UNIVERSITY OF WESTMINSTER

INFORMATICS INSTITUTE OF TECHNOLOGY

Informatics Institute of Technology

School of Computing

4C0SC006C.2 Software Development 1

Module: 4C0SC006C.2 Software Development 1

Module Leader: Mr. Pooravi Guganathan

Assessment Type: Individual

Issue Date: 30/03/2024

Student Name	IIT ID	UOW ID
R.M.W.D.Anjalee	20230179	2084393

i. Acknowledgment

I extend my deepest gratitude to Professor Mr. Pooravi Guganathan and our tutorial lecturer. Their guidance, patience, and expertise have profoundly shaped our programming journey. We are thankful for their unwavering support, invaluable advice, and passion for sharing knowledge. Their enthusiasm and dedication have inspired us to excel and pursue our goals with vigor. We are truly fortunate to have such remarkable mentors who continually ignite our curiosity and drive for learning. Their impact on our understanding and appreciation of programming is immeasurable, and we are sincerely grateful for all they have done to enrich our educational experience.

ii. Abstract

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.

Table of Contents

i. Acknowledgment ii

ii. Abstract iii

iii. List of figures v

1. Pseudo Code 6

2. Python Code 9

3. Test Plan Summary 14

3.1 Test cases 14

3.2 Screen Shots 16

3.3 Storing of the Transactions 19

3.4 Process Description 19

3.4.1 Adding Transactions 19

- Ask user for transaction details 19
- Create a new transaction and add it to Finance Tracker 19
- Save the updated Finance Tracker data to the file 19

3.4.2 Viewing Transactions 19

- Display all transactions with their details 19

3.4.3 Updating Transactions 19

- Update the transaction with new details 19

3.4.4 Deleting a transaction 19

- Delete an existing transaction 19

3.4.5 Display the summary 19

- Calculate total income, total expenses, and net income and display 19

iii. List of figures

Figure 1 16
Figure 2 17
Figure 3 17
Figure 4 17
Figure 5 18

1. Pseudo Code

```
BEGIN
  Define FILENAME as "FinanceTracker.json"

  Function Load_transactions():
      Try to open FILENAME for reading
      If FileNotFoundError:
          Set transactions as an empty dictionary
      Otherwise:
           Load transactions from the file
      Return transactions

  Function Save_transactions(transactions):
      Open FILENAME for writing
      Write transactions to the file in JSON format
      Close the file

  Function Read_bulk_transactions_from_file(filename):
      Try to open the specified filename for reading
      If FileNotFoundError:
          Print a message indicating the file was not found
          Return an empty dictionary
      Otherwise:
          Load transactions from the file
          Return transactions

  Function Add_transaction(category, amount, date):
      Load existing transactions
```

```
    If category does not exist in transactions:
        Create a new empty list for the category
    Append a new transaction (dictionary with amount and date) to the category
    Save transactions to the file
    Print "Transaction added!"

Function View_transactions():
    Load all transactions
    Iterate over each category and its transactions:
        Print category name
        Iterate over transactions, print index, amount, and date

Function Delete_transaction(category, index):
    Load existing transactions
    Calculate the new index by subtracting 1 from the provided index
    If category exists in transactions and new index is valid:
        Delete the transaction at the new index from the category
        If the category is now empty, delete it
        Save transactions to the file
        Print "Transaction Deleted!"
    Otherwise, print "Invalid category or index!"

Function Update_transaction(category, index, amount, date):
    Load existing transactions
    Calculate the new index by subtracting 1 from the provided index
    If category exists in transactions and new index is valid:
        Update the transaction at the new index with the new amount and date
        Save transactions to the file
        Print "Successfully Updated!"
    Otherwise, print "Invalid category or index!"
```

```
 Function Display_summary():
    Load all transactions
    Iterate over each category and its transactions:
        Calculate the total amount for each category
        Print category name and total amount

 Function Main_menu():
    Loop indefinitely:
        Print menu options
        Prompt user for choice
        Based on user choice:
            Add Transaction: Prompt for category, amount, and date, then call add_transaction()
            View Transactions: Call view_transactions()
            Update Transaction: Prompt for category, index, new amount,and new date,then call
  update_transaction()
            Summary of Transactions: Call display_summary()
            Delete Transaction: Prompt for category and index, then call delete_transaction()
            Exit: Print exit message and break the loop

 If __name__ == "__main__":
    Call main_menu() to start the program
END
```

