Codex: A Safe Way to Work With an AI Coding Agent

Codex is an AI coding agent. It can read a project, propose changes, and run commands in it. This guide shows you how to use it safely from the first session to a regular routine. You work on a fake practice project. You do not need the class video.

> **Check the current Codex documentation from OpenAI** before you rely on any feature, plan or limit. This guide describes safe habits that stay the same even when features change.

Part 1. The words you need

Word	Meaning
Agent	A tool that works in steps: it reads files, plans, edits and runs commands
Prompt	The instruction you give the agent
Diff	The exact lines added or removed by a change. Always read the diff before accepting
Command	Something the agent runs in the project, such as a test or an install
Approval	Your yes or no before the agent runs a command or changes a file
Sandbox	A limited area where commands run with restricted access
Repository	A project folder with its history, usually tracked with Git
Branch	A separate line of work, so changes can be reviewed before they join the main code

Part 2. Set up a practice project

1. Create a small practice project with one page and one test. Use fake data only.
2. Put the project in its own folder, and start a Git repository in it so every change is recorded.
3. Create a branch called `practice` for the agent's work.
4. Write one sentence describing what the project should do. Keep it in a file called `README.md`.

You should see: a folder with one page, one test, a README, and a clean Git status.

Part 3. Your first session

1. **Ask for a plan first.** Example: "Explain what you would change to add a hello message to the page. Do not edit any files yet."
2. **Read the plan.** If anything is unclear, ask about that step only.
3. **Approve one step at a time.** Read each proposed diff line by line.
4. **Run the tests yourself** after each accepted change.
5. **Commit with a clear message** only when the tests pass.

Pass when: the tests pass, and every changed line is one you can explain in your own words.

Part 4. Good requests and weak requests

Good request	Weak request
"Add input validation to the form on this page, and add one test."	"Make the app better."
"Explain the error on line 12 before changing anything."	"Fix everything."
"List the files you would change, then wait for my approval."	"Rewrite the whole project."
"Show me the diff for this one file."	"Just apply it."

Part 5. Safety rules

Risk	Control
A secret in the code or prompt	Store keys in environment settings. Never paste a key into a prompt or a file
A change you do not understand	Ask for a plain-English explanation of that diff before you accept it
A command that deletes or publishes	Refuse it unless you have read the exact command
Work in your main project	Use a copy or a separate branch, and review before merging
Too many files changed at once	Ask the agent to stop and list the files, then narrow the task
Installing unknown packages	Ask why each package is needed and check it on its official page first

Part 6. Reviewing a change

1. Open the diff and read each file.

2. Check that the change does only what the request asked for.
3. Check that no secret, personal data or unrelated file has changed.
4. Run the tests and read the output, not just the result.
5. Ask one question for each line you do not understand.

Part 7. Common problems and fixes

Problem	Fix
A change breaks the tests	Ask for the failing test output, then fix one thing at a time
The agent edits too many files	Stop it, ask for the list of files, then narrow the request
You cannot tell what changed	Ask for a file-by-file summary of the diff
The same error keeps returning	Ask for the cause before any new change
A command needs more access	Refuse until you know exactly why it is needed

Part 8. A regular routine

1. Start each session with a clear goal written in one sentence.
2. Ask for a plan, then approve steps one by one.
3. Review every diff and run the tests.
4. Commit, then write one line in a log: what changed and why.
5. End the session by asking for a summary of what was done and what remains.

Part 9. Checklist

Item	Done
Practice project used, not the main project	[ ]
Plan requested before any edit	[ ]
Every diff read before it was accepted	[ ]
Tests run and read after each change	[ ]
No secrets in the project or prompts	[ ]
Current Codex documentation checked	[ ]

Part 10. Quiz (answers follow)

1. Why ask for a plan before any edit?
2. What is a diff?
3. Where should a key be stored?
4. Why use a branch?
5. What do you do when a change breaks the tests?

Answers: 1. So you can check the approach before any file changes. 2. The exact lines added or removed. 3. In environment settings, never in a prompt or file. 4. So changes can be reviewed before they join the main code. 5. Read the failing output and fix one thing at a time.

Part 11. Glossary

Term	Plain meaning
Agent	A tool that works through steps on a task.
Diff	The exact changes in a file.
Approval	Your yes before a command or change runs.
Branch	A separate line of work for review.

Feature names, plans and access rules change. Check OpenAI's current developer documentation on the day you use Codex.