfully.”	Pass
Display Summary	6.0	**Display summary when there are existing transactions.** Category: Salary 1. Amount: 1000.0, Date: 2023-03-17 Category: Grocery 1. Amount: 500.0, Date: 2023-05-05	The total of the expenses are: 1000.0	The total of the expenses are: 1000.0	Pass
Exit	7.0	Option: 7	Exiting From the program	Exiting From the program	Pass

Read Bulk Transactions	8.0	Option: 5	Display bulk transactions added successfully.	Display bulk transactions added successfully.	Pass

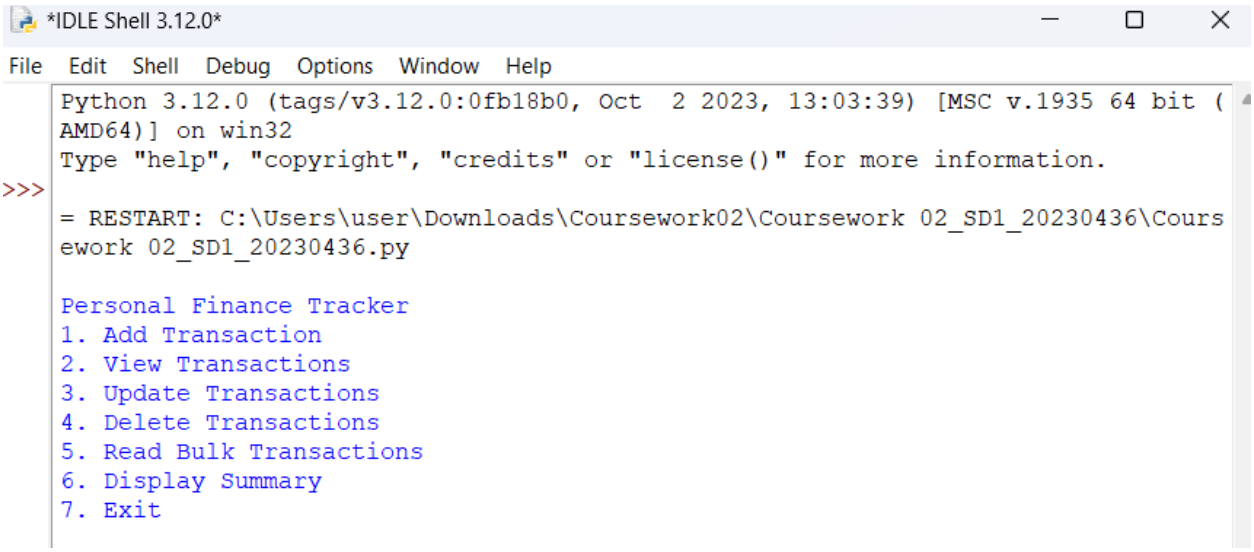

Figure 1: Test No 1

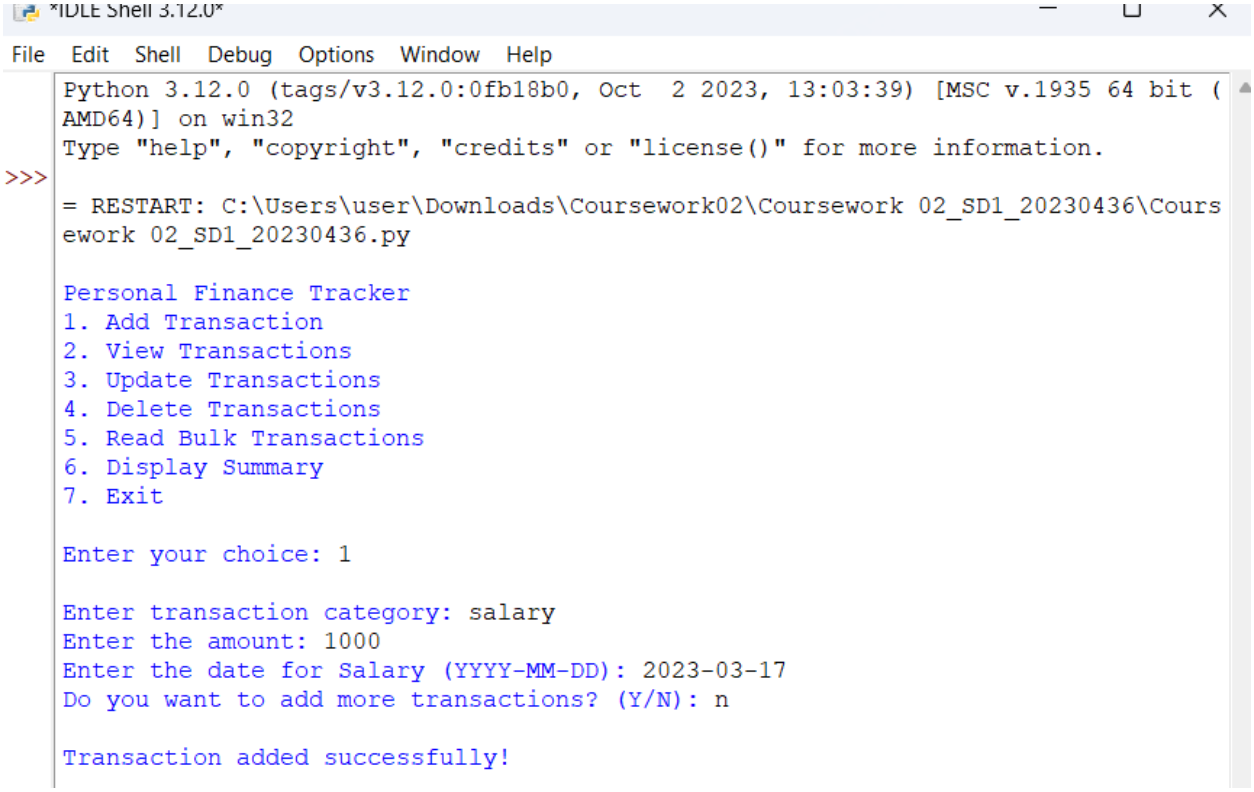

Figure 2: Test No 2.0

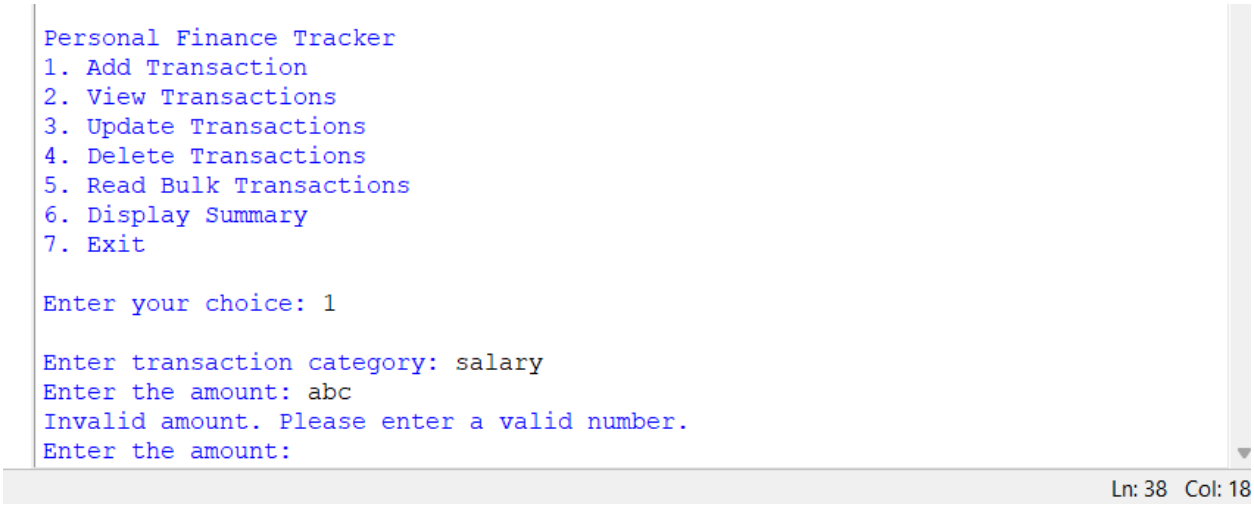

Figure 3:Test No 2.1

```
Enter your choice: 1

Enter transaction category: salary
Enter the amount: 1000
Enter the date for Salary (YYYY-MM-DD): 2023-03-17
Do you want to add more transactions? (Y/N): n

Transaction added successfully!

Personal Finance Tracker
1. Add Transaction
2. View Transactions
3. Update Transactions
4. Delete Transactions
5. Read Bulk Transactions
6. Display Summary
7. Exit

Enter your choice: 2

Category: Salary
1. Amount: 1000.0, Date: 2023-03-17

Personal Finance Tracker
1. Add Transaction
2. View Transactions
3. Update Transactions
4. Delete Transactions
5. Read Bulk Transactions
6. Display Summary
7. Exit
```

Figure 4: Test No 3.1

```
Personal Finance Tracker
1. Add Transaction
2. View Transactions
3. Update Transactions
4. Delete Transactions
5. Read Bulk Transactions
6. Display Summary
7. Exit

Enter your choice: 3

Category: Salary
1. Amount: 1000.0, Date: 2023-03-17

Enter the transaction category to update: salary
Salary transactions: [{'Amount': 1000.0, 'Date': '2023-03-17'}]

Enter the field to update (Amount/Date): amount
Enter the index of the transaction to update: 1
Enter the new Amount: 2500
Amount updated successfully!
```

