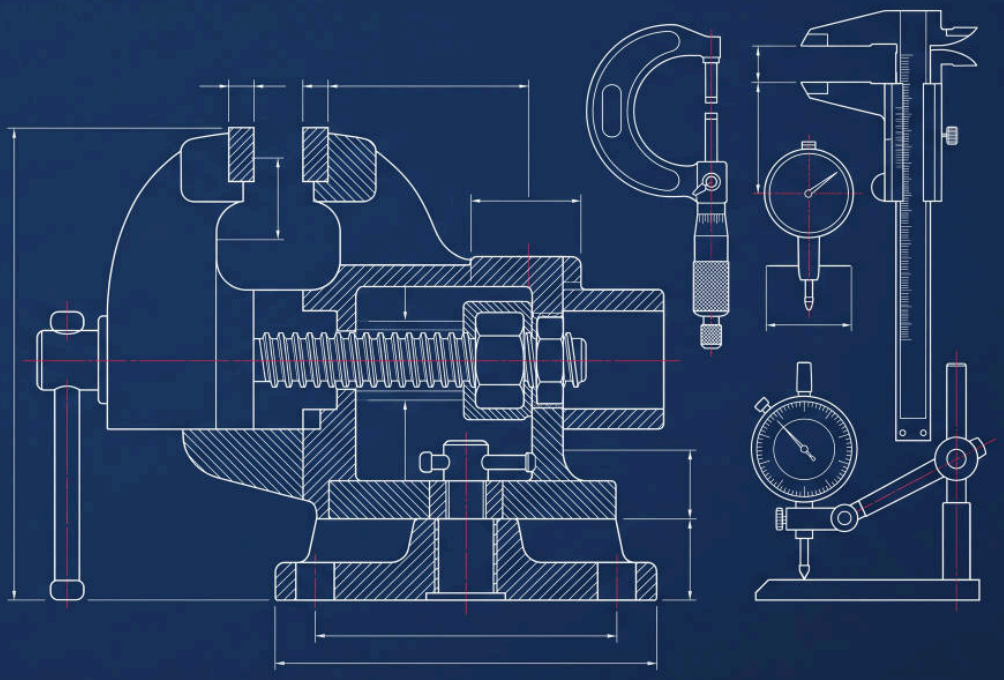

CHAPTER EIGHTEEN

Exercises

SECTIONS 18.1 TO 18.6

Seventeen chapters of argument are worth about one afternoon of measurement. These exercises are the afternoon, and several are designed to contradict what you expect.

CHAPTER EIGHTEEN

Contents

PART V · PRACTICE

Seventeen chapters of argument are worth roughly one afternoon of measurement. These exercises are the afternoon. Each one is small, each produces a number, and several are designed to contradict what you expect, because that is the only way an argument becomes knowledge.

