# Personal Finance Tracker:

**Project Overview and Objectives**

The **Personal Finance Tracker** is a comprehensive, menu-driven Python application designed to help users efficiently manage their personal expenses. It allows users to record daily expenses, categorize spending, monitor monthly budgets, and generate detailed financial reports.

**Objectives**

- Provide an easy-to-use interface for tracking expenses
- Store financial data persistently using file operations
- Generate meaningful insights through reports and statistics
- Apply modular programming principles
- Demonstrate error handling and data validation

---

**Features**

- Add, edit, and delete expenses with full validation
- Categorize expenses (Food, Transport, Entertainment, Bills, etc.)
- Save data to JSON file with automatic backups
- Load data on startup with error recovery
- Generate monthly expense reports
- View category-wise spending breakdown
- Set and track monthly budgets
- Export data to CSV for external analysis
- User-friendly menu interface

---

**Technologies Used**

- **Programming Language:** Python 3
- **Data Storage:** JSON, CSV
- **Libraries:** json, csv, datetime, os, shutil

---

**Setup and Installation Instructions**

**Prerequisites**

- Python 3.8 or higher installed

- Command line / terminal access

**Steps to Run the Application**

cd week4-finance-tracker

python run.py

---

**Project Directory Structure**

week4-finance-tracker/

|— finance_tracker/

| ├— __init__.py

| ├— expense_manager.py

| ├— file_handler.py

| ├— report_generator.py

| ├— budget_manager.py

| ├— statistics.py

| └— utils.py

|

|— data/

| ├— expenses.json

| ├— expenses_backup.json

|

|— run.py

|— README.md

---

**Code Structure Explanation**

**run.py**

- Entry point of the application
- Displays the main menu
- Handles user interaction and navigation

**expense_manager.py**

- Add, edit, delete, and search expenses
- Input validation and category enforcement

**file_handler.py**

- Handles JSON read/write operations
- Automatic backup creation
- CSV export functionality
- Robust error handling for missing or corrupt files

**report_generator.py**

- Generates monthly expense summaries
- Displays totals and trends

**budget_manager.py**

- Set and update monthly budgets
- Compare expenses against budget limits

**statistics.py**

- Provides insights like highest spending category
- Calculates averages and totals

**utils.py**

- Common helper functions (date validation, formatting, menus)

- # Technical Requirements – How They Were Met
- ## 1. File Operations (JSON & CSV)
- ✓ Expenses are saved to `expenses.json`
  ✓ Automatic backup stored as `expenses_backup.json`
  ✓ CSV export implemented for external analysis
- ## 2. Modular Code Structure
- ✓ Separate modules for expenses, reports, budgets, and files
  ✓ Clean separation of concerns
  ✓ Easy to maintain and extend
- ## 3. Error Handling
- ✓ Handles missing files on startup
  ✓ Recovers from corrupted JSON using backup
  ✓ Input validation for dates, amounts, and categories
- ## 4. Data Persistence
- ✓ All data is preserved between program runs
  ✓ Backup created before every save operation
- ## 5. Menu-Driven Interface
- ✓ Clear and user-friendly menu
  ✓ Input prompts with validation
- ## 6. Reporting and Analysis

- ✓ Monthly reports
  ✓ Category-wise breakdown
  ✓ Spending statistics
- **7. Budget Tracking**
- ✓ Monthly budget setting
  ✓ Alerts when budget is exceeded
- **8. Export Functionality**
- ✓ CSV export for Excel or Google Sheets
- **9. Code Readability**
- ✓ Meaningful function names
  ✓ Inline comments and documentation

