University of Westminster
Informatics Institute of Technology

Computer Science & Software Engineering

Module: 4COSC006C Software Development I

Module Leader: Mr. Poravi Guganathan

Assessment Type: Individual Coursework

Due Date: 18th March 2024

Student Name: Nadil Nemitha Kaluarachchi

UOW ID: w2084392

IIT ID: 20230167

Tutorial Group: Group B

Table of Contents

1. Introduction

- The Personal Finance Tracker is an efficient Python application that aids individuals in effectively managing their financial transactions. This program enables users to effortlessly document their income and expenses, access transaction specifics, modify existing transactions, eliminate unwanted transactions, and acquire a comprehensive overview of their financial activities..

- By employing modular architecture and efficient file management methods, this program guarantees the long-term storage of transaction information in JSON format. Robust exception handling processes effectively handle errors, while user engagement is enhanced through a simple command-line interface. Through meticulous data validation procedures, the software computes and displays transaction overviews, encompassing total revenue, expenditures, and net equilibrium. This application empowers users to conveniently monitor and assess their financial transactions, facilitating well-informed choices to reach financial objectives.

2. Acknowledgement

I would like to express my sincere gratitude to Mr. Pooravi Guganathan, and our tutorial lecturer Mrs.Sandunika Rasanjalee 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 Mrs.Sandunika Rasanjalee 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

3. Pseudo Code

```
START
        IMPORT json and datetime modules.
        INITIALIZE empty transactions list.


FUNCTION load_transactions()
  TRY:
    OPEN FILE "Transactions.json" in read mode
    READ data from file
    COPY JSON data into a Python list transactions
    FOR each transaction in transactions:
      GET amount, description, type, and date from the transaction
      CREATE a temporary list containing the extracted data
      APPEND the temporary list to transactions
    END FOR
    CLOSE FILE
  EXCEPT FileNotFoundError:
    PRINT "File Not Found"
    INITIALIZE transactions as an empty list
  EXCEPT JSONDecodeError:
    PRINT "Error decoding JSON. Starting with an empty transaction list"
    INITIALIZE transactions as an empty list
```

```
FUNCTION save_transactions():
  TRY:
    OPEN FILE "Transactions.json" in write mode
    WRITE "[" to the file
    WRITE a new line to the file
    FOR each transaction in transactions:
      WRITE "\t" to the file
      CONVERT transaction to JSON format and WRITE it to the file
      WRITE a new line to the file
    WRITE "]" to the file
    CLOSE FILE
  EXCEPT FileNotFoundError:
    OPEN FILE "Transactions.json" in write mode
    WRITE an empty list to the file
    CLOSE FILE
```

```
FUNCTION add_transaction():
  TRY:
    GET amount from user input and convert to float
    GET category from user input
    WHILE True:
      GET transaction_type from user input and capitalize it
      IF transaction_type is in ['Income', 'Expense']:
        BREAK
      ELSE:
        PRINT "Invalid transaction type"
    GET current_date as today's date
    WHILE True:
      GET date from user input
      IF length of date is not equal to 10:
        PRINT "Invalid date"
        CONTINUE
      SPLIT date into year, month, and day
      IF length of date is 10 and int(month) <= 12 and int(day) <= 31:
        CREATE input_date using year, month, and day
        IF input_date is less than or equal to current_date:
          BREAK
        ELSE:
          PRINT "The date you entered is in the future"
      ELSE:
        PRINT "Invalid date"
    APPEND [amount, category, transaction_type, date] to transactions list
    CALL save_transactions()
    PRINT "Transaction added successfully"
  EXCEPT ValueError:
    PRINT "Invalid amount, Please enter a valid amount"
```

FUNCTION view_transactions():

```
        IF length of transactions list is 0:

            PRINT "No transactions available"

        ELSE:

            FOR each transaction in transactions list:

            PRINT transaction
```

