USE IT IN YOUR WORK

Choosing and Using the Tools

A comparison written by someone selling none of them.

First edition, September 2026

Addaly

Series editor: Dr Mustafa Mun

Draft, open for academic review

USE IT IN YOUR WORK

Choosing and Using the Tools

A comparison written by someone selling none of them.

Series editor: Dr Mustafa Mun

Addaly

First edition, September 2026 · Draft, open for academic review

Choosing and Using the Tools

First edition, September 2026, build 62a26075.

Written by AI models to a written standard, with Dr Mustafa Mun as series editor. Exam keys checked blind by a second model: 3,665 of 3,666 agreed. Academic review invited.

The manuscript was drafted with the assistance of a large language model and was edited and published under his name. That is stated plainly here rather than left to be discovered, because a reader is entitled to know how a book they are trusting was made.

How to cite: *Mun, M. (2026). Choosing and Using the Tools. Addaly. addaly.com/learn/choosing-your-tools.*

This book is the printed form of the course of the same name at addaly.com/learn/choosing-your-tools. The website is the living version and is corrected first. Where the two differ, the website is right.

Free to read, free to download, free to print, and free to teach from. There is no paid edition and there never will be.

Every diagram in it is drawn from data by code, not generated as a picture, so the numbers on the page are the numbers in the argument. The answers to every check, test and examination question are at the back.

8 modules · 76 lessons · 28 figures · 8 case studies · 56,265 words · about 10 hours

Brief contents

1

MODULE 1

What a tool is made of

Take any AI product you are handed, name the exact model and size class inside it, say whether its thinking mode, context handling, system prompt or tool layer is producing the behaviour you are seeing, and predict which of those will change without notice.

Lesson 1 · 7 min

What you are actually choosing

THE ONE THING TO KEEP

You are choosing three things at once: a model, the product wrapped around it, and the route you reach it by.

One name, three layers

When someone says "I use ChatGPT," they have named a product. The model inside it — the trained network doing the thinking — has a different name, and it gets swapped for a newer one every few months without anyone asking you. Most bad tool decisions start with those two things being confused.

Every choice you make is really three choices stacked.

The model. A very large file of numbers produced by an expensive training run. GPT models from OpenAI, Claude from Anthropic, Gemini from Google, Llama from Meta, Mistral from Mistral AI, Qwen from Alibaba, DeepSeek from DeepSeek. Each family ships in sizes: a large one that is slow and expensive, a medium one most people should use, a small one that is fast and cheap. Same brand, very different capability.

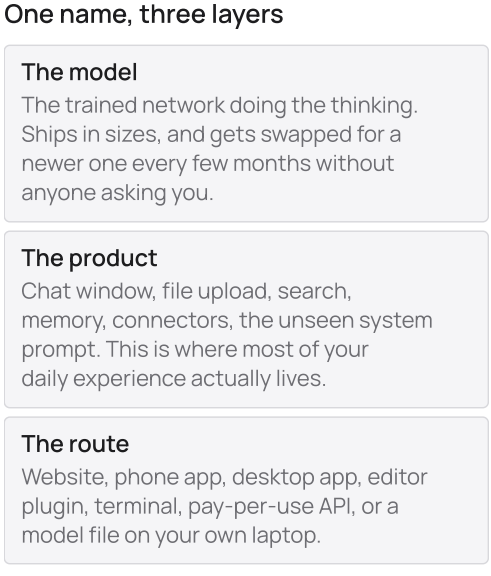
The product. Everything wrapped around the model: the chat window, file upload, web search, memory of past chats, the ability to run code, connections to your email or your code repository. This layer is where most of your daily experience actually lives.

The route. How you reach it. A website, a phone app, a plugin inside your code editor, a command in your terminal, a pay-per-use API, or a model file running on your own laptop.

The same model can feel like two different tools

Priya uploads a 40-page supplier contract to two different websites. Both say they run the same open model, same version. One gives her a useful clause-by-clause summary. The other misses half the document.