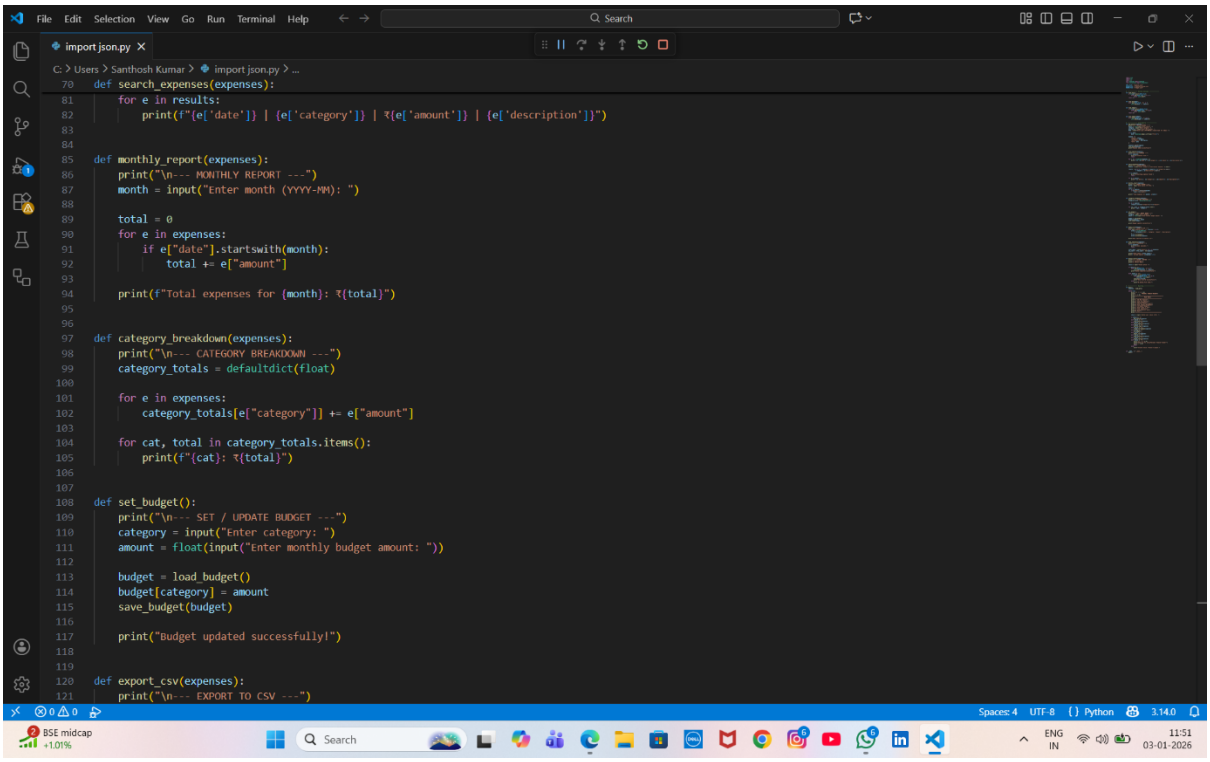
**Sample Main Menu**

```
========================================
        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
========================================
```

