MISS AI . @RealMissAI

OPENAI CODEX

Beginner's Guide 101

Your plain-English guide to OpenAI Codex. No coding background needed.
Learn what it is, how to download and set it up, how to use projects,
and how to build real things from scratch.

Get started with AI coding tools today

realmissai.com . @RealMissAI

WHAT IS OPENAI CODEX?

OpenAI Codex is an **AI coding agent**. It reads, writes, and runs code on your behalf. Think of it as a brilliant assistant who can build apps, fix bugs, automate tasks, and manage your files, all from plain-English instructions.

You do not need to know how to code. You describe what you want, and Codex figures out the how. It runs on OpenAI's GPT 5.4 model and connects to your computer, your GitHub account, and external tools like Slack and Figma.

- **Not a chat window.** It is a full coding environment with a sidebar, built-in terminal, git panel, and code review.
- **Not just for developers.** Non-technical founders, creators, and small business owners use it to build tools, automate workflows, and ship products.
- **Not the old Codex API.** The original Codex API (2021) was shut down. This is a completely new product launched April 2025, now with around 4 million weekly active users.

Official resources: openai.com/codex . developers.openai.com/codex

---

CODEX VS CLAUDE CODE

Both are AI coding agents but they take different approaches. Here is the honest comparison:

Feature	OpenAI Codex	Claude Code
Made by	OpenAI	Anthropic
Best for	Background tasks, visual work, git-heavy workflows, parallel coding	Complex reasoning, large codebases, architecture planning
Interface	Visual IDE app with sidebar, diff panel, image previews	Terminal-first. Desktop app also available.
Models	GPT 5.4, Mini, Spark (Max plan only)	Haiku, Sonnet, Opus
Git built-in	Yes. Commit, PR, diff all inside the app	Requires separate tools or VS Code
Parallel work	Yes. Native worktrees and cloud tasks	Yes. Multiple terminal instances
Config file	AGENTS.md	CLAUDE.md
Non-tech friendly	Yes. Buttons, previews, visual UI	More technical. Better for experienced users.

USING CODEX AND CLAUDE CODE TOGETHER

The most effective approach is not to pick one tool. They are genuinely complementary, and knowing when to hand off between them is what separates a good AI workflow from a great one.

Use Codex when:

- You want a visual interface with buttons, previews, and inline diffs
- You are running background or cloud tasks while you do other work
- You need parallel agents working across multiple branches simultaneously
- Git is central: commits, pull requests, CI fixes
- You want to avoid the terminal entirely