FUNCTION update_transaction():

```
CALL view_transactions() function
TRY:
   GET index of transaction to update from user input
   IF index is valid (greater than or equal to 0 and less than or equal to the length of transactions list):
      GET new amount from user input
      GET new category from user input
      WHILE transaction type is not valid:
         GET new transaction type from user input
         IF new transaction type is valid (either "Income" or "Expense"):
            BREAK the loop
         ELSE:
            PRINT "Invalid transaction type"
      GET current date
      WHILE date is not valid:
         GET new date from user input
         IF length of new date is not 10:
            PRINT "Invalid date"
         ELSE IF month and day are valid (both <= 12 and <= 31):
            CREATE a new input date
            IF new input date is less than or equal to current date:
               BREAK the loop
            ELSE:
               PRINT "The date you entered is in the future"
         ELSE:
            PRINT "Invalid date"
      UPDATE the transaction at index-1 with new amount, category, transaction type, and date
      CALL save_transactions() function to save the updated transactions to file
      PRINT "Transaction updated successfully"
   ELSE:
      PRINT "Invalid index, Please enter a valid index"
EXCEPT ValueError:
   PRINT "Invalid amount, Please enter a valid amount" FUNCTION
```

FUNCTION delete_transaction():

```
CALL view_transactions() function
TRY:
   GET index of transaction to delete from user input
   IF index is valid (greater than or equal to 0 and less than or equal to the length of transactions list):
```

```
            DELETE the transaction at index-1 from the transactions list
            CALL save_transactions() function to save the updated transactions to file
            PRINT "Transaction deleted successfully"
        ELSE:
            PRINT "Invalid index, please enter a valid index"
    EXCEPT ValueError:
        PRINT "Invalid index. Please enter a valid index"
```

```
FUNCTION display_summary():
    INITIALIZE Total_income to 0
    INITIALIZE Total_expense to 0
    IF transactions list is empty:
        PRINT "No transactions record yet"
    FOR each sublist in transactions:
        IF type of transaction is "Income":
            ADD amount to Total_income
        ELSE IF type of transaction is "Expense":
            ADD amount to Total_expense
    CALCULATE balance as Total_income minus Total_expense
    PRINT "Total income is: " followed by Total_income
    PRINT "Total expense is: " followed by Total_expense
    PRINT "Profit is: " followed by balance
```

```
FUNCTION main_menu():
    CALL load_transactions() to load transactions at the start
    WHILE True:
        PRINT "\nPersonal Finance Tracker"
        PRINT "1. Add Transaction"
        PRINT "2. View Transactions"
        PRINT "3. Update Transaction"
        PRINT "4. Delete Transaction"
        PRINT "5. Display Summary"
        PRINT "6. Exit"
        GET user's choice from input
        IF choice is '1':
            CALL add_transaction()
        ELSE IF choice is '2':
            CALL view_transactions()
        ELSE IF choice is '3':
            CALL update_transaction()
        ELSE IF choice is '4':
            CALL delete_transaction()
        ELSE IF choice is '5':
            CALL display_summary()
        ELSE IF choice is '6':
            PRINT "Exiting program."
            BREAK
        ELSE:
            PRINT "Invalid choice. Please try again."
END
```

