Coding Nexus · Follow publication

Member-only story

My Current OpenAI Codex Workflow That Writes Clean, Reliable Code

4 min read · Oct 24, 2025

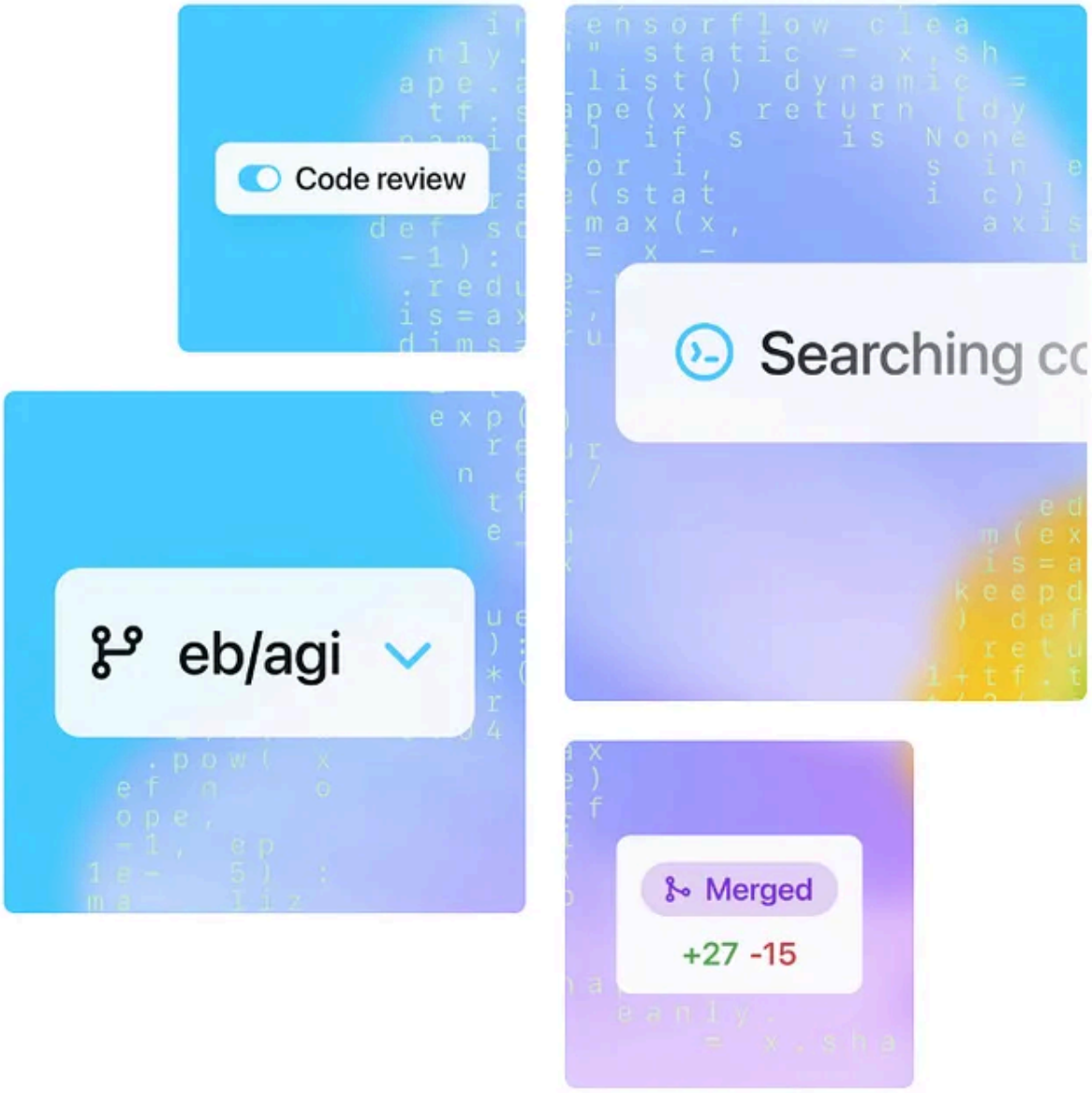
Civil Learning Following

I've been experimenting with OpenAI Codex for some time now. It's powerful — no doubt.
But if you've ever asked it to build something serious, you likely know the frustration:
The code appears fine at first... until it isn't.

