INFORMATICS INSTITUTE OF TECHNOLOGY

UNIVERSITY OF WESTMINSTER

University of Westminster
Software Development I 4COSC006C

pseudo code

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

Python code

```
  import json

# Global dictionary to store transactions
transactions = {}

# File handling functions
def load_transactions():
    global transactions
    try:
        with open('transactions.json', 'r') as file:
            transactions = json.load(file)
    except FileNotFoundError:
        print("Transactions file not found.")
    except json.JSONDecodeError as e:
        print(f"Error decoding JSON: {e}")


def save_transactions():
    global transactions
    with open('transactions.json', 'w') as file:
        json.dump(transactions, file, indent=2)

def read_bulk_transactions_from_file(filename):
    global transactions
    try:
        with open(filename, 'r') as file:
            transactions = json.load(file)
    except FileNotFoundError:
     print("File not found.")

# Feature implementations
def add_transaction():
    global transactions
    category = input("Enter category: ")
    amount = float(input("Enter amount: "))
    date = input("Enter date (YYYY-MM-DD): ")
    if category in transactions:
        transactions[category].append({"amount": amount, "date": date})
    else:
        transactions[category] = [{"amount": amount, "date": date}]
    print("Transaction added successfully.")

def view_transactions():
    global transactions
    print(json.dumps(transactions, indent=2))

def update_transaction():
    global transactions
    category = input("Enter category to update: ")
    if category in transactions:
        print(f"Current transactions under {category}:")
        for index, expense in enumerate(transactions[category], start=1):
            print(f"{index}. Amount: {expense['amount']}, Date:
{expense['date']}")
        choice = input("Enter transaction number to update: ")
        if choice.isdigit():
            choice = int(choice)
            if 1 <= choice <= len(transactions[category]):
                new_amount = float(input("Enter new amount: "))
                new_date = input("Enter new date (YYYY-MM-DD): ")
                transactions[category][choice-1] = {"amount": new_amount,
"date": new_date}
                print("Transaction updated successfully.")
            else:
                print("Invalid transaction number.")
        else:
            print("Invalid input. Please enter a valid number.")
    else:
        print("Category not found.")

def delete_transaction():
    global transactions
    category = input("Enter category to delete: ")
    if category in transactions:
        print(f"Current transactions under {category}:")
        for index, expense in enumerate(transactions[category], start=1):
            print(f"{index}. Amount: {expense['amount']}, Date:
{expense['date']}")
        choice = input("Enter transaction number to delete: ")
```

```python
        if choice.isdigit():
            choice = int(choice)
            if 1 <= choice <= len(transactions[category]):
                del transactions[category][choice-1]
                print("Transaction deleted successfully.")
            else:
                print("Invalid transaction number.")
        else:
            print("Invalid input. Please enter a valid number.")
    else:
        print("Category not found.")

def display_summary():
    global transactions
    for category, expenses in transactions.items():
        total_amount = sum(expense['amount'] for expense in expenses)
        print(f"{category}: Total Amount - {total_amount}, Count -
{len(expenses)}")

def main_menu():
    load_transactions()
    while True:
        print("\n===== Personal Finance Tracker =====")
        print("1. Add Transaction")
        print("2. View Transactions")
        print("3. Update Transaction")
        print("4. Delete Transaction")
        print("5. Display Summary")
        print("6. Read Bulk Transactions from File")
        print("7. Save Transactions to File")
        print("8. 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':
            filename = input("Enter filename to read transactions from: ")
            read_bulk_transactions_from_file(filename)
        elif choice == '7':
            save_transactions()
        elif choice == '8':
            save_transactions()
            print("Exiting...")
            break
        else:
            print("Invalid choice. Please enter a number between 1 and 
8.")

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

Pseudo code

```
BEGIN
Initialize global variable transactions as an empty dictionary

Function load_transactions():
    Attempt to open transactions.json file for reading
    If the file is not found:
        Print "Transactions file not found."
    If there is an error decoding JSON:
        Print the error message
    Otherwise:
        Load the contents of the file into the transactions dictionary

Function save_transactions():
    Open transactions.json file for writing
    Write the contents of the transactions dictionary to the file in JSON format with an
indentation of 2 spaces
```

```
Function read_bulk_transactions_from_file(filename):
    Attempt to open the specified filename for reading
    If the file is not found:
        Print "File not found."
    Otherwise:
        Load the contents of the file into the transactions dictionary

Function add_transaction():
    Prompt the user to enter a category, amount, and date
    If the category already exists in transactions:
        Append a new transaction to the existing category
    Otherwise:
        Create a new category in transactions with the entered transaction
    Print "Transaction added successfully."

Function view_transactions():
    Print the contents of the transactions dictionary in JSON format with an indentation of
2 spaces

Function update_transaction():
    Prompt the user to enter a category to update
    If the category exists in transactions:
        Display the current transactions under the specified category
        Prompt the user to enter the transaction number to update
        If the entered number is valid:
            Prompt the user to enter a new amount and date
            Update the selected transaction with the new amount and date
            Print "Transaction updated successfully."
        Otherwise, print "Invalid transaction number."
    Otherwise, print "Category not found."

Function delete_transaction():
    Prompt the user to enter a category to delete
    If the category exists in transactions:
        Display the current transactions under the specified category
        Prompt the user to enter the transaction number to delete
        If the entered number is valid:
            Delete the selected transaction
            Print "Transaction deleted successfully."
        Otherwise, print "Invalid transaction number."
    Otherwise, print "Category not found."

Function display_summary():
    Iterate over each category and its corresponding expenses in transactions
    For each category, calculate the total amount and the count of transactions
    Print the category name, total amount, and count of transactions

Function main_menu():
    Load transactions from the file
    Display the main menu options in a loop:
        Prompt the user to enter a choice
        Depending on the choice, call the corresponding function
        If the choice is 8, save transactions to the file, print "Exiting...", and break out of the
loop

If the script is run as the main program:
    Call the main_menu function
END.
```