2. Python Code

```
#importing json
import json

# Global dictionary to store transactions
FILENAME = "FinanceTracker.json"



# File handling functions

# Function to load transactions from the file
def load_transactions():
    try:
        with open(FILENAME, "r") as file:
            transactions = json.load(file)
    except FileNotFoundError:
        transactions = {}# If file not found, initialize an empty dictionary
    return transactions

# Function to save transactions to the file
def save_transactions(transactions):
    with open(FILENAME, "w") as file:
        json.dump(transactions,file,indent=2)

# Function to read transactions from a file
def read_bulk_transactions_from_file(filename):
    try:
        with open(filename, "r") as file:
            transactions = json.load(file)
        return transactions
    except FileNotFoundError:
        print(f"File '{filename}' not found.")
        return {}# If file not found, return an empty dictionary



# Feature implementations

# Function to add a new transaction
def add_transaction(category, amount, date):
    transactions = load_transactions() # Load existing transactions
    if category not in transactions:
        transactions[category] = []# If category doesn't exist, create a new one
    transactions[category].append({"amount": amount, "date": date})# Add new transaction
    save_transactions(transactions)# Save updated transactions to the file
    print ("Transaction added!")

# Function to view all transactions
def view_transactions():
    all_transactions = load_transactions()# Load all transactions
    for category, transactions in all_transactions.items():
        print(f"{category}:")
        for index, transaction in enumerate(transactions, start=1):
            print(f"  {index}. Amount: {transaction['amount']}, Date: {transaction['date']}")
        print()

def delete_transaction(category, index):
    transactions = load_transactions()# Load existing transactions
    new_index = index -1
    if category in transactions and 0 <= new_index < len(transactions[category]):
        del transactions[category][new_index]# Delete transaction
        if len(transactions[category]) == 0:
            del transactions[category]# If no transactions left in the category, delete the category
        save_transactions(transactions)# Save updated transactions to the file
        print("Transaction Deleted!")
    else:
        print("Invalid category or index!")

# Function to update a transaction
def update_transaction(category, index, amount, date):
    transactions = load_transactions() # Load existing transactions
    new_index = index -1
    if category in transactions and 0 <= new_index < len(transactions[category]):
        transactions[category][new_index] = {"amount": amount, "date": date} # Update
transaction
        save_transactions(transactions)# Save updated transactions to file
        print("Successfuly Updated!")
    else:
        print("Invalid category or index!")

# Function to display summary of transactions
def display_summary():
    all_transactions = load_transactions()# Load all transactions
    for category, transactions in all_transactions.items():
        total_amount = sum(transaction['amount'] for transaction in transactions)# Calculate
total amount
        print(f"{category}: Total Amount: {total_amount}")

# Main menu function
def main_menu():
    while True:
        print("\n1. Add Transaction")
        print("2. View Transactions")
        print("3. Update Transaction")
        print("4. Summarry of Transaction")
        print("5. Delete Transaction")
        print("6. Exit")
        choice = input("Enter your choice: ")

        if choice == "1":
            category = input("Enter category: ")
            amount = float(input("Enter amount: "))
            date = input("Enter date (YYYY-MM-DD): ")
            add_transaction(category, amount, date)

        elif choice == "2":
            view_transactions()

        elif choice == "3":
            view_transactions()
            category = input("Enter category: ")
            index = int(input("Enter index to update: "))
            amount = float(input("Enter new amount: "))
            date = input("Enter new date (YYYY-MM-DD): ")
            update_transaction(category, index, amount, date)



        elif choice == "4":
            display_summary()

        elif choice == "5":
            view_transactions()
            category = input("Enter category: ")
```

```
            index = int(input("Enter index to delete: "))
            delete_transaction(category, index)

        elif choice == "6":
            print("Thanks for using FinanceTracker!")
            break

        else:
            print("Invalid choice!")

# Entry point of the program
if __name__ == "__main__":
    main_menu()  # Call the main_menu function to start the program
```

