Project Title: **Hydration Coach System**

Submitted By: **JHANVI NAG**

Registration No. : **25MEI10012**

Course: **Introduction to problem solving and programming.**

Academic Year: **2025-26**

---

1. Introduction

The Hydration Coach is a beginner-friendly Python application designed to help users calculate their daily water intake based on body weight, provide hydration reminders, and demonstrate core programming concepts. This project applies problem-solving strategies, algorithmic thinking, modular design, and testing.

---

2. Problem Statement

People often forget to drink enough water throughout the day. Lack of hydration affects concentration, mood, and health. This project aims to solve the problem by calculating personalized water requirements and generating periodic reminders.

---

3. Functional Requirements

- Accept user data (name, weight, interval).
- Calculate water requirement in ml, liters, and ml remainder.
- Provide hydration reminders at defined time intervals.
- Display results clearly.
- Provide test cases for verification.

---

4. Non-functional Requirements

- Usability: Simple and readable for beginners.
- Reliability: Produces consistent output.
- Portability: Works on Google Colab, VS Code, or IDLE.
- Maintainability: Clean modular structure.

---

5. System Architecture

User → user_input.py → main.py → calculator.py → reminders.py → Output

---

6. Design Diagrams

Use Case Diagram

- User inputs details.
- System calculates water.
- System gives reminders.
- User views results.

Workflow Diagram

1. Start

2. Get user input

3. Calculate water

4. Show output

5. End

Sequence Diagram

User → Main → User_Input → Calculator → Reminders → Console Output

Class/Component Diagram:

HYDRATION_COACH (PACKAGE)

- calculator.py
- reminders.py
- user_input.py
- main.py

ER Diagram:

(Not applicable – no database used.)

---

7. Design Decisions & Rationale:

Modular Approach: Easier debugging and readability.

Simple Dictionaries for User Data: Beginner-friendly.

Functions Instead of Classes: Reduces complexity.

No External Dependency: Ensures easy execution.

---

8. Implementation Details:

Variables, expressions, statements used throughout.

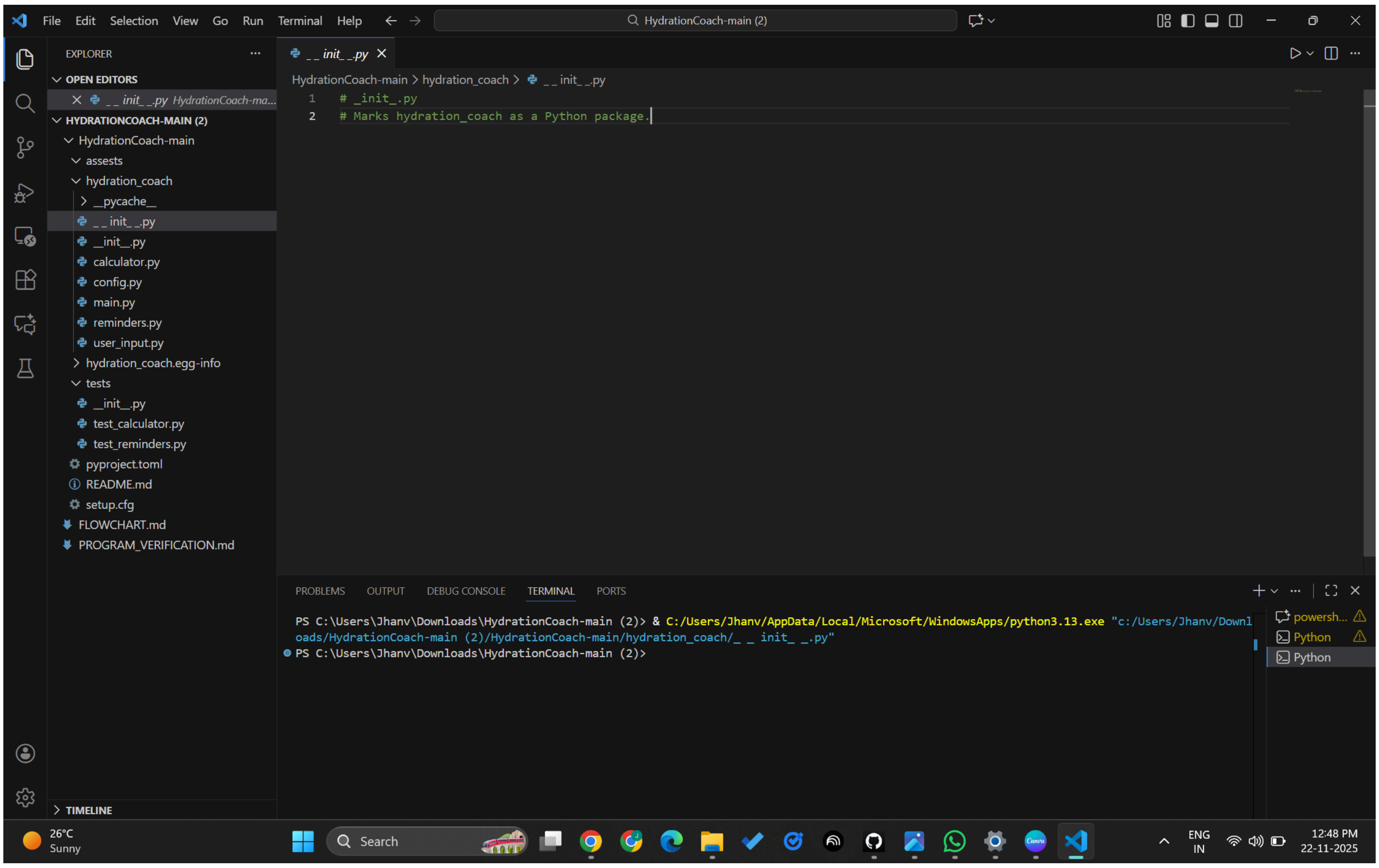
Tuple assignment in liters_and_ml().

Conditionals validate input.

