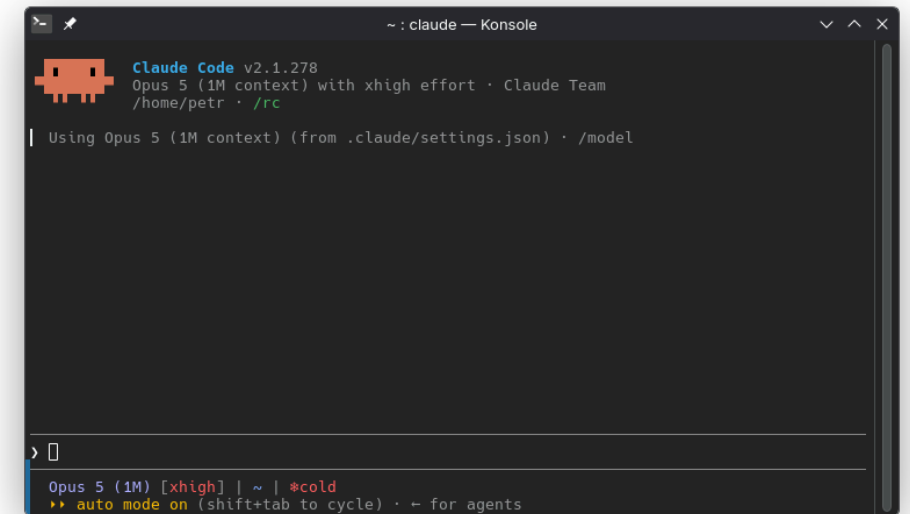


Petr's AI School

02 - Meet Claude Code



What surprised you?

Homework 01 - set up Claude Code

- What did you tell it to do?
- What was the output?



Today



Let's discover what Claude Code has to offer.

- starting claude code
- command line
- keyboard shortcuts
- models
- context
- modes
- referencing @files
- tools
- status line

Start in a folder

Windows: right-click `ai-school`, **Open in Terminal**, type `claude`

VS Code: File, Open Folder, then the Spark icon ✦

Mac: drag the folder onto Terminal, type `claude`

Desktop app: Code tab, Select folder

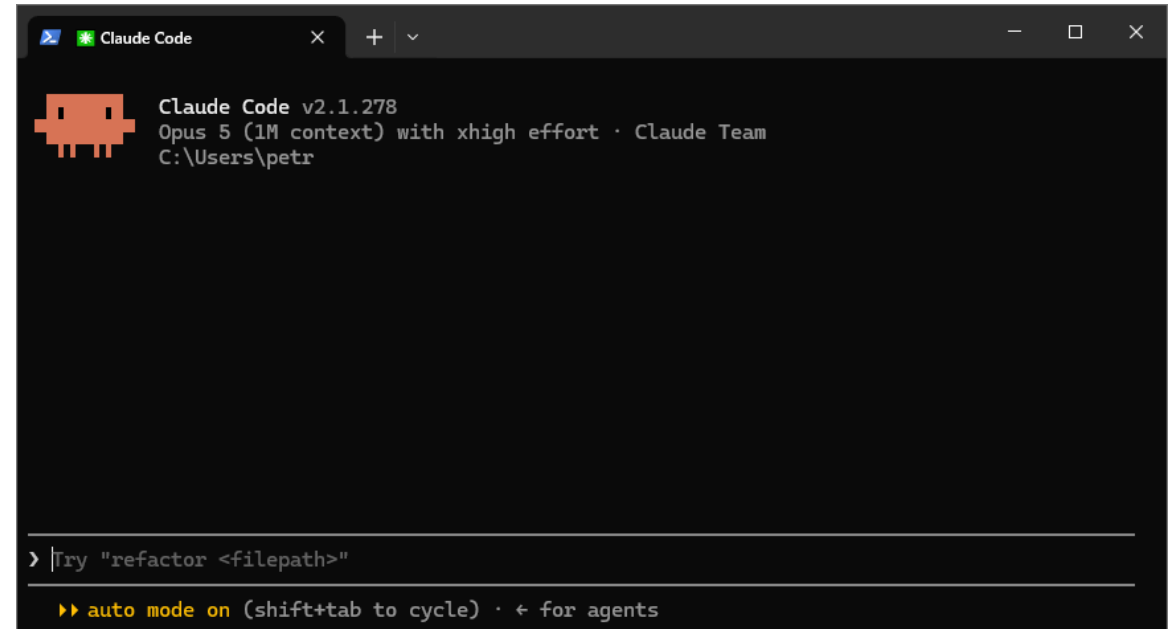
Same folder, different window.