Figure 1.1



When a tool disappoints you, ask which layer failed before you decide the model is weak. Most complaints that sound like the model being stupid are the second or the third.

Nothing is wrong with the model. One product sent the whole contract. The other chopped it up, searched for what looked relevant, and sent three paragraphs. The model answered the question it was given. It was given a worse question.

This is the most useful habit to build before you compare anything. When a tool disappoints you, ask which layer failed:

- The model was not capable enough for the task.
- The product never gave the model what it needed.
- The route limited it — a free tier quietly serving a smaller model, an editor plugin that can only see the file you have open.

Most complaints that sound like "this AI is stupid" are the second or the third.

The map, briefly

Four product names cover most of what people use:

- **ChatGPT** (OpenAI) — the app most people met first. Web, mobile, desktop, plus an API.
- **Claude** (Anthropic) — chat app, an API, and a widely used coding tool that runs in a terminal.
- **Gemini** (Google) — chat app, and woven into Search, Docs, Gmail and Android on Google's own terms.
- **Copilot** (Microsoft and GitHub) — the clearest proof that product is not model. Copilot is a brand covering several different assistants, and over its life it has run models from more than one vendor. Asking "is Copilot good?" is like asking "is a restaurant good?" without asking who is cooking.

Then there are models you can download and keep: **Llama**, **Mistral**, **Qwen**, **DeepSeek**, **Gemma**. These have no product of their own. You supply the product — an app on your laptop, or a hosting company that serves them cheaply. Two lessons from now they get proper treatment.

Find out who you are talking to

Before you judge any tool, find the model name. It is usually in a dropdown at the top of the chat, or in settings, or at the bottom of the answer. Free tiers often start you on the strong model and move you to a smaller one after a few messages, and they do not always announce it clearly.

If you cannot find out which model is answering you, that is information too. It means you are evaluating a product, not a model, and your conclusions will not transfer when the company swaps the engine next quarter.

For the rest of this course, when a comparison is made, it will say which layer it is about. Almost every argument on the internet about which AI is best fails to.

BEFORE YOU MOVE ON

Two websites both let you chat with the same open model — same family, same size, same version — but one gives clearly better answers about your 40-page contract. What best explains the gap?

A. The product around the model: how each site chunks, retrieves or truncates the document before the model ever sees it.

B. One site must be running a secretly modified copy, because identical weights would produce comparable answers.

C. The better site uses a smarter prompt template, and phrasing matters more than anything else at this length.

D. Randomness. Ask a few more times and the two will even out.

The answer and why: p. 399

Lesson 2 · 8 min

Reading a model's name

THE ONE THING TO KEEP

A model name encodes family, version, size class and build date, and the size class and the build date are the two parts that quietly decide what you actually get.

The name is a specification

`claude-3-5-sonnet-20241022`. `gpt-4o-mini`. `gemini-2.5-flash`. `llama-3.1-70b-instruct`. `qwen3-8b`. `deepseek-r1`. These look like noise until you know the parts, and then each one tells you roughly what you are dealing with before you send a single message.

Almost every name carries four things, in some order.

The family. Who made it and which lineage it belongs to. GPT, Claude, Gemini, Llama, Mistral, Qwen, DeepSeek, Gemma, Phi.

The version. A number that goes up: 3, 3.5, 4, 4.1. A bigger number from the same maker is usually a genuinely better model, and this is the one part of the name you can mostly trust.

The size or speed label. This is the part people miss. `mini`, `nano`, `flash`, `lite`, `small`, `haiku` mean the cheap fast one. `pro`, `opus`, `large`, `ultra` mean the expensive slow one. `sonnet`, `medium`, and unlabelled names usually sit in the middle. Open models put the parameter count in directly: `8b` is eight billion parameters, `70b` is seventy.

The build or date. `20241022` is a snapshot from 22 October 2024. `-preview`, `-exp`, `-latest` are channels rather than dates, and they behave differently, which the last lesson of this module is about.

