B.SC. Data Science
Eric Anton Hauck
Matriculation Number 102305503
April 2025

Object Oriented and Functional Programming with Python

(DLBDSOOFPP01)
Task: Create a Habit Tracking App

Conception Phase

Motivation

Trying to build better habits or break bad ones isn’t easy, it is something most people struggle with. That is where a habit tracker comes in. It helps you keep an eye on your routines and stay motivated by showing your progress over time. Whether you want to exercise more, drink enough water, or just get more organized, tracking habits makes your goals feel more real and manageable.

Project goal

A client approached me and requested to build a Python backend for a habit tracking app. The basic functions required are following in the next paragraph.

Functionality

Users can define multiple habits in the application with each habit having a name and a periodicity. Habits can be completed. If the user missed to complete a habit at least once during the specified period, the user “breaks” the habit. Completing habits consecutively for a predefined number of times, gives a habit “streak”. Habits that are entered in the app are not only stored between secessions but can also be analyzed giving the following outputs: return a list of all currently tracked habits, return a list of all habits with the same periodicity, return the longest run streak of all defined habits, and return the longest run streak for a given habit.

Requirements

The client requests no user interface, just the functionality of the program, written in Python version 3.7 or later. Installation and run instructions as well as documented code is necessary. The app should come with 5 predefined habits with 4 weeks of example tracking data.

Technical framework

The application will be built using Python 3.7+, following a modular architecture using the object-oriented programming paradigm for the structural design of a habit class and the habitManager class. Habit data will be stored using a relational database with SQLite3, a lightweight solution that enables easy storage and retrieval of user-defined habits, completed habits, and current streaks. The database schema includes three tables: Habit (habit definitions), Completed (completion log) and Streak (tracking progress).

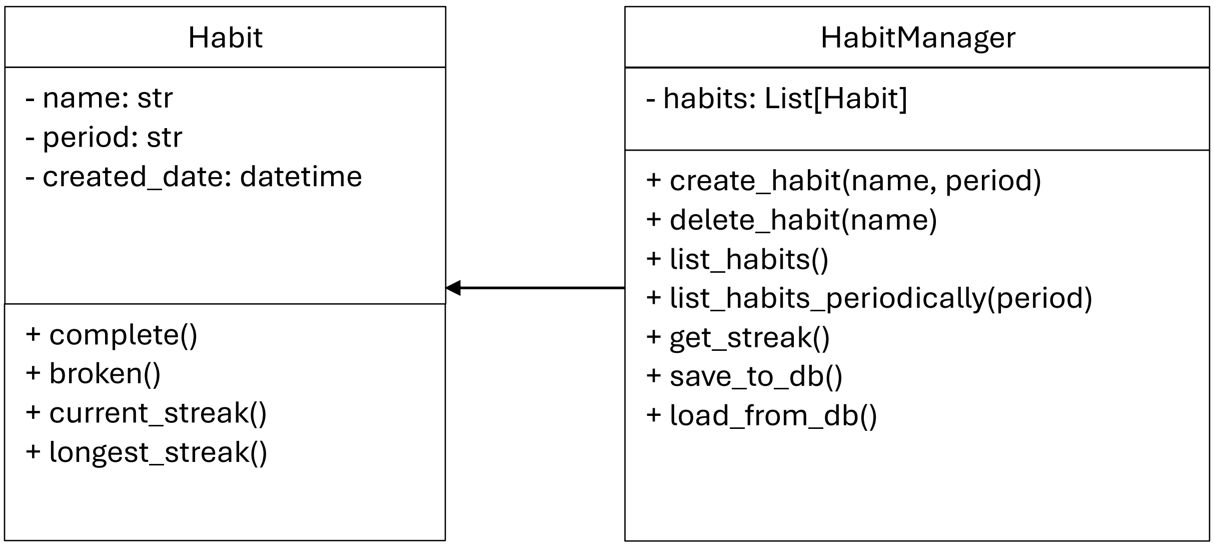
User interaction is handled via the command-line interface, offering menu-driven navigation for creating, completing, deleting, and analyzing habits. For analytics the project will apply the functional programming paradigm, using Python's built-in tools such as map, filter, lambda functions, and pure functions where appropriate.

Upon starting the application, data is retrieved from the database and used to initialize habit classes as objects. An update function is then triggered, checking the current date and time to verify whether any habits have been missed and updating the streak table accordingly. After this initialization step, the user can interact with the application via a command-line interface. Depending on the selected function, the save_to_db() method is called to either insert new records or update existing entries in the database.

