A FREE COURSE

AI, properly understood

What a language model actually is, what kinds exist, what an agent adds, how to choose between cloud and local, what you are really paying for, and how to get usable work out of it — in plain language.

No account required. No prior knowledge assumed.
Made by a small team who work on AI tools and shared free.

Contents

1. Start here
2. What a language model actually is
3. The kinds of model that exist
4. Model, agent, and the thing in between
5. Cloud or local: the real choice
6. What you are actually paying for
7. Asking properly
8. Settings that actually change the answer
9. Put it together
10. Appendix

Nine short sections, two hours end to end. Read in order if you are new to this; skip to section 6 if you already use these tools daily and want the settings that actually matter.

MODULE 0

Start here

Most people have used a chat bot and nothing else. That is enough to begin.

This course assumes no technical background. Every term is explained the first time it appears, and there is a glossary at the back for the ones that come up twice. You do not need an account, a subscription, a credit card or a new computer. You do need about two hours, spread over as many sittings as you like.

What you will be able to do by the end

- Explain, in your own words, how a language model produces text — and why that explains both its usefulness and its mistakes
- Judge whether a particular task is a good fit for AI *today*, rather than being told by advertising
- Check an answer instead of trusting it, and know what kind of answer cannot be checked
- Ask for something specific enough to get a usable result
- Run a model on your own computer, offline, without an account
- Work out what you are actually paying for, and what it costs over several years

What is not covered

This is not a course about building software, training models or coding. It is not about "prompt engineering" as a career. It does not tell you which product to buy, because the useful knowledge outlives every product on the market this year.

How to use it

Each section is short and ends with two things: **one idea to keep**, and **one thing to try**. The trying matters more than the reading. If you only read, you will finish with opinions. If you do the small exercises, you will finish with judgment.

If you are short of time, read sections 1 and 2, then jump to 6 and 7. Those four give you the most usable understanding per minute.

> **A note on honesty.** Where this course says AI cannot do something, that is a statement about today, not forever. Where it says something is uncertain, it is because the measurement genuinely is uncertain. You should distrust any AI writing that never admits a limitation, including this one.

What you need

Nothing beyond a computer you already own — Windows, Mac or Linux. Sections 1 to 4 need only a web browser. Section 5 asks you to install one free program, and that is the only install in the course.

> **Keep this idea:** Most confusion about AI comes from one missing fact — what the machine is actually doing. Section 1 fixes that, and everything after it gets easier.

MODULE 1

What a language model actually is

A large language model does one thing: it predicts what comes next.

Given some text, it produces the most plausible continuation, one small piece at a time. That is the whole engine. It is why the output reads fluently, why it can write in your style, why it can summarise and translate — and, crucially, why it can be completely wrong while sounding entirely certain.

The mechanism, without mathematics

Text is broken into pieces called **tokens**. A token is roughly three-quarters of a word: "understanding" might be two tokens. The model reads a sequence of tokens and produces a probability for what token should follow.

It picks one. Now the sequence is one token longer, so it does the whole thing again. Repeat a few hundred times and you have a paragraph. Nothing in that loop knows whether the sentence is *true* — it only knows which continuation is most likely given what has come before.

This is why the same question can produce a different answer on a second attempt. The model samples from probabilities rather than looking something up, so there is an inherent wobble.

Why it sounds right when it is wrong

Because "sounds right" is literally the objective. A model trained on enormous amounts of text learns what plausible writing looks like: how citations are formatted, how a confident executive summarises a market, how a legal paragraph opens. So when it invents a citation, it invents one with the correct shape — author, year, plausible journal, believable page numbers.

This failure has a name, **hallucination**, and the misleading part is that it is not a bug in the ordinary sense. It is the mechanism working as designed in a situation where the designed behaviour is not what you wanted.

A useful rule: **the more a wrong answer would look like a right one, the more you must check it.** Numbers, dates, quotations, names, citations, legal or medical specifics. Fluent invention is most convincing exactly where it is most damaging.