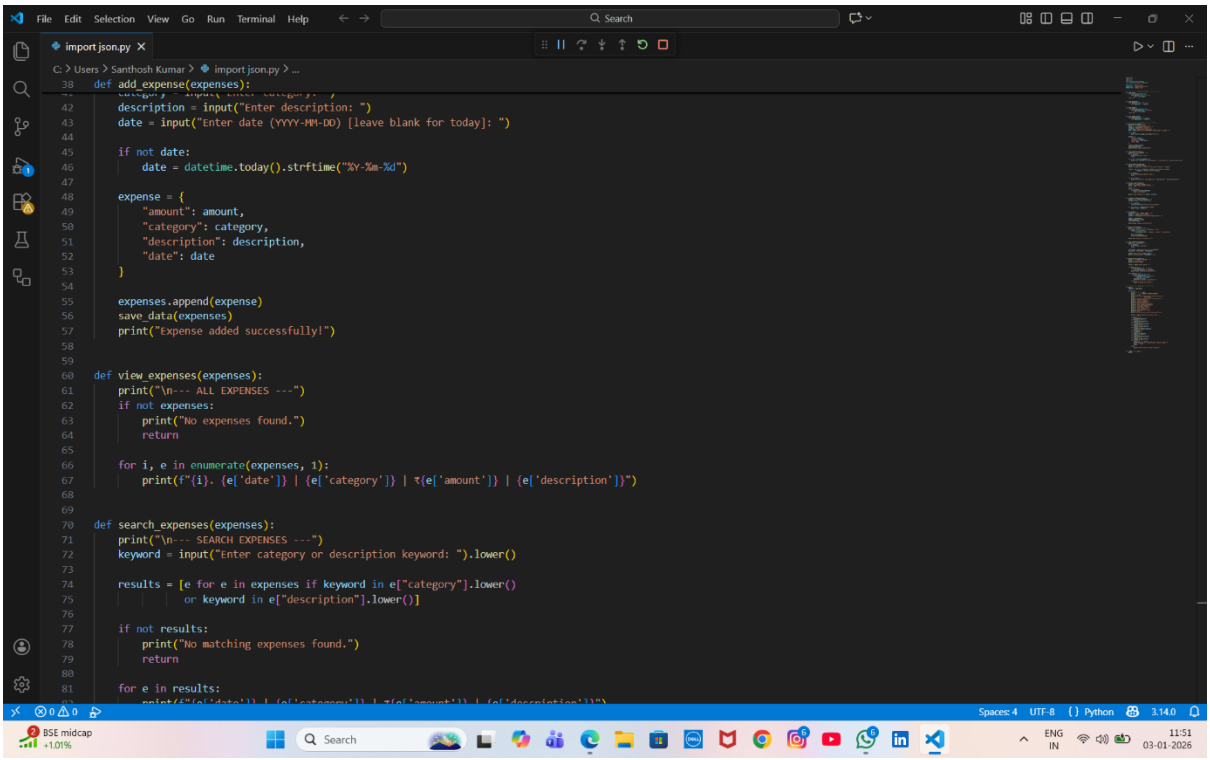
**Future Enhancements**

- GUI version using Tkinter or PyQt
- Data visualization with charts
- Cloud storage integration
- User authentication

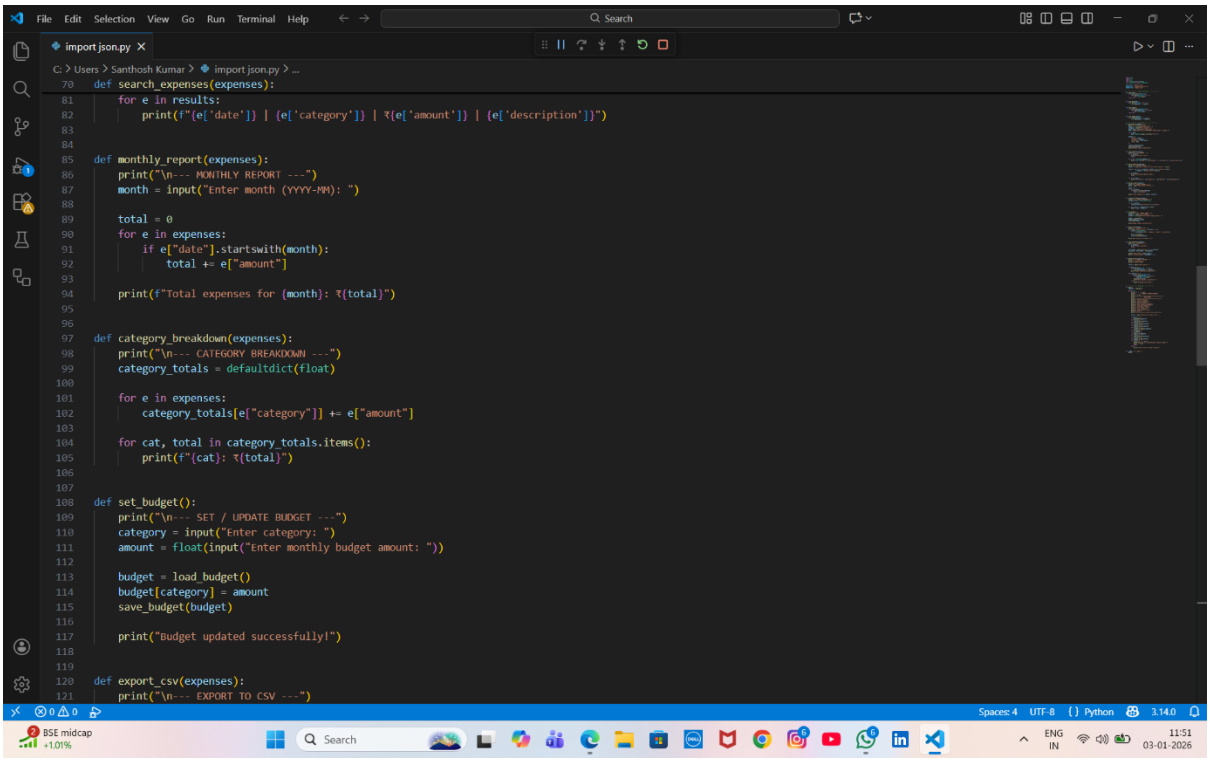
# . Personal Finance Tracker Code:



