Zewinger, Moritz

Matriculation Number: 14141201

Object Oriented and Functional Programming with Python (DLBDSOOFPP01)

Habits Phase 3 – Finalization - Abstract

Introduction & Objectives

The goal of this project was to develop a backend command-line interface (CLI) application for a habit tracking system using Python 3.7 or later. The focus was placed on producing clean, self-explanatory code, supported by comprehensive documentation. A detailed README.md file provides a high-level overview as well as installation and usage instructions.

The application was designed with an object-oriented structure, making use of classes and functions to ensure clarity, modularity, and maintainability. To demonstrate the system's functionality, it includes pre-defined example habits that help users understand how to interact with the application and extend it for their own use.

Data persistence was implemented to guarantee that user information is retained even after restarts or potential failures. Additionally, a dedicated analytics module enables users to analyze their habits, including streaks and completions, providing valuable insights into their progress and consistency.

The project also incorporates automated testing using "pytest" to validate the core functionality of the modules and ensure reliability. Finally, the use of supporting frameworks and libraries contributed to a smoother development process and a more robust final implementation.

Concept and Technical Design Overview

To start the project, a technical concept was developed. This was essential as it created an initial architectural overview that served as a foundation for all subsequent development steps and ensured that future changes could be incorporated easily without disrupting the overall structure.

Several core classes were defined in the concept and later implemented with additional attributes but no changes to the number or use case of the classes in general, ensuring the functionality of the application. As the project expanded, new features were integrated efficiently through functional programming techniques; this complemented the object-oriented design by enabling concise and reusable logic. The main design choice was based on a lightweight, fast, and user-friendly application.

Different libraries were used, some just for basic functionality, such as “datetime”, and a first implementation of the CLI with “click” was started, later then reverted, which will be explained in the “Reflection” section. The data storage (“sqlite3”) was one of the first clear implementations, since it has a good use case for small, multi-environment applications; there is no need to set up a whole database server, which was a clear overhead for the project.

Reflection: Challenges & Solutions

One of the first challenges was the choice of how to implement the CLI-based user interaction. Click was the first framework that was tried; it was easy to implement, but the exact interaction flow did not match the desired concept. The user should have a clear overview and should be able to use the desired functionality without much typing; the application should do most of the work while the user does only the essentials. Click is a great framework, and a use case for other applications is planned, but for the habit tracking application, which targets individual, regular people, a simple main loop was chosen since the application is more of a proof of concept and not an enterprise-scale implementation.

The implementation of the streak logic was not exactly accurate at the start, since it just matched the last completions without looking at the current time or date. In combination with the dummy data, this led to habits marked as a streak even though the last completion was not in the desired periodicity (e.g., weekly) referenced to the current date.

Creating an application without any project planning is a mess; features can be forgotten, or false implementations can occur. To work against that, a GitHub Project Kanban Board was created to keep track of each task and the desired functionality.

Result & Value Added

The final product is a solid proof of concept that can be easily used by individuals and documented for developers. Every feature asked for was implemented based on a valid concept and clean architecture that allows for extendibility through individuals. The application can be easily tested with the “pytest” module and can run on any operating system that has Docker installed, which significantly improves portability. Overall, this project provided valuable hands-on experience in combining object-oriented and functional programming principles to build scalable and well-documented Python applications.

Link & Formalities

The project demonstrates practical application of Python’s OOP and FP paradigms in developing a maintainable and testable backend system.

Source code: https://github.com/Mo3zart/OOFPP_Habits.git