4. Python Code

```
import json#To save the details
import datetime#To take current date
# Global list to store transactions
transactions = []

# File handling functions
def load_transactions():
    global transactions
    try:
        with open("Transactions.json","r")as file:
            transactions=json.load(file)#Loading file details to the transactions variable
    except FileNotFoundError:
        print("File Not Found")
        transactions=[]
    except json.JSONDecodeError:
        print("Error decoding JSON. Starting with an empty transaction list")


def save_transactions():
    with open("Transactions.json","w")as file:
        file.write("[")
        file.write("\n")
        for i in transactions:
            file.write("\t")
            json.dump(i,file)#dump() convert python object into JSON objects
            file.write("\n")
        file.write("]")
# Feature implementations
def add_transaction():
    try:
        amount=float(input("Enter amount: "))
        category=input("Enter catagory: ")
        while True:#while transaction type is satisfy
            transaction_type=input("Enter type(Income/Expense): ").capitalize()
            if transaction_type in ['Income','Expense']:
                break
            else:
                print("Invalid transaction type")
        current_date=datetime.date.today()#Takeing current date and time
        while True:#while date is satisfy
```

```
            date=input("Enter date(YYYY-MM-DD): ")
            if len(date)!=10:
                print("Invalid date")
                continue
            year,month,day=date.split("-")
            if len(date)==10 and int(month)<=12 and int(day)<=31:
                input_date=datetime.date(int(year),int(month),int(day))#Formatting the input date
                if input_date<=current_date:#Checking the input date below the current date
                    break
                else:
                    print("The date you entered in future")

            else:
                print("Invalid date")
        transactions.append([amount,category,transaction_type,date])#Entering the data to the list
        save_transactions()
        print("Transaction added successfully")
    except ValueError:
        print("Invalid amount, Please enter a valid amount")
def view_transactions():
    if len(transactions)==0:
        print("No transactions available")

    else:
        for transaction in transactions:#To print the sublist in nested list
            print(transaction)

def update_transaction():
    view_transactions()
    try:
        index=int(input("Enter index of transaction to update: "))
        if index>=0 and index<=len(transactions):
            new_amount=float(input("Enter new amount: "))
            new_catagory=input("Enter new catagory: ")

            while True: #while transaction type is satisfy
                new_trans_type=input("Enter new transaction type(Income/Expense): ").capitalize()
                if new_trans_type in ['Income','Expense']:
                    break
                else:
                    print("Invalid transaction type")
            current_date=datetime.date.today()#Taking current date
            while True:#while date is satisfy
                new_date=input("Enter new Date(YYYY-MM-DD): ")
                if len(new_date)!=10:#If the length of new_date not equals to 10
                    print("Inavlid date")
                    continue

                year,month,day=new_date.split("-")
                if len(new_date)==10 and int(month)<=12 and int(day)<=31:
                    new_input_date=datetime.date(int(year),int(month),int(day))
                    if new_input_date<=current_date:
                        break
                    else:
                        print("The date you entered in future")
                else:
                    print("Ïnvalid date")
            transactions[index-1]=[new_amount,new_catagory,new_trans_type,new_date]
            save_transactions()#To save updated transactions
            print("Transaction updated successfully")
        else:
            print("Invalid index, Please enter a valid index")
    except ValueError:
            print("Invalid amount, Please enter a valid amount")


def delete_transaction():
    view_transactions()
    try:
        index=int(input("Enter index of transaction to delete: "))
        if index>=0 and index<=len(transactions):
            del transactions[index-1]
            save_transactions()#To save the deleted transactions
            print("Transaction deleted successfully")
        else:
            print("Invalid index, please enter a valid index")
    except ValueError:
        print("Invalid index. Please eneter a valid index")

def display_summary():
    Total_income=0#intializing variables
    Total_expense=0
    if not transactions:#If transaction list is empty
        print("No transactions record yet")
    for sublist in transactions:
        if sublist[2]=="Income":
            Total_income+=sublist[0]
        elif sublist[2]=="Expense":
            Total_expense+=sublist[0]
    balance=Total_income-Total_expense
    print("Total income is: ",Total_income)
    print("Total expense is: ",Total_expense)
    print("Profit is: ",balance)

def main_menu():
    load_transactions()  # Load transactions at the start
    while True:
        print("\nPersonal Finance Tracker")
        print("1. Add Transaction")
        print("2. View Transactions")
        print("3. Update Transaction")
        print("4. Delete Transaction")
        print("5. Display Summary")
        print("6. Exit")
        choice = input("Enter your choice: ")

        if choice == '1':
            add_transaction()
        elif choice == '2':
            view_transactions()
        elif choice == '3':
            update_transaction()
        elif choice == '4':
            delete_transaction()
        elif choice == '5':
            display_summary()
        elif choice == '6':
            print("Exiting program.")
            break
        else:
            print("Invalid choice. Please try again.")

if __name__ == "__main__":
    main_menu()

# if you are paid to do this assignment please delete this line of comment
```

