TASK

Capstone Project – Task Manager

Visit our website

Introduction

Congratulations on making it this far! This capstone is a milestone in your learning. You will combine the knowledge that you have gained and apply it to solve a problem in this project. This project will consolidate the knowledge that you've learned by doing the various preceding tasks, requiring you to synthesise and apply your learning to build a useful task management application. Be creative – you'll be tasked with a set of criteria to meet, but the rest is up to you! It is worth spending time and effort to make this a project that you can be proud of and add to your developer portfolio.

DEVELOPER PORTFOLIO

Developers who have the edge are those who find ways to apply their newfound skills from the get-go. As you may know, a **developer portfolio** (a collection of online creations that you have made) allows you to demonstrate your skills rather than just telling people about them. It's a way of bringing your CV to life and introducing yourself to the world. As you learn more skills and put these into practice, each project that you complete will become more efficient and eye-catching.

These capstone projects give you the means to create projects for your very own developer portfolio, allowing you to walk away from this course not only with a certificate but, more importantly, with a head start to your tech career!

The task at hand

This project will begin with a focus on working with files and string manipulation. You will also have to use conditional logic and loops in this task. Once you have a basic working program, you will use lists or dictionaries and functions to extend the functionality of your Task Manager, making it more useful and powerful.

INSTRUCTIONS

A key focus of this project will be ensuring that your code is correct, well-formatted, readable, and that it adheres to the **PEP 8 style guide**. In this regard, **make sure that you do the following** (and double-check before submitting your work to avoid losing marks unnecessarily!):

1. Identify and remove all syntax, runtime, and logical errors from your code.
2. Make sure your code is readable. To ensure this, add comments to your code, use descriptive variable names, and make good use of whitespace and indentation.

3. Make sure your code is modular. Create functions to perform specific units of work.
4. How you choose to write code to create the solution to the specified problem is up to you. However, make sure you write your code as efficiently as possible.
5. Use defensive coding to validate user input and make provisions for errors that may occur using exception-handling techniques.
6. Make sure that all outputs that your program provides to a user are easy to read and understand. Labelling all data you output (whether in text files or to the screen) is essential to make the data your program produces more user-friendly. For example, compare the readability of the outputs in the images below. Notice how using spacing and labelling the output makes the second output much more user-friendly than the first:

Output 1: Raw

```
admin, Register Users with taskManager.py, Use taskManager.py to add the usernam
es and passwords for all team members that will be using this program., 10 Oct 2
019, 20 Oct 2019, No
admin, Assign initial tasks, Use taskManager.py to assign each team member with
appropriate tasks, 10 Oct 2019, 25 Oct 2019, No
```

Output 2: User-friendly

```
----------------------------------------------------------------------

Task:                   Assign initial tasks
Assigned to:            admin
Date assigned:          10 Oct 2019
Due date:               25 Oct 2019
Task Complete?          No
Task description:
 Use taskManager.py to assign each team member with appropriate tasks

----------------------------------------------------------------------
```

We have broken the instructions of this capstone project into three parts. The parts build on one another sequentially. Please ensure you have completed each preceding part before moving onto the next. All the best!

Practical task - Part 1

Follow these steps:

- In this task, you will be creating a program for a small business to help it manage tasks assigned to each member of the team. Copy the template program provided, **task_template.py**, and rename it **task_manager.py**. This template has been provided to make this task a little easier for you. Your job is to open and then modify the template to achieve the task set out below. Remember to save your work as you go along.
- Initially, this program will work with two text files, **user.txt** and **tasks.txt**. Open each of the files accompanying this project and note the following:
    - **tasks.txt** stores a list of all the tasks the team is working on. Open this file and review its contents. Note that this text file already contains data for two tasks. The data for each task is stored on a separate line in the text file. Each line includes the following data about a task in this order:
        - The username of the person to whom the task is assigned.
        - The title of the task.
        - A description of the task.
        - The date that the task was assigned to the user.
        - The due date for the task.
        - Either a “Yes” or “No” value that specifies if the task has been completed.
    - **user.txt** stores the username and password for each user that has permission to use your program (**task_manager.py**). Open the **user.txt** file and review the contents. Note that this text file already contains one default user that has the username “admin” and the password “adm1n”. When writing the username and password for a new user to this file, use the following format:

- The username followed by a comma, a space, and then the password.
- Only record one username and corresponding password per line.

Your program should allow users to do the following:

- Login. Prompt the user to enter a username and password. A list of valid usernames and passwords is stored in the **user.txt** text file. Display an appropriate error message if the user enters a username that is not listed in **user.txt** or enters a valid username but not a valid password. The user should repeatedly be asked to enter a valid username and password until they provide appropriate credentials.
- The following menu should be displayed once the user has successfully logged in:

```
Select one of the following options:
r - register a user
a - add task
va - view all tasks
vm - view my tasks
e - exit
```

- If the user chooses "**r**" to register a user, they should be prompted for a new username and password. The user should also be asked to confirm the password. If the value entered to confirm the password matches the value of the password, the username and password should be written to **user.txt** in the appropriate format.
- If the user chooses "**a**" to add a task, they should be prompted to enter the username of the person the task is assigned to, the title of the task, a description of the task, and the due date of the task. The data about the new task should be written to **tasks.txt**. The date on which the task is assigned should be the current date. Also, assume that whenever you add a new task, the value that indicates whether the task has been completed or not defaults to "No".
- If the user chooses "**va**" to view all tasks, display the information for each task on the screen in an easy-to-read format.

