

# Week 2: Coding Basics

Session agenda

**Sessions:** September 28 and 30, 10:30–11:50 a.m.

**Leads:** Shurui Liu and Fred Rajasekaran

## Before class

- Install the ChatGPT desktop app and Visual Studio Code (VS Code).
- Create a GitHub account and make sure you can sign in.
- Windows users: follow the [WSL setup guide](#).
- Before Day 2, check that you can access FarmShare with your SUNet ID and Stanford two-step authentication. See [Getting Connected](#).

## Day 1: From a coding agent to a GitHub repository

Monday, September 28 · 80 minutes

**Goal:** Build a small program with a coding agent, understand how it works, and track and share it with Git and GitHub.

| Time      | Activity                                                                                                                                                                                                                                                                                              |
|-----------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 0–5 min   | <b>Warm-up and setup.</b> Finish the preparation above and resolve initial setup issues.                                                                                                                                                                                                              |
| 5–10 min  | <b>What is a coding agent?</b> Discuss how an agent can read, edit, and run code, and where human review fits in.                                                                                                                                                                                     |
| 10–20 min | <b>Set up a coding agent.</b> Install a desktop agent, select a model, and explore the interface. Experienced participants may use their preferred tools.                                                                                                                                             |
| 20–30 min | <b>First project.</b> Create a project folder. Ask Codex to implement a prime sieve and a prime-counting function, then explain the code. Check a few small examples and generate a plot to reuse on Day 2.                                                                                           |
| 30–35 min | <b>Discussion.</b> Where did things go wrong? How did you check the output? What did the agent’s explanation clarify?                                                                                                                                                                                 |
| 35–55 min | <b>Live terminal and Git demonstration.</b> Shurui works through the workflow on a volunteer’s laptop, starting with a fresh project folder. Practice navigation, repository creation, commits, and pushing to GitHub.                                                                                |
| 55–80 min | <b>Command-line agents and LaTeX.</b> Install the Codex or Claude Code CLI and discuss why a CLI is useful on a remote server. Demonstrate the Overleaf–Git workflow with an accessible project, then ask the agent to suggest corrections to typos or mathematical mistakes. Review its suggestions. |

## Terminal and Git topics

- Navigate with `cd` and inspect files with `ls -alh`; introduce package managers such as `apt` and Homebrew.
- Practice `git init`, `git add`, `git rm`, `git commit`, `git status`, and `git log`.
- Explain `git fetch`, `git merge`, `git pull`, and `git push`. A pull fetches remote changes and integrates them according to the repository’s configuration.
- Review changes and commit deliberately. Explain why force-pushing can overwrite shared history.

**Day 1 checkpoint:** Create a local repository and a private GitHub repository; push the sieve code; write a short `README.md` yourself, then commit and push it.

## Day 2: Running code on Stanford FarmShare

Wednesday, September 30 • 80 minutes

**Goal:** Connect to a remote machine, move code and results between machines, and understand when to use a scheduled compute job.

| Time      | Activity                                                                                                                                                                                                                               |
|-----------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 0–5 min   | <b>Warm-up and discussion.</b> What computing resources are available, and when is a laptop enough?                                                                                                                                    |
| 5–15 min  | <b>Connect with SSH.</b> Log in to FarmShare, identify the remote machine and current directory, and distinguish the remote shell from the local one.                                                                                  |
| 15–30 min | <b>Transfer files with SCP.</b> Copy a practice file from the laptop to FarmShare, verify the transfer, and remove only that practice copy.                                                                                            |
| 30–45 min | <b>Clone the project with Git.</b> Set up GitHub authentication and clone the private repository on FarmShare. Keep credentials out of the repository.                                                                                 |
| 45–55 min | <b>Run the sieve.</b> Run a small, quick example and inspect the output. Discuss file permissions, privacy, and CPU and memory use on a shared system.                                                                                 |
| 55–65 min | <b>Bring results back.</b> Copy the generated plots from FarmShare to the laptop and open them locally.                                                                                                                                |
| 65–70 min | <b>Introduction to Slurm.</b> Explain login nodes versus compute nodes and demonstrate submitting a small batch job. Use <code>top</code> or <code>htop</code> to inspect resource use.                                                |
| 70–80 min | <b>Troubleshooting and optional tools.</b> Address setup issues. If time permits, introduce <code>tmux</code> , Python 3, <code>uv</code> , or a CLI agent on FarmShare, including memory use, device authentication, and logging out. |

**Where to run code:** Keep login-node work brief and within the shared resource limits. Use a Slurm job for long-running or CPU- or memory-intensive computations. See the [FarmShare overview](#) and [Running Jobs](#).

**Day 2 checkpoint:** Clone and run your project on FarmShare, retrieve a plot, and explain how you would submit a larger computation through Slurm.

### Exercises

1. **Propose a project.** Think of a software project you would actually want to use. Post a short description in the course Slack, including its purpose and a first achievable milestone.
2. **Explore a codebase with an agent.** Choose a repository, read its `README.md`, clone it, and ask the agent to explain its structure and intended workflow. Summarize the main components, required dependencies, and one part of the code you now understand. Examples include mathematical reasoning agents such as `rethlas` or `danus`, or coding-agent projects such as `deepseek-harness` or `opencode`. For a larger challenge, explore Mathlib (with a Lean toolchain) or `ver1` (with its reinforcement-learning dependencies).