Final Product

How to start

Before to start make sure that you have Python 3.10+ on your device or you can download it: https://www.python.org/downloads/

Important packages:

- bcrypt~=4.0.1 "pip install bcrypt"
- pytest~=7.4.3 "pip install -U pytest"
- questionary~=1.10.0 "pip install questionary"
- inquirerpy~=0.3.4 "pip install inquirerpy"
- plotext~=5.2.8 "pip install plotext "
- freezegun~=1.2.2 "pip install freezegun "
- SQLiteAsJSON~=1.04 "pip install SQLiteAsJSON"

Documentation

main.py

The program begins by executing this file and then enrolls a new user while establishing routines for the associated user. The process of updating the user's details and routines can be achieved by calling the functions that have been imported in this file.

edit_profile.py

Function:

edit_Profile(trc_number)

The function requires a numerical value, which was retrieved from the database during the login process. Following this, the name, last name, and password are evaluated for their respective rules. Should they meet the stipulated rules, the previous values will be replaced with the newly inputted values.

analyze_habit.py

Function:

plot_habit

This function displays the progress of the user's habits graphically.

computation.py

Functions:

comp(start_date,final_date,period,practic_time,habit_name,total_practice_time,status):

This function returns the progress of a habit, status and the number of remaining hours to complete a habit.

comp_period(start_date,final_date)

This function returns the difference between the first day and the final day.

habit.py

Habit Class

This class used to represent a habit.

Attributes:

habit_name: string , name of habit

period: string, weekly or daily

practice_time: string, the total number of hours that a daily or weekly habit was practiced.

datetime_of_creation: string, time of define or create a special habit

trace_number: integer, trace_number: An integer value that serves as a random number used for tracking a user between two classes.

status: It has 4 options,started, completed, broken or completed before the specific date.

datetime_of_completion:string , time of completion of a habit.

total_practice_time:string, the total number of hours allotted to the user to perform a daily or weekly habit.

last_alter_date:string, the last time a habit was updated.

last_practice_time:string, this feature shows the number of hours a daily or weekly habit was practiced last time.

conn: establishing a connection to the database

Methods:

get_specific_habit(self,habit_name)

This method shows detailed information for a specific habit

delete_habit_in_database(self,habit_name,trace_number)

This method deletes the history of practice time for a specific habit.

save_habit_in_dataBase(self)

This method stores a new habit in the database.

save_new_practice_time(self,habit_name,trace_number,new_excersise_time_hour)

This method stores new practice time for a specific habit.

see_last_alter_habits(self,period,alter_day)

This method displays the updated habits since a specific time.

weekly_habit(self)

This method displays all weekly habits.

daily_habit(self)

This method displays all daily habits.

all_habit(self)

This method displays habit progress both numerically and graphically.

delete_habit(self,habit_name)

This method deletes a specific habit from the database.

longest_habit(self,period)

This method shows the longest streak of a daily or weekly habit.

edit_habit(self):

This method updates the following attributes for a specific habit.habit_name, period, practice_time, status, date of creation, datetime_of_completion,total_practice_time

reset_history(self,habit_name,trace_number):

If the time or state of a previously defined habit changes to the current day or the initial state, respectively, this function will remove the previously recorded times from the database and update the most recent state of the habit change to reflect the current day.

initialize.py

Functions:

lunch_dbx()

Create a file called "test.db" if it does not already exist, and connect it to the SQLite database.

create_tablex(connx)

Create a "testtable" table within the database file(test.db).

lunch_db()

Create a file called "databaseFile.db" if it does not already exist, and connect it to the SQLite database.

create_table(conn)

Create a "User" and "Habit" table within the database file(databaseFile.db).

user_registeraion()

Get the necessary information from the user in order to register user and return an array.

intermediate.py

Functions:

make_array(trc_number)

This function creates a raw array that contains only a random number.

check_input(input_string)

This function examines the input string to ensure that it has a minimum of four characters. Additionally, it ensures that the string is free of any numerical digits or spaces. Furthermore, the function capitalizes the first letter of the string.

habit_name()

This function asks the user for the name of a habit and then ensures that the string is free of any numerical digits or spaces. The function capitalizes the first letter of the string.

dig_input()

This function verifies that the input number has a value greater than zero and does not contain any strings or spaces.

make_habit.py

Function:

habit_making()

This function prompts the user to enter the name of the habit. If this habit has not been previously defined, the system will request the user to provide any additional necessary information. Ultimately, the function generates an array and return it.

to_do.py

User class

Attributes:

firstname: string , first name of the use

surname: string, surname of the user

trace_nubmer: integer, random number

password: string, user password

connect: establishing a connection to the database

Methods:

check_user(self)

Logging the user and also preventing duplicate usernames and surnames for registration.

save_user_in_database(self)

This method stores a new user in the database.

chk_habit.py

Function:

chk_habit(habit_name,trace_number)

This function receives a habit name, and if the habit has not been defined before, it returns false, and vice versa.

chk_trace.py

Function:

chk_trace_number(trace_number)

This function receives a random number, and if the number has not been defined before, it returns false, and vice versa.

How does habit tracker program executes?