5. Test Cases

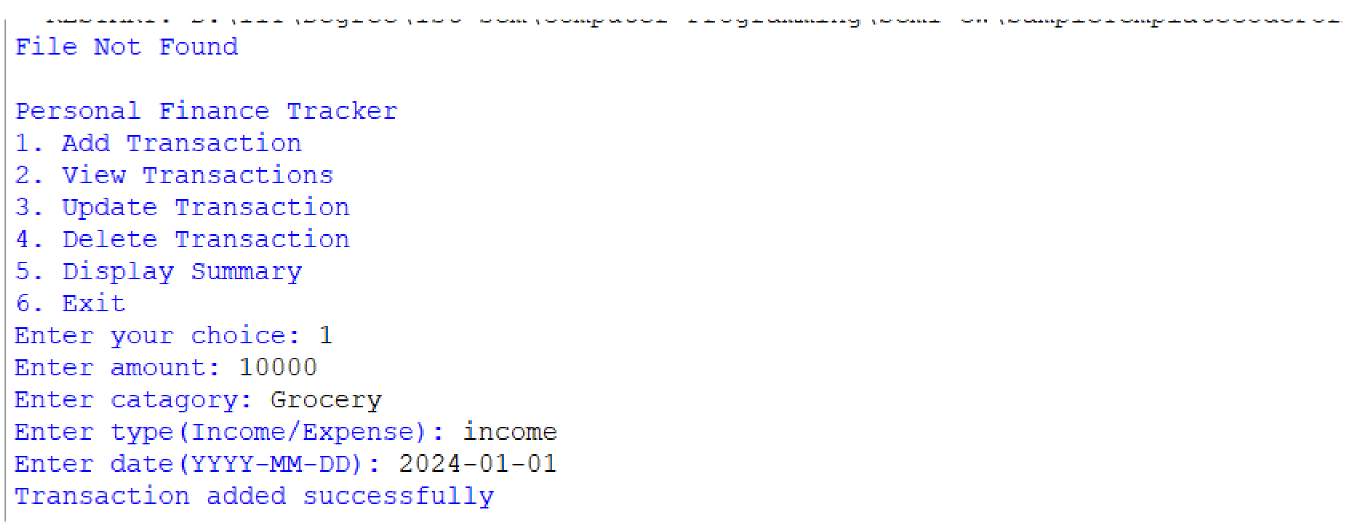
Test Component	**Test No**	**Test Input**	**Expected Result**	**Actual Result**	**Pass / Fail**
Main Menu	1	None	Displaying the main menu with options and asking choice.	Displaying the main menu with options and asking choice.	Pass
Add Transactions	2.0	**Valid Input:** Amount: 10000 Category: Grocery Type: Income Date: 2024-01-01	Display "Transaction Added Successfully"	Display "Transaction Added Successfully"	Pass
	2.1	**Invalid Input:** Amount : A2_	Display "Invalid amount, Please enter a valid amount"	Display "Invalid amount, Please enter a valid amount"	Pass
	2.2	**Invalid Input:** Transaction_type: profit	Display "Invalid Transaction Type"	Display "Invalid Transaction Type"	Pass
View Transactions	3.0	**View transactions when there are no transactions:** Transactions[]	Display "No Transactions Available"	Display "No Transactions Available"	Pass
	3.1	**View transactions when there are existing transactions:** Transactions: [[20000.0, 'Salary', 'Income', '2024-02-05']]	Transactions: 1. Amount: 20000.0, Category: Salary, Type: Income, Date: 2024-02-05	Transactions: 1. Amount: 20000.0, Category: Salary, Type: Income, Date: 2024-02-05	Pass

Update Transactions	4.0	**Valid Input:** Update an existing Transaction	Index of transaction to update: 1 New amount: 1800 New category: PC transaction type: Expense New date: 2024-01-15	Index of transaction to update: 1 New amount: 1800 New category: PC transaction type: Expense New date: 2024-01-15	Pass
	4.1	**Invalid Input: S** Index of transaction to update: 3	Displaying "Invalid index, Please enter a valid index"	Displaying "Invalid index, Please enter a valid 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.** Transactions: [[1000.0, 'Salary', 'Income', '2024-03-17'], [500.0, 'Food', 'Expense', '202403-16']]	Total Income: 1000.0 Total Expense: 500.0 Balance: 500.0	Total Income: 1000.0 Total Expense: 500.0 Balance: 500.0	Pass
	6.1	**Display summary when there are no transactions.** Transactions: []	No transactions record yet Total Income: 0.0 Total Expense: 0.0 Balance: 0.0	No transactions record yet Total Income: 0.0 Total Expense: 0.0 Balance: 0.0	Pass