Modules interact via imports

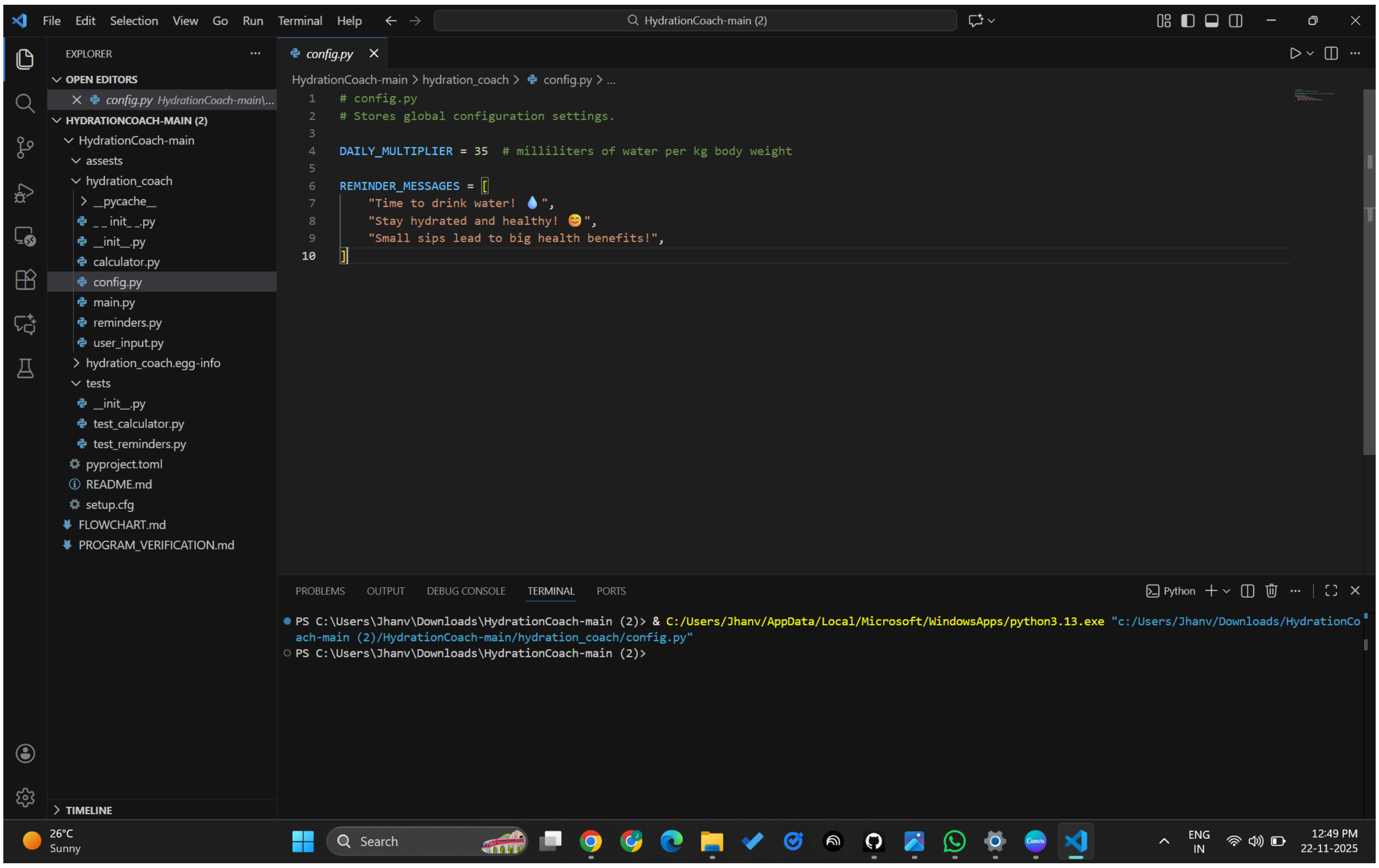
All functions tested.

9.SCREENSHOTS/RESULTS+FLOWCHART



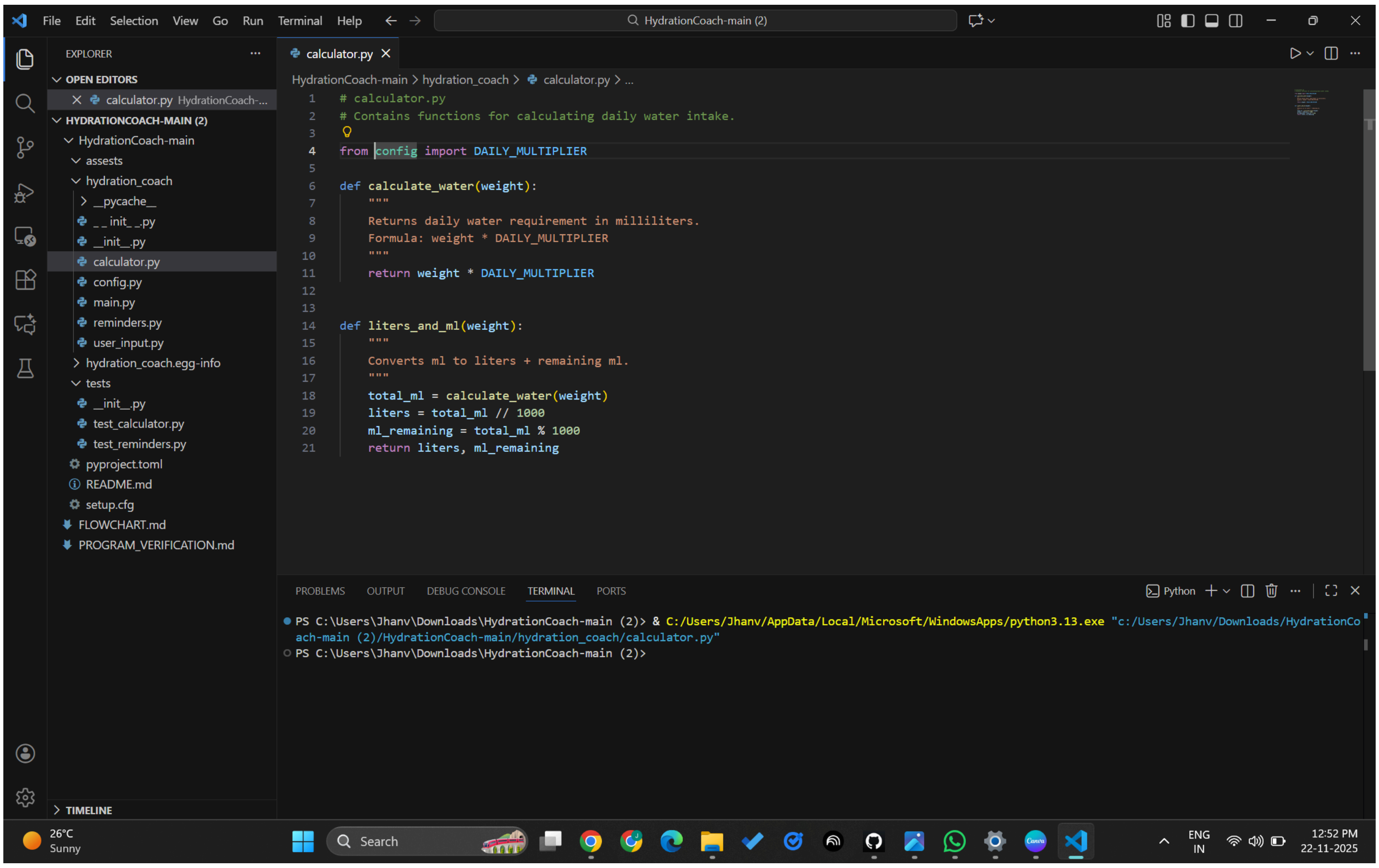
```
# _init_.py
# Marks hydration_coach as a Python package.
```

```
PS C:\Users\Jhanv\Downloads\HydrationCoach-main (2)> & C:/Users/Jhanv/AppData/Local/Microsoft/WindowsApps/python3.13.exe "c:/Users/Jhanv/Downl
oads/HydrationCoach-main (2)/HydrationCoach-main/hydration_coach/_ _ init_ _.py"
PS C:\Users\Jhanv\Downloads\HydrationCoach-main (2)>
```