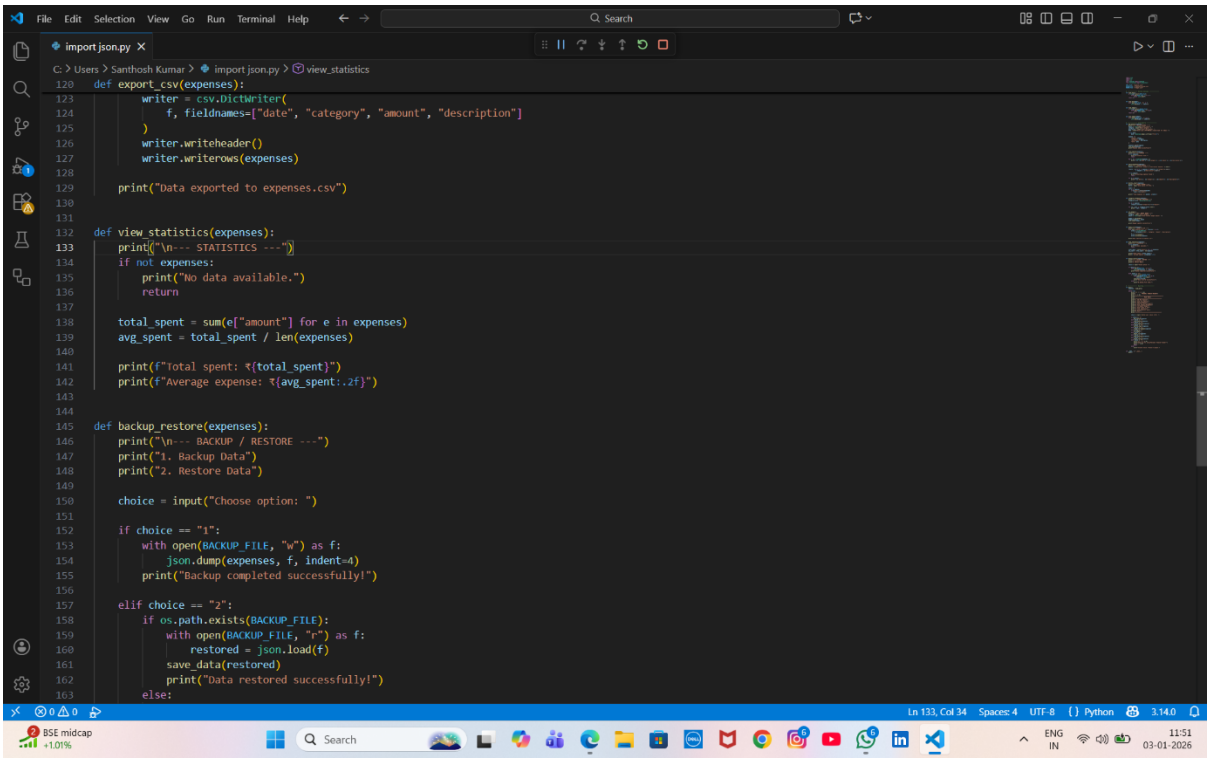
```python
def search_expenses(expenses):
    for e in results:
        print(f"{e['date']} | {e['category']} | ₹{e['amount']} | {e['description']}")


def monthly_report(expenses):
    print("\n--- MONTHLY REPORT ---")
    month = input("Enter month (YYYY-MM): ")

    total = 0
    for e in expenses:
        if e["date"].startswith(month):
            total += e["amount"]

    print(f"Total expenses for {month}: ₹{total}")


def category_breakdown(expenses):
    print("\n--- CATEGORY BREAKDOWN ---")
    category_totals = defaultdict(float)

    for e in expenses:
        category_totals[e["category"]] += e["amount"]

    for cat, total in category_totals.items():
        print(f"{cat}: ₹{total}")


def set_budget():
    print("\n--- SET / UPDATE BUDGET ---")
    category = input("Enter category: ")
    amount = float(input("Enter monthly budget amount: "))

    budget = load_budget()
    budget[category] = amount
    save_budget(budget)

    print("Budget updated successfully!")


def export_csv(expenses):
    print("\n--- EXPORT TO CSV ---")
```



