DAY FIVE

05 /10

Meet Claude Code — *an agent in your terminal*

For four days Claude has lived in a chat window. Today it moves into the place developers actually work: the command line, with the ability to read a whole codebase, edit files, and run commands — under your direction.

BY THE END OF DAY 5, YOU WILL BE ABLE TO TEACH OTHERS TO —

- Explain what Claude Code is and is not
- Describe the explore → plan → act → verify loop
- Name the work Claude Code is strongest at
- Install it and start a first session
- Say why a human stays in the loop
- Point a fresh session at a real repository

01 Why today matters

ORIENTATION

Until now, every surface has been conversational — you describe, Claude responds, you copy the result somewhere. Today that changes. Claude Code is Claude working directly inside a software project: it can read every file, make edits across many of them, run commands, and follow a task through to completion — all from the terminal, and all with you watching and approving.

This is the first of four "builder" days. You do not need to be a working engineer to teach it — but you do need the mental model: this is an agent, not a chat box. It takes actions in the real world of your files. That power is exactly why the human-in-the-loop habit from earlier days becomes non-negotiable here.

WHO THIS DAY IS FOR

Anyone who works with code, or works alongside people who do. Even non-coders benefit: Claude Code is also a remarkably good way to understand an unfamiliar codebase by simply asking it questions.

02 What Claude Code actually is

CORE CONCEPT

Claude Code is an agentic coding tool that lives in your terminal. "Agentic" is the key word: instead of handing back text for you to act on, it acts — within a project directory you point it at, and within limits you set.

THE MENTAL MODEL

A capable colleague who shares your screen.

Picture pairing with a fast, well-read teammate who can see your whole project. You describe what you want; they explore the code, propose an approach, make the changes, and run the tests — pausing to check with you at the moments that matter. That is the shape of a Claude Code session.

2.1 What it can do

- ✓ Read and understand an entire codebase, including its history
- ✓ Write new code and edit existing files across the project
- ✓ Run commands — tests, builds, linters, git operations
- ✓ Carry a multi-step task from description to working result
- ✓ Explain unfamiliar code in plain language

2.2 What it is not

It is not autonomous in the "walk away and trust it" sense, and it is not a replacement for understanding your own project. It is a powerful instrument played by a human who keeps their hands on it. Teach it as amplification, never as abdication.

KEY IDEA TO INSTALL

The chat surfaces give you answers. Claude Code gives you actions — taken inside your real files, at your direction. Everything else today follows from that one difference.

03 Getting it running

CORE CONCEPT

Installation is a one-time step. There are two routes; pick whichever suits the room.

3.1 The native installer (no extra tools needed)

The self-contained installer is the simplest path — it needs nothing else installed first.

```
# macOS / Linux / WSL
curl -fsSL https://claude.ai/install.sh | bash

# Windows (PowerShell)
irm https://claude.ai/install.ps1 | iex
```

3.2 The npm route (for Node.js users)

If learners already work in the Node.js ecosystem, the package install is familiar. It requires Node.js 18 or newer.

```
npm install -g @anthropic-ai/claude-code
```

> ONE THING TO NEVER DO
>
> Do not run the npm install with `sudo`. It causes file-ownership problems that are painful to undo. If you hit a permissions error, fix the npm directory instead — the docs walk through it.

3.3 Starting a session

Once installed, the whole interface is one command. Move into a project folder and run it:

```
cd my-project
claude
```

The first run walks through a quick sign-in with your Anthropic account. After that, you are in a session — talking to Claude in plain language, inside that project. There are also editor integrations — a VS Code extension, a JetBrains plugin, a Slack integration, and a desktop app — but the terminal is the place to teach it first, because it shows the loop most clearly.

04 The loop: explore, plan, act, verify

CORE CONCEPT

Every good Claude Code session — large or small — moves through the same four beats. Teaching the loop is teaching the tool.

> BEAT ONE
>
> **Explore.**
>
> Before changing anything, Claude reads. It looks at the relevant files, the structure, sometimes the git history, to build an accurate picture of how the project actually works. You can ask it to do only this — "explain how authentication works here" — and stop.

BEAT TWO

Plan.

For anything beyond a trivial change, ask for the plan first: which files, what approach, in what order. This is the Day 4 habit — reason before acting — and it is where you catch a misunderstanding before any code is touched.

BEAT THREE

Act.

With the plan agreed, Claude makes the edits and runs the commands. It shows you what it is doing as it goes. Significant or irreversible actions pause for your approval.

BEAT FOUR

Verify.

The work is not done when the code is written — it is done when it is checked. Run the tests, review the changes, confirm the result. Claude can help verify, but the judgement is yours.

> *"Explore, plan, act, verify. If a session skips a beat, that is usually where it goes wrong."*

05 Staying in the loop

CORE CONCEPT

Because Claude Code takes real actions, the human's role shifts from author to director and reviewer — and that role is essential, not optional.

