The AI Bootcamp

29

5 Questions to Find Work Worth Automating

In this chapter you'll learn

A five-question audit of your own week, the sum that kills most candidates, and one build worked end to end

The AI Bootcamp

THE PROBLEM

Most AI education is a pile of disconnected tips. A LinkedIn carousel here, a YouTube tutorial there, a $2,000 cohort course that goes out of date before the refund window closes. None of it is built to be read in order, so nothing compounds. You finish twelve videos and can recite twelve facts, not perform one new skill.

THE FIX

The AI Bootcamp is built the other way. 90 skills, packed 2 to 3 per chapter, 36 chapters, 3 tracks. Every chapter states what it needs from you and what it hands to the chapter after it, so skill 47 only works because skill 12 already landed. Read it in order and by the end you are not collecting facts about AI, you are running work through it.

LETTER FROM THE EDITOR

We built this because we kept meeting smart people who felt behind on AI for no reason other than nobody had organized the material for them.

This book does not assume you are technical, and it does not water anything down either. Every chapter earns its place: a real skill, a real exercise, a one-page companion you can flip back to in month three when you have forgotten the details but still need the answer.

If a chapter loses you, or a chapter changes how you work, I want to hear about it either way.

Kris

Editor, AI Central

Write to me at **editor@thecentral.ai**

THE 3 TRACKS

AI 101

Get running

Pick a tool, set it up, write a clear instruction, use it safely at work.

AI 202

Prompt properly

The framework behind every good instruction, and catching AI when it is wrong.

AI 303

Build without watching it

Connect AI to your tools, and automations that run while you are not looking.

IN THIS LESSON YOU'LL LEARN

By the end of this chapter you can look at the recurring work in your own week and say which piece is worth automating and which piece only feels like it is, price a candidate honestly before you build it, and take one survivor all the way through a build and a test against a week you already did by hand.

Chapter 28 gave you the shape of an automation on something small and safe. Now you want to point it at real work, and this is exactly where people waste a fortnight: they aim at whatever annoys them most, build it, use it for three weeks, and quietly stop. The task that irritates you and the task worth automating are different tasks, and telling them apart is a skill you can learn in an hour.

CONTENTS

29.1	The task you find most annoying is usually not the one to automate
29.2	Two of the five questions are gates, not scores
29.3	Do the sum before you build, and expect it to say no
29.4	One candidate, audited, built, then tested against a week you already did

TOPICS REQUIRED

Chapter 28. You need one small automation already running, so you know what a trigger, an action and a result feel like in practice. Chapters 26 and 27 matter too: this chapter assumes your assistant can already reach at least one place your work actually lives, and that you have already agreed what it is allowed to do there.

AFTER THIS CHAPTER

Chapter 30 assumes you can audit a piece of recurring work and price it before building. It then applies exactly that method to the two richest sources of recurring work you have, which is why this chapter keeps its examples away from them.

29.1 The task you find most annoying is usually not the one to automate

Irritation measures what a task costs you to sit through. It says nothing about whether a machine can do it. From the inside those two feelings are identical, which is why most first real builds are aimed at the wrong target.