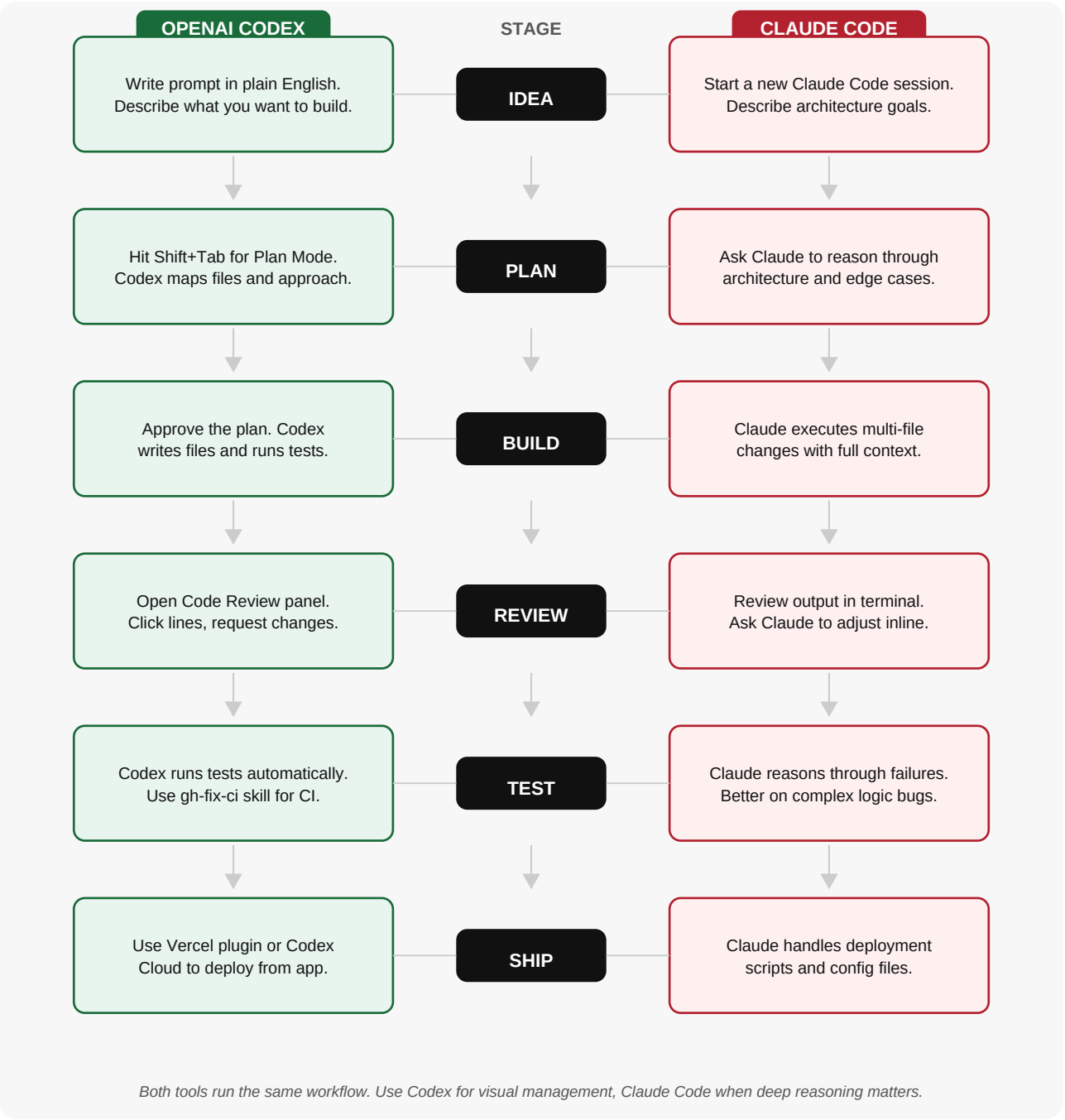
Use Claude Code when:

- You have a complex reasoning problem across a large codebase
- You need deep architecture planning across many files at once
- You want the highest quality reasoning on a difficult bug or refactor

The two-agent review loop: Claude Code generates the implementation plan and reasoning. Codex takes that plan and executes it, managing files, tests, and deployment. Claude Code does the final review pass for quality. Both CLIs support non-interactive mode so you can script the handoff.

FROM IDEA TO LIVE: THE FULL PROCESS FLOW

How a complete build goes from first idea to deployed product using both tools. Left column is Codex. Right column is where Claude Code adds value.



Both tools run the same workflow. Use Codex for visual management, Claude Code when deep reasoning matters.

4 WAYS TO USE CODEX

Codex runs in four different ways depending on your setup:

1	**Desktop App** An Electron app that looks like a coding environment. Best starting point for non-technical users. Visual UI, image previews, built-in git, and inline code review. Download from openai.com/codex.
2	**VS Code Extension** Install Codex as a VS Code extension if you already use VS Code. It runs directly inside your existing setup.
3	**CLI (Command Line)** Install via terminal: npm i -g @openai/codex. For users comfortable with the command line. Full docs at developers.openai.com/codex/cli
4	**Codex Cloud** Run tasks on OpenAI's servers. Start a job, close your laptop, come back to a finished result. Requires GitHub connected. Docs at developers.openai.com/codex/cloud

DOWNLOADING AND INSTALLING THE DESKTOP APP

Start here. The desktop app is the easiest entry point.