- **Approve consequential steps.** Claude pauses before significant or hard-to-undo actions. Read what it proposes; do not reflexively wave it through.
- **Review the diff.** The changed code is right there. Reviewing it is how you stay the engineer of record — and how you catch the subtle thing.
- **Work in version control.** A project under git means every change is visible and reversible. This is the safety net that makes confident delegation possible.
- **Scope the task.** A clear, bounded request — "add input validation to this one form" — goes far better than a vague, sprawling one. Day 3's specificity lesson applies directly.

TEACHING SHORTCUT

Frame the human's job as code review, continuously. Most developers already know how to review a colleague's pull request. Claude Code is that — just faster, and with a teammate who never gets tired of your questions.

LAB 05 ~35 MIN

First session on a real repo

Each learner installs Claude Code, points it at a small real project, and runs one full loop — explore, plan, act, verify — on a genuinely tiny change.

1. **Install** via the native installer or npm, then run `claude` inside a small project folder and complete sign-in.
2. **Explore.** Ask: "Explain what this project does and how it is structured." Read the answer. This alone is often a revelation.
3. **Plan.** Pick a tiny change — a clearer error message, a small comment, a trivial helper — and ask Claude to outline its plan before touching anything.
4. **Act.** Approve the plan and let Claude make the change. Watch what it does and where it pauses.
5. **Verify.** Review the diff and run whatever check the project has. Confirm the change is what you intended — and nothing else moved.

a learner who has felt the loop once — and understands viscerally that their job in it is to direct and to check, not to watch passively.

TEACHING NOTES

How to teach Day 5 well

OPEN WITH THIS

Ask the room: "What is the difference between someone telling you how to fix your car, and someone fixing it?" That gap — advice versus action — is the entire jump from the chat surfaces to Claude Code. Everything today is a consequence of it.

PACE & EMPHASIS

The loop in Section 04 is the heart of the day — spend your time there. Installation (03) will have a few stragglers; pair people up rather than letting it eat the room. Do not let non-coders opt out: the "explore" beat is for them.

DISCUSSION PROMPTS

· Where in your work is there a codebase nobody fully understands? · Which beat of the loop are you most tempted to skip, and why? · What does "code review, continuously" change about how you would work?

COMMON MISCONCEPTIONS TO PRE-EMPT

"It is just chat that can also code."
No — it takes actions in your real files. That is a difference in kind, not degree.

"Agentic means I can walk away."
It means it acts, not that it is unsupervised. You direct and you review — every session.

"You have to be a real engineer to get value."
Asking it to explain a codebase is one of its best uses, and needs no coding at all.

IF YOU ONLY HAVE 30 MINUTES Teach the agent-vs-chat distinction (02) and the four-beat loop (04). Do Lab steps 1–2 together as a group demo on one screen. The loop is the irreducible core; installation can be homework.

Day 5 Cheat Sheet

Claude Code	An agentic coding tool that lives in your terminal — reads your codebase, edits files, runs commands.
Agentic	It takes actions, not just gives answers — within a project directory and within limits you set.
Native installer	`curl -fsSL https://claude.ai/install.sh | bash` — self-contained, needs nothing else.
npm install	`npm install -g @anthropic-ai/claude-code` — needs Node.js 18+. Never use `sudo`.
Start a session	`cd` into a project, run `claude`, sign in once.
The loop	Explore → Plan → Act → Verify. Every good session runs all four beats.
Plan first	For anything non-trivial, get the approach agreed before code is touched.
Stay in the loop	Approve consequential steps, review the diff, work under git, scope tasks tightly.
Editor integrations	VS Code extension, JetBrains plugin, Slack, desktop app — same tool, different doorway.

Check for understanding

Five questions. Learners should be able to answer all five before Day 6.

1. In one sentence, what makes Claude Code different from the Claude chat surfaces?
2. Name the two installation routes and the one prerequisite that only one of them needs.
3. What are the four beats of a Claude Code session, in order?
4. Give two concrete habits that keep a human properly “in the loop.”
5. Name a high-value use of Claude Code that involves no code-writing at all.

> **Answer notes —** 1) It takes actions inside your real project files — editing, running commands — rather than only returning text. 2) The native installer (self-contained, no prerequisites) and npm (needs Node.js 18+). 3) Explore, plan, act, verify. 4) Any two: approve consequential steps, review the diff, work under version control, scope tasks tightly. 5) Asking it to explore and explain an unfamiliar codebase.

Day 5 in five lines

- Claude Code is an agentic tool in your terminal — it acts inside your real project, not just a chat box.
- Install once via the native installer or npm, then run `claude` in any project folder.
- Every session runs the same loop: explore, plan, act, verify.
- Because it takes real actions, the human's job is to direct and to review — continuously.
- Even without writing code, it is a powerful way to understand an unfamiliar codebase.

TOMORROW — DAY 6 → **Claude Code in Practice — workflows, project memory, and extending it**