TODO CLI

This Todo CLI application uses two python libraries:
1. OS
2. Pandas

Why pandas?

As a beginner, pandas was the first think that flashed in my mind when it came to playing with data- filtering it, storing it and editing it etc. (on a small scale). The DataFrames (two-dimensional, size-mutable data structures with labeled axis) allows data management for textual as well as numerical data types.

Had I used SQLite or any other similar library, the code would have become unnecessarily complicated. Despite the fact that it would have been a safer option, in case this code ever made it beyond, it was still an if situation. SQLite would have made it pointlessly longer and more tiresome.

I went a little unconventional in my approach of using pandas. The data fed remains even when the app restarts. This is because what is being given is stored in files. Each todo list has its own list, which the code creates on its own. Every time any new todo is entered, it is saved into the file by itself. So, even if the terminal is killed and the code is rerun, the data does not disappear.

Pros of using Pandas:

- **Wide Range of Functions:** A wide range of functions are available which can be further chained together to perform a specific task, i.e. it offers a wide pool of functions and methods for easy data manipulation, making it a flexible tool. This makes it easy to use while keeping the code simple and clean.
  It is used in my code to define multiple functions (load, save, backup, todoadd, todoshow, backup, todoundo, todoedit, todocomplete, todofilter, todoview).

- **Easy Data Analysis:** It does an effective work at converting the raw data (entered by the user) into a tabular form (DataFrames) which can be used for further data manipulation.

Cons of using Pandas:
While pandas has cons of its own (memory consumption, limited performance etc.), since this is a small-scale project (as mentioned earlier), there is no visible harm in using pandas in this code (in my observation).

Why this code?

As an Indian student with too much in plate, I realized I needed to manage my time, manage my schedule better- to come out more efficient, be more effective and reach my potential. There were multiple applications available online, but there was an issue: what I wanted the app to have was absent in most, and where I did find it, it was paid. Then, I remembered, I had taken a course on Udemy (which I never got the time to finish).
It had been recently replaced by another one, in which there was a section of making a todo application. But there was an issue, I did not remember my credentials. So, using what I remembered of pandas and OS, I wrote a code. Myself. Nothing fancy, nothing copied, nothing catchy, something just sufficient to manage my tasks- with all features/tools I needed. Exactly why this TODO CLI even came into existence in the first place.

Future Upgrades

- **Date and time feature**: to set deadlines and a reminder. Filtering out todos that were done post deadline, completed on time, etc.
- **A reminder**: A customizable optional reminder that can be set a few days/hours/minutes before the todo hits the deadline
- **A GUI interface:** A GUI interface (using Tkinter, PyQt, Kivy or any such library for an app-style GUI and Flask or Django for a web-based GUI.)

Version History

Version 0.0 (The original version): Uploaded on June 12th, 2025. Raw draft.

Version 0.1.0: Uploaded on June 28th, 2025. Removed unnecessary in-line comments

Version 0.1.1: Uploaded on June 29th, 2025. Reduced an extremely long while loop with a smaller one by defining function for the various features outside the loop.

Version 0.1.2: Uploaded on June 30th, 2025. Structured the variable names and function names.

Citations

pypi.og

Viso.ai

incentius.com

learnenough.com – technical guides/python

Medium – Rohan Patil