☒ *Your turn*

Let's open your `Downloads` folder - there is going to be definitely something interesting.

The screen

- **Top:** which model, which folder.
Whatever is in that folder, it can read.
- **The `>` line:** where you type.
- **Bottom:** which mode you are in. The work shows above the prompt as it happens.



Watch the tools

☒ *Your turn*

Type: `what is in this folder?`

🔴 Bash(ls)

📁 budget.xlsx notes.md offer.pdf

🔴 Read(notes.md)

📄 Read 42 lines

- Every 🔴 line is a **tool**.
- The model asked for it. The harness ran it. The result went back to the model.
- `Ctrl` + `0` shows the whole transcript, every tool call.

The loop

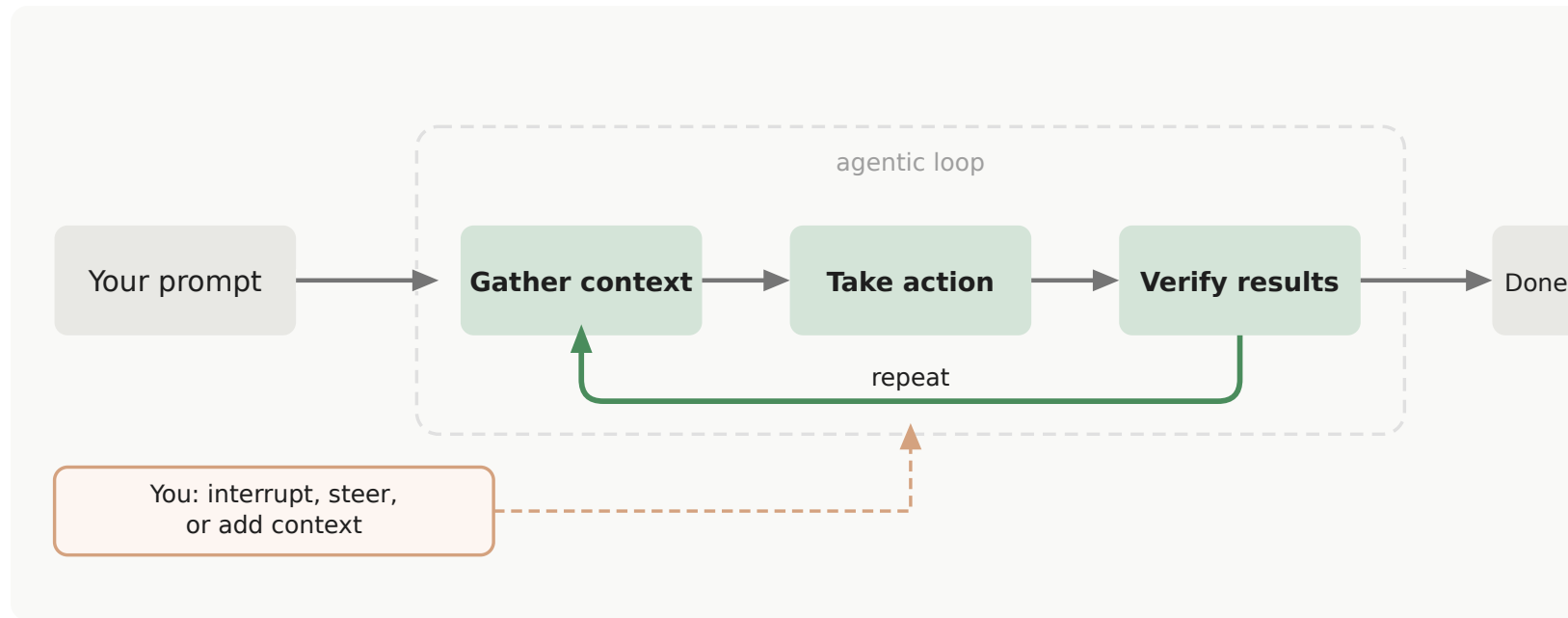


Diagram: Anthropic, Claude Code docs

Gather context, take action, check the result, repeat. You can press `Esc` at any point.

Model + harness + tools, seen live

Chat

Model only.

Cannot see your files.

