Personal information

Time Management

Kabir Bissessar 1007073 CS-A1123

6.05.2022

General Description

This program that allows users to time their activities, for example when studying for different courses.

I have attempted the Hard difficulty for this project. This includes a graphical interface, where the Timer's time is shown on the screen, the user can time multiple activities with the same timer and the program will remind you to take a break after 30 minutes.

Instructions for the user

In the program, the user can add up to 5 activities and track the time for any number of activities simultaneously.

An activity can be added in the text box under "Your Activities". Write the activity in the format "<activity name>/<total time (in hours)>" and then click “add new” button.

When the activity is added, a checkbox with the activity name will be created. The user can select this checkbox, then start the timer, to track the time for that activity.

The timer has three buttons: “Start Timer”, “Stop Timer” and “Reset Timer”.

- “Start Timer” will start the timer, and this will be displayed on the screen.
- “Stop Timer” will pause the timer, and the current elapsed time will remain displayed on the screen. If “Start Timer” is pressed at this point, the timer will continue from the time that it stopped at.
- “Reset Timer” will set the timer display to 0. If any activities were being timed (selected), the time (in seconds) will be added as an entry to the activity's data.

The program also contains the buttons: “Next day”, “Show today's times”, “Show past times” and “Read file”.

- “Next day” will change the date of the activities being tracked (the program always starts with the current date at the time of launch). This will write all of the day's data to a file with the name format “YYYY-MM-DD.json”.
- “Show today's times” will print, in the terminal, the times tracked during the current day.

- “Show past times” will print, in the terminal, the times tracked during previous days. This will only happen if times have been tracked and the “Next day” button has been clicked during the runtime of the program.
- “Read file” will read a json file that is inputted in the text field above it. This json file should be present in the same directory as the source code - in this case the “src” directory.

The user can select multiple activities to be timed, before clicking “Start timer”. However, after the timer has started, if the user selects a new activity, the timer will stop and the time data will be written to the respective activities, before the new activity is selected.

External libraries

Apart from PyQt5, no external libraries were used in this project.

Structure of the program

- Activity object
- Time object
- FileData object
- GUI object

The Activity object stores data about the activities added by the user, such as the name and the total time the user wants to spend on the activity. It also contains Time objects, which is how it stores time data for the user. The methods are:

- calculate_progress() – to calculate the user’s progress towards their time goal
- serialize() – serialize the data stored, so that it can be saved in json format.

The Time object stores the time, in seconds, that has been tracked. The methods are:

- calculate_time(): formats time into hours, minutes and seconds
- serialize(): serializes the time, so that it can be saved in json format.

The FileData object stored the filename of a file to write to. The methods are:

- write_to_file() – writes to the desired file
- read_file() – static method that reads a json file and returns the data as a Python dictionary.

The GUI object configures and runs the graphical interface. The functionality of the buttons on the screen are powered by methods in this class. The methods are:

- configure_layout() – formats the interface to display all of the widgets

- ui_components() – adds the text boxes and time display to the interface
- init_buttons() – adds the buttons to the interface
- add_activity() – creates Activity objects and displays them as checkboxes
- add_time() – adds time data to selected activities
- check_interrupt() – checks if new activity is selected after timer has started
- display_time() – shows time on the screen
- next() – writes current json data to file and changes the date
- today() – prints current time data to terminal
- past() – prints past time data to terminal
- read() – reads json data from file
- write() – writes json data to file

Algorithms

The program detects whether activities are selected when the timer is stopped. If so, the time data is added to those activities.

Whenever new time data is added to an activity, it increases the time spent on that activity. This is then used to calculate the user's progress towards their desired goal.

Data structures

Json objects were chosed as the main data structure for the project, as they can easily by formatted into Python dictionaries. My program needed to save time in hours, minutes and seconds (ints), dates (strings) and activity names (strings). It was straightforward to format this data into dictionaries, so json was a natural choice.

Files

My program handles json files, since json objects were used (for the reasons listed in the Data Structures section above).

The known shortcomings and flaws in the program

- De-selecting activities while the timer is running can cause strange behaviours regarding time data.
- The formatting and style of the program is not the best (for example, adding activities changes the position of "Your Activities" text and the buttons are at different heights in the program)

3 best and 3 worst areas

Three best areas:

- The program tracks time accurately and displays it on the screen
- Activity progress can be seen whenever time is tracked for it
- The program reminds you to take a break after 30 minutes

Three worst areas:

- Design – the program could have been styled to look better
- The program prints data to terminal, instead of in a window
- The user cannot change activities after they have been created

Changes to the original plan

I was originally going to attempt the Medium difficulty, however, I decided to attempt the Hard difficulty after building the graphical interface.

The program was developed according to plan before the first checkpoint, however afterwards the progress was not linear, as before, but exponential (most of the functionality was implemented close to the deadline).