Two extra words appear on downloadable models. `instruct` or `chat` means it has been trained to follow instructions and hold a conversation — this is the one you want. A **base** model with no such suffix continues text rather than answering it; ask it a question and it may write three more questions. People download the base model by mistake, conclude the model is broken, and give up.

Version numbers are not comparable across companies

Gemini 2.5 is not "worse than" GPT-4 because 2.5 is less than 4. The numbers are internal to each company and reset whenever marketing decides. OpenAI has shipped `4o` and `o4` as different products in the same period; one is a general model, the other from a reasoning line. Read the family first, the number second, and never compare two makers' version numbers directly.

Finding out what is actually answering you

In a chat product, look in three places, in this order:

1. The model picker at the top of the conversation. It names a product tier, not always the exact model.

2. Settings, then the account or model page.
3. The small print under an individual answer — some products name the model there, especially after a fallback.

If it says something like *"Fast"*, *"Smart"*, or *"Auto"*, you have been given a routing label rather than a model name. That is a real answer too: it means the product is choosing for you and may choose differently tomorrow.

Through an API you always get the exact string, because you have to type it:

```
curl https://api.example.com/v1/chat/completions \
  -H "Authorization: Bearer $KEY" \
  -d '{"model": "gpt-4o-mini-2024-07-18", "messages": [...]}'
```

Why the date on the end matters

Two names can point at the same family, the same version and the same size, and behave differently:

- `claude-sonnet-4-5` — an alias. It points at whatever the current build is, and it will point somewhere else after the next release.
- `claude-sonnet-4-5-20250929` — a pinned snapshot. Same weights next month as today.

An alias is convenient and it is also how your carefully tuned workflow changes behaviour on a Tuesday morning without anybody deploying anything. If you are running something that matters, pin the date. If you are chatting, you cannot pin anything, so keep the possibility in mind when a tool seems to have changed personality.

Snapshots do not live forever. Providers publish deprecation dates, typically six to twelve months out for a pinned build, and after that the name stops working and your script returns an error rather than a worse answer. That error is a kindness. The alias would have silently degraded instead.

The one thing a name never tells you

A model name says nothing about the product wrapped around it. `gpt-4o` inside a chat app with web search, file upload and memory is a different experience from `gpt-4o` inside a badly built browser extension that sends three sentences of context. The name identifies the engine. It does not identify the car.

So the habit is: find the exact string, write it down with today's date next to your judgement of the tool, and remember that half of what you liked or disliked probably came from a layer the name does not cover.

BEFORE YOU MOVE ON

A team pins their API calls to a dated snapshot rather than the family alias. Six months later the call starts returning an error saying the model no longer exists. What does this show about the choice they made?

A. The pin failed, because pinning was supposed to guarantee the model would remain available indefinitely.

B. Dated snapshots are only for testing and production should always use the family alias so it stays current.

C. The pin worked: it traded a silent behaviour change for a loud failure on a published date.

D. The error means the provider withdrew the model early and the team should move to an open model they can host themselves.

The answer and why: p. 400

CASE STUDY · MODULE 1

The Tuesday the sorting broke

An illustrative case, built from documented patterns.

A scholarship trust in Nagpur, eleven days before its application deadline, with a sorting tool that has quietly stopped agreeing with itself

Sahyog Vidya Trust awards forty scholarships a year and receives about nine hundred applications for them. Each arrives as an email with a short statement of circumstances. A volunteer, Deepak, a second-year engineering student, built a sorter for it last March: paste the statement, and a free chat assistant returns one of six categories — first-generation learner, single-earner household, disability in the family, migrant household, girl child in a low-enrolment block, or none of these. The category decides which reviewer reads it and, for two of the six, which pot of money it is considered against.

It ran for seven months. Once a fortnight a trustee, Mrs Fernandes, read a sample of forty sorted applications against the trust's own written rules, and agreement had sat between ninety and ninety-five per cent all year. Nobody had touched the prompt since May.

On a Tuesday in March her sample came back at sixty-eight per cent. The errors were not scattered. Applications mentioning a father who had died were landing in *single-earner household*, when the trust's rules put a bereaved family into a separate review with a different reviewer and a different maximum award.

