HABIT TRACKER

Manual

Contents

1 Introduction

1.1 Overview

Habit Tracker is a simple command line application for managing and analysing self-defined periodic habits. The application features a clean command line interface and is controlled by calling functions with their respective arguments, much like well known UNIX tools, for example grep. It runs on a lightweight relational database stored in a single file, so no need to set-up a full SQL server.

1.2 Installation

To use the **Habit Tracker** you need an installation of **Python 3.9.0**.

1. Download the application package from Github and extract the contents.
2. Install **pipenv** with `pip install pipenv` if you don't have it already installed.
3. Open a command prompt in the extracted folder and execute `pipenv install` to install the dependencies.
4. Run `pipenv shell` to enter the new virtual environment.
5. You are ready to use the **Habit Tracker**.

1.3 Creating a User

Before you can start creating habits, you must first create a new user.
To log-in type

```
python user.py login "username"
```

with `"username"` substituted with your chosen name. If everything worked correctly, you should see the message *Logged in successfully* on your command prompt.

2 Main Program Usage

2.1 General Application Control

After creating a user, you are ready to use the main application. The features of the **Habit Tracker** for managing and analysing habits, are used by calling the

functionality of the program through the command line, exactly like for creating a user. In general all of the commands require to be executed inside the virtual environment created with **pipenv**. If you have closed the command prompt after installation or user creation, just follow steps 3 to 4 of the installation instructions. This will activate the virtual environment. Following is a comprehensive list of the applications commands.

2.2 Managing Habits

The **Habit Tracker** supports commands for creating, deleting and checking habits periodically.

2.2.1 Creating a Habit

For creating a new habit, type

```
python tracker.py manage create "Habit" "Period"
```

with `"Habit"` being the habit, that should be created (e.g. Workout) and `"Period"` the time interval in which a habit has to be checked. The **Habit Tracker** supports the periods **Daily** and **Weekly** which must be entered into the command prompt in this exact format.

If everything worked correctly, the message *New habit "Your Habit" created* and *The habits ID is: "Your Habits ID"* is printed to the screen. For managing and analysing habits, the **name** or **ID** can be used interchangeably.

2.2.2 Deleting a Habit

For deleting a habit, type

```
python tracker.py manage delete "Habit"/"ID"
```

with `"Habit"` being the habits name or `"ID"` being the habits ID. If everything worked correctly, the message *Habit "Your Habit" deleted* is printed to the screen.

2.2.3 Checking a Habit

For checking a habit, type

```
python tracker.py manage check "Habit"/"ID"
```

with `"Habit"` being the habits name or `"ID"` being the habits ID. If the habit is checked for the first time, you should see *Streak for "Your Habit" increased*.

In general the **Habit Tracker** tracks the current streak, the longest streak and how many streak breaks have occurred. So depending on the habit's period and the last check date, the **Habit Tracker** updates the current streak, longest streak and the breaks. In addition you get a message if the habits streak is broken or increased.

For daily habits, the habit must be checked daily to maintain the streak. Weekly habits have to be checked in a timespan of seven days including the day the habit has been checked the last time to maintain the streak.

2.3 Analysing Habits

For inspecting and analysing your habits, you can use the applications analytic functions. The **Habit Tracker** has six analytic commands for various purposes.

2.3.1 List All Habits

To get a comprehensive list of all your created habits, type

```
python tracker.py analyse all
```

This command outputs all habits IDs and names in the format **ID : Name**

2.3.2 List Similar Habits

To get a list of all habits with a certain period, type

```
python tracker.py analyse similar "Period"
```

with `"Period"` being the filter criteria.

2.3.3 Show the Longest Streak

To show the longest streak of all habits overall, type

```
python tracker.py analyse longest
```

This command returns the habit or habits with the longest streak and the value of the longest streak in the format **Habit : Longest Streak**.

If you want to know the longest streak of a specific habit, type

```
python tracker.py analyse longest "Habit"/"ID"
```

with `"Habit"` being the habits name or `"ID"` being the habits ID.

2.3.4 Show the Current Streak

To show the current streak of a habit, type

```
python tracker.py analyse current "Habit"/"ID"
```

with `"Habit"` being the habits name or `"ID"` being the habits ID.

2.3.5 List Tracking Data

To list all dates on which a habit has been checked, type

```
python tracker.py analyse tracking "Habit"/"ID"
```

with `"Habit"` being the habits name or `"ID"` being the habits ID.

2.3.6 List All Breaks

To list all habits with streak breaks, type

```
python tracker.py analyse breaks
```

This command returns a list of all habits with streak breaks and the number of times a streak was broken.

Important side note: The command shows only the streak breaks which have happened until this point in time. The command do not evaluate if a habit's streak would be broken the next time a check occurs.

2.3.7 Supplementary Commands

Aside from the six analytic functions intended for regular program usage, the application features two supplementary commands for debugging purposes.

The first supplementary command is for returning the complete database table in which the habits are stored. For showing all data entries regarding a habit, type

```
python tracker.py analyse _show_habits
```

This command returns all entries of every user, that are contained in the habit table.

The second command is for returning all stored tracking data entries for all habits. To show all check dates for all habits, type

```
python tracker.py analyse _show_tracking
```

3 Administrative Commands

For user management, resetting the application and entering default data, the **Habit Tracker** features additional functions. Following is a list of all administrative commands.

3.1 User Management

To allow multiple users to use the application, the **Habit Tracker** has commands for logging in, logging out and showing the currently logged in user.

3.1.1 Login

To create a new user or log-in to get access to existing habits, type

```
python user.py login "username"
```

with `"username"` as your chosen name. The login command always logs-in as the user-name given by argument, regardless if a user is currently logged in. If everything worked correctly, you should see the message *Logged in successfully* on your command prompt.

3.1.2 Show the Current User

To return the user-name of the user logged-in at the moment, type

```
python user.py whoami
```

3.1.3 Logout

To log-out from the application, type

```
python user.py logout
```

If everything worked correctly, you should see the message *Logged out successfully* on your command prompt.

3.2 Resetting the Application

To reset the **Habit Tracker** to the default state and delete all stored data, type

```
python admin.py reset
```

If everything worked correctly, you should see the message *Restored default state* on your command prompt.

3.3 Inserting Default Data

For testing the analytic functions, the application can generate random data entries for a predefined list of habits. For inserting the default data, type

```
python admin.py testdata
```

After running the command you are logged-in as "testuser". With this user-name you have access to the default data entries and can test the analytic functions. If everything worked correctly, you should see *Entered default data* on your command prompt.

Important side note: After testing the analytic functions, log-out and then log-in with your own user-name. The user-name "testuser" is solely for testing the analytic functions, not for regular program usage.

3.4 Unit Test Suite

The **Habit Tracker** contains a unit test suite to evaluate correct command execution. For starting the unit test, type

```
python test_tracker.py
```

If everything worked correctly, you should get an OK back to the command prompt.