PORTFOLIO

Assignments for the course: Object Oriented and Functional Programming with Python (DLBDSOOFPP01)

TABLE OF CONTENTS

1. TOPICS AND TASKS

Within the framework of this course, the following topic must be selected.

1.1. Task 1: Create a Habit Tracking App

Creating good habits and breaking bad ones is no easy task. To keep track of certain habits or achieving personal goals, more and more people rely on so called habit trackers to help them throughout the day. If you check out any popular app store, you'll find no shortage of habit tracking applications in a wide range of quality and prices.

Recently, a client has approached you and wants you to help them build a basic Python backend for a habit tracking app, which they want to roll out later this year. To make this project feasible, you must focus on the essential functionality of such an application. You are not asked to provide any sort of graphical user interface, just the basic functionality of a habit tracker using object-oriented and functional programming in Python, according to the following specifications.

Formally, a habit is a *clearly defined task that* must be *completed periodically* (e.g., brush your teeth every day or go to the dentist once every year). The basic building blocks of a tracking app are as follows:

- A user can define multiple habits in the application. A habit has a task specification and a periodicity.
- A *task can be completed*, i.e., "checked-off", by a user at any point in time.
- Each task needs to be checked-off *at least once* during the period the user defined for the respective habit. If a user misses to complete a habit during the specified period, the user is said to *break the habit*.
- If a user manages to complete the task of a habit x consecutive periods in a row, I.e., without breaking the habit, we say that the user established a *streak of x periods*. For instance, if a user wants to work out every day and does so for two full weeks, they establish a 14-day streak of working out.
- The habits users enter in the app are not only stored but can also be analysed. Users want answers to several questions like: what's my longest habit streak? What's the list of my current daily habits? With which habits did I struggle most last month?

Acceptance criteria for your habit tracker

Habit trackers can be complicated pieces of software. The scope of this project is limited due to time-constraints. The following acceptance criteria should help you to get a clearer picture of what's expected of you for this project.

- Your solution should be built using *Python version 3.7 or later,* so that you can focus on writing modern Python code. The tools and libraries used to implement your habit tracker application are entirely your own choice. However, you are not permitted to directly use or modify existing, third-party habit tracking tools that you might find on the internet of elsewhere.
- The software you submit must come with detailed, self-contained *installation and run instructions* so that it is clear how to install the project and use your habit tracker. A well-written README.md or similar is enough, given that you provide all the information properly. Make sure your code is documented with Python docstrings, too.
- The concept of a habit should be encoded as a class using object-oriented programming. Depending on your design, you may or may not need more classes for this project.
- Your tracker should be able to let users create at least two habit periods, namely *weekly and daily* habits.

- Your solution comes with *5 predefined habits* (at least one weekly and one daily habit). It should be clearly documented how new habits can be created with your solution. Also, make transparent how a user can complete a task within a given period.
- For each habit, your system tracks when it has been created, and the date and time the habit tasks have been completed.
- For each predefined habit, you should provide example tracking data for a period of 4 weeks. This example data will later be used for testing purposes as so called “test fixture”. It will also be helpful to validate your approach as you program your solution, as you can more easily load habits when resorting to predefined data.
- You need some way of storing, or persisting, habit data in between user sessions. Storing and loading data can either be done with a simple file-based solution, e.g., by reading and writing JSON files with Python’s built-in *json* module, or with a relational database solution using tools like *sqlite3*. Both approaches can work well but going with a DB is likely more professional and can teach you more going forward.
- Your solution has an *analytics* module that allows users to analyse their habits. The functionality of this analytics module must be implemented using the *functional programming paradigm*. You are free to consider implementing other functionality as well, but these are the minimal requirements. Provide functionality to
    - 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.
- Your solution has to have a clean API that users understand. You do this by exposing a command line interface (CLI) tool to the user that allows them to create, delete and analyse their habits. There are great tools like *fire* or *click* to build CLIs, but you can also use a simple main loop for your program, e.g., with the built-in *input* command to create an interactive menu. If you’re experienced in building graphical user interfaces, you can alternatively build out a GUI, too. An idea could be to use *tkinter* or similar Python-based tools or go for a more modern web-development approach with *flask* or *django*. Just note that you won’t gain any extra points with a more visually pleasing solution. In the end, the submission will be graded according to the acceptance criteria and the quality of your Python code.
- The critical parts of your solution, in particular the validity of your habit tracking components and the analytics module, should be tested by providing a *unit test suite* that can be run following the instructions provided with the solution. It is recommended that you work with either *pytest* or *unittest* for this.