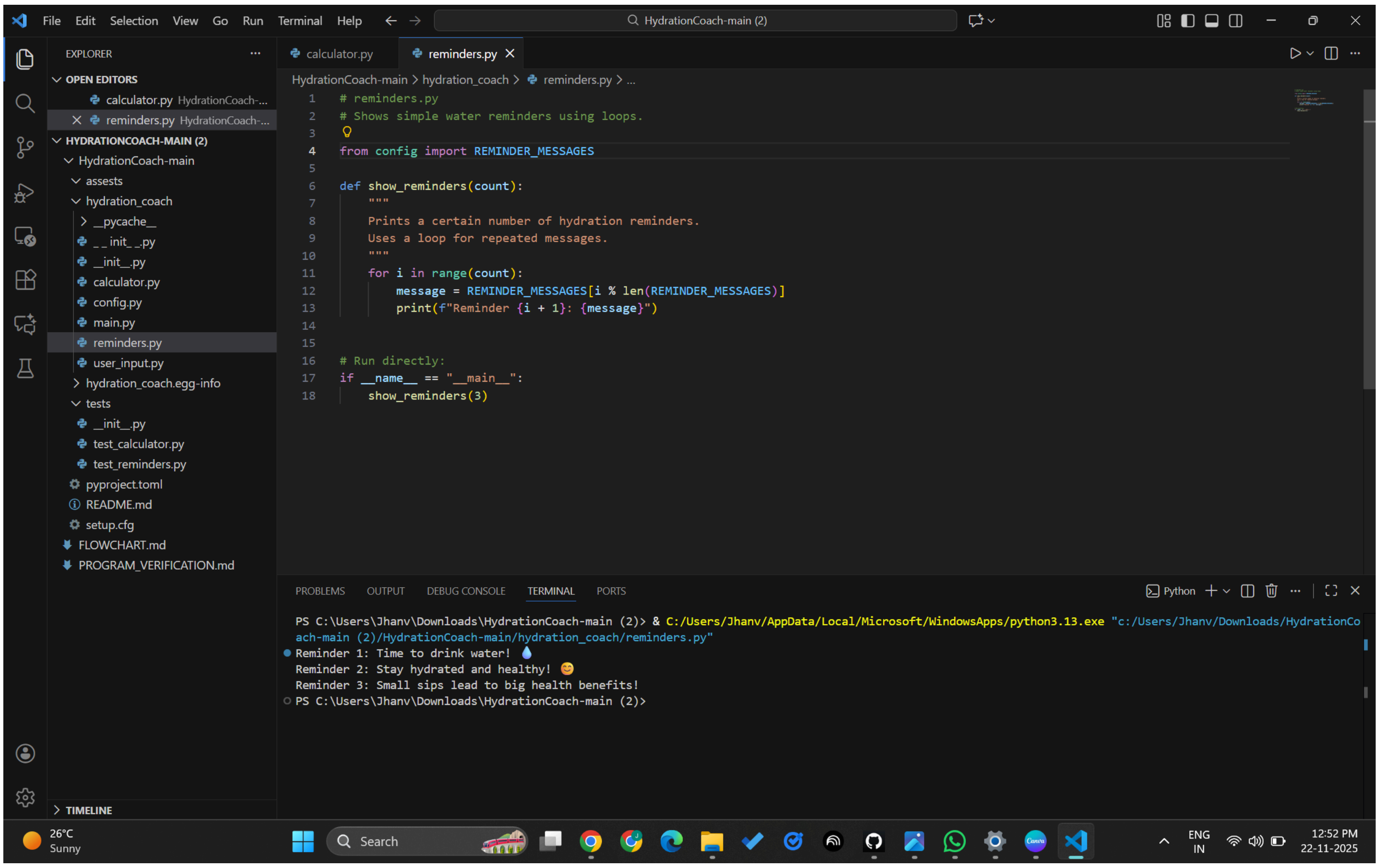
```
# config.py
# Stores global configuration settings.

DAILY_MULTIPLIER = 35  # milliliters of water per kg body weight

REMINDER_MESSAGES = [
    "Time to drink water! 💧",
    "Stay hydrated and healthy! 😊",
    "Small sips lead to big health benefits!",
]
```

```
PS C:\Users\Jhanv\Downloads\HydrationCoach-main (2)> & C:/Users/Jhanv/AppData/Local/Microsoft/WindowsApps/python3.13.exe "c:/Users/Jhanv/Downloads/HydrationCo
ach-main (2)/HydrationCoach-main/hydration_coach/config.py"
PS C:\Users\Jhanv\Downloads\HydrationCoach-main (2)>
```