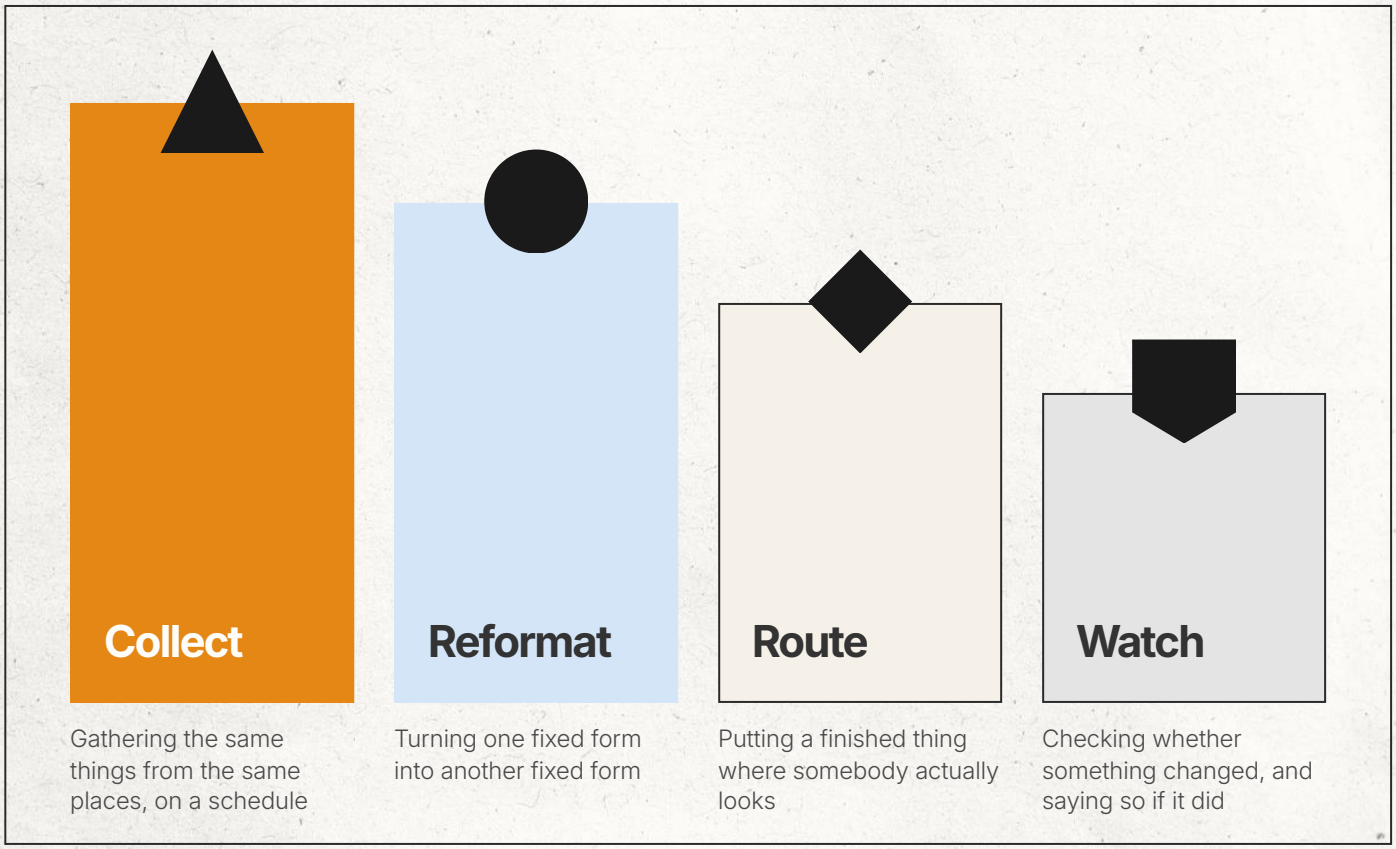


Fig 29.1 Four shapes recurring work takes. The shape predicts whether an automation will survive contact with your actual week far better than the subject matter does. Bar heights here are visual rhythm, not a ranking.

29.1

Log it, do not recall it. One week of notes beats memory. The jobs you forget are the small ones, and the small ones are the frequent ones.

Frequency hides. A 6-minute job you do every working day costs you more per year than a half-day job you do quarterly.

Not mail, not diary. Those two are Chapter 30, on purpose. Leave them off this list for one more chapter.

You already know which task you would automate. It is the one you complain about. It is probably the monthly thing with the awkward spreadsheet, or the report nobody reads, and you have been waiting 28 chapters to get rid of it.

