Task Manager

Description:
Create a to-do list application that allows users to add, edit, and delete tasks. The application should also allow users to mark tasks as complete and save the task list to a file. Start with a console-based version and optionally add a GUI.

Features:

1.Add, Edit, and Delete Tasks:

Allow users to manage their tasks through the console.In the code, this feature is implemented through the TaskManager class, which has methods for adding, editing, and deleting tasks. The add_task method adds a new task to the list, the edit_task method edits an existing task, and the delete_task method deletes a task.

2.Mark Tasks as Complete:

Provide an option to mark tasks as done.The mark_task_as_complete method in the TaskManager class marks a task as complete. This method is called when the user clicks the "Mark Task as Complete" button in the GUI.

3.Task List Display:

Show the current list of tasks, along with their status (complete or incomplete).The display_tasks method in the TaskManager class displays the current list of tasks, along with their status. This method is called when the user starts the application or when the user clicks the "Display Tasks" button in the GUI.

4.Data Persistence:

Save the task list to a file and load it on startup.The save_tasks method in the TaskManager class saves the task list to a file, and the load_tasks method loads the task list from a file. These methods are called when the user starts the application and when the user exits the application.

Code Explanation:

The code consists of three main classes: TaskManager, Task, and TaskManagerGUI.

TaskManager class

This class represents a task manager that can add, edit, delete, and mark tasks as complete. It has the following methods:

- **__init__**: Initializes the task manager with a filename to store tasks.
- **add_task**: Adds a new task to the list.

- **edit_task**: Edits an existing task.
- **delete_task**: Deletes a task.
- **mark_task_as_complete**: Marks a task as complete.
- **display_tasks**: Displays all tasks.
- **save_tasks**: Saves tasks to a file.
- **load_tasks**: Loads tasks from a file.

Task class

This class represents a single task. It has the following attributes and methods:

- **description**: The task description.
- **completed**: A boolean indicating whether the task is complete.
- **__init__**: Initializes the task with a description.
- **mark_as_complete**: Marks the task as complete.
- **__str__**: Returns a string representation of the task.

TaskManagerGUI class

This class represents the GUI for the task manager. It has the following methods:

- **__init__**: Initializes the GUI with a filename to store tasks.
- **display_tasks**: Displays all tasks in the GUI.
- **add_task_callback**: Called when the "Add Task" button is clicked.
- **edit_task_callback**: Called when the "Edit Task" button is clicked.
- **delete_task_callback**: Called when the "Delete Task" button is clicked.
- **mark_task_as_complete_callback**: Called when the "Mark Task as Complete" button is clicked.
- **run**: Starts the GUI event loop.

Main code

The main code creates an instance of the **TaskManagerGUI** class and calls its **run** method to start the GUI.

Here's a high-level overview of how the program works:

1. The user creates a new task manager GUI instance, passing a filename to store tasks.
2. The GUI is initialized with a list box to display tasks and four buttons: "Add Task", "Edit Task", "Delete Task", and "Mark Task as Complete".
3. When the user clicks a button, the corresponding callback method is called.

4. The callback method interacts with the **TaskManager** instance to perform the desired action (e.g., add a new task, edit an existing task, etc.).
5. The **TaskManager** instance updates the task list and saves it to the file.
6. The GUI is updated to reflect the changes.

Conclusion

In conclusion, the provided code implements a simple task manager application that allows users to add, edit, and delete tasks, mark tasks as complete, and save the task list to a file. The application has a console-based version and an optional GUI version using the Tkinter library.

The code is organized into three main classes: **TaskManager**, **Task**, and **TaskManagerGUI**. The **TaskManager** class manages the list of tasks, the **Task** class represents a single task, and the **TaskManagerGUI** class represents the GUI for the task manager.

The application provides a user-friendly interface for managing tasks, with features such as adding, editing, and deleting tasks, marking tasks as complete, and displaying the task list. The task list is saved to a file and loaded on startup, providing data persistence.

Overall, the code provides a solid implementation of a task manager application, with a clear and organized structure, and a user-friendly interface.