To-Do List Project Documentation

Introduction

The To-Do List Application is a simple command-line tool designed to help users manage their tasks efficiently.

This document describes a simple Python project designed to function as a simple to-do list manager. Users can add, delete, and view tasks within the application.

Functionality

The project is having following core functionalities:

- Add Tasks: Users can enter new tasks with descriptive text. These tasks are stored in a list for future reference.
- List Tasks: Users can view all currently stored tasks displayed with an index number for easy reference.
- Delete Tasks: Users can select and remove tasks from the list by entering the corresponding task index number.

Code Structure

The application is structured as a Python script with the following key components:

- tasks list: This list stores all the user-entered tasks as strings.
- addTask function: This function prompts the user for a new task description, adds it to the tasks list, and provides a confirmation message.
- listTasks function: This function checks if any tasks are stored. If empty, it informs the user. Otherwise, it iterates through the tasks list and displays each task with its corresponding index number.
- deleteTask function: This function first displays the current task list for reference. It then prompts the user to enter the index number of the task to be deleted. The function validates the user input and removes the corresponding task from the tasks list if found. It provides confirmation or error messages depending on the outcome.
- Main Loop (if name == " main ":): This block contains the core application logic. It starts with a welcome message and presents a menu with four options: Add Task, Delete Task, List Tasks, and Quit. The loop continues prompting the user for choices and executes the corresponding functions (addTask, deleteTask, listTasks) based on the selection.

Limitations

This is a basic to-do list application with the following limitations:

- Tasks do not have deadlines or priorities.
- There is no functionality to mark tasks as completed.
- The application data is not persisted and resets upon exiting.

Future Enhancements

- Implement due dates and priorities for tasks.
- Add the ability to mark tasks as completed.
- Explore data persistence options to save and load tasks between sessions.
- Consider creating a graphical user interface (GUI) for a more user-friendly experience.

Conclusion

This Python application provides a functional to-do list management tool. The included documentation serves as a guide for understanding the functionalities and limitations of the program. The outlined enhancements suggest potential directions for future development.

This project provides a simple and intuitive interface for managing tasks. It can be easily extended and customized to fit specific requirements.