Be careful with that instinct. What you are feeling is the cost of doing the task, and that cost is mostly boredom, interruption and the dread that builds up on Sunday evening. None of those tell you whether the steps are stable enough for something to follow them without you. Some of the most hated work in a professional week is hated precisely because it is fiddly and different every time, which is the same property that makes it a poor candidate.

So build the list from evidence rather than memory. For one week, keep a note of every piece of work you did more than once, or that you know you will do again on the same day next week. Write the day, and write how many minutes it actually took, not how many it feels like it took. You want 6 or so candidates by Friday. Anything you cannot remember doing by Friday afternoon was not costing you much.

As the list fills, sort it by shape rather than by subject. Almost all recurring office work is one of 4 things: collecting the same material from the same places, reformatting something from one fixed form into another, routing a finished thing to where it gets read, or watching something and speaking up when it changes. Two tasks that share a shape behave the same way when you automate them, even when one is a sales report and the other is a compliance check.

Two further marks are worth putting beside each candidate. First, does the task end in something you could show a colleague, a file, a message, a row in a sheet? Work with a visible finished object is far easier to build for, and far easier to check afterwards, than work that ends in you simply knowing something. Second, where does the input come from? If a machine produced it, you are in good shape. If a person typed it, you are not, and 29.2 explains why that one matters more than anything else on the list.

Go and start the log now, before you read the next section. The audit is worthless run against a list you invented from memory.

29.2 Two of the five questions are gates, not scores

Frequency, minutes per run, how stable the steps are, what a mistake costs, and whether the inputs arrive in a consistent form. Add all 5 into one number and a big enough saving can outvote a failure you cannot afford, so 2 of them do not get a vote. They get a veto.