Set of limited tools from Anthropic.

Work

Model + tools inside one folder you hand over.

Tools living partly in Anthropic Cloud.

Code

Model + every tool on your computer.

You set the mode.

☆ *Key point*

Code is ultimate power over the tools. 💪

Slash commands

Type `/` and a menu opens. Keep typing to filter it.

Type	Does
<code>/help</code>	lists everything
<code>/clear</code>	new conversation
<code>/resume</code>	pick an old one
<code>/model</code>	change the model
<code>/effort</code>	how hard it thinks

Type	Does
<code>/context</code>	how full the whiteboard is
<code>/usage</code>	how much of your plan is left
<code>/memory</code>	your instructions files
<code>/statusline</code>	set up the status line
<code>/exit</code>	leave

☒ *Your turn*

Type `/` and look. Then run `/help`.

Six keys

Key	Does
<code>Esc</code>	stop what it is doing
<code>Esc</code> <code>Esc</code>	clear what you typed; on an empty line: rewind file changes
<code>Shift</code> + <code>Tab</code>	switch permission mode
<code>↑</code>	your previous prompt
<code>Tab</code>	accept a suggestion
<code>?</code>	on an empty line: every shortcut

Nice to know: `Ctrl` + `C` clear or exit · `Ctrl` + `L` redraw · `Shift` + `Enter` new line · `!` runs a terminal command · `Ctrl` + `V` paste an image (Windows: `Alt` + `V`) · `Alt` + `P` model · `Ctrl` + `0` transcript

Everything else is on the cheat sheet in the wiki.

Pointing at files

- Type `@` and the first letters of the name, then `Tab`
- Drag a file from Explorer into the window
- Paste a screenshot: `Ctrl` + `V` (Windows: `Alt` + `V`)
- Folders work too: `@` and a folder name points at the whole folder

☒ *Your turn*

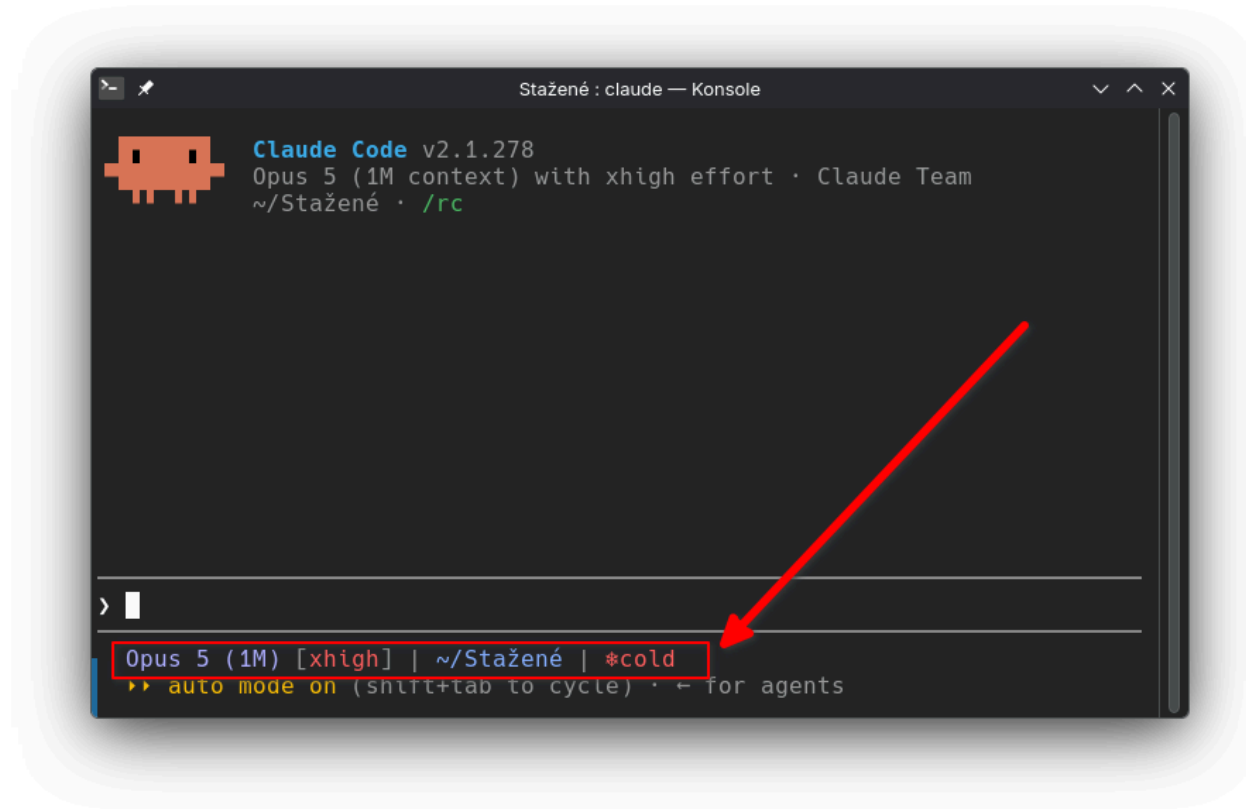
Type: `summarise @notes.md in three bullets`