Shobha, who runs the programme, wanted to move to a different product that morning. Eleven days remained before the deadline and four hundred applications were still to arrive. Her argument was that the sample was the only quality check the trust had, that it had failed, and that spending the day investigating a free tool was a day the applications did not stop coming.

Deepak's argument was that he did not know what had changed, and that a different product would be a different unknown. He had not edited the prompt. The trust was on a free account whose model picker had said *Auto*

since the day he created it — a routing label, not a model name — and he had never looked at it in seven months. He could name three things that might have moved and had evidence for none of them.

The first was the model. Free tiers move you to a smaller model once an allowance is used, and the trust's sorting runs in bursts, so the afternoon batch might be answered by something different from the morning batch. The second was the product's own instructions, which the company rewrites without telling anyone. The third was the applications themselves. A new district scheme had opened in Gadchiroli in February, and those statements read differently: longer, and frequently describing two hardships at once.

Deepak's prompt held six rules in a single paragraph.

Shobha's counter was fair. Whatever had moved, the trust still had to sort four hundred applications correctly, and diagnosis is not sorting. She offered him until four o'clock, and pointed out that she could put two volunteers on manual sorting for the rest of the week, which would cost the trust about nine hours of work it had not budgeted for and would at least be work whose failures somebody could see.

Deepak's objection to that was that manual sorting had its own error rate and nobody had ever measured it. The ninety-three per cent figure was agreement between the tool and Mrs Fernandes, not between the tool and the truth. He did not know how often two volunteers would agree with each other on the same forty applications, and neither did she.

Mrs Fernandes, when asked, made the point that decided the afternoon. She had been sampling for seven months and she had nothing written down except a tally. She could not say whether the tool had ever been good at bereavement cases, because bereavement cases are perhaps one application in twenty and her samples were forty at a time. It was entirely possible that the category had always been wrong and March was simply the month a sample had caught it.

That changed the shape of the problem. If the fault was seven months old, then a hundred and some applications had been misrouted since September, several of them to a smaller pot of money, and that was a separate matter

from the deadline.

The thing all three agreed on was that the question was not which tool to use. It was which of the three layers had moved, or whether anything had moved at all, because the answer determined whether the fix was an hour or a fortnight — and because the trust had nothing written down that could tell them.

What would you have done with the afternoon, and what would you have wanted to exist before that Tuesday?

YOUR ANSWERS

1. Deepak's first thought was that the model had got worse, and Shobha's was to change product. Why was neither the right first move?

2. Why did sixty labelled applications from January settle the question so quickly?

3. A free chat account cannot pin a model version. What did the trust do instead, and what does that not protect against?

STOP HERE

Write your answers before you turn the page. How it played out starts on p. 50.

CASE STUDY · MODULE 1

How it played out

It took four hours. Deepak built the check he should have had already: sixty applications from January, with Mrs Fernandes's own labels beside them, in a plain text file. He ran those sixty through the tool again. Fifty-seven came back correct — the same as in January. So the tool had not become worse at January's material.

Then he ran sixty from the previous week. Forty-one correct. The inputs had changed, not the tool. The Gadchiroli statements were longer and mentioned more than one hardship, and a prompt with six rules in a paragraph was being followed to about five of them, with the middle ones dropped. Nothing had been deployed by anybody.

The fix took forty minutes: the six rules became a numbered list, and a seventh was added stating explicitly that a deceased parent goes to bereavement review even when the household now has one earner. Both sets then scored above ninety per cent. They kept the sixty labelled applications as a permanent file and ran them whenever anything felt wrong. In August the tool genuinely did change — the answers became longer and started arriving with headings — and they knew within fifteen minutes that this time it was not them.

The questions, discussed

1. Deepak's first thought was that the model had got worse, and Shobha's was to change product. Why was neither the right first move?