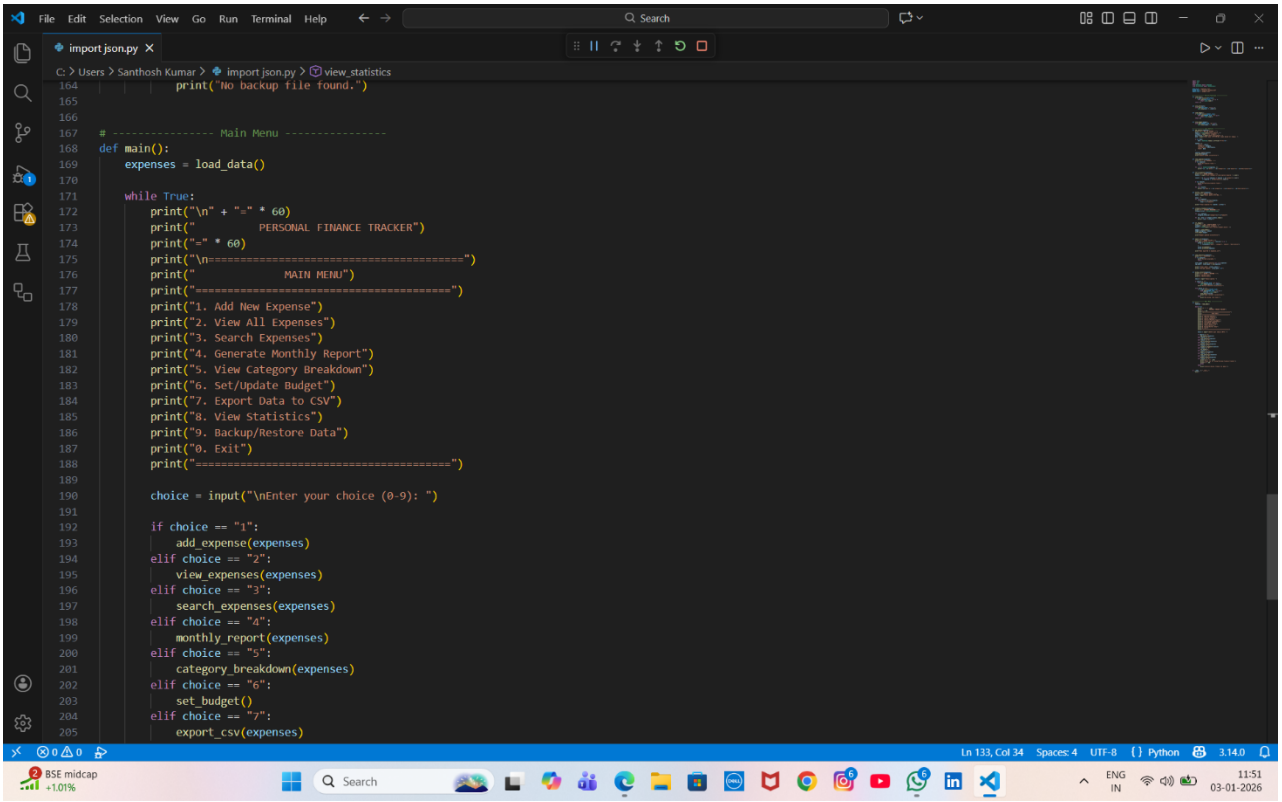
```python
def add_expense(expenses):
    description = input("Enter description: ")
    date = input("Enter date (YYYY-MM-DD) [leave blank for today]: ")

    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:
```