1	**Go to openai.com/codex** Find the download button. Choose Mac or Windows. The file is around 150MB. The install takes about one minute with no setup decisions to make.
2	**Sign in with your ChatGPT account** Codex runs on your existing ChatGPT subscription: Plus ($20/mo), Pro ($100/mo), Business, or Enterprise. Free accounts have a limited trial window. Use the same email as ChatGPT so you are not juggling two logins.
3	**Grant file permissions** Codex asks for read and write access. You can allow once or always. Codex does not read files on your computer unless you point it to a specific folder. It only works inside the project folder you choose.
4	**Optional: bring your ChatGPT history** In ChatGPT, go to Settings > Data Controls > Export Data. OpenAI emails you a zip file with every past conversation. Paste a short summary of the most relevant ones into your first Codex thread as context. Three sentences about your business and goals is enough.
5	**You are in** The app opens to a clean project sidebar on the left. You are ready to create your first project.

UNDERSTANDING PROJECTS AND THREADS

This is the most important concept to understand before you start building. Get this right and everything else clicks.

Projects

A project in Codex maps directly to a folder on your computer. When you add a project, you are pointing Codex at a real folder on your hard drive. Everything Codex creates, edits, or generates lives inside that folder as real local files you can open in any app. To add a project, click + in the sidebar or hit Cmd+O.

Linking Codex to Local Files

Because a project is just a folder, Codex has direct access to every file inside it. Drop existing documents, images, spreadsheets, or code into the folder and Codex can read and work with them immediately:

- Drop a client brief PDF in and ask Codex to generate a proposal from it
- Add your brand assets folder and Codex will reference them when building anything visual
- Put your existing website files in and ask Codex to update or redesign them
- Drop a CSV of data and ask Codex to analyse, clean, or visualise it

> Files Codex creates are saved directly into the folder in real time. Open them in Finder, upload them, or share them immediately with no export step.

One project per thing. Here is the rule:

One project per client	Keep each client's work isolated so the agent only has relevant context
One project per product	Your website is one project. Your app is another.
One project per automation	A recurring email workflow lives in its own project
One project per tool	A proposal generator, a content calendar, each gets its own folder

Threads

Inside each project you create threads. A thread is one conversation about one specific task. Hit Ctrl+N to create a new thread.

- **One thread per task.** 'Build the homepage' is one thread. 'Fix the contact form' is another.
- Threads persist when you close the app. Your context and work history are never lost.
- If a thread gets too long, type /compact to summarise it and keep the context clean.
- Type /fork to explore a different direction without touching the original thread.

YOUR FIRST SESSION: LEVEL 0 TO RUNNING IN 10 MINUTES

Follow this exact sequence the first time you open Codex:

1 — Create your first project folder

On your desktop or Documents, create a new folder. Name it clearly: 'my-website', 'client-smith', or 'content-tools'. In Codex hit Cmd+O and select that folder. It appears in your sidebar.

2 — Create a new thread

With your project selected, hit Ctrl+N. A blank thread opens. This is your workspace for one task.

3 — Enter Plan Mode first

Hit Shift+Tab. You will see Plan Mode activate at the top. Always do this before asking Codex to build anything. It makes Codex outline the approach before touching any files. Skipping this is the number one beginner mistake.

4 — Write your prompt using the 4 pillars

Goal, context, constraints, done when. Example: 'Goal: build a single-page website for my photography business. Context: brand colour is dark green, logo is in this folder. Constraints: plain HTML only. Done when the page loads with a working contact form.'

5 — Review and approve the plan

Codex produces a written plan. Read it. If anything looks wrong, say so before approving. Once you approve, it starts building.