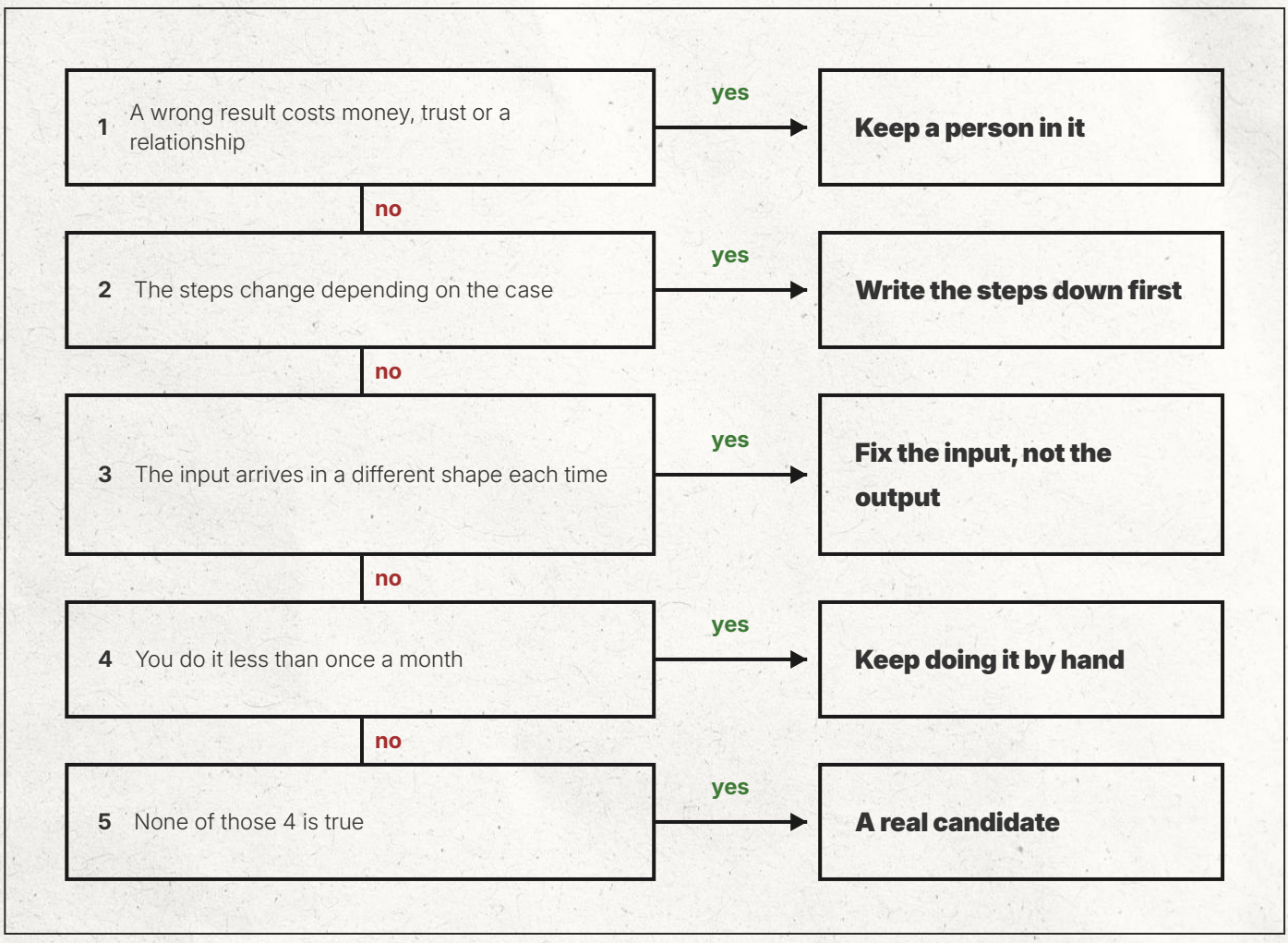


Fig 29.2 Read down and stop at the first line that is true. Most of the work you would most like rid of stops at line 2 or line 3, and a candidate list that shrinks here is the audit working rather than failing.

29.2

The 2 gates. What a mistake costs, and whether the input is consistent. Neither can be outvoted by a large saving.

The 3 magnitudes. How often, how many minutes, and how stable the steps are. These are the numbers you multiply in 29.3.

The writing test. If you cannot write the steps as a numbered list a competent colleague could follow, there is nothing to automate yet.

Five questions decide this, and how irritating the work is does not appear among them. Two of the five are gates: fail one and the candidate is out no matter how much time it would save. The other three are magnitudes, and those are the ones you multiply in the next section.

The first gate is what a mistake costs. Ask what actually happens on the morning the automation gets it wrong and nobody notices. If the answer is that a number goes out to a client, a payment is coded to the wrong account, or somebody senior reads something untrue, that is not a task you take your hands off this month. It can still be automated, with a person approving each run, and putting a person in the right place is Chapter 34's whole subject. What you must not do is treat a big time saving as a reason to skip that step.

The second gate is the shape of the input, and it is the single most reliable predictor on the page. An automation is only ever as steady as the material arriving at the top of it. A system-generated export with the same columns in the same order every week is close to ideal. A PDF that a supplier types by hand, in a slightly different layout each month, is not, and no amount of cleverness downstream fixes that reliably. When you hit this one, the correct move is to go upstream: ask for the export instead of the summary, or replace the emailed request with a short form. The cheapest automation available to most professionals is a form that makes the input consistent, and it needs no assistant at all.

Then the magnitudes. Stability of the steps is measured by writing them down: open a blank page and describe the task as a numbered list, the way you would for somebody covering while you are on leave. If you get to step 4 and find yourself writing it depends, you have found the boundary of what can be built, and the useful move is to automate the steps above the word depends and stop there. That is not a compromise. It is usually the right answer, and 29.4 shows it happening.

Frequency and minutes per run are the easy two, with one trap in each. Count runs per year rather than per week, because the year is the unit the arithmetic works in. And time one real run with a timer instead of estimating, because the estimate is always the version where nothing went wrong and nobody interrupted you.

