

Homework — bring one task you already do (Friday 16 October)

For this homework, choose one small, ordinary task that you already do and describe how you do it today. The goal is not to invent an impressive AI project. The goal is to understand a real workflow well enough that, on Friday, we can decide together:

- which parts should stay with you,
- which parts could be automated,
- whether any part actually needs an agent, and
- how you would know if the result were wrong.

“This does not need an agent” is a perfectly good conclusion.

Pick a small task you already know well

Good examples are things like:

- tidying up documents someone sends you,
- finding a few useful papers or articles on a topic,
- summarizing the numbers from a spreadsheet each month,
- organizing files,
- checking a recurring report, or
- running specific set of tools for some tasks.

Pick something you already do and, ideally, something where you already use ChatGPT or Claude for at least part of the work. Small, repetitive tasks are often best because you know what can go wrong.

Small and slightly boring is better than ambitious. If you have done the task many times, you already know where it tends to go wrong and what a correct result looks like.

Answer five questions

	Question	What we want to learn
1	What do you do?	Describe the task, how often you do it, and how long it takes.
2	How do you do it now?	Write the real steps you currently follow.
3	Draw it	Show the steps and any loops or places where you have to go back. Boxes and arrows, one per step.
4	What goes in, what comes out, and who does each step?	Decide what you would keep for yourself and what an agent might do.
5	What could go wrong, and would you notice?	Identify a quiet failure and a quick way to detect it.

Keep it small

Your task is probably the right size if:

- it starts from one identifiable file, folder, input, or request;
- you could tell within about five minutes whether the result is right;
- the workflow has fewer than about ten steps.

If the whole task is really just one good prompt to ChatGPT, write that down. That is useful information too.

A task becomes more interesting for agent design when it involves **several steps, files moving between steps, repeated work, or something that needs to be checked.**

Which one — chat, Cowork, or Claude Code?

Prof. Murillo's rule from last Friday: pick by what the agent has to touch.

	what it can work with	Good fit
Chat (browser/phone)	Content you paste or upload; a temporary sandbox	One-off questions, drafting, planning
Cowork (desktop)	Folders you grant access to, plus browser tasks	Multi-step file work without a terminal
Claude Code (Code tab / terminal)	Project files, scripts, tests, and a shell	Running code or making repeatable workflows with explicit project instructions

What to submit

Complete Questions 1–5 in this document. You can type directly into it or print it and write by hand. A photo of a hand-drawn workflow for Question 3 is completely fine.

About 30 minutes is enough. Half a page to one page of answers is usually plenty.

Bring your completed homework and **one real input file** from the task on Friday. Remove sensitive details if necessary.

On Friday, we will give your workflow to Claude Code or Cowork and use it to decide what—if anything—is worth building.

Your answers

Name: _____ Date: _____

Type into this file or print it and write on it. Either way, bring it on Friday — Claude Code can read a .docx directly.

1 · What do you do?

In 2–4 sentences, describe the task and what “finished” looks like. Include frequency, time spent, and the tools you use today.

How often (every week? every month?)	How long it takes now (30min? hours?)	What you use today (ChatGPT, Claude, Word, Excel, or your eyes?)

2 · How do you do it now?

Number the steps. Four to eight is plenty. Include the dull parts — the copying and pasting, the files you keep open, and what you check before finishing.

1.
2.
3.
4.
5.
6.
7.
8.

3 · Draw it

Boxes and arrows, one per step. A photo of a sketch on paper is a perfect answer. If a step sometimes sends you back to an earlier one, draw that arrow too, that loop is usually where the time goes.

draw here, or attach a photo

4 · What goes in, what comes out, who does it — and where

For each step in Question 2 (or a group of similar steps), identify what goes in, what comes out, who should do it, and where it might happen. “Not sure” is a valid answer.

Start with the human decisions: which steps would you keep for yourself, and why? An agent checking its own work is not always an independent check.