Your application needs to be built, documented, and delivered according to the following three phases.

1.1.1. Conception phase

This phase represents the most important part of the design process. Anything that is overlooked or forgotten in this phase has a negative effect on the implementation later and will lead, in the worst case, to useless results.

The first step is to create a **written concept**, to describe everything you need to build your habit tracking application. To write it, imagine you're working in a company creating a habit tracking app, you're the lead developer and you're trying to explain the technical foundations to your colleague sitting next to you. So, there's no need to go into too much detail about what habits are etc., you can directly dive into the proposed solution. Apart from your "Habit" class do you need anything else? How does habit data get stored and retrieved? How do users interact with the application and what's the general user flow of your app? To help visualizing your concept, at least **one diagram** (e.g., in UML) will be created and inserted to the written concept to show the interaction of the components and the process.

This step of creating a concept first is perhaps the most important of the entire design process. **It is crucial to take/plan enough time for this phase before the next steps can be taken.** It is therefore essential to follow the sequence of the respective steps carefully.

It is important not only to consider what the client probably expects from your habit tracking app, but also how users should interact with it, how they create new habits, complete tasks, and check for their progress. The written concept must explain why the structure and process have been designed in the particular manner. Think about which tools you can use to implement each component and the communication between the components.

A conceptual text (1-3 DIN A4 pages) as PDF has to be prepared for the submission, explaining these analyses and considerations, together with the **diagram(s)** showing the interaction of the components and the process. The PDF will be uploaded in the PebblePad template, the text field in the template can be left empty.

Throughout the process, online tutorials are offered, and they provide an opportunity to talk, share ideas and/or drafts, and obtain feedback. In the online tutorials, exemplary work can be discussed with the tutor. Here, everyone has the opportunity to get involved and learn from each other's feedback. **It is recommended to make use of these channels to avoid errors and to make improvements.** You should only submit work after making use of the above-mentioned tutorial and informative media. This will be followed by a feedback from the tutor and the work on the second phase can begin.

1.1.2. Development phase/reflection phase

In this phase you **start implementing your habit tracking app**:

- You set up the frameworks and tools that you described in the conception phase.
- You implement the components outlined in your design diagram.
- Your code is commented, and you document its usage.
- Users can create, manage, and inspect habits they define in a convenient way.
- Users can analyse their habits.

In this phase you must submit an **explanation of your design and implementation procedure** as a composite **presentation PDF with about 5-10 slides**. The file should contain **visual elements** that facilitate comprehension, it needs to be structured and **explain which tools you decided to use**. While in the first phase you were scoping out an internal design document shared with your colleagues, think of this presentation as a customer-facing document. You as core developer explain to your potential customer base what components and features your application has and how to use them. The PDF will be uploaded in the PebblePad template, the text field in the template can be left empty.

Throughout the process, online tutorials and other channels provide the opportunity to profoundly discuss ideas and/or drafts and to get sufficient feedback, tips, and hints. **It is recommended to use these channels to avoid errors and to improve your work.** Once this is done, you can hand in your second phase for evaluation. Following a feedback from the tutor, your work on the final draft will continue in the third phase.

1.1.3. Finalization phase

In this final phase, your goal is to **polish and refine the application**, after having received feedback from the tutor, and prepare it for final submission. Certain elements may have to be improved or changed to finalize the task and complete this portfolio course.

Your finished product, the habit tracking app, is submitted by providing all files: **the files you created**, your **Python program code**, **documentation** etc., as follows:

- Your project is hosted on a public GitHub repository. That means you must create a GitHub account, if you don't have one already, create a repository and upload all your code into this repository. On submission in PebblePad you provide a **link to the repository**. Hosting the project on GitHub is part of building your portfolio. A well written README.md is sufficient documentation for the final submission, but the code itself has to be documented as well.
- You then create a ZIP file from all files contained in the GitHub repository and put it into a folder. You have to zip this folder and insert it in your submission in PebblePad.

Important note: The content of the GitHub repository and the contents of the ZIP file uploaded to your zip folder should be *identical*. You are not allowed to modify either one after the final submission. The content of the zip folder will be sent to the examination office upon submission.

