Portfolio

Finalization phase – Habit tracker

DLBDSOOFPP01 – Object oriented and functional programming with Python

Rafael Mutis-Maschmann

rafael.mutis-maschmann@iu-study.org

Table of content

List of figures

1. Introduction and content of HabiTrack

The solution HabiTrack is a habit tracker application which shall help the user to implement new, desirables habits. Below there are the main features of the solution:

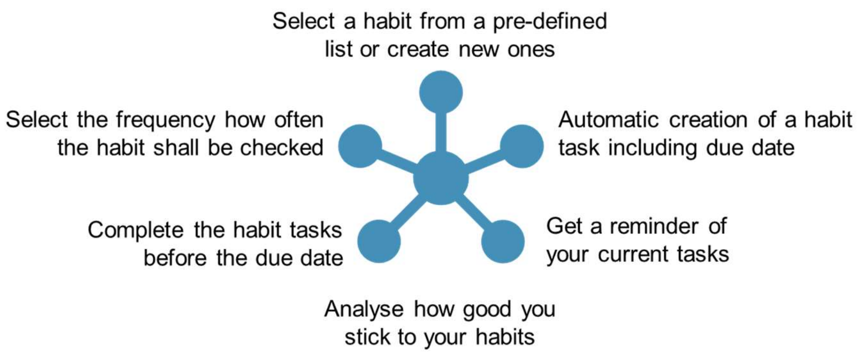
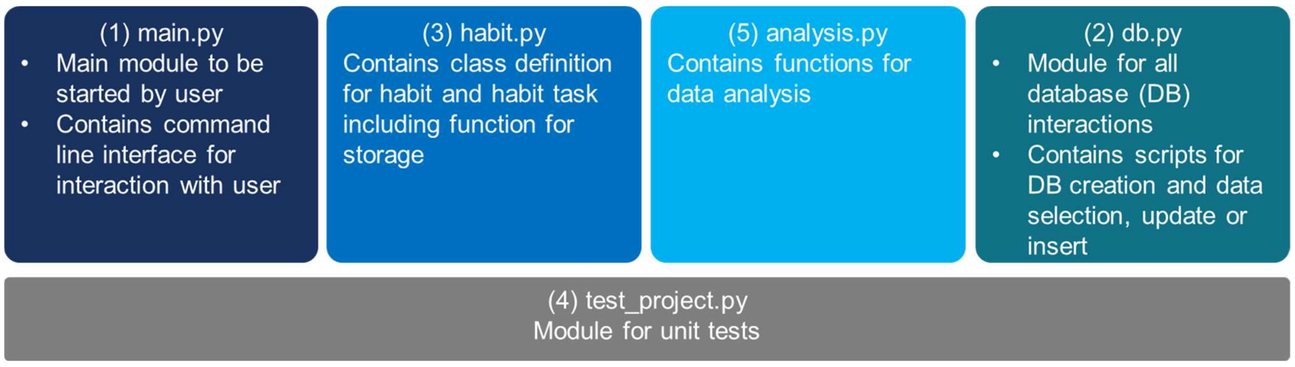

Select a habit from a pre-defined list or create new ones
Automatic creation of a habit task including due date
Get a reminder of your current tasks
Analyse how good you stick to your habits
Complete the habit tasks before the due date
Select the frequency how often the habit shall be checked

Figure 1: Main features of HabiTrack

Each habit task has a dedicated due date which marks the deadline until the task must be completed. If the task is completed within the timeframe the status is set to completed and the streak counter is increased as well. As this information is stored in each dedicated habit task it is possible not only to analyse current run streaks but also historical ones. Once the due date is exceeded the system automatically marks this task as missed. Therefore, it is not possible for the user to cheat and mark a task as completed after exceeding the due date.

2. Components of the solution and their development

The solution HabiTrack was programmed in Python 3.12. The graphic below shows the different modules, the numbers indicate the chronological order they have been created:



(1) main.py
- Main module to be started by user
- Contains command line interface for interaction with user

(3) habit.py
Contains class definition for habit and habit task including function for storage

(5) analysis.py
Contains functions for data analysis

(2) db.py
- Module for all database (DB) interactions
- Contains scripts for DB creation and data selection, update or insert

(4) test_project.py
Module for unit tests

Figure 2: Overview of Python modules in HabiTrack

(1) Due to my lack of knowledge in functional and object-oriented programming I realized a combined approach of theoretical learning and practising at the same time. Based on these constraints I followed the recommendations of the tutor using packages like cli or sqlite3 and not to build up a dedicated GUI. I started practising with the cli-package in the main-module to see how interaction with user can be done.

(2) After gaining some experience in the questionary I tried to store and load the data in the sqlite3 database. It was quite challenging to prepare retrieved information from the database

for user interaction by using the sqlite3-functions, e.g. returning the current habits and display them as selection options in a questionary. Therefore I used the read_sql_query-function as it was a much easier way for me preparing the data. Concerning the table structure I decided to use the habit name in the habit table as primary key: It is not possible to define two habits the same name to keep unambiguity in the data model.

(3) Afterwards I concentrated on the object-oriented part by creating two classes for habits and tasks as well as a store-function to call a respective database function. The retrieval or update of any existing habit task is realized via functions rather than classes which seem to be better manageable for me. At that time I revised and specified the module structure in order to define a clear objective for each module and better decide where to implement a new code snippet.

(4) Once the other modules showed good progress (basic functionality was available but e.g. analysis function still not complete) I created the test-module for unit tests. It was mainly a copy and paste job to build up the user behaviour of one complete month but also a good chance to review and optimize the code of the basic functions in terms of re-usage.

(5) The analysis-module was implemented as a separate module at the very end thanks to feedback of the tutor. After completion of analysis module I implemented these functions as well in the test-module.

3. Conclusion

The creation of such a project was for me personally sometimes challenging due to my lack of experience in programming but on the other hand it was very enriching to see how code was working successfully after several attempts. I am quite proud of current solution as it works without any blocking bugs. However there are following ideas concerning improvement of the current solution:

- Additional check when completing a habit task: The current implementation marks a habit task automatically as missed once due date is exceeded. This helps preventing cheating. On the other hand, the user can complete a habit several times during one check interval. An additional check of e.g. minimum time to be passed until next completion is possible could prevent this behaviour.
- Analysis module: Usage of machine learning algorithms to predict the status of current open habit tasks in order to warn the user (be carefully, you probably miss this habit task). What needs to be taken into account is a sufficient amount of training data for proper prediction.
- Analysis module: Currently the complete data is treated equally for analysis purposes. However, it would be more interesting to consider stronger actual data e.g. the data from last year. Temporal clustering of analysis results per month including graphical representation would be also quite interesting.

Link to GitHub Repository: https://github.com/rmutis/HabiTrack