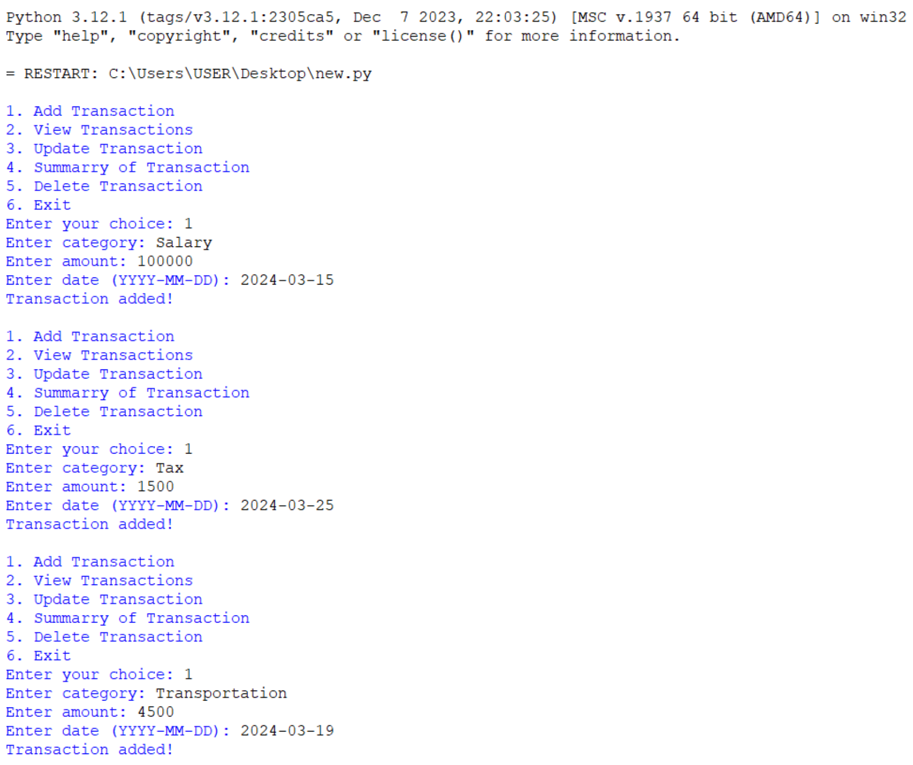
3. Test Plan Summary

3.1 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	2.1	Choice : 1 category: Salary amount: 100000 date (YYYY-MM-DD): 2024-03-15	Display “Transaction added!”	Display “Transaction added!”	Pass
	2.2	Choice : 1 category: Tax amount: 1500 date (YYYY-MM-DD): 2024-03-25	Display “Transaction added!”	Display “Transaction added!”	Pass
	2.3	Choice : 1 category: Transportation amount: 4500 date (YYYY-MM-DD): 2024-03-19	Display “Transaction added!”	Display “Transaction added!”	Pass
View Transactions	3	Choice : 2	Display all saved transactions.	Display all saved transactions.	Pass
Update Transactions	4.1	Choice :3 category: Tax index to update: 1 new amount: 2000 new date (YYYY-MM-DD): 2024-03-05	Display “Successfuly Updated!”	Display “Successfuly Updated!”	Pass
Display Summary	5	Choice : 4	Display all the Transaction categories and their Total amount	Display all the Transaction categories and their Total amount	Pass