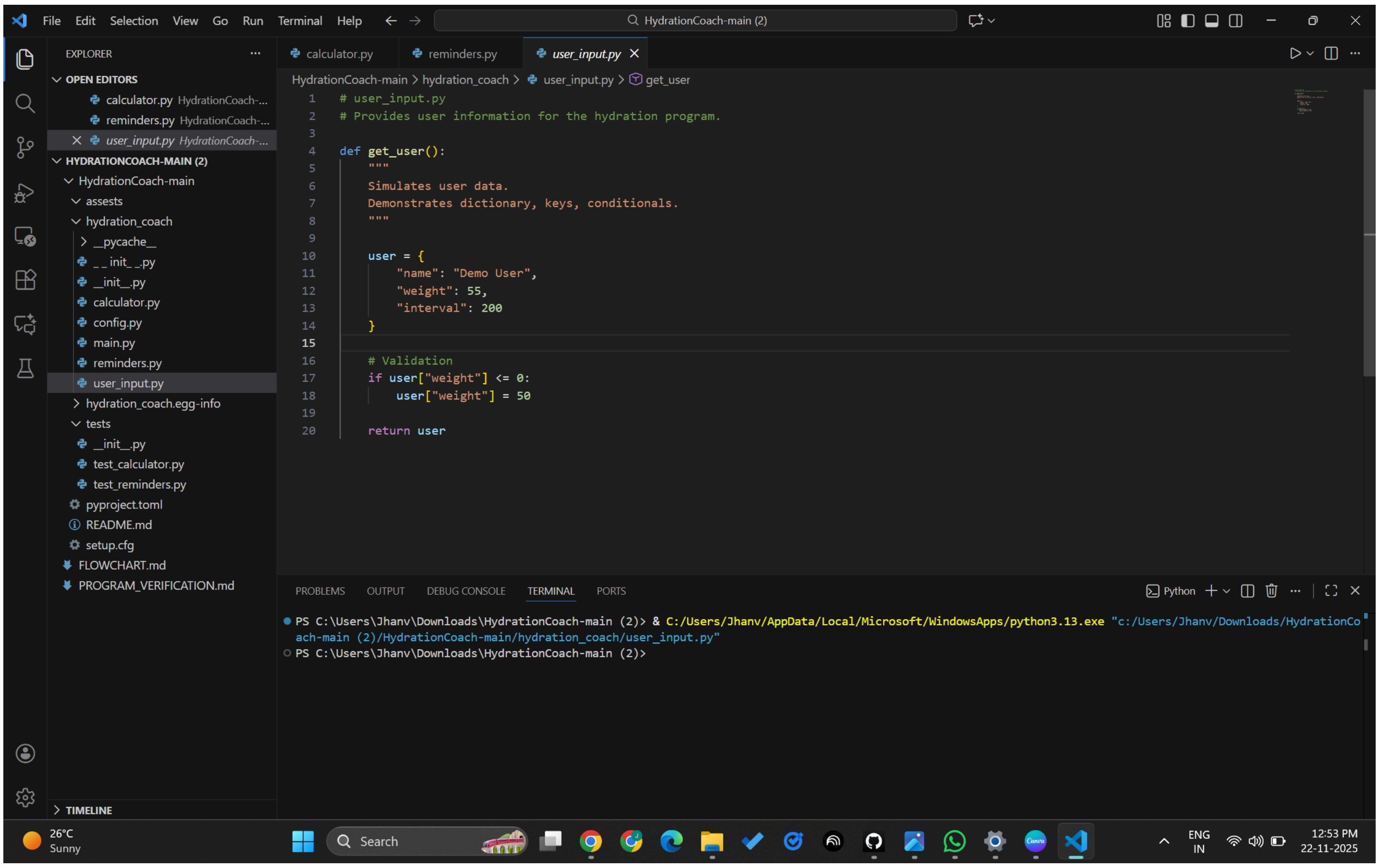
```python
# calculator.py
# Contains functions for calculating daily water intake.

from config import DAILY_MULTIPLIER

def calculate_water(weight):
    """
    Returns daily water requirement in milliliters.
    Formula: weight * DAILY_MULTIPLIER
    """
    return weight * DAILY_MULTIPLIER


def liters_and_ml(weight):
    """
    Converts ml to liters + remaining ml.
    """
    total_ml = calculate_water(weight)
    liters = total_ml // 1000
    ml_remaining = total_ml % 1000
    return liters, ml_remaining
```

```
PS C:\Users\Jhanv\Downloads\HydrationCoach-main (2)> & C:/Users/Jhanv/AppData/Local/Microsoft/WindowsApps/python3.13.exe "c:/Users/Jhanv/Downloads/HydrationCo
ach-main (2)/HydrationCoach-main/hydration_coach/calculator.py"
PS C:\Users\Jhanv\Downloads\HydrationCoach-main (2)>
```