6 — Review changes

When finished, hit Cmd+Option+B to open the code review panel. You see exactly what changed. Click any line and ask for adjustments in plain English.

WRITING GOOD PROMPTS: THE 4 PILLARS

Use these four elements every time for best results:

GOAL	What are you trying to achieve?	*Build a landing page for my photography business with a contact form*
CONTEXT	What files, folders, or background info?	*Brand colours are red and black. Logo is in the /assets folder*
CONSTRAINTS	What rules or limits should Codex follow?	*Use plain HTML and CSS only. No frameworks*
DONE WHEN	How will Codex know it succeeded?	*Done when the page loads without errors and the form submits*

PRICING: WHAT DOES IT ACTUALLY COST?

Codex is included with your existing ChatGPT subscription. There is no separate purchase.

Plan	Monthly Cost	What You Get	Best For
Free	$0	Limited trial access	Testing it out
Plus	$20/mo	Standard usage window, all core features	Individuals and freelancers
Pro	$100/mo	Higher usage limits, priority access	Power users and small teams
Business / Enterprise	Custom	Team management, SSO, data privacy controls	Companies and organisations

Usage limits explained: Codex uses a dual-window system. You have a 5-hour window and a weekly window, both running at the same time. Every request uses from both. When either hits zero you are paused until it resets. Check your usage anytime at chatgpt.com/codex/settings/usage or type /status inside Codex.

Token-saving tip: Use the Mini model for simple tasks like summarising or formatting. Save GPT 5.4 for the complex builds. This alone can extend your session budget significantly.

WHAT CODEX CAN AND CANNOT DO

Setting the right expectations before you start will save you frustration.

Codex CAN do this well	Codex CANNOT do this
Build websites, apps, tools, and dashboards from a description	Read your mind. Be specific about what you want.
Read, summarise, rewrite, and analyse documents and data files	Access the internet in real time unless you connect a web search MCP
Write, run, and fix code across many programming languages	Guarantee bug-free code. Always review what it builds.
Connect to external services like Gmail, Slack, and Google Calendar via MCP	Work on files outside your project folder unless you add them
Schedule and automate recurring tasks and workflows	Deploy to a live server without a plugin like Vercel (you need to set that up first)
Draft emails, proposals, content, and reports from templates	Remember things between sessions unless you use AGENTS.md or provide context
Manage your GitHub: commit, push, create pull requests, fix CI	Replace your own judgment. Review every significant piece of output before publishing.

MCP: HOW CODEX CONNECTS TO THE OUTSIDE WORLD

MCP stands for Model Context Protocol. In plain English: it is how you give Codex the ability to talk to other apps and services.

Without MCP, Codex only knows about the files in your project folder. With MCP, Codex can read your emails, check your calendar, post to Slack, search the web, talk to your database, and much more.

Think of MCP connectors like apps on a phone. You install the ones you need and Codex gains those powers. You do not need to understand how they work under the hood.

Connector	What Codex can do with it
Gmail	Read emails, draft replies, categorise inbox, send messages
Google Calendar	Read today's events, create new events, reschedule meetings
Slack	Send messages, read channel history, post summaries
GitHub	Commit code, open pull requests, read issues, fix CI failures
Figma	Read design files and translate them into working code
Notion / Google Drive	Read documents, create pages, update databases
Web Search	Search the internet for real-time information during a task

How to install an MCP connector: Open Codex, click the puzzle icon in the sidebar to open the MCP marketplace. Find the connector you want and click Install. Codex will walk you through authentication (usually just signing in to that service). Once connected, you can reference that service naturally in any prompt.

AGENTS.MD: CODEX'S PERMANENT MEMORY

AGENTS.md is a simple text file that lives in your project folder. Every time Codex starts a new thread in that project, it reads this file first. It is how you give Codex standing instructions that never need repeating.

If you have ever had to tell Codex the same thing twice, that thing belongs in AGENTS.md.