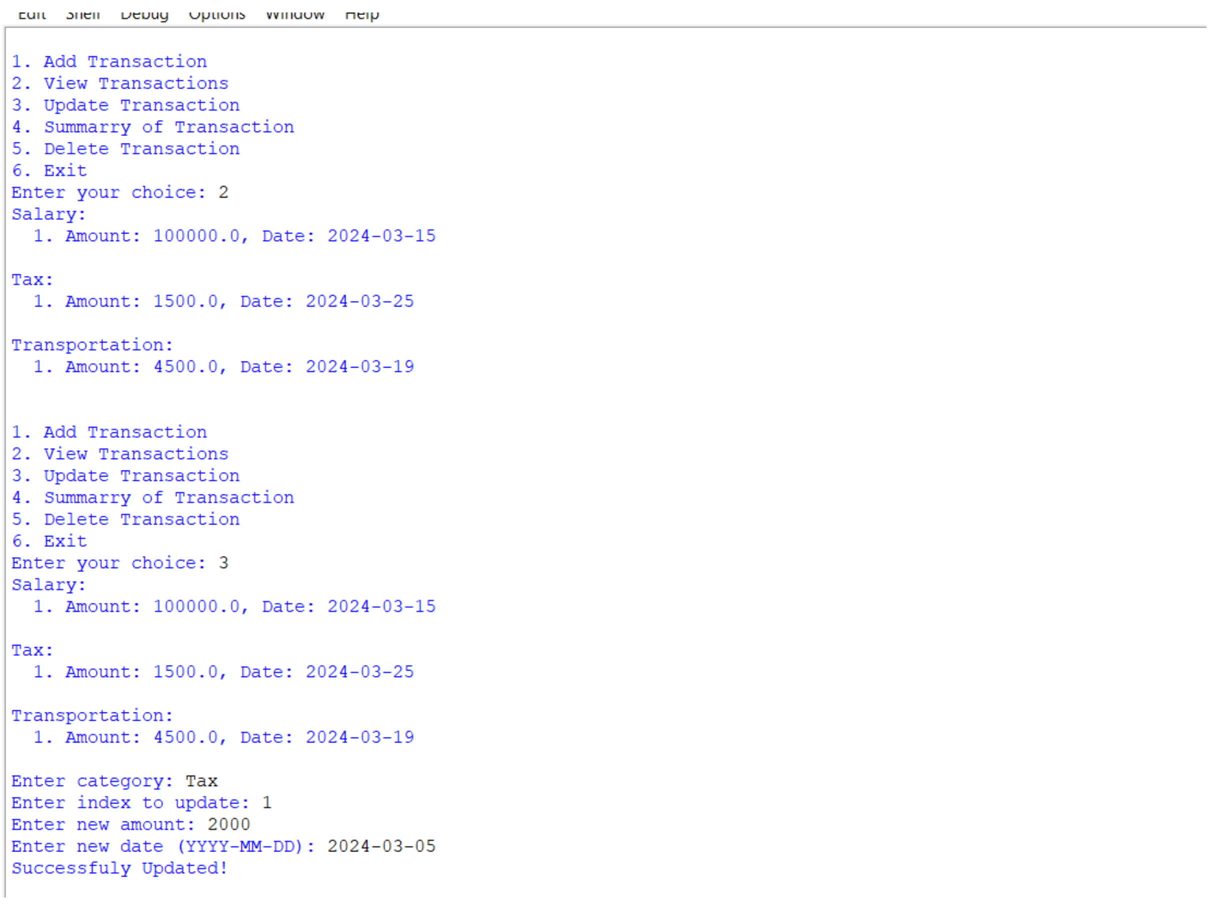
Delete Transaction	6	Choice : 5 Category: Transportation Index to delete: 1	Delete the selected transaction and Display “Transaction deleted!”	Delete the selected transaction and Display “Transaction deleted!”	Pass
Exit	7	Choice : 6	Exiting From the program and display “Thanks for using FinanceTracker ”	Exiting From the program and display “Thanks for using FinanceTracker ”	Pass

3.2 Screen Shots



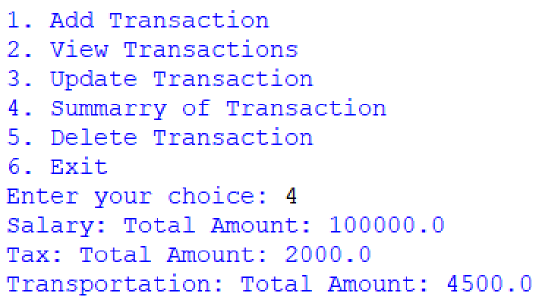
```
Python 3.12.1 (tags/v3.12.1:2305ca5, Dec  7 2023, 22:03:25) [MSC v.1937 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.

= RESTART: C:\Users\USER\Desktop\new.py

1. Add Transaction
2. View Transactions
3. Update Transaction
4. Summarry of Transaction
5. Delete Transaction
6. Exit
Enter your choice: 1
Enter category: Salary
Enter amount: 100000
Enter date (YYYY-MM-DD): 2024-03-15
Transaction added!

1. Add Transaction
2. View Transactions
3. Update Transaction
4. Summarry of Transaction
5. Delete Transaction
6. Exit
Enter your choice: 1
Enter category: Tax
Enter amount: 1500
Enter date (YYYY-MM-DD): 2024-03-25
Transaction added!

1. Add Transaction
2. View Transactions
3. Update Transaction
4. Summarry of Transaction
5. Delete Transaction
6. Exit
Enter your choice: 1
Enter category: Transportation
Enter amount: 4500
Enter date (YYYY-MM-DD): 2024-03-19
Transaction added!
```