Live demo: your status line

☒ *Your turn*

Type: `/statusline` show the model, the folder name and how full the context is in %. Like "Opus | C:/Users/Petr | 85%"

It set itself up by writing two files. You saw each step. Nothing appears? The wiki guide **Set up a status line** has the manual way.



Models and effort

Model: which brain

`/model` or `Alt` + `P`

- **Fable** the expert, slow
- **Opus** hard work
- **Sonnet** everyday work
- **Haiku** quick and cheap

Effort: how long it thinks

`/effort low` up to `/effort max`

- the default: medium on Opus 5.5, high on the others
- low for chores
- max when it is hard and you can wait

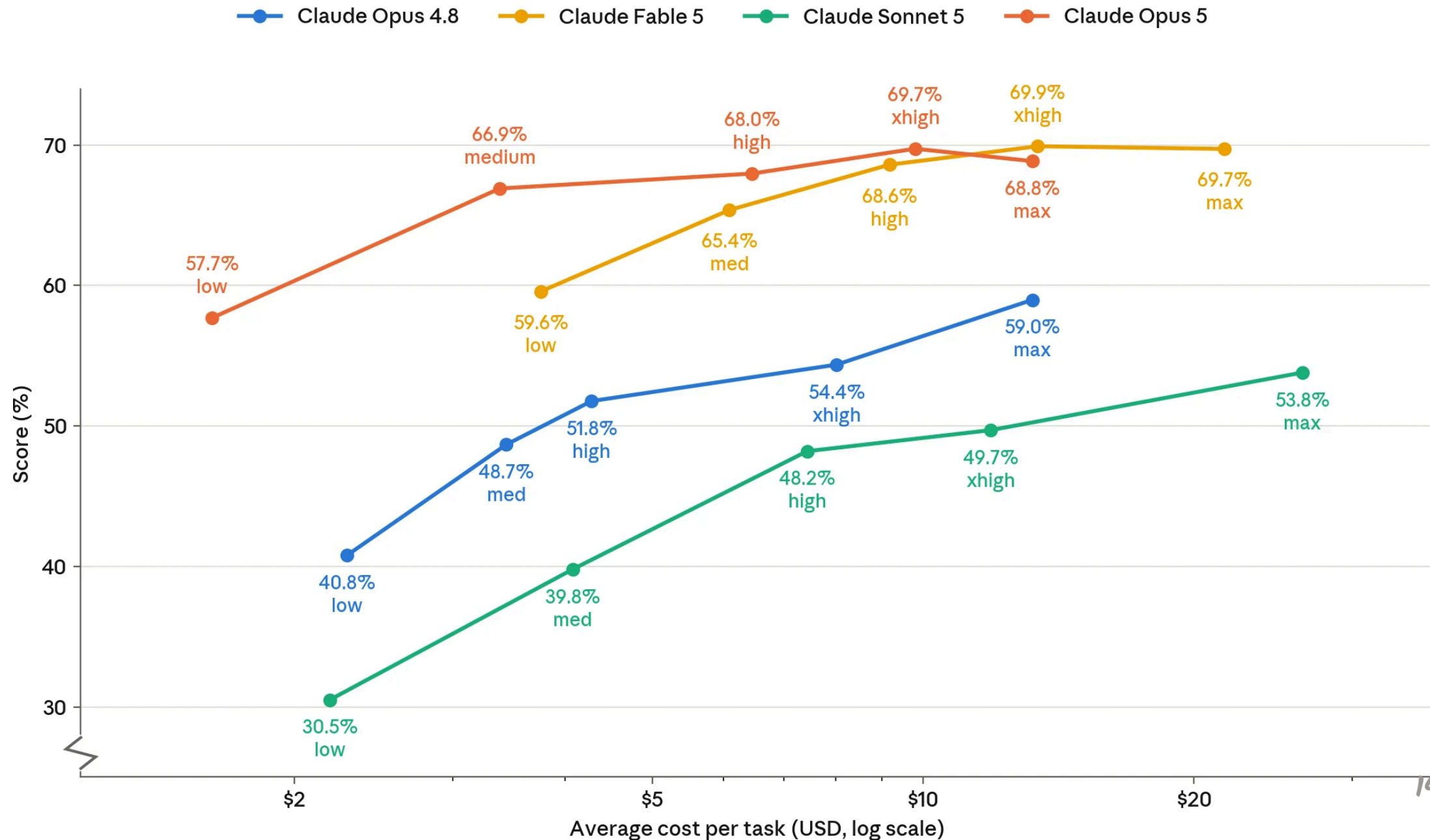
Risky task? A stronger model, not a cheaper one.