Now the honest part. Some genuinely awful work fails this audit and stays yours. The quarterly thing with the varying inputs and the expensive mistakes is exactly the sort of task that fails on 3 of the 5 questions at once, and it will keep failing next year. Run the 5 lines against the 6 candidates from your log and expect 1 or 2 to survive. If none survive, you have not wasted the week: you have avoided a build you would have abandoned in March.

29.3 Do the sum before you build, and expect it to say no

Minutes the automation actually removes, times runs a year, against time to build it plus a year of keeping it alive. Written out plainly that sum kills most candidates, and killing them on paper is far cheaper than killing them in March.

The gross sum

How long the whole task takes, times how often you do it. The number people quote when they pitch an automation.

TRAITS

· Counts the whole task, including the parts you still do by hand

· Counts a clean run, not an average one

· Stops counting on the day you finish building

PROS

+ Quick, and roughly right about which tasks are large

+ Fine for putting a shortlist in order

CONS

– Leaves out upkeep entirely, which is where builds actually die

– Flatters whatever you happen to find annoying

The net sum

Minutes the automation genuinely removes, times runs a year, minus what it costs to build and a year of keeping it working.

TRAITS

· Counts only the steps you stopped doing yourself

· Counts runs a year, so 4 quarterly runs look like 4

· Carries an upkeep line even before you can see the work

PROS

+ Kills the expensive occasional jobs before you build them

+ Promotes small frequent jobs, which are also the stable ones

CONS

– Needs an upkeep number you are partly guessing at

– Reads as pessimistic in month one, and is not

Fig 29.3 The same candidate, priced twice. Almost the whole gap between the 2 cards is the upkeep line, and the upkeep line is the one nobody writes down before building.

29.3

The sum. Minutes removed per run, times runs a year. Subtract build time, then subtract a year of upkeep.

Runs a year. Weekly is about 48. Every working day is about 250. Quarterly is 4, and 4 almost never pays.

It breaks quietly. Zapier switches a Zap off once it errors 95% of the time and has run more than 20 times in 7 days. Power Automate turns off a flow that fails continuously for 14 days.

Here is the sum, and it fits on one line. Minutes removed per run, times runs a year, minus the hours it takes to build, minus a year of upkeep. Write both sides down before you open any tool.

Try it on the candidate everyone reaches for first: the quarterly pack. It takes 3 hours, you do it 4 times a year, and you would happily never do it again. Gross saving, 12 hours. Now be honest about what the automation actually removes, which is maybe half of it, so 6 hours. It is a complicated build against inputs that vary, so call it 6 hours to make. And because 3 months pass between runs, something will have moved every single time, so add 3 hours of repair across the year. First year: 3 hours in the red. Year two is barely better, because after a 3-month gap you have to relearn your own build before you can fix it.

Now the candidate nobody gets excited about: 6 minutes, every working day, about 250 times a year. Gross 25 hours. The automation removes 5 of the 6 minutes, so 21 hours. It is simple, so 2 hours to build and 2 hours of upkeep. First year: 17 hours saved, and it improves after that. The unglamorous daily job beats the hated quarterly one by roughly 20 hours, and it will still be running at Christmas.

Runs a year is the term that dominates, and there is a second reason to like it beyond the multiplication. Frequent work is stable work. A job you do 250 times a year has been standardised by repetition whether you meant to standardise it or not, so its steps hold still. A job you do 4 times a year drifts between runs, because the business changed underneath it and nobody told the automation.

The upkeep line is the one readers most want to leave blank, so here is the evidence that it is real. Zapier automatically turns a Zap off when it errors 95% of the time it runs and has run more than 20 times in the last 7 days. Microsoft's Power Automate turns off a cloud flow whose trigger or actions fail continuously for 14 days, and turns off any flow that has not been triggered at all for 90 days. Read those as what they are: two of the largest automation platforms found breakage and abandonment common enough that they wrote standing policies for it. Nobody writes a policy for a rare event. Note also which term of your sum has moved lately and which has not. Building has genuinely got cheaper, because a low-code studio will now assemble a first draft of a workflow from a plain-language description of what you want, sometimes turning days into an afternoon. Upkeep has not moved at all. A thing that was cheap to build is not thereby cheap to own, and being able to make 5 of these in a morning is precisely why the sum matters more than it used to.