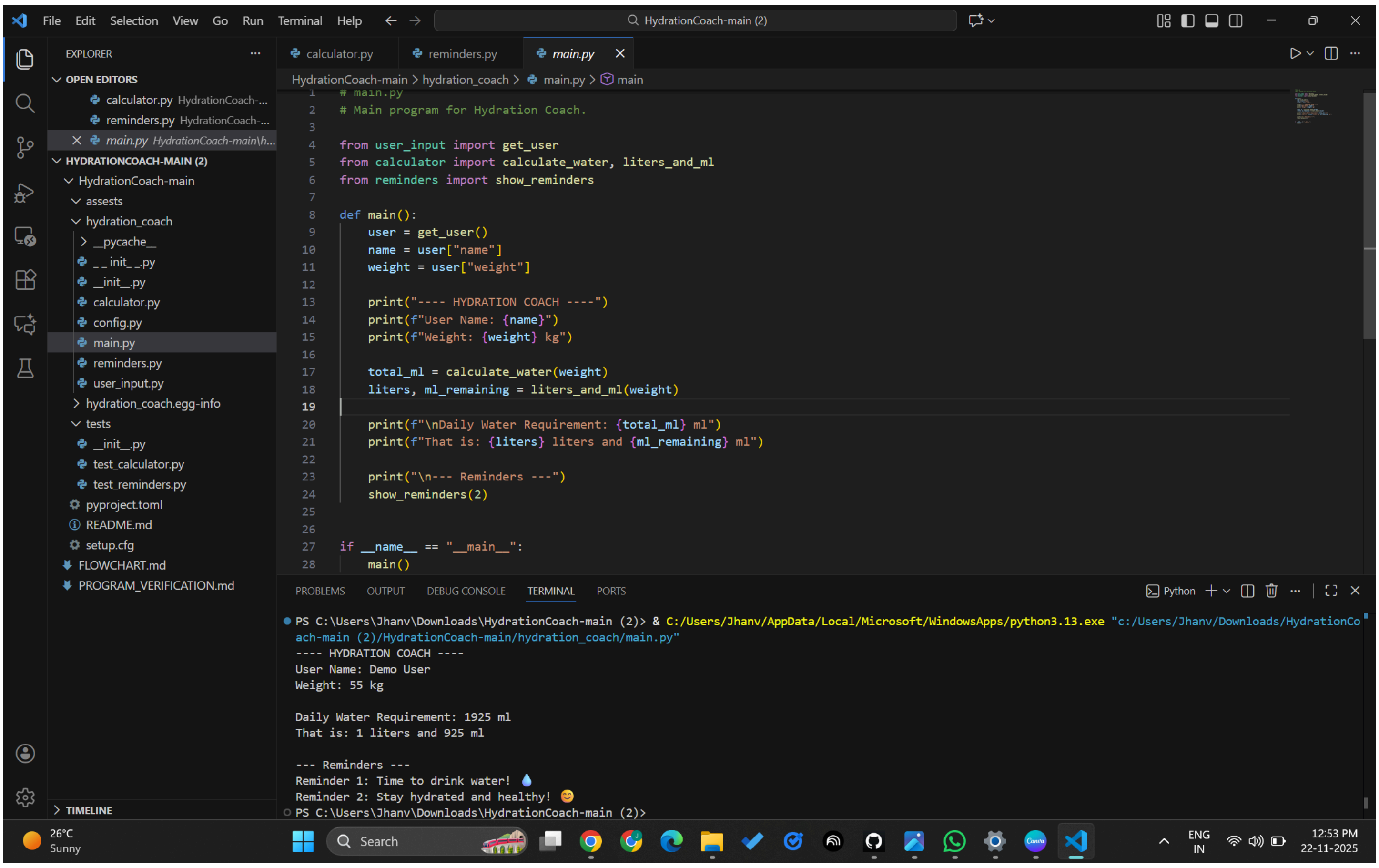
```python
# reminders.py
# Shows simple water reminders using loops.

from config import REMINDER_MESSAGES

def show_reminders(count):
    """
    Prints a certain number of hydration reminders.
    Uses a loop for repeated messages.
    """
    for i in range(count):
        message = REMINDER_MESSAGES[i % len(REMINDER_MESSAGES)]
        print(f"Reminder {i + 1}: {message}")


# Run directly:
if __name__ == "__main__":
    show_reminders(3)
```

```
PS C:\Users\Jhanv\Downloads\HydrationCoach-main (2)> & C:/Users/Jhanv/AppData/Local/Microsoft/WindowsApps/python3.13.exe "c:/Users/Jhanv/Downloads/HydrationCo
ach-main (2)/HydrationCoach-main/hydration_coach/reminders.py"
Reminder 1: Time to drink water! 💧
Reminder 2: Stay hydrated and healthy! 😊
Reminder 3: Small sips lead to big health benefits!
PS C:\Users\Jhanv\Downloads\HydrationCoach-main (2)>
```