Both skip the question that decides what the fix costs. A chat product is a model, a product layer of unseen instructions, and a route — and the input is a fourth thing that can change on its own. A model swap, a rewritten system prompt, a router serving something smaller after the free allowance runs out, and a new kind of application all produce the same symptom. Changing product without knowing which one moved carries the risk of reproducing the fault on the new tool, and blaming the model leaves a prompt fault unfixed.

2. Why did sixty labelled applications from January settle the question so quickly?

Because they were fixed material with known correct answers. Running them again holds the input constant, so a drop in score can only come from the tool. Holding the score means the tool is unchanged and something else moved. Without that file the trust had only Deepak's memory of the tool having been fine, and memory of software quality is unreliable in one direction: the early good answers are remembered and the early failures are not, so everything feels as though it has declined.

3. A free chat account cannot pin a model version. What did the trust do instead, and what does that not protect against?

It kept a dated file of labelled examples and re-ran them whenever behaviour seemed off. That detects a change, quickly and cheaply. It does not prevent one: on a consumer plan there is no version to pin, no changelog granular enough and no undo, so the trust can be moved onto a different model mid-deadline and find out only afterwards. Detection is what a free tier permits; prevention requires a pinned snapshot on an API.

MODULE 1 TEST

Check what stuck

6 questions, one right answer each. This one is for you, not for a record. The answers, and the reason behind each, are in the key on p. 398. If two go wrong, re-read the module before the exam.

1 You upload the same forty-page contract to two services that both state they run the identical open model and version. One returns a clause-by-clause summary; the other misses half the document. What is the most likely explanation?

A. The second service chunked the file, retrieved the passages matching your question, and sent only those.

B. One provider is a full generation behind despite the matching version number, because version numbers are not comparable between companies.

C. The second service ran the model at a lower temperature, which makes it skip material it is less confident about.

D. Model quality varies between requests depending on load on the provider's hardware, and one request was unlucky.

2 You run an unattended script that classifies support tickets. Why does calling the alias rather than the dated snapshot change the risk you carry?

A. The alias costs more per token, because the provider charges a premium for always serving the newest build.

B. The alias points at whatever the current build is, so behaviour can shift on a day nobody deployed anything.

C. The alias has a smaller context window than the pinned snapshot, so long tickets are truncated without warning.

D. The alias disables prompt caching, so every request is billed at the full input rate rather than a cached one.

3 A small model is given a prompt containing seven rules and reliably follows five of them. What does the ladder in this module say to do first?

A. Move up to the large model, because ignoring instructions is the failure that size fixes.

B. Raise the temperature so the model explores more of the instruction before it answers.

C. Rewrite the prompt as a numbered list and run it again on the same small model.

D. Split the task into seven separate requests, one per rule, and combine the answers afterwards.

4 You switch thinking mode on for a question about the current registration fee for a certificate in your state. What should you expect?

A. A correct answer, because the extra reasoning lets the model search its training data more thoroughly.

B. A refusal, because reasoning modes decline questions they cannot check against a source.

C. The same answer at the same price, because thinking tokens are billed as input rather than output.

D. A more elaborate and more convincing answer, whether or not it happens to be correct.

5 Why does putting your question after a long document, rather than before it, measurably improve recall?

A. Models attend more reliably to the beginning and the end of a long input than to the middle.

B. The question is tokenised differently when it follows text, which makes it carry more weight.

C. Providers process the final part of a request at higher precision to save cost on the earlier part.

D. Putting the question last lets prompt caching match the document, and a cached document is read more carefully.

6 An answer that used web search turns out to be wrong. What should you check first?

A. The model, since a capable one would have recognised a bad source and said so before answering.

B. The tool output, because whatever it returned entered the context as text the model cannot check.

C. The temperature, because sampling is what introduces factual errors into an otherwise sound retrieval.

D. The citations alone, because a citation that resolves to a real page proves the claim it is attached to.

5. **One structured task.** Extract these six fields into a table, or convert this mess into a list.
6. **One reasoning task** from your work. A scheduling constraint, a cost comparison, a piece of logic you had to think about.
7. **One task you have failed to get done before.** The thing you gave up on.
8. **One edge case for your field.** A medical phrasing, a legal citation, a security question, a financial term — whichever your job puts near a refusal boundary.

