Object Oriented and Functional Programming with Python (DLBDSOOFPP01)

Project title: Habit Tracking Application (Python CLI)

Ayaulym Myrzatay

GitHub Repository:

https://github.com/ayaaiden/Ayaulym_Myrzatay_92003661_DLBDSOOFPP01

Berlin 2025

Introduction

The Habit Track project is one of the most significant projects I have worked on. It features a fully functional Python backend that enables users to set, track, and rate their daily and weekly habits via a command-line interface. The steps included careful planning, correcting mistakes, repeating the process several times, and applying both functional and object-oriented programming principles. In the final stage of the Python project, it was crucial to ensure that the system not only worked but was also reliable, well-tested, and organized. Phase 3 required meticulous attention to carefully review each part, from the logic for data persistence and analytics to the user interaction with the system and the tests that cover it.

Technical Part

The habit tracking application was developed using Python 3.10 and is organized into distinct parts. Such as:

- Habit_tracker.py: It contains the Habit class and logic for tracking habit streaks.
- Tracker.py: It handles loading and saving JSON-based data and habit management.
- Analytics.py: The functional programming-based module for analyzing habits.
- Main.py: The entry point with CLI for interaction.
- Test_analytics.py and test_habit.py: Unit Test suites using pytest.

Functional Analytics Module:

- Implementation using map(), filter() and lambdas.
- Includes: the list of habits, filter by frequency, longest streak(per habit or all).

Data Management:

- Created a habit.json with five predefined habits (both daily and weekly). With habits such as exercising, meditation, reading, etc.
- Includes 4 weeks of tracked completions to test streak logic.

Habit Streack Logic:

- The Get_longest_streak () method integrates using time deltas.
- Adjusted logic for both "daily" and "weekly" patterns.

Testing and Debugging:

- Used pytest to test analytics functions and Habit behavior.
- Fixed parsing issues, function naming mismatches, and JSON formatting errors.

Project Structure:

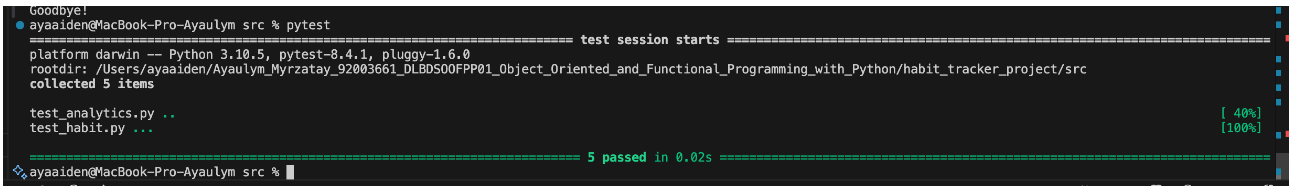
- __pycache__ and unused files were removed.

    - Added meaningful docstrings and inline documentation

Tests

Each unit in the test_analytics.py like test_get_all_habits_by_frequency() is a unit test. Each test checks one small piece of the programming, and the asserts statements validate expected behavoir.

Pytest made the code automated, readable, and reusable. All the important functionality was tested using pytest.



```
Goodbye!
ayaaiden@MacBook-Pro-Ayaulym src % pytest
=========================================================================== test session starts ===========================================================================
platform darwin -- Python 3.10.5, pytest-8.4.1, pluggy-1.6.0
rootdir: /Users/ayaaiden/Ayaulym_Myrzatay_92003661_DLBDSOOFPP01_Object_Oriented_and_Functional_Programming_with_Python/habit_tracker_project/src
collected 5 items

test_analytics.py ..                                                                                                                                              [ 40%]
test_habit.py ...                                                                                                                                                 [100%]

============================================================================ 5 passed in 0.02s ============================================================================
ayaaiden@MacBook-Pro-Ayaulym src % 
```

Conclusion

Throughout the development of the Habit tracking application, various practical issues were encountered, such as date parsing and file path bugs, which led to learning how to debug in a real–world development environment. The debugging process involves verifying method access, import paths, and handling JSON correctly across all nested methods. The project evolved significantly from the original concept of Phase 1.