How to create one:

- Open your project folder in Codex
- Create a new thread and ask: 'Create an AGENTS.md file for this project'
- Codex will create the file and ask you what rules to include
- Add your answers and it builds the file for you

What to put in AGENTS.md:

Category	Example
Project overview	This is a website for a photography business in Auckland. The audience is brides and event planners.
Tech stack	We use plain HTML, CSS, and vanilla JavaScript only. No frameworks.
Brand rules	Primary colour is #2D4A3E. Font is Inter. Logo is always in the top left.
Tone of voice	Write copy in a warm, professional tone. No jargon. Short sentences.
Do not touch	Never modify the /legacy folder. Never change the homepage hero image.
Build command	Run 'npm run build' to test. Run 'npm run deploy' to publish.

WHEN THINGS GO WRONG: BASIC TROUBLESHOOTING

Codex will not always get it right the first time. That is normal. Here is how to handle the most common situations:

What happened	What to do
Codex went in the wrong direction	Stop it immediately. Type 'stop' or close the thread. Open the Code Review panel (Cmd+Option+B), revert the changes, then start a new thread with a clearer prompt. Next time, use Plan Mode first.
The output looks wrong or broken	Do not accept it. Tell Codex specifically what is wrong: 'The button is overlapping the text on mobile. Fix only that.' Be specific. Vague feedback produces vague fixes.
Codex seems stuck or keeps looping	Type /compact to summarise the thread and free up context. If still stuck, fork the thread with /fork and try a different approach from the plan.

You hit your usage limit	Check your window at chatgpt.com/codex/settings/usage. Switch to the Mini model to stretch your remaining budget. Your limit resets on a rolling 5-hour and weekly basis.
A file got changed that you did not want changed	Open the Code Review panel (Cmd+Option+B). Find the file. Click Revert on that specific file. Codex does not permanently delete files unless you ask it to.
Codex cannot find a file you mentioned	The file must be inside the project folder. If it is elsewhere on your computer, move it or copy it into the folder first, then ask again.

GLOSSARY: KEY TERMS IN PLAIN ENGLISH

You will encounter these terms throughout the app and in documentation. Here is what they all mean without the jargon:

Agent	An AI that takes actions on your behalf, not just answers questions. Codex is an agent because it reads files, writes code, and runs commands, not just chats.
Thread	One conversation with Codex about one task. Like a chat session that stays open and remembers everything you said.
Project	A folder on your computer that Codex works inside. All files it creates or edits live in this folder.
Context window	How much information Codex can hold in its 'working memory' at once. Long threads use more context. Use /compact when threads get long.
Plan Mode	A mode where Codex thinks and plans before acting. Activated with Shift+Tab. Always use this for anything important.
MCP	Model Context Protocol. How you connect Codex to external apps like Gmail or Slack. Think of it as installing an app.
AGENTS.md	A text file in your project folder that gives Codex standing instructions. Its permanent memory for that project.
Worktree	A parallel copy of your project on a separate branch. Lets two agents work on the same codebase at once without conflicts.
Sub-agent	A smaller helper AI that Codex can spin up to do a specific task, like research or searching your codebase, separately from the main thread.
Skill	A reusable set of instructions stored in a file that Codex loads automatically when relevant. Like a preset behaviour.
Token	The unit Codex uses to measure how much processing it does. Think of tokens as fuel. More complex tasks use more tokens.
CI / GitHub Actions	Automated tests that run when you push code to GitHub. Codex can read these logs and fix failures automatically.
Diff / Code Review	A view showing exactly what changed in your files. Green lines were added. Red lines were removed. Accessible with Cmd+Option+B.
Pull Request (PR)	A way of proposing a set of code changes for review before they go live. Codex can create these for you automatically.
Vercel	A popular free hosting service for websites. Codex can deploy your website directly to Vercel using its plugin.