Last, distrust your own estimate of the saving. The most careful measurement we have of people judging their own AI-assisted productivity comes from METR, which in 2025 randomised 246 real tasks among 16 experienced open-source developers. With AI tools allowed, they took 19% longer. They had predicted a 24% speed-up beforehand, and afterwards still believed they had been sped up by 20%. That study is about assisted coding rather than scheduled automation, so do not stretch it further than it goes. Take from it only the part that transfers: people are confidently wrong about their own time, in the flattering direction. Which is why you time one real run rather than estimating it.

29.4 One candidate, audited, built, then tested against a week you already did

Everything above, run once on a real piece of work. Watch which parts get built and, more usefully, watch the part that deliberately does not.

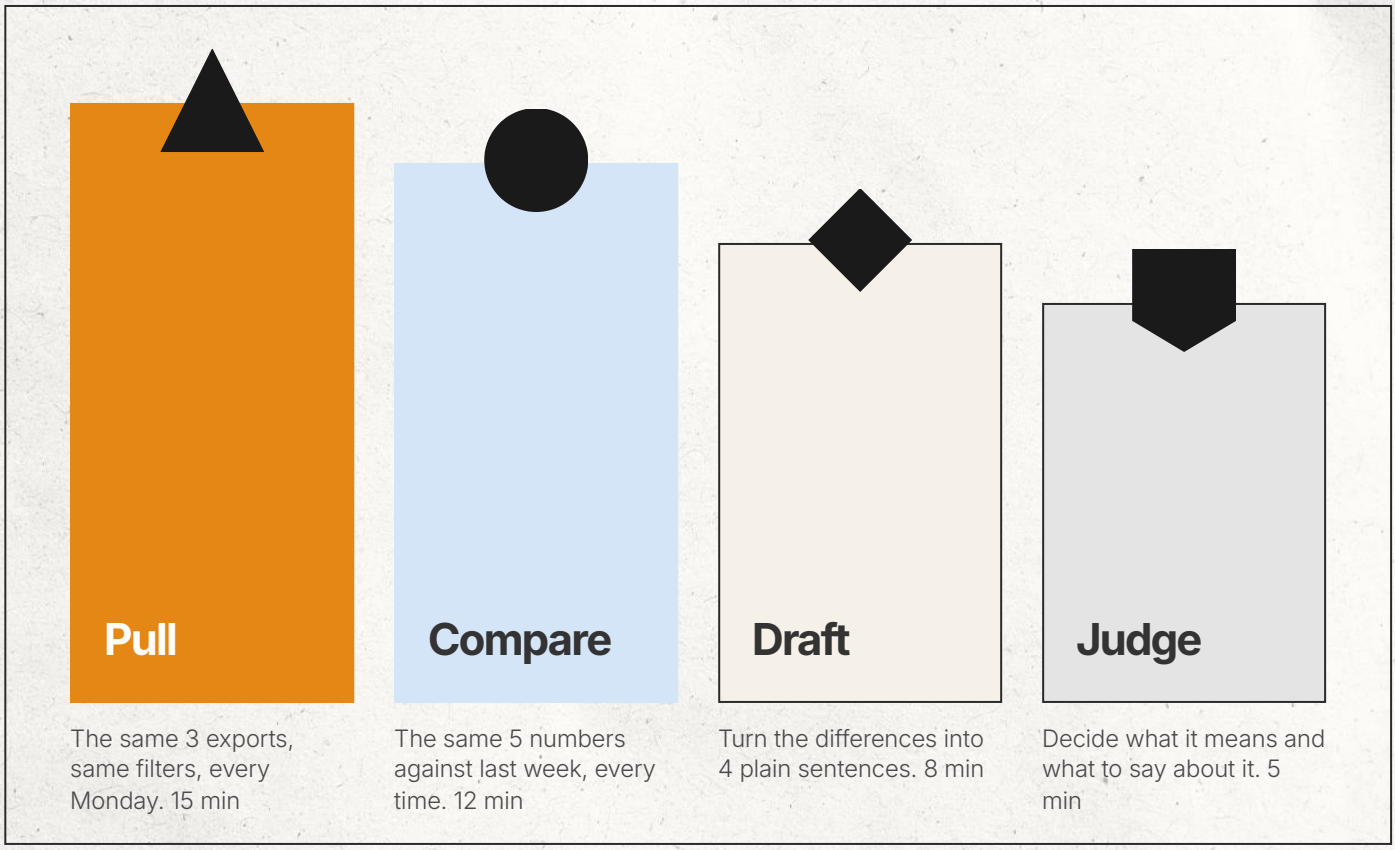


Fig 29.4 One 40-minute Monday job, timed step by step. These bar heights are the real minutes each step takes. The first 2 steps became the build. The last step is the reason the job was worth doing at all, and it stays yours.

29.4

What got built. Pull and compare. 27 of the 40 minutes, and the 2 steps that never vary.

What did not. The judgement. Automating it would have saved 5 minutes and removed the point of the task.

The test. Run it against last Monday, a week you already did by hand, and compare line by line before trusting a single output.

Take a real one. Every Monday morning you pull 3 standard exports, check the same 5 numbers against the previous week, write 4 sentences of commentary, and send the result into the leadership pack. Timed properly it is 40 minutes, and you have done it most Mondays for 2 years.

Start with the gates. What does a mistake cost? A wrong number in a leadership pack is genuinely expensive, so this candidate very nearly stops at line 1. It survives for a specific reason: the error would be visible to you in the output, on the same morning, before anything leaves your hands. That is the condition to look for. Not that mistakes are impossible, but that they are visible and cheap to catch at the point where you already look. Then the input. All 3 exports are machine-generated, same columns, same order, every week. That is the strongest signal in the whole audit, and it is why this candidate is worth the next hour.