18.1 How to Use These Exercises

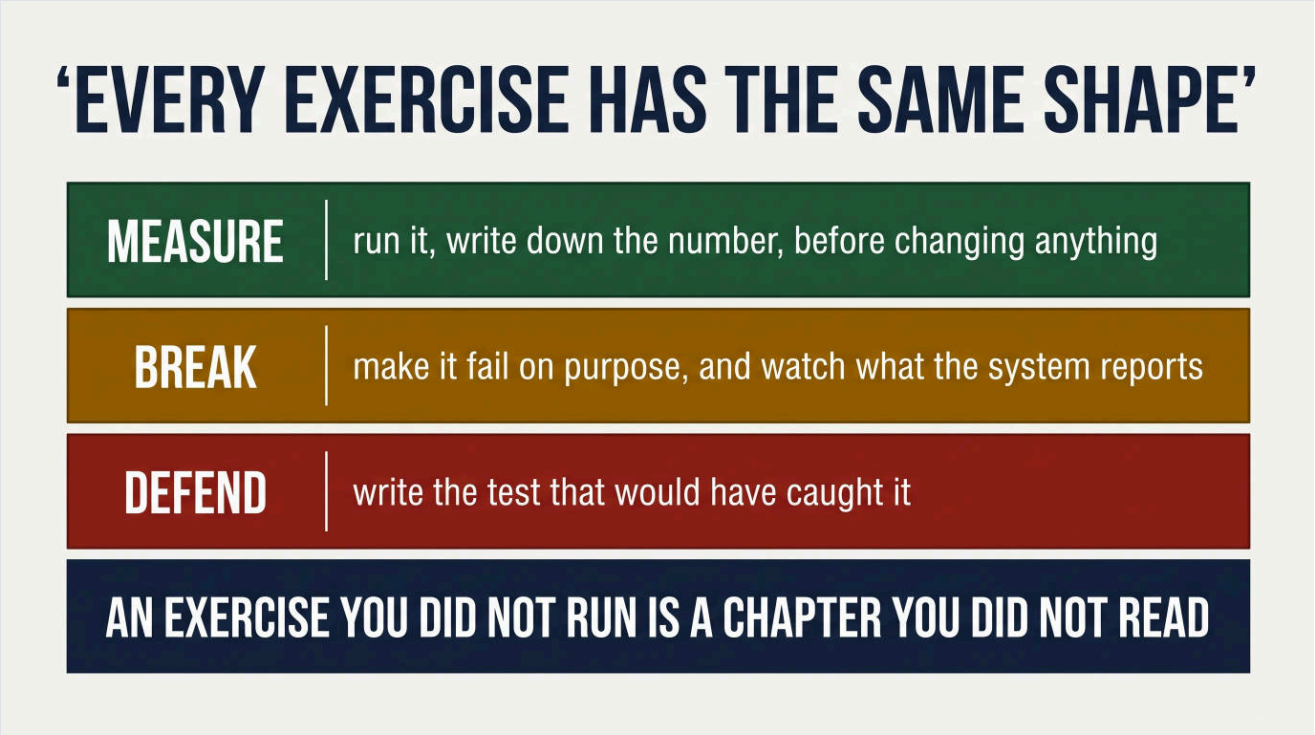

Figure 18.1 Three moves, in order. Skipping the first makes the third meaningless.

Every exercise here has the same three-part shape, and the order is not decorative.

Measure. Run the thing and write the number down before you change anything. A change with no baseline is an opinion, and section 14.1 established that a one-point movement on a small sample is noise you can talk yourself into either direction.

Break. Make it fail deliberately, and watch what the system reports. Most of this book's arguments are about failure modes that are silent, and the fastest way to learn whether yours are silent is to cause one.

Defend. Write the test that would have caught it. This is the step that converts an afternoon into a permanent property of the codebase, and it is the step everyone skips.

> **An exercise you did not run is a chapter you did not read.** The arguments in this book are all falsifiable, and several of them surprised me when I computed them. If none of these exercises surprises you, the most likely explanation is that you measured the thing you already believed rather than the thing that was claimed.

What you need

The `docassist` repository from the book, a corpus of twenty to fifty real documents you actually care about, and about thirty real questions someone has genuinely asked of them. **Not synthetic questions**, for the reason section 14.2 gave: they measure the generator rather than the system.

