Reflection on Using Aider for Calculator Enhancement

Most Effective Aider Techniques

- **/ask**: Helped me quickly understand the project's structure and features. However, answers were often too long, so I sometimes had to skip to the part I needed. - **/diff**: Very important to review the exact code changes before committing them. - **/add**: Allowed me to select which files to include in the context. I also learned that **/add *** includes everything (even node_modules), so I avoided that. - **/code**: The core of Aider since it generated the actual implementations. - **/commit**: Helped me easily save progress and maintain version control checkpoints.

Limitations Encountered

- Sometimes Aider's code was incomplete or caused the app to break temporarily (blank screen). I could usually restore it by asking Aider to fix or revert changes, but it was frustrating. - **/undo** did not always behave correctly. - Explanations were very verbose, which made reading tiring.

Comparison to Traditional Coding Workflow

- Aider made repetitive coding tasks faster and helped with complex features. - It reduced time spent on boilerplate code and gave quick suggestions. - Compared to tools like Cursor or GitHub Copilot, I did not find Aider significantly better, because those tools are more integrated into the editor environment.

Suggestions for Improvement

- Show changes directly inside the editor, not only through **/diff**. - Improve the reliability of **/undo**. - Provide better integration as an extension (e.g., "Aider Composer"), which would be easier to use than the terminal.

Conclusion

Overall, Aider is a powerful assistant that simplifies both simple and complex coding tasks. Although it has some rough edges, it can significantly speed up prototyping and iterative development when used carefully.