☒ *Your turn*

Type `/model` , look, press `Esc`.

DeepSWE v1.1: test-time compute scaling

PETR-AI-SCHOOL



More effort buys a higher score, at a much higher cost

Claude models at each effort level: score on an independent test against the cost of running it



Source: Artificial Analysis Intelligence Index v4.3, artificialanalysis.ai, read 23 September 2026. Points left to right: low, medium, high, xhigh, max. Haiku 4.5 has no effort setting.

Context: the whiteboard

- `/context` shows how full it is. Your status line shows the % all the time.
- `/clear` hangs up a new whiteboard. The old one is saved.
- `/resume` brings an old one back.
- `/compact` summarises the whiteboard to make room. You should never need it.
- From the terminal: `claude -c` continues the last one.

☒ *Your turn*

`/context` , then `/clear` , then `/resume` and pick the one you just left.

Permission modes: `Shift` + `Tab`

Mode	Label at the bottom	Runs without asking
Manual	 <code>manual mode on</code>	reading only
Accept edits	 <code>accept edits on</code>	reading and file edits
Plan	 <code>plan mode on</code>	reading, then it writes a plan
Auto	 <code>auto mode on</code>	everything a second model calls safe

You start in **Auto** on paid plans. The habit: **Plan**, then **Auto**.

☒ *Your turn*

`Shift` + `Tab` to Manual. Type `rename notes.md to notes-old.md`. Read the question. Press `Esc` to say no. Then `Shift` + `Tab` until Plan.

Your turn

☒ *Your turn*

In Plan mode, type:

Make a file called `index.md` that lists every file in this folder with one line about each.
Do not change the other files.

1. Read the plan. Argue with it if you want.
2. Choose **Yes, and use auto mode**.
3. Watch the  lines.
4. Open `index.md` in Explorer. It is there.
5. Press `Esc` `Esc` and pick the rewind. It is gone.

Nothing to fear

☆ *Key point*

- `Esc` stops it - **don't be afraid to use it**
- `Esc` `Esc` undoes its file edits (not moves or deletes)
- It works only in the folder you opened
- `/clear` resets the whiteboard
- A usage limit pauses you. Nothing breaks
- Keep a copy of anything precious

Chat, Work, Code: which one?

- **Chat** when there are no files. A question, a draft, a translation. Few tools.
- **Work** when the task lives in one folder and you want a result quickly and want to go away from it.
- **Code** when you want the full toolbox: many folders, programs, the web. And you watch each step. Reuse, build.

☆ *Key point*

Same brain in all three. More tools, more it can do.

Homework

1. Make the status line yours. Tell `/statusline` what you want, in your own words.
2. One real task. Copy a messy folder of yours. Plan, read, then Auto: sort the files into subfolders, unsure ones into `_review`.
3. Find one slash command we did not cover. Bring what it does.
4. Bring an answer: which tool call surprised you?



Tip

Work on a copy. The original stays where it is. Esc Esc cannot undo moving files into folders.

Terms from today

Slash command · Session · Effort · File mention · Permission mode · Context window ·
Compaction · Tool · Agent loop · Harness · Model · Claude model family · Benchmark ·
Chat, Work and Code · Terminal · Project

Guides: Reading the Claude Code screen · Working safely

All of them are linked from the lecture page in the wiki.

Questions?