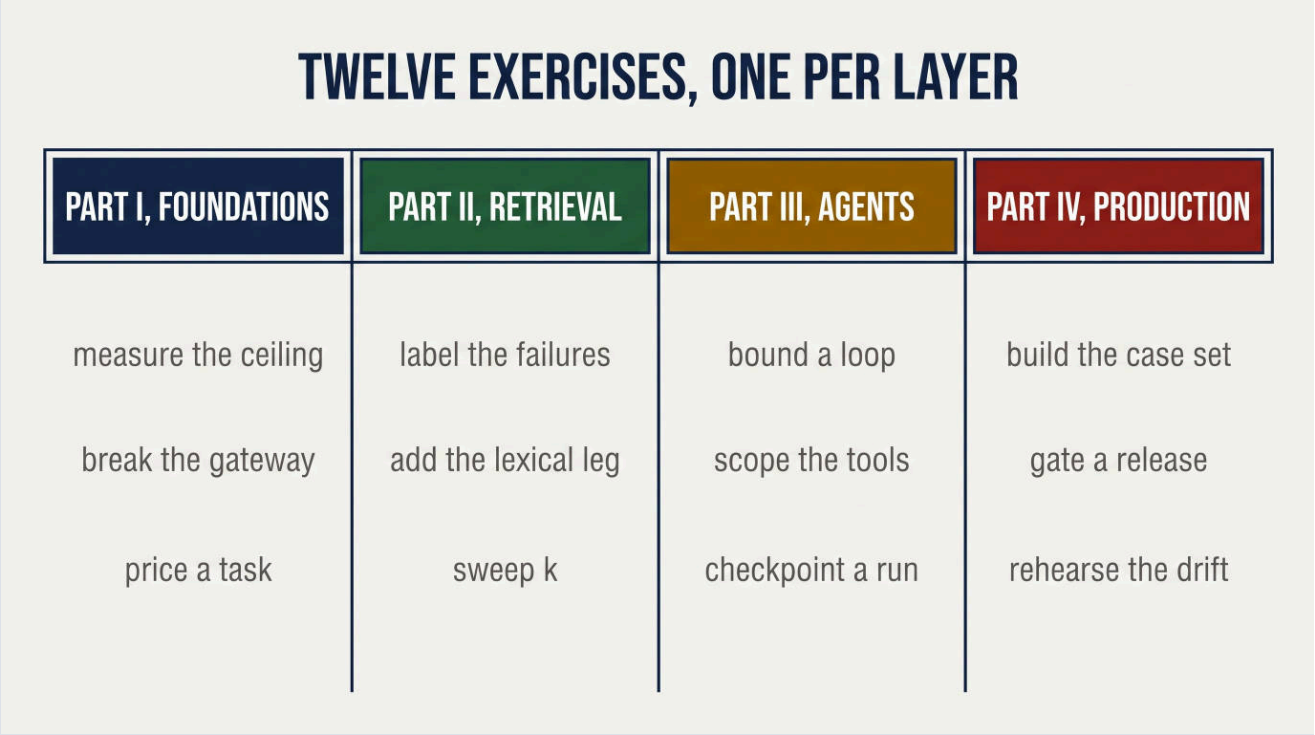

Figure 18.2 Fourteen exercises across the four parts. Three each for Parts I and II, four each for Parts III and IV, roughly a day each.

18.2 Part I: Foundations

EXERCISE 1.1 · THE CEILING IN YOUR OWN NUMBERS

Measure. Time thirty real generations end to end and take the median. Count the worker processes your API actually runs. The arithmetic reaches ahead to Part II's section 8.1, but the measurement needs nothing you have not already built, which is why it sits first.

Compute. `SyncCeiling(workers, seconds).requests_per_second`, then `workers_needed_for(50)`.

Defend. Put the second number in your capacity documentation. Section 8.1 got 700 workers; if yours is smaller, say why.

EXERCISE 1.2 · BREAK THE GATEWAY ON PURPOSE

Break. Point your gateway at a socket that accepts connections and never responds. Watch what happens to a request, to the worker pool, and to your health check.

Expect. Section 3.6's cascade, including the health check timing out alongside the request and a healthy pod being killed.

Defend. Add the deadline, the jitter, and the separate `/alive` probe. Re-run and confirm the pool survives.

EXERCISE 1.3 · PRICE ONE TASK HONESTLY

Measure. For one common task, record token counts and cost across fifty runs, and the fraction that produced a usable answer.

Compute. Cost per *successful* task, per section 1.6, not cost per call.

Surprise to look for. The gap between the two numbers is your success rate, and it is usually worse than the team's estimate of it.

18.3 Part II: Retrieval

EXERCISE 2.1 · LABEL FIFTY FAILURES BY STAGE

Measure. Take fifty questions your system answered badly. For each, decide where it broke: not in the corpus, not chunked usefully, not retrieved, not ranked into the window, retrieved but ignored, or generated wrongly.

Expect. Section 7.2's distribution: most failures happen before the model is called.

Defend. This labelling *is* the deliverable. It decides what you work on for the next month, and it is the single best-value afternoon in this book.

EXERCISE 2.2 · ADD THE LEXICAL LEG

Measure. Recall at 20 on your question set, dense only.

Change. Add BM25 and fuse with reciprocal rank fusion. Eighty lines, no new infrastructure.

Expect. The gain concentrated on queries containing identifiers, part numbers or clause references. If you see no gain, check whether your questions contain any.

EXERCISE 2.3 · SWEEP K UNTIL IT HURTS

Measure. Recall and answer accuracy at k = 2, 4, 8, 12, 20.

Expect. Section 7.10's trap: recall keeps climbing while answer accuracy peaks and then falls. Find your own peak.

Defend. Set k from the accuracy curve, and write down that recall alone would have chosen differently.

18.4 Part III: Agents

EXERCISE 3.1 · COMPUTE YOUR OWN COMPOUND RELIABILITY

Measure. Estimate per-step accuracy by sampling fifty agent steps and labelling each correct or not.