Figure 5: Test No 4.0

```
Personal Finance Tracker
1. Add Transaction
2. View Transactions
3. Update Transactions
4. Delete Transactions
5. Read Bulk Transactions
6. Display Summary
7. Exit

Enter your choice: 3

Category: Salary
1. Amount: 2500, Date: 2023-03-17

Enter the transaction category to update: salary
Salary transactions: [{'Amount': '2500', 'Date': '2023-03-17'}]

Enter the field to update (Amount/Date): amount
Enter the index of the transaction to update: 5
Invalid transaction index.
```

Figure 6: Test No 4.1

```
Enter your choice: 1

Enter transaction category: salary
Enter the amount: 1000
Enter the date for Salary (YYYY-MM-DD): 2023-03-17
Do you want to add more transactions? (Y/N): n

Transaction added successfully!

Personal Finance Tracker
1. Add Transaction
2. View Transactions
3. Update Transactions
4. Delete Transactions
5. Read Bulk Transactions
6. Display Summary
7. Exit

Enter your choice: 4

Category: Salary
1. Amount: 1000.0, Date: 2023-03-17
Enter the transaction category to delete from: salary
Salary transactions: [{'Amount': 1000.0, 'Date': '2023-03-17'}]
Enter the index of the transaction to delete: 1
Transaction deleted successfully!
```

Figure 7: Test No 5.0

```
Personal Finance Tracker
1. Add Transaction
2. View Transactions
3. Update Transactions
4. Delete Transactions
5. Read Bulk Transactions
6. Display Summary
7. Exit

Enter your choice: 6

Category: Salary
1. Amount: 1000.0, Date: 2023-03-17

Category: Grocery
1. Amount: 500.0, Date: 2023-05-05

The total of the expenses are:  1000.0

Personal Finance Tracker
1. Add Transaction
2. View Transactions
3. Update Transactions
4. Delete Transactions
5. Read Bulk Transactions
6. Display Summary
7. Exit
```

Figure 8: Test No 6.0

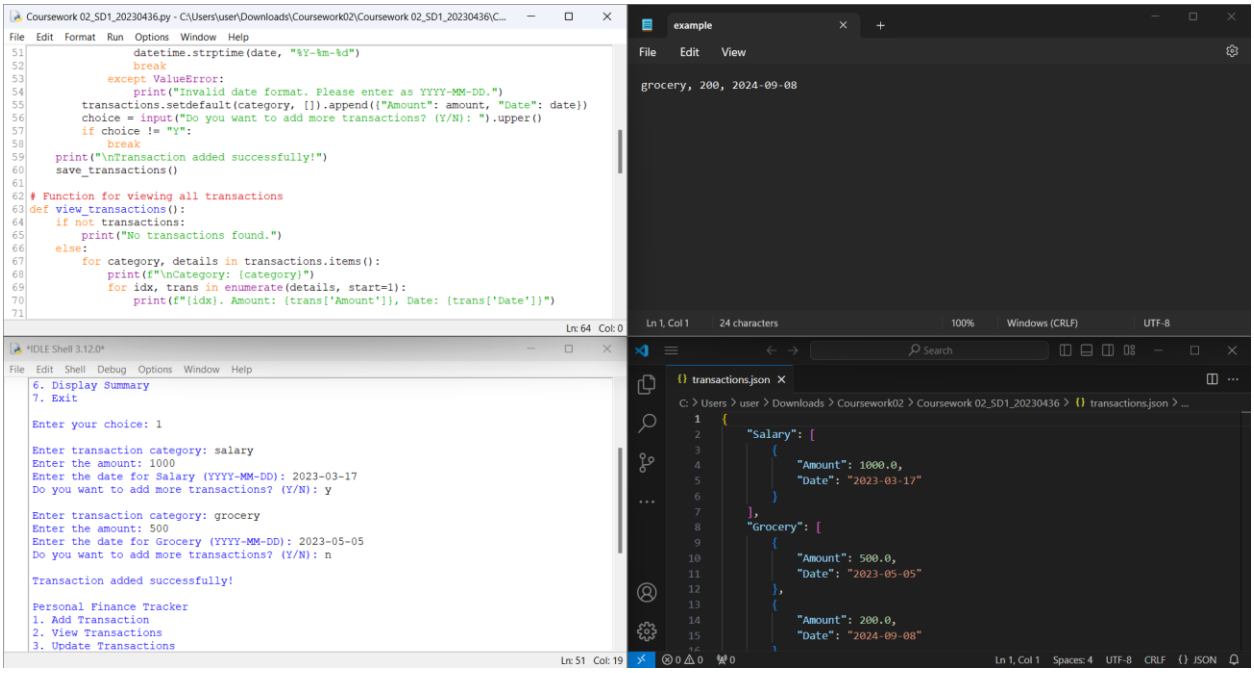

Figure 9: Test No 8.0