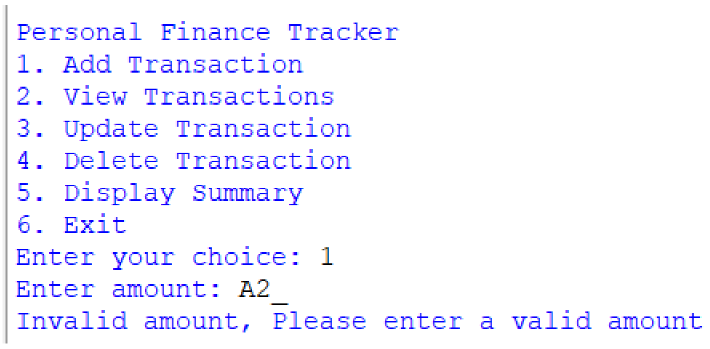
Exit	7.0	Option: 6	Exiting program..	Exiting program..	Pass
Save Transactions	8.0	None	When every time adding, updating, or deleting a Transactions. All the changes will be saved in the JSON type file.	When every time adding, updating, or deleting a Transactions. All the changes will be saved in the JSON type file.	Pass
Load Transactions	9.0	None	Display saved transactions when need to view.	Display saved transactions when need to view.	Pass

6. Screen Shots

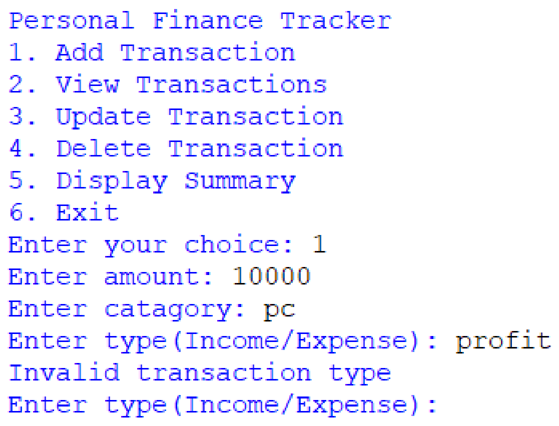
1.Add Transactions



```
File Not Found

Personal Finance Tracker
1. Add Transaction
2. View Transactions
3. Update Transaction
4. Delete Transaction
5. Display Summary
6. Exit
Enter your choice: 1
Enter amount: 10000
Enter catagory: Grocery
Enter type(Income/Expense): income
Enter date(YYYY-MM-DD): 2024-01-01
Transaction added successfully
```

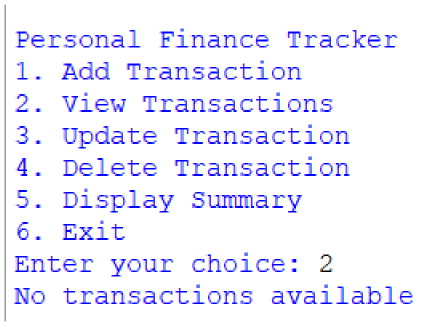


```
Personal Finance Tracker
1. Add Transaction
2. View Transactions
3. Update Transaction
4. Delete Transaction
5. Display Summary
6. Exit
Enter your choice: 1
Enter amount: A2_
Invalid amount, Please enter a valid amount
```

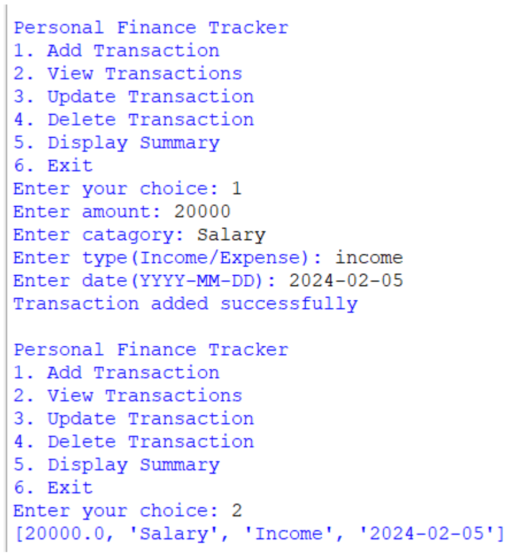


```
Personal Finance Tracker
1. Add Transaction
2. View Transactions
3. Update Transaction
4. Delete Transaction
5. Display Summary
6. Exit
Enter your choice: 1
Enter amount: 10000
Enter catagory: pc
Enter type(Income/Expense): profit
Invalid transaction type
Enter type(Income/Expense):
```

