Abstract

The habit tracker app's goal is to provide a Command Line Interface (non-GUI) based habit tracker that enables daily and weekly habit tracking for users. Since the usefulness of this design exceeds expectations, the initial notion to depict the behaviours through a data frame has proven to be effective expectations. The application of the object-oriented programming (OOP) methodology went above and beyond expectations and was fully utilised.

For main.py, the idea was straightforward. First, if you are an existing user, you can simply log in and your habit tracker list will be displayed. But if you are a new user, you are required to create your user. Once your user is created, 8 predefined habits will be added to your habit tracker. Once user is logged in, he/she can perform various functions which are mentioned below.

To keep this file's code legible, use classes and assign objects. To catch details and enhance user conveniences, I was able to do so by utilising a sophisticated while loop. For instance, without having to restart anything, the user can add several habits repeatedly. When a command fails to execute, the user or the application receives messages explaining why this failure has occurred without crashing the application. This is achieved by using try & except statements for every function in the code.

The foundation of the programme is the **Habit Tracker** class. This class call different functions based on user input. This class has methods for adding, deleting, restoring, managing, and analyzing habits. User can also view their habits by simply calling analyze function.

Code Structure

Object-oriented and functional programming principles will serve as the foundation for the code's structure. To make code more understandable and easier to track, I decided to create a single class of Habit, create separate functions (each function e.g., Create/Login User, Create, Delete, Restore & Analyze habits have separate files) and main function to display user menu. Habit Tracker class will call above functions based on user input.

The adoption of the data frame structure is the largest success. The table clearly displays streaks, measured in days and weeks, as well as the history for each behavior. In this approach, the integration of the DateTime, and Rich modules was essential. The computation of the current streaks was simple thanks to DateTime library. Rich library enables a uniform appearance for the table and assisted with auxiliary duties like classifying habits based on Habit Status.

Below is the brief description of each function.

❖ **Create New User:**

- ✓ Include all the functionality related to user creation and storage.

❖ **Login:**

- ✓ This function will get input from the user and verify the credentials with Users table in database.

❖ **Create New Habit:**

- ✓ This function will take input from the user and create a new habit in the Habit Tracker table.
- ✓ If user try to create a habit which already exist user will get an error message.

❖ **Managed Habit:**

- ✓ This function will take input from the user and checked-off the specific habit.

❖ **Modify, Delete & Restore Habit:**

- ✓ Modify function will perform modification in existing habits. e.g., Change the habit name and its description, change the habit period (daily to weekly or weekly to daily) & Change Max Day.
- ✓ Delete function will soft delete the habit. Habit will remain in the table, but their status will be changed to "**Deleted**".
- ✓ Restore function will restore deleted/completed habits.

❖ **Analyze:**

- ✓ This function will perform all the other functions which will be displayed on the menu such as:
  - View All Tracked Habits
  - View All Deleted Habits
  - View All Completed Habits
  - View All Daily Habits
  - View All Weekly Habits
  - View Smallest Run Streak Daily Habit
  - View Smallest Run Streak Weekly Habit

- View Smallest Run Streak Among All Habits
- View Longest Run Streak Daily Habit
- View Longest Run Streak Weekly Habit
- View Longest Run Streak Among All Habits
- View Longest Run Streak Given Habit

❖ **Reset Overdue Habits & Mark Habits Complete Automatically**

✓ Reset Overdue Habits & Mark Habits Complete is a background process which starts running when program starts.

✓ The first purpose of this process is to reset the habits whose due date is overdue.

✓ The second purpose of this process is to marks the habit complete whose Max_Streak = Max_Days

Conclusion

The app runs smoothly. There are no faults that the user can observe in my tests. The outcomes are as anticipated by the project. The user can add, remove, and manage habits.

The user can mark their habits. When a habit is broken, the software automatically starts it again from the day it started and keeps track of as many streaks as possible in the "Break" column.

The main table displays the requested Analytics, which are arranged such that similar periodicities are displayed together.

GitHub Link

https://github.com/ma-muhammadali/Habit-Tracker.git