WHAT COULD YOU BUILD? REAL EXAMPLES

Five things a non-technical person could build with Codex from scratch, plus exactly how to do each one.

A Simple Business Website

Project folder: my-website

Goal: A one-page website for a local service business with a contact form and booking link.

How to do it:

1. Create project folder 'my-website'. Drop in your logo and any brand assets.
2. New thread. Plan mode on. Prompt: 'Build a one-page site for a cleaning business. Include a services section, pricing table, and contact form. Brand colour is navy blue. Done when it renders cleanly in a browser.'
3. Review and approve the plan Codex produces.
4. Codex builds HTML, CSS, and form logic. Review changes in the diff panel.
5. Ask Codex to deploy using the Vercel plugin (install via /plugins).

End result: *A live, publicly accessible website. No hosting knowledge required.*

An Automated Email Responder

Project folder: email-automation

Goal: A tool that reads incoming emails and drafts personalised replies by category.

How to do it:

1. New project: 'email-automation'. Connect Gmail via the MCP marketplace inside Codex.
2. New thread. Plan mode. Prompt: 'Read my last 10 unread emails. Categorise each as enquiry, complaint, or partnership, then draft a reply. Done when drafts are in my Gmail drafts folder.'
3. Codex maps the email categories and writes reply templates from real examples.
4. Once approved, set it up as a weekly automation via the Automations panel.

End result: *A semi-automated inbox that drafts replies for you. You review and send.*

A Content Calendar Generator

Project folder: content-calendar

Goal: A spreadsheet-based planner that generates 30 days of post ideas from a brand brief.

How to do it:

1. New project: 'content-calendar'. Drop a text file in with your brand voice and topics.
2. New thread. Plan mode. Prompt: 'Read brand-brief.txt. Generate a 30-day social media calendar for Instagram and LinkedIn. Each day needs a topic, caption angle, and hashtag set. Done when output is a CSV I can import into a scheduler.'
3. Codex reads your brief, generates the plan, and outputs a CSV into the project folder.
4. Open the CSV and import directly to Buffer, Later, or any scheduling tool.

End result: *30 days of content ideas in under 5 minutes, formatted and ready to schedule.*

A Client Proposal Generator

Project folder: proposals

Goal: A tool that takes a client brief and outputs a formatted proposal PDF.

How to do it:

1. New project: 'proposals'. Drop a previous proposal PDF in as a template reference.
2. New thread. Plan mode. Prompt: 'Read client-brief.txt and template.pdf. Generate a new proposal matching the template structure. Fill in scope, pricing, and timeline. Done when output is a readable PDF.'
3. Codex reads both files, extracts the structure, and writes the new proposal.
4. Review the draft in the thread. Ask for edits. The PDF saves into your project folder.

End result: *A client-ready proposal document generated in minutes from a short brief.*

A Personal Dashboard App

Project folder: my-dashboard

Goal: A web app showing daily tasks, calendar events, and top priorities in one view.

How to do it:

1. New project: 'my-dashboard'. Connect Google Calendar via the MCP marketplace.
2. New thread. Plan mode. Prompt: 'Build a minimal dashboard. Pull today's events from Google Calendar. Show a checklist for tasks and a top-3 priorities section. Done when it loads in a browser with live calendar data.'
3. Codex builds the app, connects the calendar API, and styles the layout.
4. Deploy via the Vercel plugin. Bookmark and open each morning.

End result: *Your own personal command centre, built to your exact spec.*

MUST-HAVE SKILLS

Skills are reusable instruction files stored at ~/.agents/skills/ that Codex loads automatically. Install via the Skills Marketplace (puzzle icon) or type $skill-installer [name].