Now write the steps down as a numbered list, and watch where the list stops being writable. Steps 1 and 2, the pull and the comparison, are completely fixed: same files, same filters, same 5 comparisons, no exceptions in 2 years. Step 3, turning differences into sentences, is half fixed, because the sentence pattern repeats even though the content does not. Step 4 is where you decide what the movement means and whether to flag it, and no numbered list survives contact with that. So the candidate is not the task. It is the first 2 steps of the task, and possibly a first draft of the third. Naming that boundary correctly is most of the skill in this chapter.

Then the arithmetic, honestly. The automation removes the pull and the compare, 27 minutes, but adds about 3 minutes of reading and checking what it produced, so call it 24 minutes removed per run. 48 Mondays a year is 1,152 minutes, a little over 19 hours. The build is about 2 hours, most of which is describing the job in plain language and then fixing the 3 things it got wrong. Upkeep: half an hour a quarter, plus an hour on the day one of the exports changes shape, so 3 hours across the year. First year, about 14 hours net, and the second year is better because the build cost is already paid. That is a clear yes, and notice that it is a clear yes at 24 minutes a week rather than at the 40 you would have claimed.

The build itself is one paragraph, because Chapter 28 already taught the shape. The trigger is a time, early Monday. The action fetches the 3 exports and works out the 5 comparisons. The result is a short table of this week against last, plus a first-draft sentence for each number that moved more than a set amount, delivered back to you and to nobody else. Sending it onward to the team is Chapter 31's job, and pulling in a fourth and fifth system is Chapter 33's. Resist both this week. One automation, one output, one reader, who is you.