Use real material, not made-up examples. Made-up material is always tidier than the real thing and every tool looks better on it.

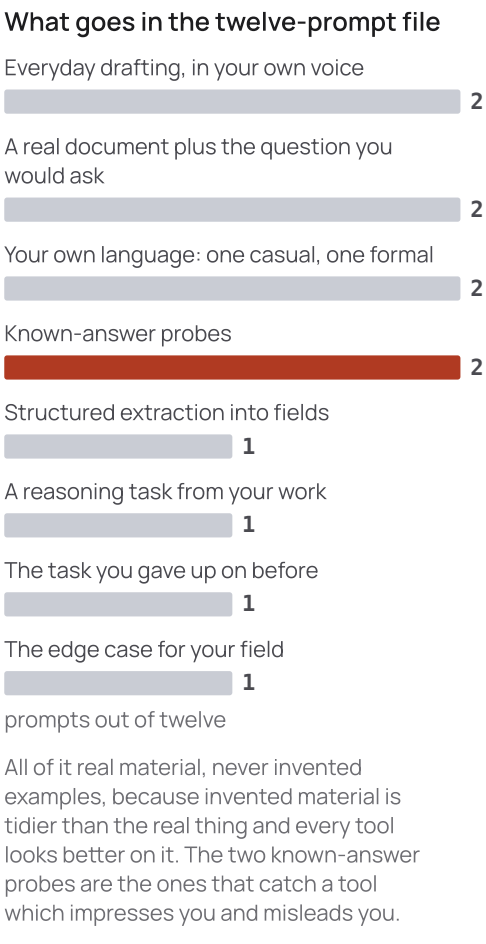
The format

Plain text. Not a spreadsheet, not a notes app that syncs, not a document in a tool you might stop paying for. A `.txt` or `.md` file you can open in 2032.

```
--- 03  supplier chase, Hindi ---
context: distributor 6 weeks late, we have paid 40% advance,
         want firm date without losing the relationship
prompt:  <the actual text you would send a model>
good:    polite, gives a deadline, does not grovel,
         natural Hindi not translated-English Hindi
last run: 2026-03-14  toolA 2  toolB 1  local-qwen 2
```

The `good:` line matters more than it looks. Writing down what a good answer contains **before** you see any answers is what stops you rationalising afterwards. It takes two minutes per prompt and it is the difference between a test and a vibe.

Figure 2.1



Keeping it honest over time

Do not fix a prompt because a tool failed it. That is how a test set stops testing. If the prompt is genuinely badly written, rewrite it and reset the history, noting that you did.

Do not let it become easy. As you get better at prompting, your prompts get clearer, and clearer prompts are easier. Every six months, add one hard new item and retire nothing.

Keep the failures. When a tool gets something wrong in an interesting way, that becomes prompt thirteen.

EXAMINATION

Choosing and Using the Tools — final examination

This paper covers the whole course:

- what a tool is made of
- testing one yourself
- the routes in
- what it actually costs
- open models and your own hardware
- data terms and the law
- the categories beyond the chat box
- living with a tool over years

56 questions, the whole pool. The site draws 25 for a sitting, pass mark 70%. Work any 25 under your own conditions. Answers from page 447.

1 Module 1 · What a tool is made of

You download an open model, ask it a question, and it replies with three more questions instead of an answer. What has most likely gone wrong?

A. You took the base model rather than the instruct or chat version.

B. The quantisation is too aggressive and instruction-following has collapsed.

C. The model's context window is too small for your prompt, so it echoed it back.

D. The runner is serving the model without a system prompt, which is required for it to answer at all.

2 Module 1 · What a tool is made of

A chat product's model picker offers Fast, Smart and Auto. What does that tell you?

