

Homework — Programming Today

Estimated time: ~60 min · No coding required · A free editor, a free GitHub account, and an AI assistant of your choice

Three short tool-tour exercises. The goal: take the abstract picture from the theory page (an editor + a repo + an AI assistant, glued together by git) and see *it yourself* in the actual tools professionals use every day in 2026. You won't write a single line of code — you'll just install, click around, and notice the parts. Take a screenshot at each step.

Before you start

You only need a laptop and a browser. Tasks 1 and 2 use the free tier of public tools (VS Code is free; GitHub is free for public browsing). Task 3 works with any AI chat assistant you can reach — *Claude*, *ChatGPT*, *Gemini*, the *Cursor* playground, or whichever one your organisation already allows. You are not writing or running any code.

Task 1 — Install VS Code and poke around (no coding required)

VS Code is, as of 2026, the editor most professional engineers actually open every morning. Today you'll just install it and learn where the five core panels are — no project, no code, just the geography.

1. Download VS Code from code.visualstudio.com. Pick the build for your operating system. Install it the normal way (drag to *Applications* on macOS, run the installer on Windows, use your package manager on Linux).
2. Open VS Code. You'll see an empty welcome tab — that's fine.

3. Look at the thin vertical strip on the very left edge. It has a stack of icons. Click each one in turn and note what panel opens:
 - The **Explorer** (folder icon) — this is where your files would show up if you opened a project folder.
 - The **Search** panel (magnifying glass) — project-wide find & replace.
 - The **Source Control** panel (branching icon) — empty until there's a git repo open, but this is where every commit you ever make starts.
 - The **Run and Debug** panel (play-with-bug icon) — for stepping through code line by line.
 - The **Extensions** panel (four-squares icon) — the marketplace; this is how VS Code becomes a Python editor, a Rust editor, a SQL editor, etc.
4. Click the gear icon at the very bottom-left and open **Settings**. Scroll for ten seconds and notice there are *thousands* of options. You don't need to change anything.
5. Open the **Command Palette**: **Cmd + Shift + P** (macOS) or **Ctrl + Shift + P** (Windows/Linux). Type a few letters of a word like *theme* or *format* and notice it offers you a filtered list of every command in the editor. This is the single most-used keyboard shortcut in VS Code.
6. Take a screenshot of the editor with the Command Palette open.

✓ **Self-check: you can name each of the 5 sidebar panels (Explorer, Search, Source Control, Run and Debug, Extensions); you can open the Command Palette and find one command by typing a keyword. You now know the geography of the tool every engineer in your team will assume you know.**

Task 2 — Read open-source code without writing any

Most software in the world lives in a public git repository on GitHub, and you can read all of it without an account. Today you'll browse one of the most-watched repositories on the platform and notice that it is, after all, just files of text.

1. Open github.com. In the top search bar, type `react`. The top result should be **facebook/react** — click it.
2. You're now on the repo's home page. The big file tree on the left is the project. Browse it:
 - Open `README.md` — this is what GitHub renders by default; it's the project's front door.
 - Open the `packages/` folder.
 - Click into `packages/react/`, then `src/`, then a file ending in `.js` (e.g., `React.js`).

Notice: it's just a text file. Long, dense, full of comments — but text.
3. Go back to the repo home page. Look near the top right for the **Contributors** count, and on the file tree for the timestamp on the most recent commit (the "last updated" column).
4. Click the **Insights** tab at the top, then **Pulse** in the left sidebar. Pulse summarises activity over the past week: pull requests opened/merged, issues opened/closed.
5. Take a screenshot of the Pulse page.

✓ **Self-check: you can answer (a) approximately how many people have contributed to this repo (you saw the contributors count), (b) when was the most recent commit, (c) write one sentence describing how active the project has been in the past week based on what Pulse shows you (e.g., "about 20 PRs merged and 30 issues opened — very active"). You do NOT need to understand any of the code — the point is that a repo is a public, browsable, time-stamped record of every change ever made to a piece of software.**

Task 3 — Try an AI coding assistant on a real question

AI assistants are, in 2026, the third tool every engineer keeps open all day (after the editor and the browser). Today you'll use one for the exact thing it's best at: explaining a confusing concept in plain language, then turning the explanation into a concrete example.

1. Open whichever AI chat assistant you can reach: claude.ai, chatgpt.com, gemini.google.com, or the playground at cursor.com. Any of them works.
2. Paste in this exact question:

In one paragraph, explain what `git rebase` does and how it differs from `git merge`.

Read the response carefully.

3. Ask a follow-up:

Show me a 4-line shell example for each.

The assistant should produce two short code blocks. Read them — don't run them.

4. Ask one more follow-up of your own choosing — something like *"When would I prefer one over the other on a team project?"* or *"What's the danger of rebasing a branch that other people are already working on?"*.
5. Take a screenshot of the conversation.

✓ **Self-check:** you can answer (a) does the explanation roughly match what you read in this lesson's theory page (and Lesson 04's reading list, if you've done it), (b) did the assistant give you example commands that you could, in principle, run on a real git repo, (c) did the follow-up answer add useful context or just repeat itself? You've now used an AI assistant the way working engineers use one all day: not to write code *for* you, but to compress an hour of Stack-Overflow-reading into a two-minute conversation. You do NOT need to run any of the commands.

What "done" looks like

You finish this homework when you have, in your notes:

- A screenshot of VS Code with the Command Palette open — plus your own short list naming the five sidebar panels you clicked through.

- A screenshot of the **Insights → Pulse** tab on the `facebook/react` repo — plus your own one-sentence answer on contributor count, last-commit time, and "how active is this project this week".
- A screenshot of your AI-assistant conversation about `git rebase` vs. `git merge` — plus a one-line note on whether the explanation matched the theory page.

You've now *touched* the three tools that defined the theory page: the editor (Task 1), the public repository (Task 2), and the AI assistant (Task 3). The fourth tool — git itself — is the invisible glue that ties them together; you'll see it in action when you start your first project. Next lesson we zoom out one more level — from *how* code gets written to *how* the resulting systems are shaped: monoliths, microservices, serverless, event-driven.