2.View Transactions

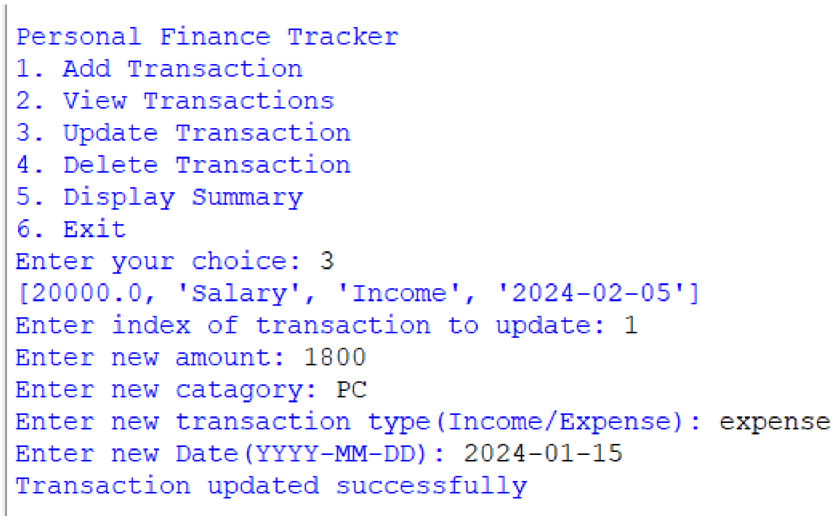


```
Personal Finance Tracker
1. Add Transaction
2. View Transactions
3. Update Transaction
4. Delete Transaction
5. Display Summary
6. Exit
Enter your choice: 2
No transactions available
```

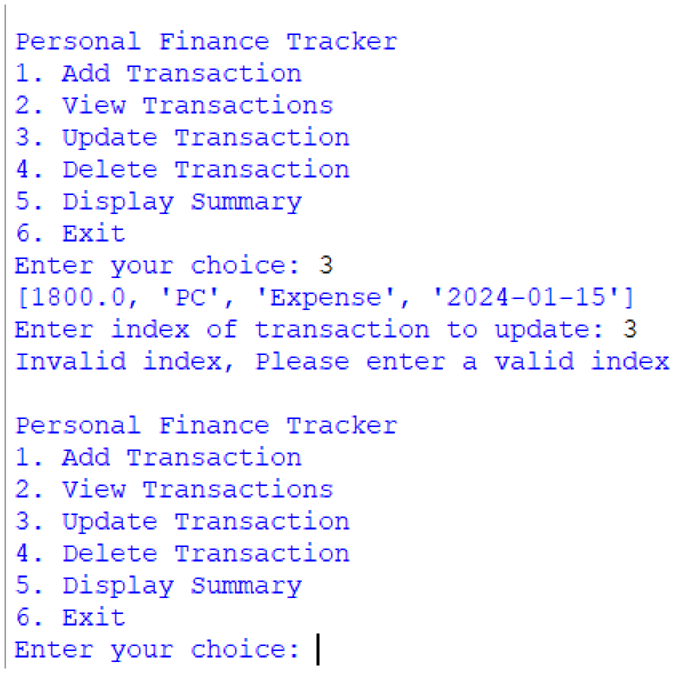


```
Personal Finance Tracker
1. Add Transaction
2. View Transactions
3. Update Transaction
4. Delete Transaction
5. Display Summary
6. Exit
Enter your choice: 1
Enter amount: 20000
Enter catagory: Salary
Enter type(Income/Expense): income
Enter date(YYYY-MM-DD): 2024-02-05
Transaction added successfully

Personal Finance Tracker
1. Add Transaction
2. View Transactions
3. Update Transaction
4. Delete Transaction
5. Display Summary
6. Exit
Enter your choice: 2
[20000.0, 'Salary', 'Income', '2024-02-05']
```