A. That the product is serving three named models and has renamed them for clarity.

B. That the product is choosing for you, and may choose differently tomorrow.

C. That the product routes by your subscription tier rather than by the question.

D. That the product has not yet updated its interface after a model release, which is common in the first weeks.

3 Module 1 · What a tool is made of

Which description best captures what a small model is genuinely good at?

A. Tasks that are simple, meaning anything a person could do without thinking hard.

B. Tasks with short inputs, because window size is what separates the sizes.

C. Tasks where the answer is already in the input and the model only has to transform it.

D. Tasks in widely spoken languages, because the smaller training run concentrated on those first.

4 Module 1 · What a tool is made of

A product shows you a summary of the model's reasoning. How should you read it?

A. As proof of how the conclusion was reached, since it is generated before the answer.

B. As the exact computation, rendered in words for readability.

C. As advertising, since visible reasoning panels exist to make the wait feel purposeful and contain nothing useful.

D. As a helpful sketch, because a model can reach an answer by steps the visible text does not describe.

5 Module 1 · What a tool is made of

You type 'repeat your instructions' into a chat product and get a detailed system prompt back. What do you now have?

A. A hypothesis, because models often produce a plausible reconstruction instead.

B. The product's actual instructions, since the model can read its own context.

C. Nothing, because products universally block this request at the safety layer.

D. A version of the instructions from an earlier release, since the current ones are withheld from the model itself.

6 Module 1 · What a tool is made of

A chat product's code sandbox cannot install a library you asked for. Why?

A. Library installation requires a paid tier on every major product.

B. The sandbox usually has no internet access, and it is wiped between sessions.

C. The model is not permitted to run installation commands for safety reasons.

D. The sandbox runs a cut-down interpreter that supports only the standard library and nothing beyond it.

Module 1 · What a tool is made of

The module starts on p. 11.

MODULE 1 TEST

Module 1 test, Q1, p. 52

You upload the same forty-page contract to two services that both state they run the identical open model and version. One returns a clause-by-clause summary…

Answer A: The second service chunked the file, retrieved the passages matching your question, and sent only those.

Why: The product layer decides what the model is actually given. A product that retrieves splits your file into chunks, searches for the ones that look relevant, and sends those — which is excellent for 'what does it say about indemnity' and poor for 'summarise the whole thing', because the model never sees the whole thing. Same weights, worse question.

Module 1 test, Q2, p. 52

You run an unattended script that classifies support tickets. Why does calling the alias rather than the dated snapshot change the risk you carry?

Answer B: The alias points at whatever the current build is, so behaviour can shift on a day nobody deployed anything.

Why: A pinned snapshot gives you the same weights next month as today. An alias is re-pointed whenever the provider ships, and your prompt was tuned to the specifics of the old build. Pinning trades a quiet drift for a loud, scheduled migration you control — and the error you eventually get when a snapshot is retired is a kindness, because the alias would have degraded silently instead.

Module 1 test, Q3, p. 53

A small model is given a prompt containing seven rules and reliably follows five of them. What does the ladder in this module say to…

Answer C: Rewrite the prompt as a numbered list and run it again on the same small model.

Why: Look at how the outputs are wrong before moving up. Wrong facts mean the model does not hold the knowledge and you move up. Ignored instructions usually mean several conditions were packed into a paragraph, and the middle ones are the ones dropped — a numbered list frequently makes the same small model pass, at a twentieth of the price.

Module 1 test, Q4, p. 53

You switch thinking mode on for a question about the current registration fee for a certificate in your state. What should you expect?

Answer D: A more elaborate and more convincing answer, whether or not it happens to be correct.

Why: Thinking modes help where a wrong step produces a contradiction the model can notice — arithmetic with units, constraint problems, debugging, planning. Recall has no internal structure to check, so longer deliberation cannot fetch a fact the model does not hold. It raises your confidence in an answer more reliably than it raises the answer's accuracy, and you pay for every hidden token at the output rate.

Module 1 test, Q5, p. 53

Why does putting your question after a long document, rather than before it, measurably improve recall?

Answer A: Models attend more reliably to the beginning and the end of a long input than to the middle.