Context: what the model can see

The **context window** is how much text the model can hold in view at once — your question, the conversation so far, any document you pasted. Measured in tokens.

Two consequences people discover the hard way:

- **It forgets what falls out.** In a very long conversation, the earliest parts drop out of the window. The model is not being rude; it genuinely cannot see them any more.
- **It cannot read a whole book in one go** unless the window is enormous. Long documents are handled by cutting them up and processing the pieces, which is why summarising a 300-page report can lose the thread of the argument.

What "8B" means, and why size matters

Models are described by parameter count: 8B means eight billion parameters. Parameters are the learned numbers that shape its predictions — think of them as the model's accumulated sense of how language fits together.

Bigger models generally reason better over several steps and hold more nuance. They also need more memory and produce words more slowly, because every word generated requires reading the whole model. There is no free lunch: capability and cost rise together.

Two numbers matter when you run one yourself:

WHAT IT CHANGES	EFFECT
Parameter count	More capable, needs more memory, slower per word
Quantisation	Smaller file, slightly less accurate; "Q4" means roughly 4 bits per parameter

Quantisation is why a model that nominally needs 16 GB can sometimes run in 5. It compresses the parameters with a small loss in quality. Q4 is the usual compromise on ordinary computers; Q8 is closer to the original but roughly twice the size.

Training data, and the cutoff

A model learns from text collected up to some date. After that date it knows nothing, and it does not reliably know that it knows nothing — asked about last week's news it may answer as if the cutoff were today. Never ask a bare model for current facts; give it the facts and ask it to work with them.

Writing is not understanding

The most useful sentence in this section: the model has no model of the world, no beliefs, and no intention to help or mislead. It has a statistical sense of which words follow which. That produces remarkably useful output and explains every one of its failures.

Keep this idea: a language model predicts the next token. Fluent, confident and sometimes wrong are all the same fact seen from different angles.

TRY THIS

Ask any chat assistant these three things in one conversation: 1. "Summarise this paragraph in one sentence" — paste in something you wrote. It will do this well. This is the mechanism at its best. 2. "Give me three academic references about the history of the pencil, with years and journals." Then check whether those references exist. Usually at least one will not. 3. "What is 4,317 multiplied by 892?" Then check the answer on a calculator. You will see the same engine succeed, invent and miscalculate. The difference is not intelligence — it is whether the task rewards prediction or requires computation.

MODULE 2

The kinds of model that exist

"What AI should I use?" is usually the wrong question. The right one is "what am I trying to do?" Different tasks want different kinds of model, and using the wrong kind is the most common reason people conclude that AI "doesn't work".

The main families

KIND	WHAT IT IS FOR	WHERE YOU MEET IT
Chat / instruct	Conversation, drafting, summarising, explaining	Every assistant you have used
Reasoning	Multi-step problems, logic, maths, planning	"Thinking" modes; slower and dearer per answer
Vision	Reading images, screenshots, scans, charts	Photo of a receipt, diagram, handwritten note
Speech	Transcribing audio; generating speech	Meeting notes, subtitles, voice assistants
Embedding	Finding things by meaning rather than keywords	"Chat with your documents", search
Code	Writing and repairing software	Coding assistants

A single product often bundles several of these, which is why one app can feel like it does everything. Underneath, it is calling different models for different jobs.

Chat models

The default. Trained to follow instructions and hold a conversation, and usually the most useful general-purpose tool you have. When people say "AI wrote me a letter", this is what did it.

Their strength is language: tone, structure, register, translation, rewriting, explaining something at three different levels of difficulty. Their weakness is anything that requires computation or verification, from section 1.

Reasoning models

Some tasks are too long-chained for a single pass of prediction: work through this spreadsheet and find the inconsistency; plan a project with dependencies; prove this step follows from that one. Reasoning models are trained to work through intermediate steps before answering.