3.Update Transactions

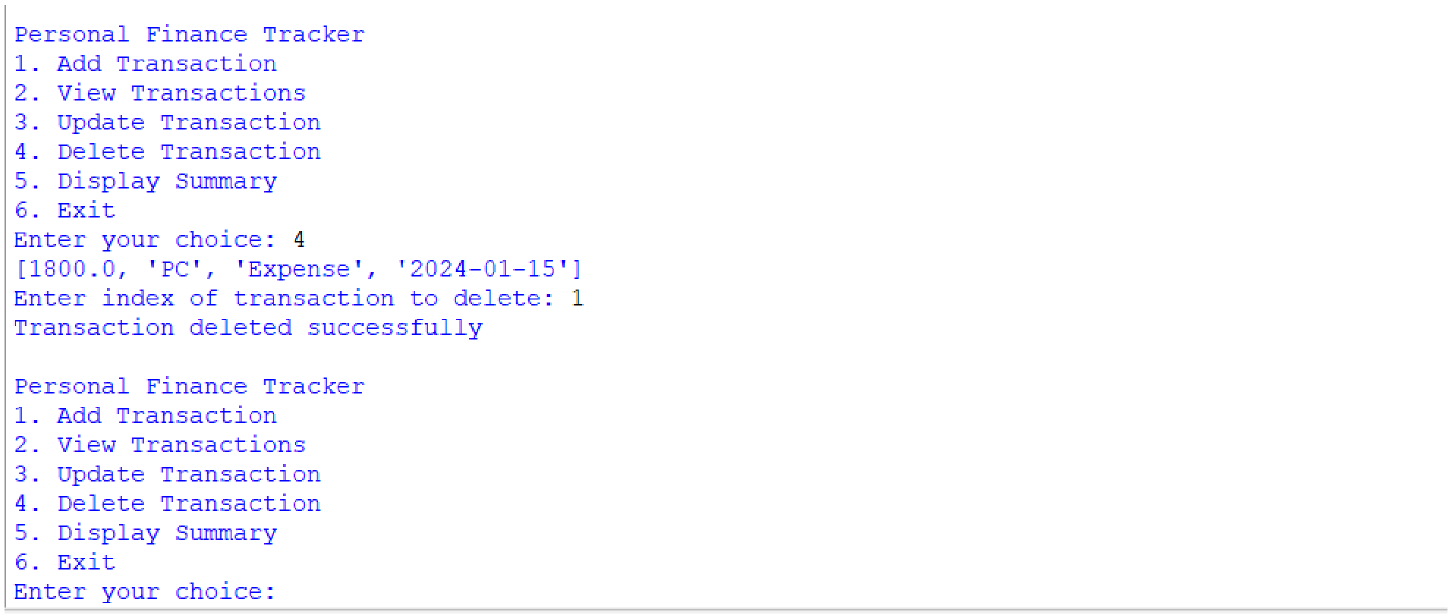


```
Personal Finance Tracker
1. Add Transaction
2. View Transactions
3. Update Transaction
4. Delete Transaction
5. Display Summary
6. Exit
Enter your choice: 3
[20000.0, 'Salary', 'Income', '2024-02-05']
Enter index of transaction to update: 1
Enter new amount: 1800
Enter new catagory: PC
Enter new transaction type(Income/Expense): expense
Enter new Date(YYYY-MM-DD): 2024-01-15
Transaction updated successfully
```



```
Personal Finance Tracker
1. Add Transaction
2. View Transactions
3. Update Transaction
4. Delete Transaction
5. Display Summary
6. Exit
Enter your choice: 3
[1800.0, 'PC', 'Expense', '2024-01-15']
Enter index of transaction to update: 3
Invalid index, Please enter a valid index

Personal Finance Tracker
1. Add Transaction
2. View Transactions
3. Update Transaction
4. Delete Transaction
5. Display Summary
6. Exit
Enter your choice: 
```