Compute. `compound_reliability(p, n)` for your typical step count.

Surprise to look for. Compare five steps at your rate against ten steps at two points better. Section 9.1 found the shorter loop wins; check whether it wins for you.

EXERCISE 3.2 · LET A LOOP RUN AWAY, THEN BOUND IT

Break. Remove the step budget and give the agent a question it cannot answer from the corpus. Watch the cost.

Expect. Not a wrong answer: a run that does not return. Note how long it took you to notice.

Defend. Add all three ceilings and the no-progress detector, then find an input that trips each one *separately*. If you cannot trip the cost ceiling without the step ceiling, your ceilings are not independent.

EXERCISE 3.3 · SURVIVE A DEPLOY MID-RUN

Break. Start a long run and restart the worker halfway.

Expect. Without checkpointing, the whole run is lost and paid for twice.

Defend. Add per-node checkpoints, repeat, and confirm the resumed run does not redo completed work. Section 10.7's `rounds=2` after approval is the property you are looking for.

EXERCISE 3.4 · INJECTION, END TO END

Break. Put an instruction in a document your system will retrieve: something asking it to call a tool it should not, or to reveal its system prompt.

Grade behaviourally, per section 14.9: it is fine for the model to mention the attempt; it is not fine for the tool call to fire.

Defend. If it fired, the fix is the output policy, never a sentence added to the system prompt.

18.5 Part IV: Production

EXERCISE 4.1 · BUILD EIGHTY CASES FROM REAL TRAFFIC

Do. Pull eighty questions from your traces. Label the expected chunk and the facts that must appear, per section 14.2, not the expected prose.

Hold out twenty you never tune against.

Defend. Run them nightly. This is the artefact everything else in Part IV depends on.

EXERCISE 4.2 · GATE A CHANGE YOU BELIEVE IN

Do. Take a prompt change you are confident improves things. Run it through the five gates of section 14.6.

Surprise to look for. Check cost per successful task. Section 5.2 found a longer prompt that scored better and destroyed money; a quality-only gate approves that change every time.

EXERCISE 4.3 · REHEARSE THE INCIDENT WITH NOTHING TO ROLL BACK

Break. Simulate a silent provider change: point at a different model version without telling the on-call person.

Measure. How long until anything notices? Which of section 14.5's six proxies moved first?

Defend. Write the runbook, then have someone who did not write it execute it. Section 16.6's third property is the test.

EXERCISE 4.4 · DELETE ONE CUSTOMER COMPLETELY

Do. Pick a tenant. Delete their documents, chunks, vectors, memory facts, superseded ancestors, cached answers and traces.

Verify. Search the trace store for their text afterwards. Section 15.4 calls traces the surface everybody forgets; find out whether you are everybody.