#	goes in	comes out	who — me / the agent / the agent drafts, I check	where — chat / Cowork / Claude Code
1				
2				
3				
4				
5				
6				
7				

5 · What could go wrong — and would you notice?

Give one or two quiet failures: an output that looks plausible but is wrong. For each, describe a specific check you could perform in about a minute—not simply “I would read it carefully.”

if it got this wrong...	...I’d spot it by

Is there anything it should never touch? Your original files, an email send button, anything it is supposed to be checking. One line.

Worked example · Formatting a colleague’s document

This is deliberately ordinary and short. Your own answers can be this simple.

1 • What I do

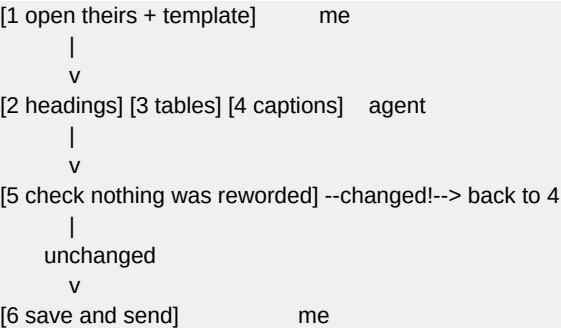
Colleagues send sections of a shared report in different formats. I apply our template styles, adjust tables, and place figure captions correctly without changing the underlying wording.

About twice a month • 30–40 minutes each • today: Word, by hand, plus Claude when the wording also needs a tidy.

2 • How I do it now

- 1. Open their file next to our template.
- 2. Apply our heading styles to their headings.
- 3. Fix the tables — widths, header row, borders.
- 4. Move figure captions below the figures and renumber them.
- 5. Read it through to check that no wording changed.
- 6. Save as <section>_formatted.docx and send it back.

3 • Drawn



4 • In, out, who, where

Steps	Input	Output	Who	Where
1	Colleague file + template	files ready	Me	Desktop
2–4	Original + template	formatted copy	Agent	Cowork/Claude code
5	Original + formatted copy	change list	Agent drafts; I check	Cowork/Claude code
6	Approved file	email sent	Me	Email

5 • What could go wrong

It “tidies” a sentence while restyling, and I send back something they did not write. I would not notice; it would read fine.

I’d spot it by having it list every wording change it made: the list should be empty, and an empty list takes two seconds to check.

Never touch: their original file. Always write a new one.

Where: Cowork. The files sit in a folder on my machine and there is no code to run.

Chat would mean uploading and downloading every file by hand.

Claude Code: it happens every week or month, and I want it done identically every month, so the rules live in a CLAUDE.md the agent re-reads each time, or write skills for the tasks.

Why this example matters

The agent handles the repetitive middle; a person starts and approves the result. The check is an observable artifact (a change list), not a vague assurance that everything looks fine. A short script may be preferable to an agent when the steps are fully deterministic.

On Friday — build something you can run again

You will not need a special folder, a setup file, or anything installed beyond Claude itself. Everything below is a prompt you paste, with your own answers pasted underneath. Work wherever you are — Cowork, the Code tab, or chat.

By the end you will have a skill: a short page of written instructions that Claude loads by itself whenever this task comes round again. Next time, the whole workflow is one sentence.

1. Hand over your answers.

I want to design a small agent workflow for a task I already do.
Here are my answers to five questions about it.

1. WHAT I DO:
 <paste your answer to question 1>
2. HOW I DO IT NOW:
 <paste your numbered steps from question 2>
3. IN / OUT / WHO / WHERE, PER STEP:
 <paste your table from question 4>
4. WHAT COULD GO WRONG, AND HOW I'D SPOT IT:
 <paste your answer to question 5>

Read all of it. Before you propose anything, ask me the three questions whose answers would most change your design. Then wait.

The waiting is the point. Nearly every brief is missing the same thing — a way it could go wrong that nobody would catch — and you want that found now, not after you have built something.

