

Personal Finance Tracker Code :

```
import json
import csv
import os
from datetime import datetime
from collections import defaultdict

DATA_FILE = "expenses.json"
BACKUP_FILE = "expenses_backup.json"
BUDGET_FILE = "budget.json"

# ----- Utility Functions -----
def load_data():
    if os.path.exists(DATA_FILE):
        with open(DATA_FILE, "r") as f:
            return json.load(f)
    return []

def save_data(data):
    with open(DATA_FILE, "w") as f:
        json.dump(data, f, indent=4)

def load_budget():
    if os.path.exists(BUDGET_FILE):
        with open(BUDGET_FILE, "r") as f:
            return json.load(f)
    return {}

def save_budget(budget):
    with open(BUDGET_FILE, "w") as f:
        json.dump(budget, f, indent=4)

# ----- Core Features -----
def add_expense(expenses):
    print("\n--- ADD NEW EXPENSE ---")
    amount = float(input("Enter amount: "))
    category = input("Enter category: ")
    description = input("Enter description: ")
```

```
def add_expense(expenses):
    description = input("Enter description: ")
    date = input("Enter date (YYYY-MM-DD) [leave blank for today]: ")

    if not date:
        date = datetime.today().strftime("%Y-%m-%d")

    expense = {
        "amount": amount,
        "category": category,
        "description": description,
        "date": date
    }

    expenses.append(expense)
    save_data(expenses)
    print("Expense added successfully!")

def view_expenses(expenses):
    print("\n--- ALL EXPENSES ---")
    if not expenses:
        print("No expenses found.")
        return

    for i, e in enumerate(expenses, 1):
        print(f"{i}. {e['date']} | {e['category']} | ₹{e['amount']} | {e['description']}")

def search_expenses(expenses):
    print("\n--- SEARCH EXPENSES ---")
    keyword = input("Enter category or description keyword: ").lower()

    results = [e for e in expenses if keyword in e["category"].lower()
               or keyword in e["description"].lower()]

    if not results:
        print("No matching expenses found.")
        return

    for e in results:
        print(f"{e['date']} | {e['category']} | ₹{e['amount']} | {e['description']}")
```

The image shows a Windows 11 desktop environment. The primary focus is the Visual Studio Code (VS Code) editor, which is open to a file named `importjson.py`. The script is a Python program designed for managing expenses. It features several functions: `search_expenses` for querying a list of expenses by date, category, amount, or description; `monthly_report` for generating a report of expenses for a specific month; `category_breakdown` for analyzing expenses by category; `set_budget` for setting and updating a budget for different categories; and `export_csv` for saving the expense data to a CSV file. The script uses standard Python syntax, including loops, conditionals, and dictionary operations. The VS Code interface includes a sidebar with file explorer, search, and source control views. The Windows taskbar at the bottom displays various application icons, including the Start menu, search bar, and several open applications like Chrome, Edge, and various utility tools. The system tray shows the date and time as 11:51 on 03-01-2026.

```

File Edit Selection View Go Run Terminal Help
importjson.py X
C:\Users> Santhosh Kumar > importjson.py ...
70 def search_expenses(expenses):
71     for e in results:
72         print(f"{e['date']} | {e['category']} | ₹{e['amount']} | {e['description']}")
73
74
75 def monthly_report(expenses):
76     print("\n--- MONTHLY REPORT ---")
77     month = input("Enter month (YYYY-MM): ")
78
79     total = 0
80     for e in expenses:
81         if e["date"].startswith(month):
82             total += e["amount"]
83
84     print(f"Total expenses for {month}: ₹{total}")
85
86
87 def category_breakdown(expenses):
88     print("\n--- CATEGORY BREAKDOWN ---")
89     category_totals = defaultdict(float)
90
91     for e in expenses:
92         category_totals[e["category"]] += e["amount"]
93
94     for cat, total in category_totals.items():
95         print(f"{cat}: ₹{total}")
96
97
98 def set_budget():
99     print("\n--- SET / UPDATE BUDGET ---")
100     category = input("Enter category: ")
101     amount = float(input("Enter monthly budget amount: "))
102
103     budget = load_budget()
104     budget[category] = amount
105     save_budget(budget)
106
107     print("Budget updated successfully!")
108
109
110 def export_csv(expenses):
111     print("\n--- EXPORT TO CSV ---")