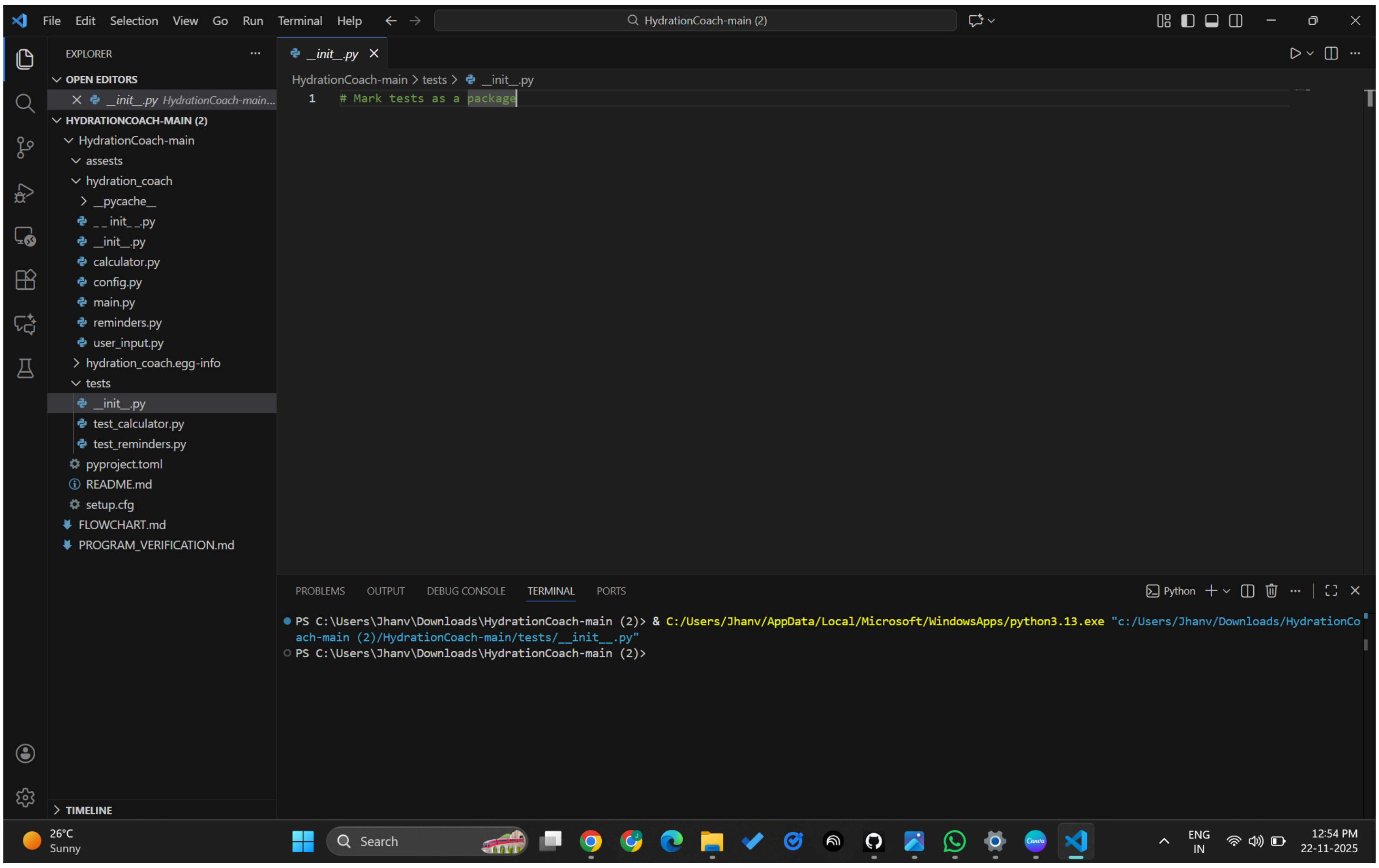
```python
# user_input.py
# Provides user information for the hydration program.

def get_user():
    """
    Simulates user data.
    Demonstrates dictionary, keys, conditionals.
    """

    user = {
        "name": "Demo User",
        "weight": 55,
        "interval": 200
    }

    # Validation
    if user["weight"] <= 0:
        user["weight"] = 50

    return user
```

```
PS C:\Users\Jhanv\Downloads\HydrationCoach-main (2)> & C:/Users/Jhanv/AppData/Local/Microsoft/WindowsApps/python3.13.exe "c:/Users/Jhanv/Downloads/HydrationCoach-main (2)/HydrationCoach-main/hydration_coach/user_input.py"
PS C:\Users\Jhanv\Downloads\HydrationCoach-main (2)>
```



```python
# main.py
# Main program for Hydration Coach.

from user_input import get_user
from calculator import calculate_water, liters_and_ml
from reminders import show_reminders

def main():
    user = get_user()
    name = user["name"]
    weight = user["weight"]

    print("---- HYDRATION COACH ----")
    print(f"User Name: {name}")
    print(f"Weight: {weight} kg")

    total_ml = calculate_water(weight)
    liters, ml_remaining = liters_and_ml(weight)

    print(f"\nDaily Water Requirement: {total_ml} ml")
    print(f"That is: {liters} liters and {ml_remaining} ml")

    print("\n--- Reminders ---")
    show_reminders(2)


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

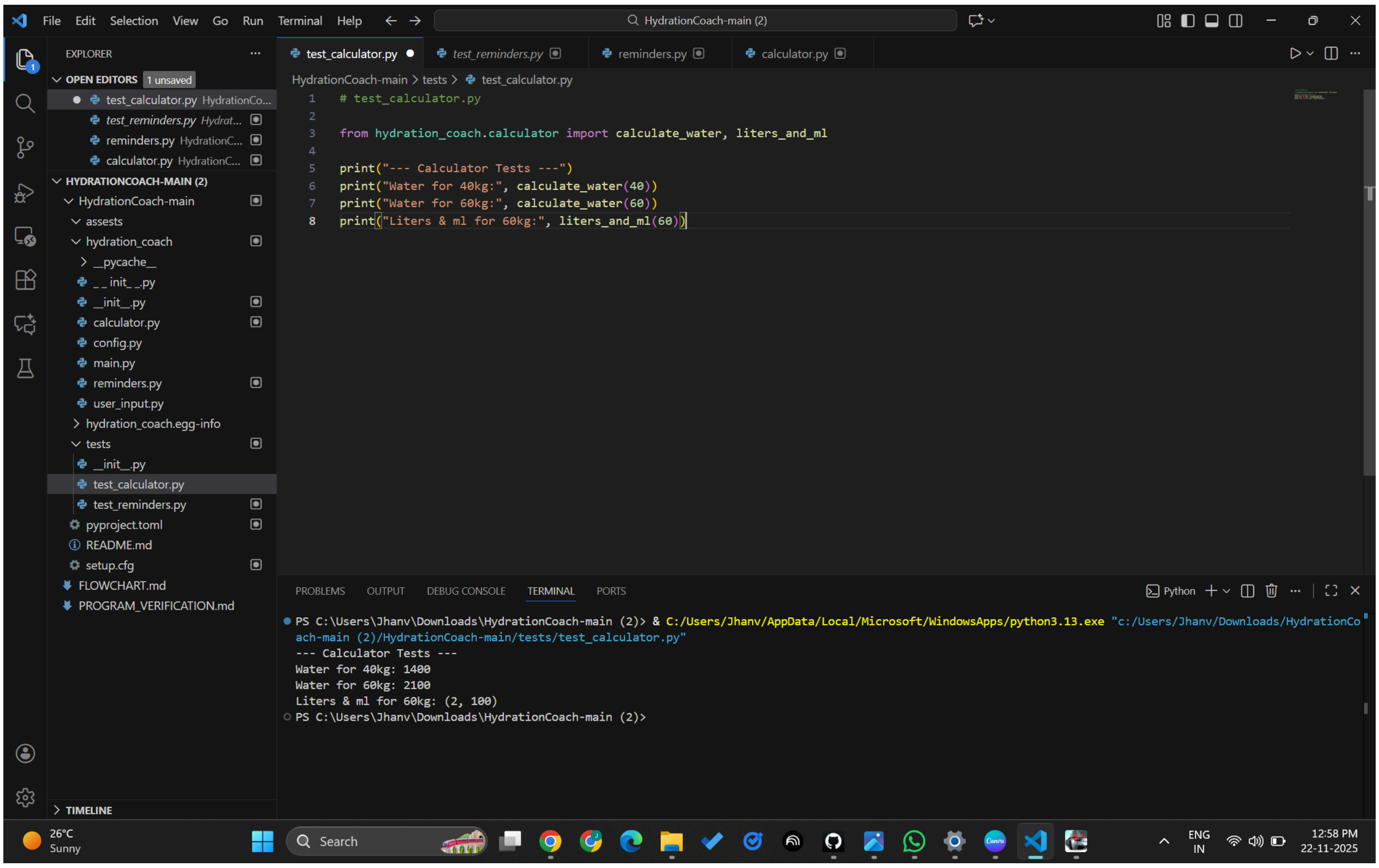
```
PS C:\Users\Jhanv\Downloads\HydrationCoach-main (2)> & C:/Users/Jhanv/AppData/Local/Microsoft/WindowsApps/python3.13.exe "c:/Users/Jhanv/Downloads/HydrationCoach-main (2)/HydrationCoach-main/hydration_coach/main.py"
---- HYDRATION COACH ----
User Name: Demo User
Weight: 55 kg

