The Claude Behavior Guide

The full CLAUDE.md file with 215K+ GitHub stars, plus the exact steps to drop it into your project and make Claude follow it.

Why I'm sending you this

I've run into the same four things over and over with Claude Code: it overcomplicates a simple task, it ignores an instruction I gave two messages ago, it tells me something is done when it isn't, or it invents an API that doesn't exist. None of that is a Claude problem exactly. It's a default-behavior problem, and there's a one-file fix for it.

Someone turned Andrej Karpathy's public complaints about LLM coding agents into a single markdown file called CLAUDE.md, built around four rules. It now sits at over 215,000 stars on GitHub. You don't install anything new. You just drop the file into your project and Claude reads it the same way it reads any other project file.

This guide has the full file text, what each rule actually does, and the setup so you can have it running in the next five minutes.

01

How it actually works

1. Claude Code already reads a file called CLAUDE.md at the root of your project for context before it does anything else.
2. This particular CLAUDE.md doesn't describe your codebase. It describes how Claude should behave: state assumptions, keep changes minimal, touch only what you asked about, and define success criteria before it starts.
3. Because it loads with every session, you don't re-explain these rules in every prompt. Claude applies them by default until you remove or edit the file.

02

The one thing almost everyone gets wrong

People paste this file in and expect it to replace clear prompting. It doesn't.

> **WATCH OUT**
>
> The fourth rule, Goal-Driven Execution, only works if you give Claude something to verify against. 'Fix the bug' still gets you a guess. 'Write a test that reproduces the bug, then make it pass' gets you a loop Claude can actually finish on its own. The file changes Claude's defaults, it doesn't read your mind.

---

03 The file

SOURCE CODE

The full repository, including the Cursor rule version and setup docs:

https://github.com/multica-ai/andrej-karpathy-skills

CLAUDE.MD

```
# CLAUDE.md

Behavioral guidelines to reduce common LLM coding mistakes. Merge with project-specific
instructions as needed.

**Tradeoff:** These guidelines bias toward caution over speed. For trivial tasks, use
judgment.

## 1. Think Before Coding

**Don't assume. Don't hide confusion. Surface tradeoffs.**

Before implementing:
- State your assumptions explicitly. If uncertain, ask.
- If multiple interpretations exist, present them - don't pick silently.
- If a simpler approach exists, say so. Push back when warranted.
- If something is unclear, stop. Name what's confusing. Ask.

## 2. Simplicity First

**Minimum code that solves the problem. Nothing speculative.**

- No features beyond what was asked.
- No abstractions for single-use code.
- No "flexibility" or "configurability" that wasn't requested.
- No error handling for impossible scenarios.
- If you write 200 lines and it could be 50, rewrite it.

Ask yourself: "Would a senior engineer say this is overcomplicated?" If yes, simplify.

## 3. Surgical Changes

**Touch only what you must. Clean up only your own mess.**

When editing existing code:
- Don't "improve" adjacent code, comments, or formatting.
- Don't refactor things that aren't broken.
- Match existing style, even if you'd do it differently.
- If you notice unrelated dead code, mention it - don't delete it.

When your changes create orphans:
- Remove imports/variables/functions that YOUR changes made unused.
- Don't remove pre-existing dead code unless asked.

The test: Every changed line should trace directly to the user's request.

## 4. Goal-Driven Execution

**Define success criteria. Loop until verified.**

Transform tasks into verifiable goals:
- "Add validation" -> "Write tests for invalid inputs, then make them pass"
- "Fix the bug" -> "Write a test that reproduces it, then make it pass"
- "Refactor X" -> "Ensure tests pass before and after"
```

```
For multi-step tasks, state a brief plan:
1. [Step] -> verify: [check]
2. [Step] -> verify: [check]
3. [Step] -> verify: [check]

Strong success criteria let you loop independently. Weak criteria ("make it work")
require constant clarification.

---

These guidelines are working if: fewer unnecessary changes in diffs, fewer rewrites due
to overcomplication, and clarifying questions come before implementation rather than
after mistakes.
```

SITES SHOWN IN THE VIDEO

- andrej-karpathy-skills on GitHub
  https://github.com/multica-ai/andrej-karpathy-skills

Set it up in Claude Code or Codex

1. Open your project folder in Claude Code. If you've never set one up, the root is just the top-level folder you point Claude Code at when you start a session.
2. If you don't already have a CLAUDE.md at the project root, clone the file in directly: git clone https://github.com/multica-ai/andrej-karpathy-skills.git, then copy the CLAUDE.md out of that folder into your own project root. Or skip the clone and pull the file straight with curl -o CLAUDE.md https://raw.githubusercontent.com/multica-ai/andrej-karpathy-skills/main/CLAUDE.md
3. Already have a CLAUDE.md for this project? Don't overwrite it. Append instead: echo "" >> CLAUDE.md followed by curl https://raw.githubusercontent.com/multica-ai/andrej-karpathy-skills/main/CLAUDE.md >> CLAUDE.md
4. Using Codex instead of Claude Code? Put the same file in your project folder, or in .codex/skills/claude-behavior/ with a one-line pointer added to your AGENTS.md so Codex knows to read it.
5. Close and reopen your Claude Code session so it reloads project files. The file needs to be present when the session starts, it isn't picked up mid-session.
6. Ask Claude to repeat its own guidelines back to you in one sentence per rule. If it can summarize all four, the file loaded correctly.

💡 QUICK TIPS

- No API key is needed for this one, it's a plain markdown file, not a tool or MCP server.
- If you run multiple projects, keep a copy of CLAUDE.md in a personal notes folder so you can paste it into a new project in seconds instead of re-downloading it each time.

---

04 The commands I actually run

PULL THE FILE INTO A NEW PROJECT

```
curl -o CLAUDE.md https://raw.githubusercontent.com/multica-ai/andrej-karpathy-skills/main/CLAUDE.md
```

APPEND TO AN EXISTING CLAUDE.MD

```
echo "" >> CLAUDE.md && curl https://raw.githubusercontent.com/multica-ai/andrej-karpathy-skills/main/CLAUDE.md >> CLAUDE.md
```

CHECK THAT CLAUDE ACTUALLY READ IT

```
Summarize the four behavior rules in CLAUDE.md back to me in one sentence each, then tell me which one applies to [DESCRIBE YOUR CURRENT TASK].
```

TURN A VAGUE TASK INTO A VERIFIABLE GOAL

```
Before you touch any code: state your assumptions about what [FEATURE OR BUG] requires, then propose a test or check I can use to confirm it's actually done.
```

When it breaks

ERROR	FIX
Claude never mentions the guidelines or acts the same as before	The file wasn't at the project root when the session started. Confirm the path with ls CLAUDE.md, then start a new session.
You already had a CLAUDE.md and the download overwrote your project notes	Restore from git if the folder is tracked. Going forward, always use the append command (echo + curl >>), never curl -o, on a project that already has the file.
Claude still overcomplicates despite the file being present	Check whether the task was described as a vague command instead of a goal. Rule 4 only kicks in when you give it something to verify against.
curl returns an empty or HTML file instead of the markdown	You likely fetched the GitHub page URL instead of the raw.githubusercontent.com URL. Re-run the exact curl command from this guide.
Codex ignores the file	Codex doesn't auto-read CLAUDE.md the way Claude Code does. Add a line in AGENTS.md pointing to wherever you saved the file so Codex loads it on session start.

Ready to go deeper?

This playbook is the short version. Inside the hub you get the full build walkthroughs, the prompt library, and the systems I use to ship this in a weekend.

Join the Build →

https://hub.digicuratoragency.com/join