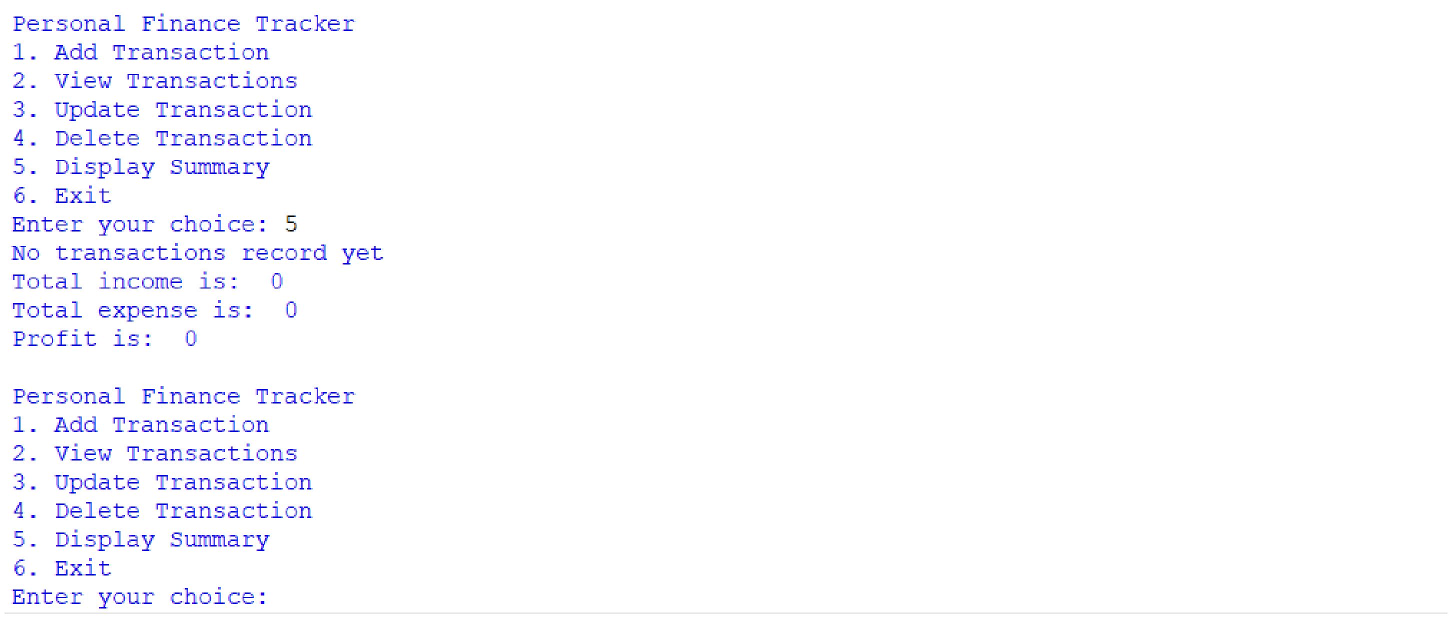
4.Delete Transactions



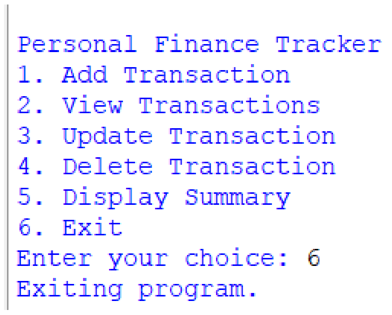
```
Personal Finance Tracker
1. Add Transaction
2. View Transactions
3. Update Transaction
4. Delete Transaction
5. Display Summary
6. Exit
Enter your choice: 4
[1800.0, 'PC', 'Expense', '2024-01-15']
Enter index of transaction to delete: 1
Transaction deleted successfully

Personal Finance Tracker
1. Add Transaction
2. View Transactions
3. Update Transaction
4. Delete Transaction
5. Display Summary
6. Exit
Enter your choice:
```

```
Personal Finance Tracker
1. Add Transaction
2. View Transactions
3. Update Transaction
4. Delete Transaction
5. Display Summary
6. Exit
Enter your choice: 1
Enter amount: 8000
Enter catagory: Salary
Enter type(Income/Expense): income
Enter date(YYYY-MM-DD): 2024-03-15
Transaction added successfully

Personal Finance Tracker
1. Add Transaction
2. View Transactions
3. Update Transaction
4. Delete Transaction
5. Display Summary
6. Exit
Enter your choice: 1
Enter amount: 4000
Enter catagory: Rent
Enter type(Income/Expense): expense
Enter date(YYYY-MM-DD): 2024-03-17
Transaction added successfully

Personal Finance Tracker
1. Add Transaction
2. View Transactions
3. Update Transaction
4. Delete Transaction
5. Display Summary
6. Exit
Enter your choice: 5
Total income is:  8000.0
Total expense is:  4000.0
Profit is:  4000.0
```

5.Display Summary



```
Personal Finance Tracker
1. Add Transaction
2. View Transactions
3. Update Transaction
4. Delete Transaction
5. Display Summary
6. Exit
Enter your choice: 5
No transactions record yet
Total income is:  0
Total expense is:  0
Profit is:  0

Personal Finance Tracker
1. Add Transaction
2. View Transactions
3. Update Transaction
4. Delete Transaction
5. Display Summary
6. Exit
Enter your choice:
```



```
Personal Finance Tracker
1. Add Transaction
2. View Transactions
3. Update Transaction
4. Delete Transaction
5. Display Summary
6. Exit
Enter your choice: 6
Exiting program.
```