```

The screenshot displays a Windows 10 desktop with a Visual Studio Code (VS Code) editor open. The editor is showing a Python file named `importjson.py`. The code is a script for a personal finance tracker, featuring a `main()` function that runs a loop until the user chooses to exit. The script includes functions for loading data, adding expenses, viewing expenses, searching, generating reports, and exporting to CSV. The terminal window at the bottom shows the script being executed, and the file explorer on the left shows the project files.

```

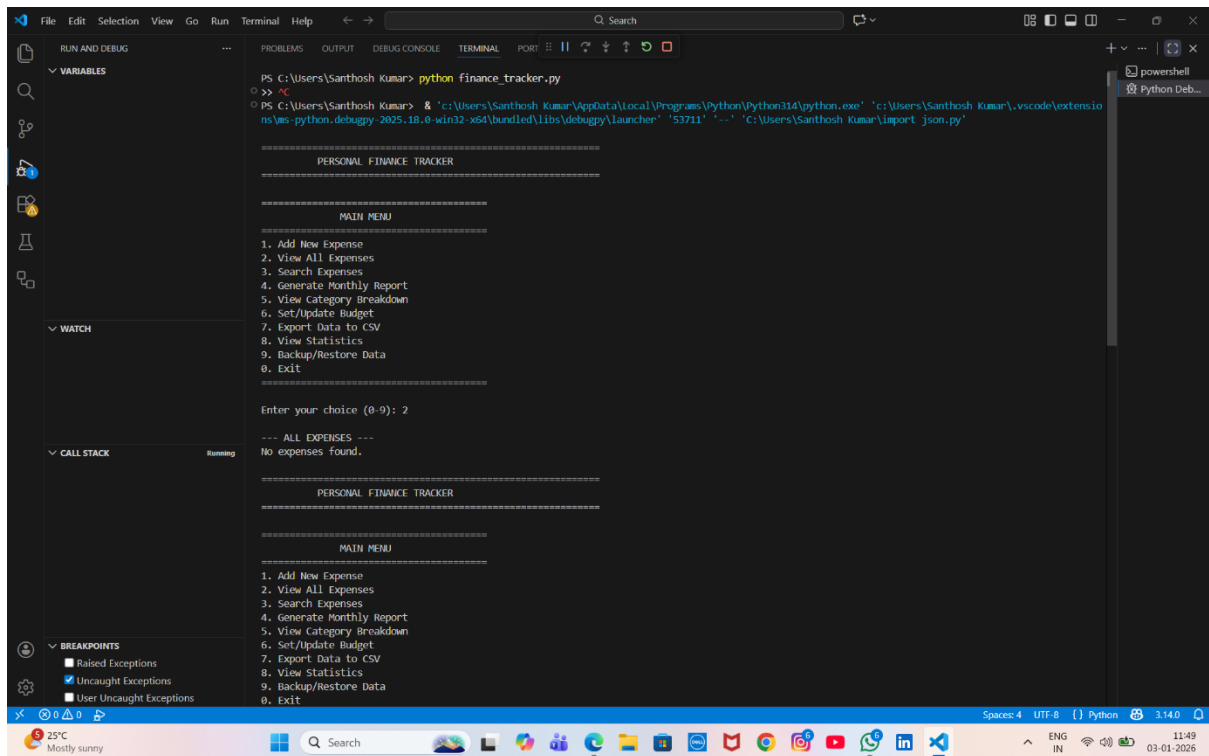
C:\Users> Santhosh Kumar > importjson.py > view_statistics
164         print("No backup file found.")
165
166
167 # ----- Main Menu -----
168 def main():
169     expenses = load_data()
170
171     while True:
172         print("\n" + "=" * 60)
173         print("                PERSONAL FINANCE TRACKER")
174         print("=" * 60)
175         print("\n=====")
176         print("                MAIN MENU")
177         print("=====")
178         print("1. Add New Expense")
179         print("2. View All Expenses")
180         print("3. Search Expenses")
181         print("4. Generate Monthly Report")
182         print("5. View Category Breakdown")
183         print("6. Set/Update Budget")
184         print("7. Export Data to CSV")
185         print("8. View Statistics")
186         print("9. Backup/Restore Data")
187         print("0. Exit")
188         print("=====")
189
190         choice = input("\nEnter your choice (0-9): ")
191
192         if choice == "1":
193             add_expense(expenses)
194         elif choice == "2":
195             view_expenses(expenses)
196         elif choice == "3":
197             search_expenses(expenses)
198         elif choice == "4":
199             monthly_report(expenses)
200         elif choice == "5":
201             category_breakdown(expenses)
202         elif choice == "6":
203             set_budget()
204         elif choice == "7":
205             export_csv(expenses)

```

The terminal window at the bottom shows the script being executed, and the file explorer on the left shows the project files.

Windows taskbar at the bottom shows the time as 11:51 AM on 03-01-2023. The system tray includes icons for network, volume, and battery.