Test it before you rely on it, and test it against a known answer. Run it over last Monday, a week you already did by hand and still have the output for, then compare line by line. If the 5 numbers match, you have evidence rather than a feeling. Then run it in parallel for 3 more Mondays while still doing the job manually. Yes, that costs you 3 Mondays of doing the work twice, and it is the cheapest insurance in this track. Write the build date on it and note which exports it depends on, because the day one of them changes is the day it starts failing, and Chapter 35 is about how you find out.

READING ENDS HERE

Everything before was reading. Everything after is doing.

3 exercises, About 20 minutes and one real decision that opens Chapter 30.

Do these before Chapter 30

3 exercises. About 20 minutes

From your week of notes, write down your 3 strongest recurring candidates. For each one give the day it happens, how many times a year, and the minutes you actually timed. Keep mail and diary work out of it, that is Chapter 30.

CANDIDATE 1: DAY, RUNS A YEAR, MINUTES	CANDIDATE 2: DAY, RUNS A YEAR, MINUTES	CANDIDATE 3: DAY, RUNS A YEAR, MINUTES

Run the 5 questions down each candidate and stop at the first line that is true. Write which line killed each one. If all 3 survive, you have been generous with yourself, so go back and answer the mistake question again properly.

Take the survivor and do the sum in full, both sides. Minutes actually removed, times runs a year, minus build hours, minus a year of upkeep. Write the net figure and then write yes or no. A no here is a good outcome and you are allowed to stop.

Chapter 29 on one page

THE 4 SHAPES	Collect, gathering the same things from the same places · Reformat, one fixed form into another · Route, putting a finished thing where it is read · Watch, noticing a change and saying so
THE 2 GATES	What a mistake costs, and whether the input arrives in a consistent form. Neither is outvoted by a large saving. A bad input is fixed upstream, not downstream
THE 3 MAGNITUDES	Runs a year · minutes per run, timed not estimated · stability, tested by writing the steps as a numbered list and seeing where you write "it depends"
THE SUM	Minutes removed per run, times runs a year, minus build hours, minus a year of upkeep. Weekly is about 48 runs, every working day about 250, quarterly is 4 and 4 rarely pays
NEEDED BEFORE CHAPTER 30	One audited candidate, built, and back-tested against a week you already did by hand

BRING TO CHAPTER 30

Your audited candidate, the sum you wrote for it, and the automation now running on it. Chapter 30 uses the same 5 questions and the same sum, so bring the working, not just the answer.

BOTTOM LINE

You can tell a recurring task worth automating from one that merely annoys you, price either of them in under 5 minutes, and take a survivor through a build and a real test.

NEXT CHAPTER

How to Automate Your Inbox and Calendar

Chapter 30 turns the audit on the 2 places you have been told to leave alone until now. Mail and diary hold more recurring work than anything else in a professional week, and they are also where a careless automation does the most damage, so they get their own chapter and the same 5 questions.

BRING WITH YOU

- ☐ The audited candidate and the sum you wrote for it in this chapter
- ☐ A rough count of how many minutes a day you spend in mail and diary, timed rather than guessed

GO FURTHER

The Ultimate AI Library

Every guide, prompt pack and playbook AI Central has published, in one place. The reference shelf this course keeps pointing you to.

explore the library ↗



WHO MAKES THIS

AI CENTRAL

AI Central Media is a London-based editorial team. Our flagship publication covers practical AI for senior professionals, and 100+ AI companies, SaaS platforms and growth teams have advertised with us since 2023.

100+

AI companies, SaaS platforms, education brands and growth teams have advertised with us since 2023

613K

Accounts reached every month, across the 7 channels we operate

151

Countries our readers are in, plus all 50 US states

PROUD PARTNER OF

GAMMA · IIElevenLabs · guidde. · Outskill · Luma AI

HubSpot · attio · Synthflow · Notion · Replit

We turn attention into pipeline.

visit thecentral.ai ↗