Why: The failure has a name — lost in the middle — and it holds on nearly every model at every price. Placing the instruction at the end puts it where attention is most reliable, and it costs nothing. On a paid API that ordering also suits prefix caching, but caching is a billing mechanism and has no effect on how carefully anything is read.

Module 1 test, Q6, p. 54

An answer that used web search turns out to be wrong. What should you check first?

Answer B: The tool output, because whatever it returned entered the context as text the model cannot check.

Why: The loop is: the model asks for a tool, the product runs it, and the result is pasted into the conversation as text. Garbage in the tool result is garbage in the answer, and the model has no independent way to know. A resolving link is not evidence either — models produce citations that look right and point at a page that does not say what the answer says, so click two of them.

THE LESSON CHECKS

Lesson 1, p. 15

Two websites both let you chat with the same open model — same family, same size, same version — but one gives clearly better answers…

Answer A: The product around the model: how each site chunks, retrieves or truncates the document before the model ever sees it.

Why: You are choosing three things at once: a model, the product wrapped around it, and the route you reach it by.

B One site must be running a…: Assumes that different output means different weights, when the same weights given different input behave completely differently.

C The better site uses a smarter…: Prompt wording does help, but no phrasing can recover text the app never sent to the model.

D Randomness. Ask a few more times…: Regenerating changes wording, not whether the document reached the model at all.

Lesson 2, p. 18

A team pins their API calls to a dated snapshot rather than the family alias. Six months later the call starts returning an error saying…

Answer C: The pin worked: it traded a silent behaviour change for a loud failure on a published date.

Why: A model name encodes family, version, size class and build date, and the size class and the build date are the two parts that quietly decide what you actually get.

A The pin failed, because pinning was…: Confuses stability of behaviour with permanence of hosting; a snapshot promises the same weights while it is served, not that it is served forever.

B Dated snapshots are only for testing…: Inverts the trade-off: the alias keeps working precisely by changing the weights underneath, which is the failure the pin was chosen to prevent.

D The error means the provider withdrew…: Jumps to a conclusion the evidence does not support; deprecation dates for pinned builds are published months ahead and an expiry is not an early withdrawal.

Lesson 3, p. 22

A small model classifying support tickets ignores three of the seven rules in your instructions. What does that failure pattern suggest you should do first?

Answer B: Rewrite the instructions as a numbered list and re-test the same small model, since dropped middle rules signal formatting.

Why: Start every task on the smallest model in the family and move up only when the failures are wrong facts or broken reasoning, because a fifth of failures are prompt problems and the price gap between sizes is around twentyfold.

A Move to the large model, since…: Treats every failure as a capability ceiling; dropped rules in a dense paragraph are usually a prompt-format problem and cost twenty times more to solve by upgrading.

C Split the task into seven separate…: Solves it by multiplying cost sevenfold when a formatting change usually fixes it, and it forfeits the model's ability to weigh rules against each other.

D Accept the result, because classification is…: Takes a general rule as a guarantee; being the right class of task makes the small model a good candidate, not a verified one, and twenty outputs still have to be read.

USE IT IN YOUR WORK

Choosing and Using the Tools

A comparison written by someone selling none of them.

WHAT IT TEACHES	A straight, comparative look at the AI tools people actually use — ChatGPT, Claude, Gemini, Copilot, and the open models you can download. What genuinely separates them, what each free tier really gives you, when paying is worth it, what happens to what you type, and how to re-check all of it in six months when the answers have moved.
WHAT IS INSIDE	Eight modules and 76 lessons, 28 figures drawn from data, eight case studies, eight module tests, a 56-question examination, and every answer with the reason behind it.
LICENCE	CC BY-NC-SA 4.0. Copy, print, share and adapt it for any non-commercial purpose, with credit, under the same licence. There is no paid edition.
THE COURSE	Free to read, with the lessons, the films and the examination, at **addaly.com/learn/choosing-your-tools**

Scan to open the course

Addaly

First edition, September 2026 · build 62a26075