Figure 1



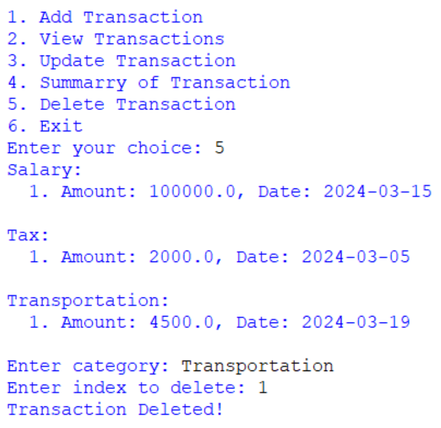
```
1. Add Transaction
2. View Transactions
3. Update Transaction
4. Summarry of Transaction
5. Delete Transaction
6. Exit
Enter your choice: 2
Salary:
  1. Amount: 100000.0, Date: 2024-03-15

Tax:
  1. Amount: 1500.0, Date: 2024-03-25

Transportation:
  1. Amount: 4500.0, Date: 2024-03-19


1. Add Transaction
2. View Transactions
3. Update Transaction
4. Summarry of Transaction
5. Delete Transaction
6. Exit
Enter your choice: 3
Salary:
  1. Amount: 100000.0, Date: 2024-03-15

Tax:
  1. Amount: 1500.0, Date: 2024-03-25

Transportation:
  1. Amount: 4500.0, Date: 2024-03-19

Enter category: Tax
Enter index to update: 1
Enter new amount: 2000
Enter new date (YYYY-MM-DD): 2024-03-05
Successfuly Updated!
```

Figure 2



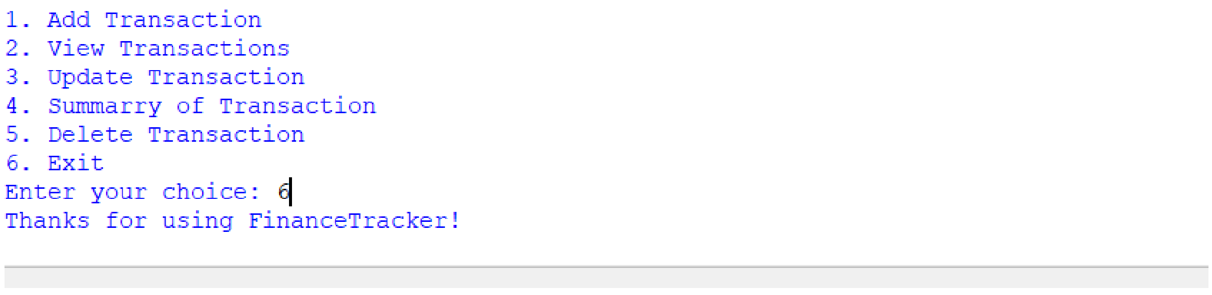
```
1. Add Transaction
2. View Transactions
3. Update Transaction
4. Summarry of Transaction
5. Delete Transaction
6. Exit
Enter your choice: 4
Salary: Total Amount: 100000.0
Tax: Total Amount: 2000.0
Transportation: Total Amount: 4500.0
```

Figure 3



```
1. Add Transaction
2. View Transactions
3. Update Transaction
4. Summarry of Transaction
5. Delete Transaction
6. Exit
Enter your choice: 5
Salary:
  1. Amount: 100000.0, Date: 2024-03-15

Tax:
  1. Amount: 2000.0, Date: 2024-03-05

Transportation:
  1. Amount: 4500.0, Date: 2024-03-19

Enter category: Transportation
Enter index to delete: 1
Transaction Deleted!
```

Figure 4



```
1. Add Transaction
2. View Transactions
3. Update Transaction
4. Summarry of Transaction
5. Delete Transaction
6. Exit
Enter your choice: 6
Thanks for using FinanceTracker!
```

Figure 5

3.3 Storing of the Transactions

The transactions are stored in a file named "FinanceTracker.json" using JSON format. The “Save_transactions” function opens this file for writing and writes the transactions to it in JSON format. Conversely, the “Load_transactions” function attempts to open the same file for reading and loads the transactions from it. Additionally, there's a function called “Read_bulk_transactions_from_file” which can read transactions from any specified filename. This setup allows for easy storage and retrieval of transaction data in a structured format.

3.4 Process Description

3.4.1 Adding Transactions

- Ask user for transaction details
- Create a new transaction and add it to Finance Tracker
- Save the updated Finance Tracker data to the file

3.4.2 Viewing Transactions

- Display all transactions with their details

3.4.3 Updating Transactions

- Update the transaction with new details

3.4.4 Deleting a transaction

- Delete an existing transaction

3.4.5 Display the summary

- Calculate total income, total expenses, and net income and display