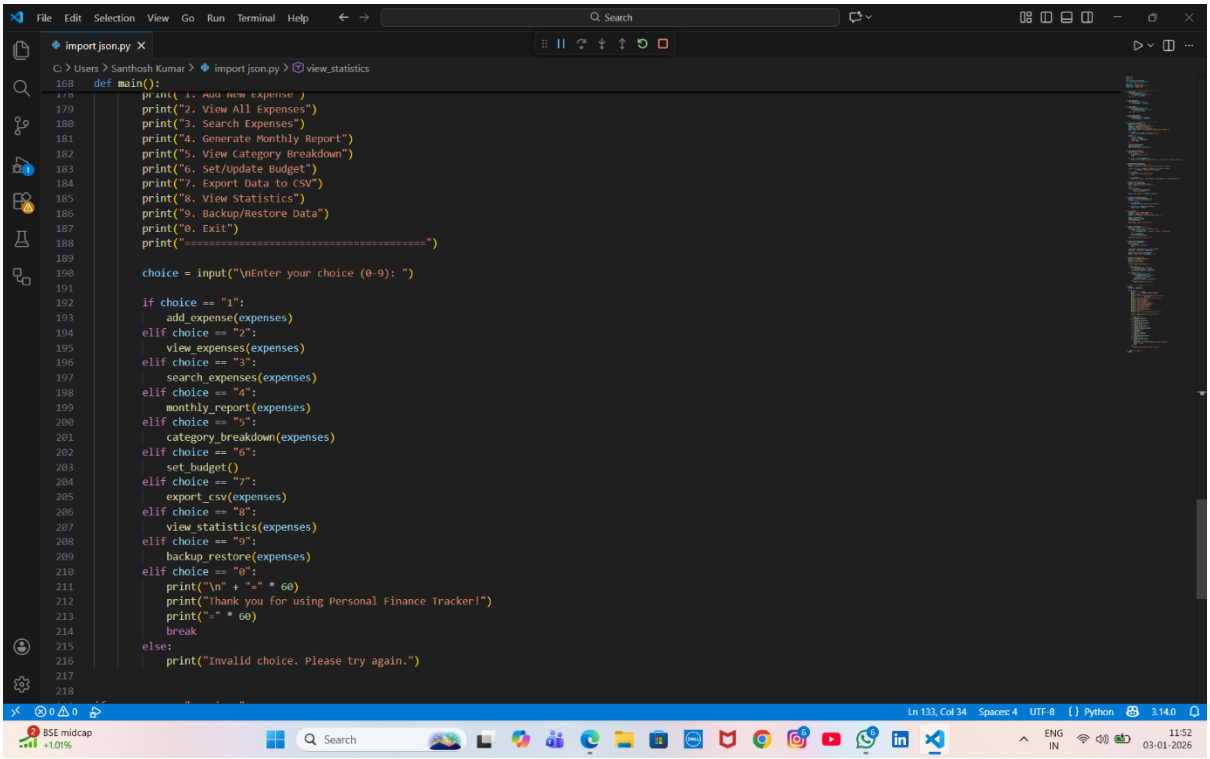


```python
def search_expenses(expenses):
    for e in results:
        print(f"{e['date']} | {e['category']} | ₹{e['amount']} | {e['description']}")


def monthly_report(expenses):
    print("\n--- MONTHLY REPORT ---")
    month = input("Enter month (YYYY-MM): ")

    total = 0
    for e in expenses:
        if e["date"].startswith(month):
            total += e["amount"]

    print(f"Total expenses for {month}: ₹{total}")


def category_breakdown(expenses):
    print("\n--- CATEGORY BREAKDOWN ---")
    category_totals = defaultdict(float)

    for e in expenses:
        category_totals[e["category"]] += e["amount"]

    for cat, total in category_totals.items():
        print(f"{cat}: ₹{total}")


def set_budget():
    print("\n--- SET / UPDATE BUDGET ---")
    category = input("Enter category: ")
    amount = float(input("Enter monthly budget amount: "))

    budget = load_budget()
    budget[category] = amount
    save_budget(budget)

    print("Budget updated successfully!")


def export_csv(expenses):
    print("\n--- EXPORT TO CSV ---")
```



```python
def export_csv(expenses):
        writer = csv.DictWriter(
            f, fieldnames=["date", "category", "amount", "description"]
        )
        writer.writeheader()
        writer.writerows(expenses)

    print("Data exported to expenses.csv")


def view_statistics(expenses):
    print("\n--- STATISTICS ---")
    if not expenses:
        print("No data available.")
        return

    total_spent = sum(e["amount"] for e in expenses)
    avg_spent = total_spent / len(expenses)

    print(f"Total spent: ₹{total_spent}")
    print(f"Average expense: ₹{avg_spent:.2f}")


def backup_restore(expenses):
    print("\n--- BACKUP / RESTORE ---")
    print("1. Backup Data")
    print("2. Restore Data")

    choice = input("Choose option: ")

    if choice == "1":
        with open(BACKUP_FILE, "w") as f:
            json.dump(expenses, f, indent=4)
        print("Backup completed successfully!")

    elif choice == "2":
        if os.path.exists(BACKUP_FILE):
            with open(BACKUP_FILE, "r") as f:
                restored = json.load(f)
            save_data(restored)
            print("Data restored successfully!")
        else:
```


