The Notes I Did Not Want to Lose, while writing analysis code

A practical guide to preserving context while building a quant code for both people and agents

Rahul

2026-07-09

Why This Started

I began coding with a paper lab notebook beside me. In my first scientific computing exercise (a Fortran 77 bisection program), the code handled computation, but the notebook held the important reasoning: why relative error was better than absolute error in one case, why an iteration cap was necessary, and what assumptions made results trustworthy. That context made the work understandable.

Today, AI agents can write code quickly, but the same old problem remains: rationale gets scattered across prompts, plans, and specs. I kept losing decisions in file sprawls of prompts and agent planning docs, including simple but important choices like default parameter values and why they were chosen. I did not need more notes. I needed better links between ideas, code, and decisions (Karpathy 2026; Rahul 2012).

How I Built It

`wiki_code_space` is my practical answer. It uses a lightweight wiki, semantic links, and repeatable update steps so reasoning stays close to implementation.

Karpathy, Andrej. 2026. *LLM Wiki*. GitHub Gist. https://gist.github.com/karpathy/442a6bf555914893e9891c11519de94f.

Rahul. 2012. *A Survey of Wiki Based Collaborative Learning Environments for the Interdisciplinary Training of the Students in Maths and Biology*. Teaching. Waterloo, Canada.

The wiki is plain Markdown in an Obsidian-friendly structure, versioned with the code, and readable by both humans and agents. It is a living layer that evolves with implementation, so context does not drift away from code over time. The prompts are designed to capture information that Graphify can interpret easily for semantic relationship inference, which supports token efficiency and faster workspace navigation in my quantitative research workflow. Because these prompts live in the wiki folder, they can be tuned to project-specific needs (Graphify Labs 2026).

Graphify Labs. 2026. *Graphify*. GitHub repository. https://github.com/Graphify-Labs/graphify.

The wiki-code-space is a reusable template for agent-first quantitative analysis code development. So, that work becomes more reproducible, understandable, and maintainable for both *humans and AI agents*.

GitHub repository: https://github.com/r2rahul/wiki-code-space [1]

[1] The repository contains a template for a wiki-code-space project, adaptable to user specific needs.

What I Learned

The idea is simple: when relationships are explicit, both humans and agents work better. The idea that is conveyed is to treat code as living memory, not just execution. The workflow is similar to how we do git commits: as one unit of work is completed in our coding workflow, the wiki is similarly LLM maintained update as one unit of conceptual decisions [2] is made.

[2] The conceptual update is guided by the prompt with deterministic set of instructions to LLM, capturing what, why and where with other user specifications

---

Acknowledgements

Code development and blog writing were assisted by Posit AI (Posit PBC 2026) and OpenAI Codex (OpenAI 2021).

Posit PBC. 2026. *Posit AI*. Web. https://posit.ai/.

OpenAI. 2021. *OpenAI Codex*. Web. https://openai.com/codex/.