Additionally, you provide a **1–2-page abstract PDF document** in which you describe your solution in terms of content and concept. This abstract presents a short break-down ("making of" of the project) about the technical approach in a clear and informative way. Think of this abstract as a quick report to your manager at the end of this project. What went well and what didn't? What pitfalls did you detect that you didn't foresee? What features did you build into the application that you're most proud of and that add value to the overall product? Please make sure the abstract contains a link to your project on GitHub.

To summarize, in this phase you will submit your project by hosting it on GitHub and uploading it to a zip folder as a ZIP file. In PebblePad you provide the link to the GitHub in the submission text field, upload the zip folder into the appropriate field and submit the abstract PDF file as final product. The abstract should contain the link to your GitHub project.

In the "Finalization phase", the online tutorials and other channels also provide the opportunity to obtain sufficient feedback, tips, and hints before the finished product is finally handed in. **It is recommended to use these channels to avoid errors and to make improvements.** The finished product is submitted **with the results from Phase 1 and Phase 2** and together with the materials mentioned above. Following the submission of the third portfolio page, the tutor submits the final feedback which includes evaluation and scoring within six weeks.

2. TUTORIAL SUPPORT

In principle, several channels are open to attain feedback for the portfolios. The respective use is the sole responsibility of the user. The independent development of a product and the work on the respective portfolio parts is part of the examination performance and is included in the overall assessment.

On the one hand, the tutorial support provides feedback loops on the portfolio parts to be submitted in the context of the conception phase as well as the development and reflection phase. The feedback takes place within the framework of a submission of the respective part of the portfolio. In addition, regular online tutorials are offered. These provide you with an opportunity to ask any questions regarding the processing of the portfolio and to discuss other issues with the tutor. The tutor is also available for technical consultations as well as for formal and general questions regarding the procedure for portfolio management.

Technical questions regarding the use of “PebblePad” should be directed to the exam office via mail.

3. EVALUATION

The following criteria are used to evaluate the portfolio with the percentage indicated in each case:

Evaluation criteria	Explanation	Weighting
Problem Solving Techniques	*Capturing the problem *Clear problem definition/objective *Understandable concept	10%
Methodology/Ideas/Procedure	*Appropriate transfer of theories/models *Clear information about the chosen Methodology/Idea/Procedure	20%
Quality of implementation	*Quality of implementation and documentation	40%
Creativity/Correctness	*Creativity of the solution approach *Solution implemented fulfils intended objective	20%
Formal requirements	* Compliance with formal requirements	10%

The design and construction of the portfolio should take into account the above evaluation criteria, including the following explanations:

Problem Solving Techniques: According to the topic a habit tracking app should be designed, implemented, and documented. It must be clear how to set up, run and use the application using the transferred files.

Methodology/idea/procedure: The concept of habits and how to track them according to their tasks and periods should follow a sound underlying object-oriented design. Analysing, summarizing, and aggregating habits over time should be suitably implemented using functional programming techniques. Which design and technology choices drive the project? Are the choices justified and do they make sense within the scope of this portfolio course?

Quality of implementation*:* Does the implementation follows a clear concept? Can it easily be used by a user that is unfamiliar with the project, e.g., by providing sufficient documentation? Is it straightforward to create and track new habits with the provided solution? Is it easy for users to analyse their habits with the tool? The acceptance criteria laid out in Chapter 2 must be respected.

Creativity/Rightness: It is evaluated whether the specific requirements have been understood and implemented in a comprehensible and innovative way. The basic functionality of the application must be covered by unit tests to ensure correctness of the software.

Formal requirements: The submission follows the acceptance criteria from Chapter 3 and the formal guidelines following in the next chapter. It is particularly important to respect the formal submission requirements outlined in Chapter 4.

4. FORMAL GUIDELINES AND SPECIFICATIONS FOR SUBMISSION

4.1. Components of the examination performance

The following is an overview of the examination performance portfolio with its individual phases, individual performances to be submitted, and feedback stages at one glance. A template in “PebblePad” is provided for the development of the portfolio parts within the scope of the examination performance. The presentation is part of this examination.