You end up with random abstractions, over-engineered functions, and strange bugs that appear out of nowhere. Been there, done that.

So I decided to stop treating Codex like a "magic coder" and start treating it like a **junior developer** — one who's extremely fast but needs guidance.

This is how this workflow came about. It's straightforward, repeatable, and guides Codex (especially GPT-5) to act like a disciplined engineer rather than an overenthusiastic intern.



. . .

The Problem: Too Smart, Too Fast

Codex loves jumping straight into code. Give it a feature request, and it'll start coding immediately — even if it doesn't fully understand your app.

That's fine for demos, but in a real project, it's a recipe for chaos.

I wanted a system that required the AI to **slow down**, think, plan, and only then begin building.

Turns out, that one change made everything smoother, cleaner, and much more predictable.

. . .

The 3-Step Workflow That Changed Everything

I refer to it as the **“Plan First, Code Later”** method. It consists of three stages, which must be followed precisely.

. . .

Stage 1: Make It Understand — Don’t Let It Code Yet

At this stage, I tell Codex:

> “Your job is NOT to write any code. Just understand what I want and ask smart questions.”

Here’s the skeleton I use:

```
# Initial Explanation Stage

Your task is NOT to implement yet. Just understand and prepare.
```

```
Here's what I need:

[FEATURE DESCRIPTION HERE]

Responsibilities:
- Analyze how this fits in the existing codebase.
- Identify dependencies and edge cases.
- List everything unclear or ambiguous.
- Don't assume anything that's not mentioned.
```

It's like doing a quick technical spec session.

Codex usually fires back with a list of questions — stuff like:

> "Are we using React hooks or class components?"
> "Should this connect to the existing auth API?"

You answer those one by one until there's no confusion. Now Codex knows exactly what you want — no guesswork.

. . .

Stage 2: The Plan — Turning the Idea Into Steps

Once everything's clear, I tell it to create a plan.

The plan lives in a file called `plan.md`.
Each step is short, clear, and has a progress emoji to track what's done.

Here's an example:

```
# Feature Plan: User Login

Progress: 0%
- [ ] Step 1: Setup authentication module
  - [ ] Create auth service
  - [ ] Handle JWT tokens
  - [ ] Connect to database
- [ ] Step 2: Frontend login page
  - [ ] Build React component
  - [ ] Connect to backend API
  - [ ] Add form validation
- [ ] Step 3: Session management
```

```
  - [ ] Secure cookies
  - [ ] Auto-renew tokens
  - [ ] Logout process
```

Nothing fancy — just enough structure to keep everything grounded. At this point, you and Codex both share a common blueprint.

. . .

Stage 3: Implementation — Now We Build

Once the plan looks good, I switch to **GPT-5 Codex (high)** and instruct it to build *exactly* what's in the plan.

Prompt goes something like this:

```
Now implement precisely as planned.

Rules:
- Keep code minimal and modular.
- Follow existing naming conventions.
- Comment clearly.
- Update the markdown plan as each step finishes.
```

Then Codex starts coding, step by step, updating progress like:

```
Progress: 60%
Auth service done
JWT handling done
Frontend integration in progress
Session logic next
```

Watching it work feels oddly satisfying.
It's like having a focused engineer who never forgets what's next.

. . .

Why It Works So Well

This workflow requires Codex to think like a developer, not merely a code generator.

By splitting the process into “Understand — Plan — Build,” you get:

- Cleaner integration with your existing system
- Zero unnecessary features or guesswork
- Code that’s consistent with your own patterns

The AI ceases hallucinating new architecture and does what you asked — efficiently.