Realized order and scheduled

The program was not developed according to the original schedule. I did not have a working graphical interface for the second checkpoint, the documentation was not done by 23.04.2022 and the program was not submitted on 3.05.2022 as initially intended.

The graphical interface was written by the second checkpoint.

The functionality of the interface was written after the second checkpoint, during 2.05.2022 to 6.05.2022

Assessment of the final result

I believe I have created a program that fulfils most of the functionality described in the the program description:
https://plus.cs.aalto.fi/y2/2022/project_topics/topics_muut_aiheet_304aika_en/

It is a program that I would personally use to track time while studying or doing other tasks (I have actually been using it towards the end of the development, while improving some of the functionality and writing the documentation).

There are some shortcomings, such as an interface that isn't very intuitive to use and the inability to edit/delete activities after they have been created. I believe that these could be enhanced, if I spend more time learning about PyQt5 and how to implement more of its features.

The data could also be saved in other formats, such as using datetime objects and txt files, instead of json.

I think the class division was done fairly well, so that the relevant tasks were implemented in relevant classed. However, most of the functionality was implemented in the GUI class, which makes the source code a bit long and difficult to read.

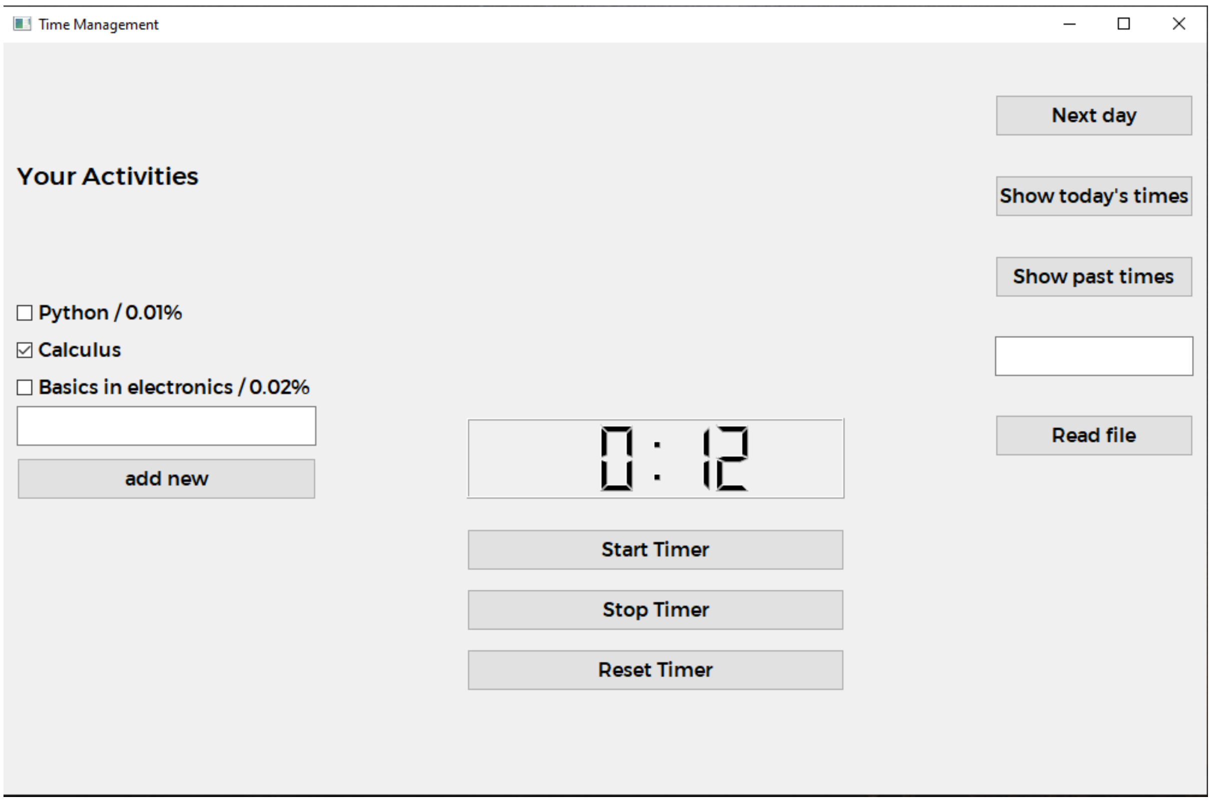
I believe the GUI class could be written a bit differently so that it is more human-readable and does not contain so many methods.

Based on the structure of the program, I believe it is suitable for making changes and expanding to add more functionality.

References

- Python Documentation – https://docs.python.org/3.9/
- PyQT5 documentation - https://doc.qt.io/qtforpython/

Attachments


Time Management
Your Activities
☐ Python / 0.01%
☑ Calculus
☐ Basics in electronics / 0.02%
add new
0 : 12
Start Timer
Stop Timer
Reset Timer
Next day
Show today's times
Show past times
Read file