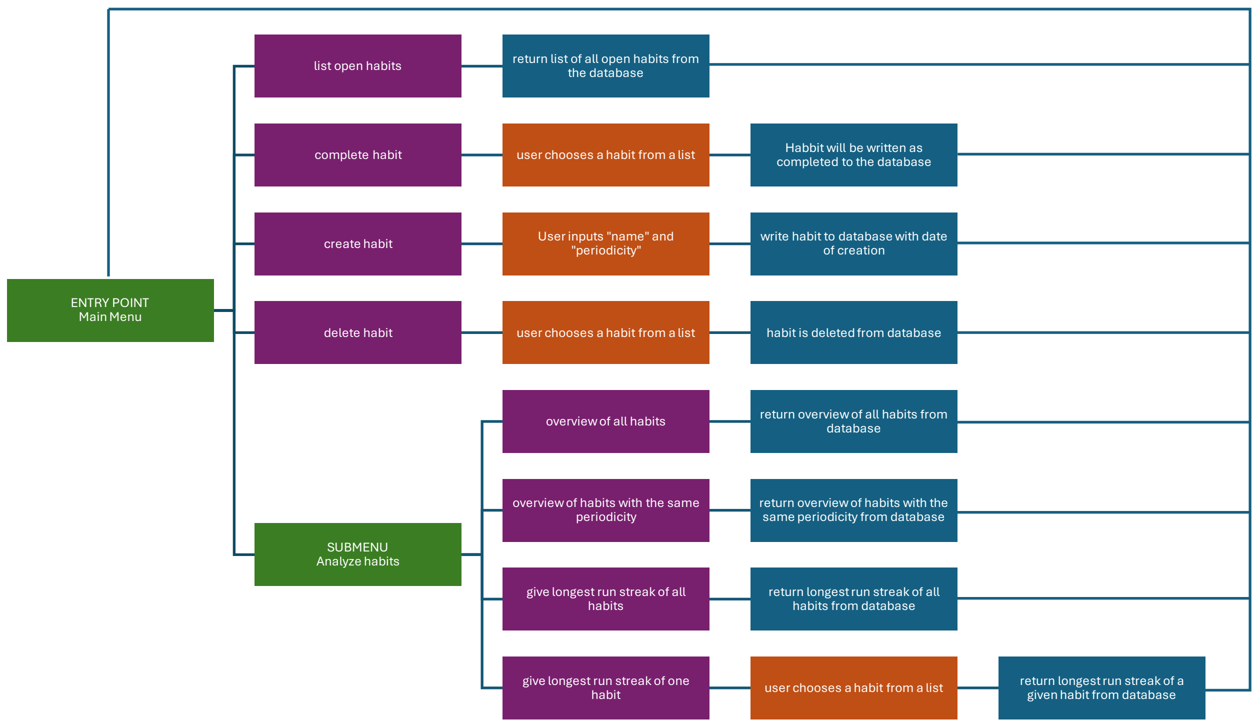
Future updates

Future updates on the program include additional analytical functions and new user functions like earning a "cheat" token for a habit streak that can be used to "skip" a habit for a period. Also a GUI will be implemented. A more flexible approach to storing the period is also considered. Instead of storing a string users can then set up individual intervals.

ULM Class Diagram

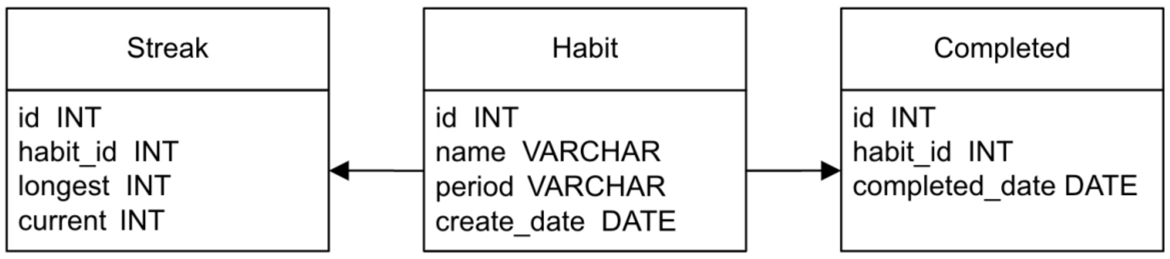


ULM User Flow Diagram


ENTRY POINT
Main Menu
list open habits
return list of all open habits from the database
complete habit
user chooses a habit from a list
Habbit will be written as completed to the database
create habit
User inputs "name" and "periodicity"
write habit to database with date of creation
delete habit
user chooses a habit from a list
habit is deleted from database
SUBMENU
Analyze habits
overview of all habits
return overview of all habits from database
overview of habits with the same periodicity
return overview of habits with the same periodicity from database
give longest run streak of all habits
return longest run streak of all habits from database
give longest run streak of one habit
user chooses a habit from a list
return longest run streak of a given habit from database

🟩 Menu point
🟪 Function user can choose
🟧 User inputs data
🟦 Program execution
If the user is in a Submenu or is prompted to input data, a command “back” takes the user back to the ENTRY POINT

Database scheme diagram


Streak
id INT
habit_id INT
longest INT
current INT
Habit
id INT
name VARCHAR
period VARCHAR
create_date DATE
Completed
id INT
habit_id INT
completed_date DATE