Stage	Intermediate result	Performance to be submitted
Conception phase	Portfolio part 1	• 1-3 pages written concept as PDF with at least one diagram to show interaction and processes.
		Feedback
Development phase/ reflection phase	Portfolio part 2	• Explanation of implementation in written form as a composite presentation PDF with about 5-10 slides.
		Feedback
Finalization phase	Portfolio part 3	• 1–2-page abstract (“making of”), including the link to the project on GitHub. • Final software product (scripts with installation manual and documentation included) hosted on GitHub • Upload the zip folder (incl. all files) • Result from phase 1 • Result from phase 2
		Feedback + Grade

4.2. Format for Digital File Submission

Conception phase	
Recommended tools/software for processing	Word processing tool of your choice for text, online UML editor like “lucidchart” for visuals.
Permitted file formats	PDF
File size	as small as possible/preview
Further formalities and parameters	For the performance-relevant submissions on “PebblePad”: Name-FirstName_MatrNo_OOFPP _Habits_Submission_Conception.pdf Example: Mustermann-Max_12345678_OOFPP_Habits_Submission_ Conception.pdf The PDF document needs to be attached in PebblePad as “Media file”, the “Conception” submission text can be left empty or give a short description of the submission.

Development/reflection phase	
Recommended tools/software for processing	Powerpoint or similar
Permitted file formats	PDF
File size	as small as possible/preview
Further formalities and parameters	For the performance-relevant submissions on “PebblePad”: Name-FirstName_MatrNo_OOFPP_Habits _Submission_Development.pdf Example: Mustermann-Max_12345678_OOFPP_Habits_Submission_ Development.pdf

Finalization phase

Recommended tools/software for processing	Word processing tool of your choice for the abstract, code must be hosted on GitHub
Permitted file formats	– PebblePad: PDF – GitHub: various file formats for your solution implemented in Python, helper code, e.g. Shell scripts, etc., and other related files like documentation, installation instructions, etc. – Zip folder: various file formats for submissions from all three phases. The code submitted via GitHub also has to be inserted in the zip folder by creating a ZIP file from all the files.
File size	as small as possible
Further formalities and parameters	**IMPORTANT:** you need to upload the zip folder that has been created especially for the submission (please follow the instructions on "myCampus"). This folder contains all the files you used to complete the task. To ensure a better overview, please create subdirectories for this purpose. The folder structure then looks like this: • Main directory (name of the zip folder) -> name:Name-Firs Name_MatrNo_OOFPP_ Habits ○ Subdirectory -> name: OOFPP_Habits_Phase1 ○ Subdirectory -> name: OOFPP_Habits_Phase2 ○ Subdirectory ->name: OOFPP_Habits_Phase3 In phase 3 you should also upload a zipped version (ZIP file with all program code) to a zip folder with the following naming convention: Name-FirstName_MatrNo_OOFPP_Habits_Submission_Final.zip Example: Mustermann-Max_12345678_OOFPP_Habits_Submission_Final.zip For the submission of your code on GitHub you are free to choose your own username. You create a public GitHub repository of your choice and upload all your code to it. That means your code should be available under the following URL schema: https://github.com/<user_name>/<repository_name> Example: https://github.com/maxmustermann/oofpp_habits_project For the performance-relevant submissions on "PebblePad", The 1-2-page abstract (making of) must always be named according to the following pattern: Name-FirstName_MatrNo_OOFPP_Habits_Submission_Abstract.pdf Example: Mustermann-Max_12345678_OOFPP_Habits_Submission_ Abstract.pdf In PebblePad, put the GitHub link into the "Information" section, upload the zip folder into the "resources" section and upload the abstract to both "Abstract" and "Final Product". Make sure your abstract contains a link to your project on GitHub, too.

4.3. Format of Abstract

Length	2 pages of text
Paper size	DIN A4
Margins	Top and bottom 2cm; left 2cm; right 2cm
Font	General Text - Arial 11 pt.; Headings - 12 pt., Justify
Line Spacing	1,5
Sentences	Justified; hyphenation
Footnotes	Arial 10 pt., Justify
Paragraphs	According to mental structure - 6 pt. after line break
Affidavit	The affidavit shall be made in electronic form via “myCampus”. No submission of the examination performance is possible before it. Please follow the instructions for submitting a portfolio on “myCampus”.

If you have any questions regarding the submission of the portfolio, please contact the exam office via mail.

Please also note the instructions for using PebblePad & Atlas!

Good luck creating your portfolio!