File Edit Selection View Go Run Terminal Help
Search
import json.py
C: > Users > Santhosh Kumar > import json.py > view_statistics
print("No backup file found.")
# ---------- Main Menu ----------
def main():
expenses = load_data()
while True:
print("\n" + "=" * 60)
print(" PERSONAL FINANCE TRACKER")
print("=" * 60)
print("\n========================================")
print(" MAIN MENU")
print("========================================")
print("1. Add New Expense")
print("2. View All Expenses")
print("3. Search Expenses")
print("4. Generate Monthly Report")
print("5. View Category Breakdown")
print("6. Set/Update Budget")
print("7. Export Data to CSV")
print("8. View Statistics")
print("9. Backup/Restore Data")
print("0. Exit")
print("========================================")
choice = input("\nEnter your choice (0-9): ")
if choice == "1":
add_expense(expenses)
elif choice == "2":
view_expenses(expenses)
elif choice == "3":
search_expenses(expenses)
elif choice == "4":
monthly_report(expenses)
elif choice == "5":
category_breakdown(expenses)
elif choice == "6":
set_budget()
elif choice == "7":
export_csv(expenses)
Ln 133, Col 34 Spaces: 4 UTF-8 Python 3.14.0
BSE midcap +1.01%
Search
ENG IN
11:51 03-01-2026


File Edit Selection View Go Run Terminal Help
Search
import json.py
C: > Users > Santhosh Kumar > import json.py > view_statistics
def main():
print("2. View All Expenses")
print("3. Search Expenses")
print("4. Generate Monthly Report")
print("5. View Category Breakdown")
print("6. Set/Update Budget")
print("7. Export Data to CSV")
print("8. View Statistics")
print("9. Backup/Restore Data")
print("0. Exit")
print("========================================")
choice = input("\nEnter your choice (0-9): ")
if choice == "1":
add_expense(expenses)
elif choice == "2":
view_expenses(expenses)
elif choice == "3":
search_expenses(expenses)
elif choice == "4":
monthly_report(expenses)
elif choice == "5":
category_breakdown(expenses)
elif choice == "6":
set_budget()
elif choice == "7":
export_csv(expenses)
elif choice == "8":
view_statistics(expenses)
elif choice == "9":
backup_restore(expenses)
elif choice == "0":
print("\n" + "=" * 60)
print("Thank you for using Personal Finance Tracker!")
print("=" * 60)
break
else:
print("Invalid choice. Please try again.")
Ln 133, Col 34 Spaces: 4 UTF-8 Python 3.14.0
BSE midcap +1.01%
Search
ENG IN
11:52 03-01-2026

# Personal Finance Tracker output :