1	**create-plan** **What:** Forces Codex to produce a written implementation plan before it opens a single file. **Why:** Prevents the wrong-direction session that wastes time and tokens. Start here.
2	**WarpGrep** **What:** A search subagent that runs 8 parallel code searches at once. **Why:** Finds things in your codebase in around 5 seconds instead of 75 seconds.
3	**gh-fix-ci** **What:** Reads failing GitHub Actions logs and automatically writes the fix. **Why:** Turns a 45-minute debugging loop into a background task.
4	**gh-address-comments** **What:** Reads every PR review comment and addresses them all in one session. **Why:** The review-response cycle is where development velocity goes to die.
5	**frontend-skill** **What:** Overrides generic AI design decisions. Forces a real colour palette and typography choice. **Why:** Without it, Codex builds UIs that work. With it, they look like someone made a decision.
6	**stop-slop** **What:** Strips AI writing patterns from READMEs, docs, and commit messages. **Why:** The code can be excellent and still feel hollow because the docs were generated.

KEY FEATURES AT A GLANCE

Project Sidebar	Every top-level item is a project (a local folder). Inside each, you create threads. Sessions persist when you close the app. No lost context.
Git Built-In	Commit, create pull requests, and review diffs all inside the app. No switching to GitHub or a separate terminal.
Worktrees	Clone your project into a parallel branch and run two agents on the same codebase simultaneously without conflicts.
Automations	Schedule tasks on a timer. Pick a project, set when to run, and let it go. Great for recurring reports, content generation, or inbox management.
Voice Input	Hit Ctrl+M to speak your prompt instead of typing. Fast for ideation and quick follow-ups.
Hooks	Trigger actions at lifecycle events, for example automatically lint code after every edit or send a Slack message when a task finishes.
Code Review Panel	Cmd+Option+B opens inline diffs. Click any line, request a change, watch Codex iterate. Most useful feature for non-technical reviewers.
Sub-Agents	Spin up smaller AI helpers for specific tasks using cheaper, faster models. Keeps your main thread clean and your costs down.
Cloud Execution	Send a task to OpenAI's servers and let it run in the background. Come back hours later to a finished pull request.

COMMON MISTAKES TO AVOID

These are the things that trip up beginners and experienced users alike:

01

Skipping plan mode

Always plan first. Most tasks worth doing require it. Execution without a plan wastes tokens, time, and often has to be undone entirely.

02

One big project for everything

Keep projects scoped tightly. One client, one product, one tool per project. Context is everything to an AI agent.

03

Repeating the same instructions

If you keep reminding Codex of a rule, that rule belongs in your AGENTS.md file. Add it once. Every future session inherits it automatically.

04

Not giving validation criteria

Tell Codex how to know it succeeded. The phrase 'Done when...' is the signal it looks for. Without it, you are the only quality gate.

05

Two threads editing the same files

If two tasks need to touch the same codebase, use Worktrees. Running two threads against the same files on the same branch will cause conflicts.

06

Automating something not working yet

Get the task right manually first, then put it on a schedule. Automating a broken workflow just runs the broken workflow faster.

07

Using the highest model for every task

Mini models are fast and cheap for simple jobs. Save GPT 5.4 with high reasoning for the hard problems. Match compute to complexity.

08

Not using AGENTS.md

AGENTS.md is the permanent memory for your project. Build commands, naming conventions, tone preferences, what not to touch. Any rule you have told Codex twice belongs here.

SHORTCUTS WORTH MEMORISING

`Shift + Tab`	Enter Plan Mode
`Cmd + O`	Open or add a new project
`Ctrl + N`	New thread in the current project
`Cmd + B`	Toggle the project sidebar
`Cmd + Option + B`	Toggle the code review panel
`Cmd + J`	Open the built-in terminal
`Ctrl + M`	Start voice input
`Cmd + K`	Command palette
`Cmd + F`	Search within a thread
`/models`	Switch AI model or reasoning level
`/fast`	Toggle fast mode for GPT 5.4
`/compact`	Summarise a long thread to save context
`/fork`	Branch a conversation without touching the original
`$skill-name`	Run or install a skill