DAY SIX

06 /10

Claude Code in Practice — *workflows that hold up*

Day 5 introduced the tool and its loop. Today is about using it like someone who has used it for months: scoping work well, reviewing with discipline, giving the project a memory, and extending what Claude Code can reach.

BY THE END OF DAY 6, YOU WILL BE ABLE TO TEACH OTHERS TO —

- Scope a task so a session goes well
- Review diffs without rubber-stamping
- Explain how MCP and subagents extend reach
- Run plan-first as a working discipline
- Use a project memory file effectively
- Set sensible permissions and boundaries

01 Why today matters

ORIENTATION

Knowing what a tool is and knowing how to use it well are different things. Yesterday's loop — explore, plan, act, verify — is the skeleton. Today adds the muscle: the working habits that separate a frustrating session from a productive one, and the configuration that makes Claude Code fit a real project instead of a toy one.

Everything here builds on Day 5. If the loop is not yet solid, revisit it first — today assumes it. Teach this day as craft: small disciplines, practised until they are automatic.

PREREQUISITES

Day 5's install and the four-beat loop. A real project to practise on — ideally one under version control, so every change is visible and reversible.

02 Scope before you start

CORE CONCEPT

The single biggest predictor of a good session is a well-scoped task. A request that is clear and bounded gives Claude a target it can actually hit; a vague, sprawling one produces vague, sprawling work.

2.1 One job at a time

"Refactor the whole app" is not a task — it is a wish. "Extract the date-formatting logic into one shared helper and update the three places that use it" is a task. Teach learners to break big intentions into bounded pieces, each of which can be reviewed on its own.

2.2 Say what done looks like

Bring Day 3 forward: state the task, the relevant context, and what a finished result looks like — "the tests still pass, and the helper has no other callers left." A session with a clear finish line knows when to stop.

KEY IDEA TO INSTALL

A Claude Code session is only as good as the task you hand it. Scope tightly, define done, review in pieces. Big ambitions are reached one bounded task at a time.

03 Plan-first and disciplined review

CORE CONCEPT

Two habits do most of the quality work. Both are simple. Both are easy to skip when you are in a hurry — which is exactly when they matter most.

HABIT ONE

Plan first, every time.

For anything beyond a trivial edit, ask Claude to lay out its approach and wait for your go-ahead. Reading a plan takes a minute; it catches the wrong assumption before any effort is spent on it. The plan is also where you learn what the change really involves.

HABIT TWO

Review the diff like you mean it.

When Claude makes changes, it shows you exactly what changed. Read it as you would a colleague's pull request — looking for the subtle thing, the unintended side effect, the "that is not quite how we do it here." Approving without reading is the one habit that quietly erodes everything else.

> *"Plan-first costs a minute. Skipping it can cost an afternoon. The maths is not close."*

THE RUBBER-STAMP TRAP

After a few good sessions, trust creeps in and review gets lazy. Resist it. The diff review is not bureaucracy — it is the thing that keeps you the engineer of record rather than a spectator to your own project.

04 Give the project a memory

CORE CONCEPT

By default, each session starts fresh. But most projects have standing facts — conventions, commands, structure, "we always do it this way" — that you would otherwise re-explain every time. Claude Code can read a project memory file kept in the repository, so those facts are simply known.

WHAT BELONGS IN PROJECT MEMORY

- ✓ How the project is structured and where key things live
- ✓ The commands that matter — how to run tests, build, lint
- ✓ Conventions and style rules the team follows
- ✓ Things to be careful with, and things never to touch

This is the Day 4 "persistent instruction layer" again, in a developer's clothing. It is standing orders for the project, checked in alongside the code so the whole team — and every future session — shares the same context.

TEACHING SHORTCUT

Call it the project's onboarding doc for a new teammate — except the teammate reads it instantly and never forgets it. If a fact would go in a human's onboarding, it belongs in project memory.

05 Extending reach — and setting limits

CORE CONCEPT

Out of the box, Claude Code works with your files and your shell. Two mechanisms extend that reach — and one discipline keeps the extension safe.

5.1 MCP — connecting to outside systems

The Model Context Protocol lets Claude Code reach beyond the local project to other tools and data sources — issue trackers, documentation, services. It is a preview of Day 8; for now, the teaching point is simply that the tool's reach is extensible, not fixed.

5.2 Subagents — delegating sub-tasks

For larger jobs, a session can delegate a focused piece of work to a subagent — a separate worker handling one part — and bring the result back. The mental model is a lead handing a well-defined task to a teammate.

5.3 Permissions and boundaries

More reach means more care. Claude Code asks before consequential actions, and you can set the boundaries: what it may do freely, what needs a check, what is off-limits. Teach learners to set these deliberately — matched to how much they trust the task and the context — rather than accepting defaults without thought.