Output:



```
PS C:\Users\Santhosh Kumar> python finance_tracker.py
>>>
PS C:\Users\Santhosh Kumar> & 'c:\Users\Santhosh Kumar\AppData\Local\Programs\Python\Python14\python.exe' 'c:\Users\Santhosh Kumar\.vscode\extensions\ms-python.debugpy-2025.18.0-win32-x64\bundle\libs\debugpy\launcher' '53711' '-...' 'C:\Users\Santhosh Kumar\import_json.py'

=====
PERSONAL FINANCE TRACKER
=====

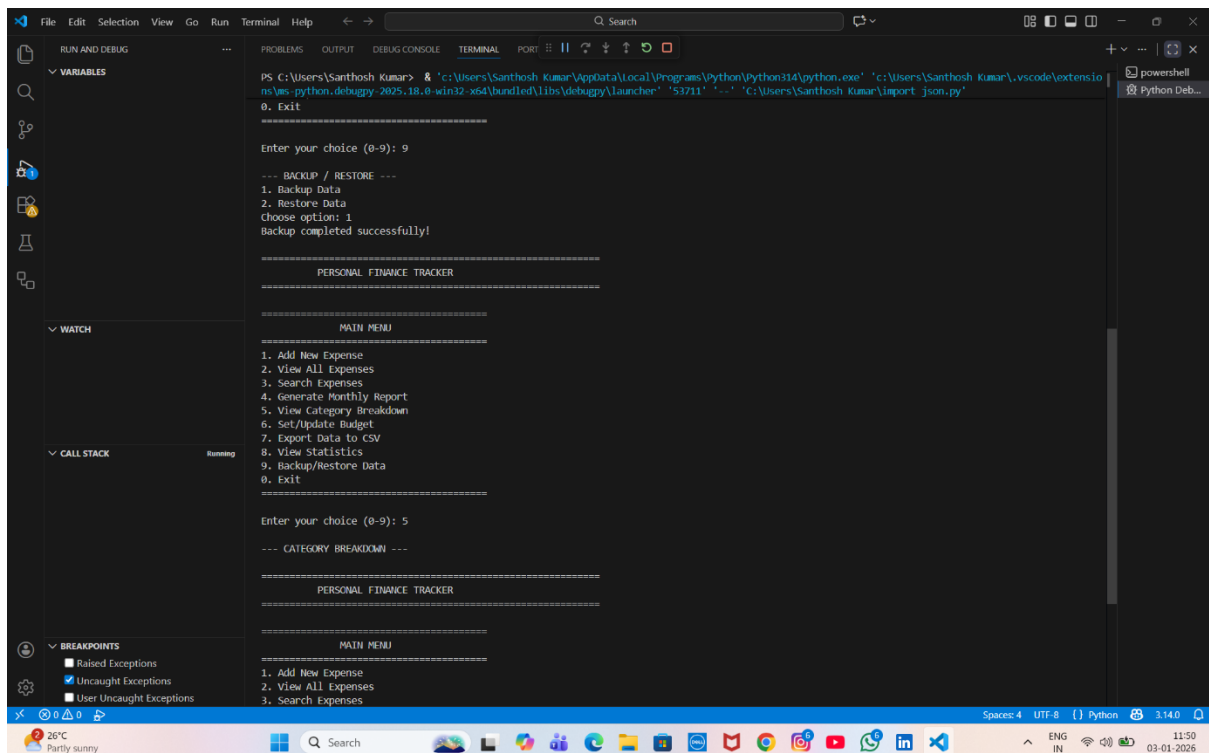
MAIN MENU
=====
1. Add New Expense
2. View All Expenses
3. Search Expenses
4. Generate Monthly Report
5. View Category Breakdown
6. Set/Update Budget
7. Export Data to CSV
8. View Statistics
9. Backup/Restore Data
0. Exit
=====

Enter your choice (0-9): 2

--- ALL EXPENSES ---
No expenses found.

=====
PERSONAL FINANCE TRACKER
=====

MAIN MENU
=====
1. Add New Expense
2. View All Expenses
3. Search Expenses
4. Generate Monthly Report
5. View Category Breakdown
6. Set/Update Budget
7. Export Data to CSV
8. View Statistics
9. Backup/Restore Data
0. Exit
```



```
PS C:\Users\Santhosh Kumar> & 'c:\Users\Santhosh Kumar\AppData\Local\Programs\Python\Python14\python.exe' 'c:\Users\Santhosh Kumar\.vscode\extensions\ms-python.debugpy-2025.18.0-win32-x64\bundle\libs\debugpy\launcher' '53711' '-...' 'C:\Users\Santhosh Kumar\import_json.py'
0. Exit
=====

Enter your choice (0-9): 9

--- BACKUP / RESTORE ---
1. Backup Data
2. Restore Data
Choose option: 1
Backup completed successfully!

=====
PERSONAL FINANCE TRACKER
=====

MAIN MENU
=====
1. Add New Expense
2. View All Expenses
3. Search Expenses
4. Generate Monthly Report
5. View Category Breakdown
6. Set/Update Budget
7. Export Data to CSV
8. View Statistics
9. Backup/Restore Data
0. Exit
=====

Enter your choice (0-9): 5

--- CATEGORY BREAKDOWN ---

=====
PERSONAL FINANCE TRACKER
=====

MAIN MENU
=====
1. Add New Expense
2. View All Expenses
3. Search Expenses
```