Daily Water Requirement: 1925 ml
That is: 1 liters and 925 ml

--- Reminders ---
Reminder 1: Time to drink water! 💧
Reminder 2: Stay hydrated and healthy! 😊
PS C:\Users\Jhanv\Downloads\HydrationCoach-main (2)>
```



HydrationCoach-main > tests > __init__.py

```python
# Mark tests as a package
```

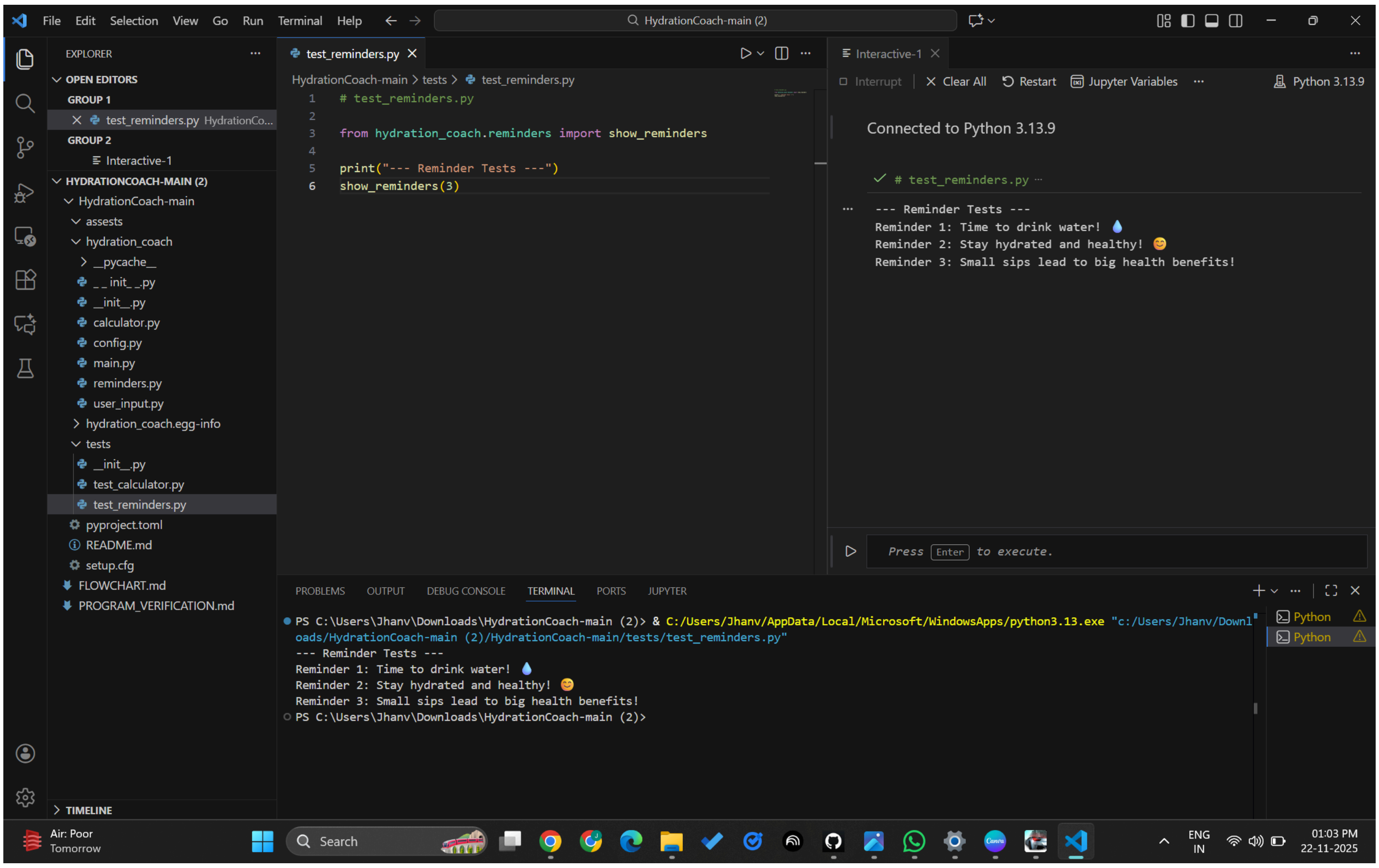
```
PS C:\Users\Jhanv\Downloads\HydrationCoach-main (2)> & C:/Users/Jhanv/AppData/Local/Microsoft/WindowsApps/python3.13.exe "c:/Users/Jhanv/Downloads/HydrationCoach-main (2)/HydrationCoach-main/tests/__init__.py"
PS C:\Users\Jhanv\Downloads\HydrationCoach-main (2)>
```



HydrationCoach-main > tests > test_calculator.py

```python
# test_calculator.py

from hydration_coach.calculator import calculate_water, liters_and_ml

print("--- Calculator Tests ---")
print("Water for 40kg:", calculate_water(40))
print("Water for 60kg:", calculate_water(60))
print("Liters & ml for 60kg:", liters_and_ml(60))
```