2. Ask whether it is worth building at all.

Now tell me honestly: is this worth building? Give me three options, smallest first —

- (a) one good prompt I paste each time,
- (b) a short script I run,
- (c) a skill you load whenever this task comes up.

For each: what would I have to do every time, and what could still go wrong? Recommend one, and say why not the smaller one.

(a) and (b) are real answers. If all three sound equally good, ask which one it would drop first — that question gets a straighter answer than asking which is best.

3. Do one step, once, on a real file.

Before we write anything reusable, do step <N> once, right now, on this real file: <drag the file in, or give me its path>.
Show me what comes out. If a few lines of code would do this more reliably than you reading the file yourself, write the code.

A real file, not a made-up one. Made-up files are tidy and yours is not, and the difference is where the whole design lives.

4. Add your check — then break it on purpose.

Add the check I described in question 5 — the one that catches <the quiet failure you named>. It must report what it finds and must not fix anything.

Then take a copy of my file, put that exact problem into it, and run the check again. I want to watch the check fail.

A check you have never seen fail is not a check. Yours will probably pass the first time for the wrong reason — it read the wrong column, the file was empty, the text wrapped onto two lines. Ten minutes here saves you from trusting it for a year.

5. Turn it into a skill.

Now write this up so I can re-run it. Produce one file called SKILL.md, starting exactly like this:

```
---
name: <short-name-with-hyphens>
description: <one sentence: what it does and when to use
  it – this is what makes you pick it up later>
---
```

Then the steps, in order, written as instructions to you rather than to me. Include:

- what file or folder the task starts from,
- the checking step, and what it should print,
- what you must never do: <your 'never touch' line from Q5>,
- what to hand me at the end, and in what format.

Keep it under a page. Show me the file.

The description does more work than the steps. It is the only part Claude reads when deciding whether this skill applies, so it should name the things you will actually say: the file type, the folder, the words you use for the task.

6. Save it where Claude will find it.

Put that SKILL.md inside a folder named <short-name-with-hyphens>, zip the folder so the folder itself is the top level of the zip, and give me the zip.

Then, in Claude: Settings → Capabilities, turn on Code execution and file creation if it is off. Then Customize → Skills → + → Create skill → Upload a skill, and upload the zip. A skill saved there is available in chat, in Cowork, and — signed in to the same account — in Claude Code. If you live in one project folder, you can instead save it as .claude/skills/<short-name>/SKILL.md and run /skills to check it is listed.

7. Test it cold.

<Start a NEW conversation, and say the task the way you would say it to a colleague:>

Can you tidy up this section for me? Here's the file.

If Claude does not reach for your skill by itself, the description is what needs fixing, not the steps. That is the real test, and it is worth doing before you leave the room.

“This does not need an agent” is a passing answer. If prompt 2 lands on “one good prompt” or “a short script you run once a month”, that is a result and a good one. Write it down with the reasoning, and skip to prompt 5 — a skill that says “paste this prompt, then check that” is still worth saving.

If you are in a group project. Last Friday’s assignment — one problem solved four or five ways, fifteen minutes of slides — needs a problem and a seed.md. The task you describe here is a good candidate for both. Question 1 is the goal and who it is for, question 2 is the working guesses about how it gets done, and question 5 is how a result gets checked and what “done” means. Bring both and we will line them up.

What to bring

- Your answers — typed into this document, or on paper. A photo of your question 3 sketch is fine.
- One real file your task starts from, messy as it is. Change names if it is sensitive; do not invent a clean one.
- Claude working on your laptop: the desktop app for Cowork, or Claude Code. Nothing else to install, and nothing to download beforehand.

What we are looking for

Whether it is honest and specific — not whether the idea is impressive.

- Question 2 describes what you actually do, awkward parts included.
- Question 4 names real files and real formats, and says “not sure” where you are not sure.
- Question 5 names something that could go wrong quietly, and a check you could actually run in a minute.
- An answer concluding “this does not need an agent”, with the reasoning, is complete.

Half a page is a good length. If it is running to three, you have picked something too big — take one piece of it instead.