- If the user chooses "`vm`" to view the tasks that are assigned to them, only display the tasks that have been assigned to the current user in an easy-to-read format.

Important: Be sure to upload all files required for the task submission inside your task folder and then click "Request review" on your dashboard.

Practical task - Part 2

- Now format your program so that:
    - Only the user with the username "admin" is allowed to register users.
    - The admin user is provided with new menu options:
        i. `vc` – this allows them to view completed tasks.
            1. `del` – this allows them to delete tasks.
- Modify the admin menu to look like this:

```
Please select one of the following options:
r - register user
a - add task
va - view all tasks
vm - view my tasks
vc - view completed tasks
del - delete tasks
e - exit
```

- Note that the menu displayed to non-admin users should now look like this:

```
Please select one of the following options:
a - add task
va - view all tasks
vm - view my tasks
e - exit
```

Important: Be sure to upload all files required for the task submission inside your task folder and then click "Request review" on your dashboard.

Practical task - Part 3

- Modify your program so that functions are used to improve the modularity of your code. Your program should include at least the following functions:
    i. `reg_user` – this is called when the user selects "`r`" to register a user.
    ii. `add_task` – this is called when a user selects "`a`" to add a new task.
    iii. `view_all` – this is called when users type "`va`" to view all the tasks listed in **tasks.txt**.
    iv. `view_mine` – this is called when users type "`vm`" to view all the tasks that have been assigned to them.
    v. `view_completed` – this is called when the admin user types "`vc`" to view all the tasks that have been deleted.
    vi. `delete_task` – this is called when the admin user types "`del`" to delete a specific task.

- Modify the function called `reg_user` to make sure that you don't duplicate usernames when you add a new user to **user.txt**. If a user tries to add a username that already exists in **user.txt**, provide a relevant error message and allow them to try to add a user with a different username.

- Add the following functionality when the user selects "`vm`" to view all the tasks assigned to them:

    i. Display all tasks in a manner that is easy to read. Make sure that each task is displayed with a corresponding number that can be used to identify the task.

    ii. Allow the user to select either a specific task (by entering a number) or input "`-1`" to return to the main menu.

    iii. If the user selects a specific task, they should be able to choose to either mark the task as complete or edit the task.

        1. If the user chooses to mark a task as complete, the "`Yes`"/"`No`" value that describes whether the task has been completed or not should be changed to "`Yes`".

        2. If the user chooses to edit a task, the username of the person to whom the task is assigned or the due date of the task can be edited (i.e. the user must be presented with both options and be able to edit one or both). The task can only be edited if it has not yet been completed.

- Add options to generate reports (`gr`) and display statistics (`ds`) to the main menu of the application. The menu for the admin user should now look something like this:

```
Please select one of the following options:

r - register user

a - add task

va - view all tasks

vm - view my tasks

vc - view completed tasks

del - delete a task

ds - display statistics
```

```
gr - generate reports

e - exit
```

- Next, add the following functionality: when the user chooses to generate reports, two text files, called **task_overview.txt** and **user_overview.txt**, should be generated.
    - **task_overview.txt** should contain:
        i. The total number of tasks that have been generated and tracked using the **task_manager.py.**
        ii. The total number of completed tasks.
        iii. The total number of uncompleted tasks.
        iv. The total number of tasks that haven't been completed and that are overdue.
        v. The percentage of tasks that are incomplete.
        vi. The percentage of tasks that are overdue.
    - **user_overview.txt** should contain:
        i. The total number of users registered with **task_manager.py**.
        ii. The total number of tasks that have been generated and tracked using **task_manager.py**.
        iii. For each user, also describe:
            1. The total number of tasks assigned to that user.
            2. The percentage of the total number of tasks that have been assigned to that user
            3. The percentage of the tasks assigned to that user that have been completed
            4. The percentage of the tasks assigned to that user that must still be completed
            5. The percentage of the tasks assigned to that user that have not yet been completed and are overdue

Incorporate this functionality by creating one or more functions with descriptive names, thus ensuring the modularity of your code.

- Add the following functionality when the user selects to display statistics: present the contents of **task_overview.txt** and **user_overview.txt** in a user-friendly format on the screen. To achieve this, implement one or more functions with descriptive names that handle this functionality. Take into consideration that if the text files (**task_overview.txt** and **user_overview.txt**) do not exist yet (because the user has not selected to generate the reports), you should initially call the code to generate these files before displaying the statistics.
- Use defensive programming techniques in your programming to efficiently handle erroneous user input.

- To demonstrate your comprehension of recursion, consider attempting the following **optional task**:
  - Modify the existing code by introducing a function named `get_valid_task_number` within the `view_mine` function.
  - This recursive function should be called whenever a user selects a task that does not exist or enters a non-integer value.
  - Keep in mind that setting the base case is crucial for the recursion to work correctly. In this case, the base case should be when the user chooses to return to the main menu by entering "`-1`".

  Note that although this task is optional, it provides an excellent opportunity to showcase your understanding of recursion, which will be a great addition to your developer portfolio.

Important: Be sure to upload all files required for the task submission inside your task folder and then click "Request review" on your dashboard.

Share your thoughts

Please take some time to complete this short feedback **form** to help us ensure we provide you with the best possible learning experience.