```
PS C:\Users\Jhanv\Downloads\HydrationCoach-main (2)> & C:/Users/Jhanv/AppData/Local/Microsoft/WindowsApps/python3.13.exe "c:/Users/Jhanv/Downloads/HydrationCoach-main (2)/HydrationCoach-main/tests/test_calculator.py"
--- Calculator Tests ---
Water for 40kg: 1400
Water for 60kg: 2100
Liters & ml for 60kg: (2, 100)
PS C:\Users\Jhanv\Downloads\HydrationCoach-main (2)>
```



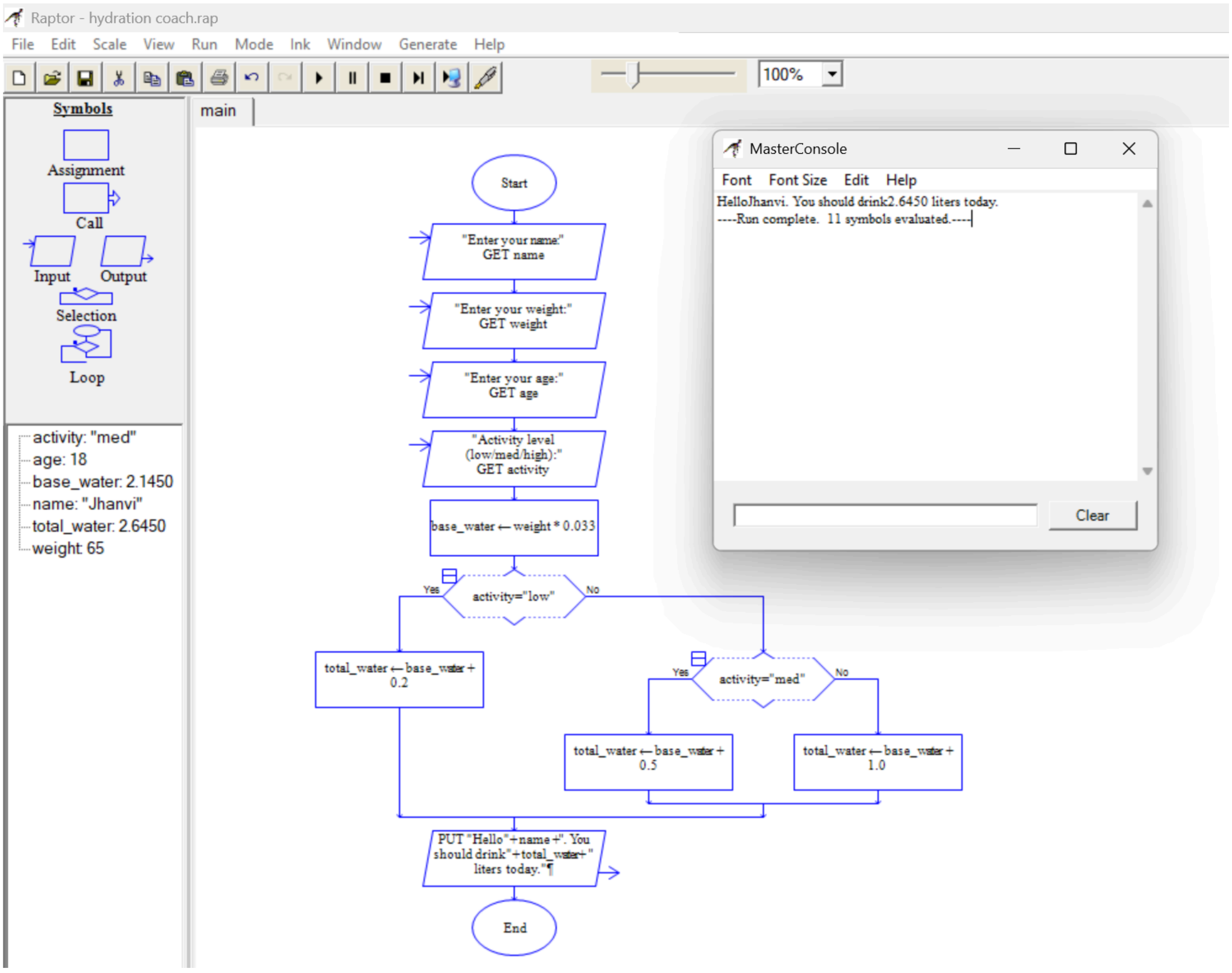
HydrationCoach-main > tests > test_reminders.py

```python
# test_reminders.py

from hydration_coach.reminders import show_reminders

print("--- Reminder Tests ---")
show_reminders(3)
```

```
Connected to Python 3.13.9

# test_reminders.py ...

--- Reminder Tests ---
Reminder 1: Time to drink water! 💧
Reminder 2: Stay hydrated and healthy! 😊
Reminder 3: Small sips lead to big health benefits!
```

Press Enter to execute.

```
PS C:\Users\Jhanv\Downloads\HydrationCoach-main (2)> & C:/Users/Jhanv/AppData/Local/Microsoft/WindowsApps/python3.13.exe "c:/Users/Jhanv/Downloads/HydrationCoach-main (2)/HydrationCoach-main/tests/test_reminders.py"
--- Reminder Tests ---
Reminder 1: Time to drink water! 💧
Reminder 2: Stay hydrated and healthy! 😊
Reminder 3: Small sips lead to big health benefits!
PS C:\Users\Jhanv\Downloads\HydrationCoach-main (2)>
```



```
activity: "med"
age: 18
base_water: 2.1450
name: "Jhanvi"
total_water: 2.6450
weight: 65
```

```
HelloJhanvi. You should drink2.6450 liters today.
----Run complete.  11 symbols evaluated.----
```

10.Testing Approach

Testing was conducted using two methods:

1. Manual Testing

Running main.py to verify correct flow of execution.

Checking console outputs for validity.

2. Functional Testing via Test Files

test_calculator.py tested:

Water calculation logic

Tuple output correctness

test_reminders.py tested:

Reminder formatting

Both test files were executed individually to check whether module imports and calculations worked correctly.

11. Challenges Faced

- Module Import Errors: VS Code initially showed errors due to incorrect folder navigation.
- Package Structure Issues: Test files could not detect hydration_coach without correct directory structure.
- Google Colab Limitations: Colab required uploads and directory adjustments for modules to run.
- Name Error for _name: Incorrect use of _name caused runtime errors.

12. Learnings & Key Takeaways

- Learned how Python modules and packages work.
- Understood importance of _init_.py for package recognition.
- Practiced algorithm design, top-down design, and modular coding.
- Learned debugging skills for import paths.
- Understood how to run Python code in VS Code and Google Colab.
- Gained exposure to preparing documentation and flowcharts.

13. Future Enhancements

- Future improvements may include:
- Adding a graphical interface (Tkinter or Web App).
- Push notifications for reminders.
- User profiles stored in a database.
- Daily/weekly hydration analytics dashboard.
- Integration with fitness trackers.

14. References

- Python Official Documentation: https://docs.python.org/
- Visual Studio Code Documentation
- W3Schools Python Guide
- Real Python Tutorials
- Stack Overflow discussions