18.6 What Done Looks Like

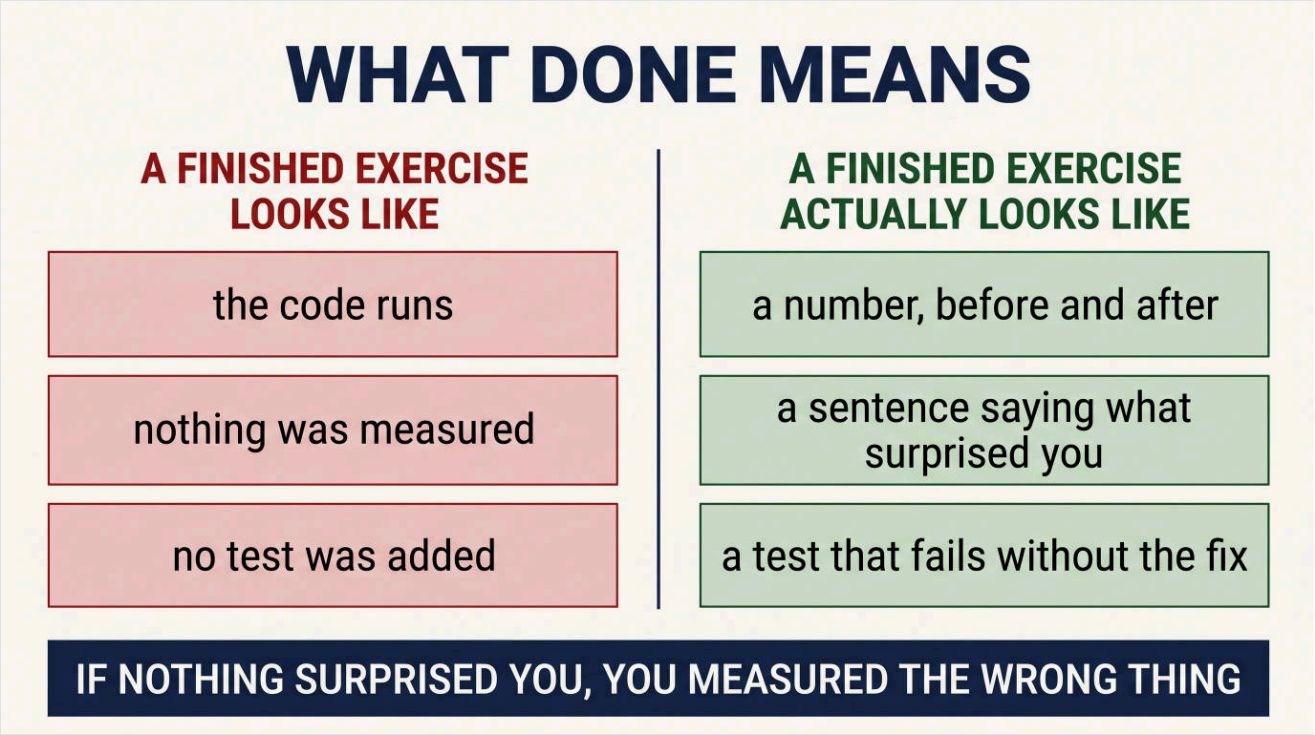

Figure 18.3 The left column is how exercises are usually finished. The right column is what makes them worth doing.

An exercise is finished when three things exist, and running code is only the first.

Artefact	Why it is the point
A number, before and after	Without a baseline you cannot tell improvement from noise, and section 14.1 says the noise is larger than you think.
A sentence about what surprised you	If nothing did, you probably measured the thing you already believed. Every finding in this book's closing table contradicted my expectation before I ran it.
A test that fails without the fix	This is the difference between an afternoon and a property. Section 14.2 makes each one a permanent regression case.

> A SUGGESTED ORDER, IF YOU ONLY HAVE A WEEK
>
> Exercise 2.1 first, always: labelling fifty failures by stage tells you which of the other thirteen exercises matter for your system. Then 1.3, because knowing cost per successful task reframes every later argument. Then 4.1, because without a case set the remaining exercises produce numbers you cannot defend.
>
> Those three are about two days and they are the ones I would defend as non-optional.

> FOUR QUICK MEASUREMENTS WITH NO EXERCISE OF THEIR OWN
>
> Four findings from the closing table are not reproduced by any exercise, because each takes minutes with the repo rather than an afternoon. Run them anyway, with your own inputs. Section 2.8's serving gap: `serving_capacity()` with the GQA and MHA configurations in `tokens/capacity.py`. Section 2.11's latency split: `LatencyModel`, comparing a trimmed prompt against a trimmed answer. Section 4.11's self-hosting decision: `utilisation_sweep()` with your hardware quote and your traffic. Section 5.2's shot count: `decide_shot_count()` with your measured accuracies. Each already exists as a test asserting the book's number; your job is to see whether the argument survives your values.

Chapter checklist

Have you...	Exercise
Computed your own synchronous ceiling and worker count for 50 req/s	1.1
Watched a slow provider take down a health check	1.2
Written down cost per successful task, not per call	1.3
Labelled fifty failures by stage	2.1
Measured the lexical leg's gain on identifier queries	2.2
Found the k where accuracy peaks and recall keeps rising	2.3
Computed compound reliability from your own per-step accuracy	3.1
Tripped each agent ceiling separately	3.2
Resumed a run after killing the worker mid-flight	3.3
Fired an injection at your own retrieval path	3.4
Built eighty labelled cases from real traffic	4.1
Gated a change you believed in on cost as well as quality	4.2
Had someone else execute your runbook	4.3
Deleted a tenant from the trace store as well as the database	4.4

What Chapter 19 offers

Exercises are bounded on purpose: each one isolates a single argument so the measurement is clean. Chapter 19 removes the bounds. It is a set of projects at three sizes, five real domains with their genuinely different hard parts, and a five-week shape that has worked, along with the scoping questions that decide whether a project is a portfolio piece or an abandoned repository.