. . .

A Small Example

Let’s say you want to add a **dark mode** toggle in your React app.

In Stage 1, Codex might ask:

> “Should I save the theme preference in localStorage?”
> “Do you want it synced to user profiles?”

You confirm those details.

Then, in Stage 2, it gives you a plan like:

```
- [ ] Add ThemeContext
- [ ] Create ToggleButton in Navbar
- [ ] Sync theme to localStorage
- [ ] Update CSS variables
```

Finally, Stage 3 is pure implementation.
No surprises. Just code that works.

. . .

The Takeaway

AI can write code — but it needs direction. If you treat Codex like a collaborator instead of a magic box, it produces better, cleaner, and more reliable results.

This workflow doesn't increase complexity; it enhances *clarity*. And in software development, clarity is everything.

So next time you use Codex, don't say "build me X." Say, "Let's understand X first."

You'll be amazed at how much the code improves.

OpenAI ChatGPT Coding AI Ai Agent

Responses (1)

Bgerby

What are your thoughts?

James Bangma
Oct 24

Very interesting. Short of programming the code. I have everything else it's necessary to do the trading in terms of what I call the squiggles and wiggles a different time frame frames different equities and different equations. My life is upside... more

Reply

More from Civil Learning and Coding Nexus

In Coding Nexus by Civil Learning

Google's New LLM Runs on Just 0.5 GB RAM—Here's How to Fine-Tune It Locally"

A few days ago, Google quietly released a little AI model called Gemma 3 270M.

Aug 15 2.1K 30

In Coding Nexus by Code Coup

Claude Desktop Might Be the Most Useful Free Tool You'll Install This Year

I didn't expect much when I first saw the announcement for Claude Desktop. Another AI wrapper, I thought. Maybe with a shiny UI.

Oct 23 953 37

In Coding Nexus by Civil Learning

The Guy Who Let ChatGPT Trade for Him—and Somehow It Worked

You know how everyone says, “Don’t let AI touch your Money”? Well, someone on Reddit decided to ignore that.

Oct 8 274 8

In Coding Nexus by Civil Learning

AGENTS.md: The File That Saves You From Dumb AI Code

If you've ever thought, "This AI code is smart but also dumb," you'll get this.

Sep 18 595 8

See all from Civil Learning

See all from Coding Nexus

Recommended from Medium

The Atomic Architect

You Haven’t Seen AI Until You Try Claude Sonnet 4.5’s New Feature — It Redefines Insane

I built a working expense tracker in eight minutes while my coffee was still hot, and it remembered every receipt when I closed my laptop...

Oct 26 325 19

In Towards AI by Teja Kusireddy

We Spent $47,000 Running AI Agents in Production. Here’s What Nobody Tells You About A2A and MCP.

Multi-agent systems are the future. Agent-to-Agent (A2A) communication and Anthropic's Model Context Protocol (MCP) are revolutionary. But...

Oct 16 3.3K 111

In Coding Nexus by Code Coup

Claude Desktop Might Be the Most Useful Free Tool You'll Install This Year

I didn't expect much when I first saw the announcement for Claude Desktop. Another AI wrapper, I thought. Maybe with a shiny UI.

Oct 23 953 37

In AI Software Engineer by Joe Njenga

Cursor 2.0 Has Arrived — And Agentic AI Coding Just Got Wild

Cursor has released version 2.0 , bringing the most powerful agentic AI we have seen yet, more autonomous than ever before,here's what's...

Oct 29 763 18

In Level Up Coding by Swathi Ganesh

13 Rare Python Packages That Make AI Feel Like Magic

Rare yet the best python packages to discover

Oct 30 200 2

In Python in Plain English by Kuldeepkumawat

15+ INSANE Things You Can Do With Google Nano Banana

What if you could change any photo in seconds—no Photoshop skills, no complicated tools, just a simple prompt?

Oct 15 236 1