> **KEY IDEA TO INSTALL**
>
> Reach and restraint travel together. Extend what Claude Code can touch only as far as you have set sensible limits — and let the size of the task set how tight those limits are.

LAB 06 ~40 MIN

A realistic feature, done properly

Each learner takes one small but real feature or fix through a full disciplined workflow — scoped, planned, reviewed, verified — with a project memory file in place.

1. **Write a project memory file** for your practice repo: structure, key commands, one or two conventions. Keep it short and true.
2. **Scope a real task.** Pick something small but genuine, and write it as task + context + what "done" looks like.
3. **Plan first.** Ask Claude for its approach. Read it. Push back or adjust before approving.
4. **Act, then review the diff** properly — line by line, as you would a teammate's pull request. Note anything you would question.
5. **Verify.** Run the project's checks. Confirm the feature works and nothing unintended moved.

a learner who has run the full craft loop once — and can feel the difference between a scoped, planned, reviewed session and a sloppy one.

TEACHING NOTES

How to teach Day 6 well

OPEN WITH THIS

Show two task descriptions side by side — “make the app better” versus a tight, bounded one — and ask the room which one they would want to be handed. The gap they just named is the day. Craft is mostly the discipline of being specific under time pressure.

PACE & EMPHASIS

Sections 02 and 03 — scoping and plan-first / review — are the load-bearing ones; give them the most time. Section 05’s MCP and subagents are a light touch and a Day 8 teaser — do not over-invest. The memory file (04) is best taught by writing one live.

DISCUSSION PROMPTS

· What is a “make it better” task in your work that needs breaking into bounded pieces? · What three facts would go in your project’s memory file today? · Where are you most tempted to rubber-stamp, and what would catch you?

COMMON MISCONCEPTIONS TO PRE-EMPT

“A bigger task just means a longer session.”
No — it means a worse one. Big intentions get broken into bounded tasks, each reviewable alone.

“Once I trust it, I can skip the diff review.”
That is exactly when review lapses. The review is what keeps you the engineer, not a spectator.

“Project memory is just documentation.”
It is active context every session reads — standing orders for the project, not a dusty README.

IF YOU ONLY HAVE 30 MINUTES Teach scoping (02) and the plan-first / review pair (03) — these are the whole craft. Do Lab steps 2–4. Project memory and extensions can be a follow-up; the disciplines are what change outcomes.

Day 6 Cheat Sheet

Scope	One bounded job at a time. “Refactor everything” is a wish; a tight task is a target.
Define done	State task, context, and what a finished result looks like — so the session knows when to stop.
Plan first	For anything non-trivial, get the approach agreed before code is touched.
Review the diff	Read every change like a colleague’s pull request. Approving unread erodes everything.
Project memory file	Standing facts — structure, commands, conventions — kept in the repo and read every session.
MCP	Connects Claude Code to outside tools and data beyond the local project. (Full treatment: Day 8.)
Subagents	Delegating a focused sub-task to a separate worker, then bringing the result back.
Permissions	Set deliberately what Claude may do freely, what needs a check, what is off-limits.
Reach + restraint	Extend what the tool can touch only as far as you have set sensible limits.

Check for understanding

Five questions. Learners should be able to answer all five before Day 7.

1. What is the single biggest predictor of a good Claude Code session, and why?
2. Describe the plan-first habit and the one thing it most reliably prevents.
3. Why is rubber-stamping a diff dangerous even after several good sessions?
4. What is a project memory file, and name two things that belong in one.
5. What two mechanisms extend Claude Code's reach, and what discipline must travel with them?

> **Answer notes —** 1) A well-scoped task — a clear, bounded request gives Claude a target it can hit; vague requests produce vague work. 2) Asking Claude to lay out its approach and waiting for approval before it acts; it catches a wrong assumption before effort is spent on it. 3) Trust makes review lazy exactly when it still matters; the diff review is what keeps the human the engineer of record. 4) A file of standing project facts kept in the repo and read each session — structure, key commands, conventions, things to avoid (any two). 5) MCP (outside tools and data) and subagents (delegated sub-tasks); deliberately set permissions and boundaries must travel with them.

Day 6 in five lines

- A session is only as good as the task — scope tightly, define what “done” means, review in pieces.
- Plan-first and disciplined diff review do most of the quality work; both are easy to skip and shouldn’t be.
- A project memory file gives every session the project’s standing facts without re-explaining them.
- MCP and subagents extend Claude Code’s reach beyond the local files.
- More reach demands more restraint — set permissions and boundaries deliberately.

TOMORROW — DAY 7 → The Claude Developer Platform — building with the API