They are better at those tasks and worse at being cheap or fast. A reasoning model asked to rewrite a short email is an expensive way to get the same answer more slowly. Use them for problems with multiple steps, not for polishing prose.

Vision models

Give a picture instead of a description. This is genuinely different from typing what you see: a vision model can read a photographed invoice, find the error in a diagram, or tell you what is in a medical report you already have — though never trust it as a clinician.

The common failure is detail at small scale: faint handwriting, dense tables, low-resolution scans. If the answer matters, crop the image to the part you care about.

Speech models

Two directions: speech to text (transcription, subtitles, meeting notes) and text to speech (narration, accessibility). Transcription is one of the most genuinely useful applications in daily work and one of the least discussed.

Despite the name, this is not the same model as a chat model. Feeding a transcript into a chat model afterwards is where the value compounds.

Embedding models

These do not answer questions. They turn text into lists of numbers that capture meaning, so that "how do I reset my router" and "troubleshooting home internet" end up close together.

They are the reason "chat with your PDF" works: your document is broken into pieces, each piece becomes an embedding, and your question is matched against them by meaning. When such a tool gives a poor answer, the failure is usually in the retrieval step rather than the chat model — a distinction worth knowing, because the fix is different.

Small models and large models

The same family comes in very different sizes, and the choice is practical rather than philosophical:

	SMALL (ROUGHLY 1–8B)	LARGE (ROUGHLY 30B AND UP)
Runs on	A normal laptop	Servers, or a computer with a lot of memory
Speed	Fast	Slow on ordinary hardware
Good at	Short writing, tidying, extracting, translating	Nuanced reasoning, long analysis

	SMALL (ROUGHLY 1–8B)	LARGE (ROUGHLY 30B AND UP)
Cost	Often free	Usually a subscription or per-use fee

A small model that answers in two seconds is often more useful than a large one that answers in ninety. For most everyday writing and reading tasks, a well-chosen small model is enough — a fact that the marketing of large models tends to obscure.

Mixture models, briefly

Some models are built from several sub-models and only use part of themselves for each word. The practical effect: memory requirement is set by the whole thing, but speed behaves more like a much smaller model. That combination is why they are popular for running locally on ordinary processors.

What about "open" and "closed"?

Closed models are rented: you send text, you get text back, you pay monthly or per use. Open models are files you can download and run. This is the subject of section 4, because it has consequences for cost, privacy and whether the tool works when the internet does not.

> **Keep this idea:** match the kind of model to the task. Most disappointment with AI comes from using a language model for a job that wanted retrieval, or a small model for a job that wanted careful reasoning.

TRY THIS

Write down three things you would actually like AI to do for you this month. For each, decide which family from the table above should do it — and notice when the honest answer is "none of them; that needs a different tool entirely".

MODULE 3

Model, agent, and the thing in between

Three words get used as if they were interchangeable, and they are not. Getting them straight tells you what a product actually is, and whether its claim about "AI doing your work" is plausible.

The model writes text. That is all.

A **model** takes text in and produces text out. In one turn, with no memory beyond the conversation you can see, no ability to open a file, no ability to click anything.

This is worth sitting with, because almost everything described as "AI" is doing far more than that, and the extra is not in the model.

The harness is the software around it

A **harness** is everything wrapped around the model to make it useful:

- **Tools.** Ability to read a file, write a document, search, calculate, run code. The model cannot do arithmetic reliably, but it can call something that can.
- **Steps.** A plan: first extract the figures, then total them, then write the summary. One pass of prediction cannot reliably do a five-step job.
- **Context assembly.** Deciding what to put in front of the model: your document, the earlier steps, the rules it should follow.
- **Checking.** Comparing the result against the requirement, and going back if it failed. This is the part that separates a demo from a tool.

When people say a model "can't do maths" and a product "does my accounts", the difference is almost always the harness, not a different model.

An agent is a harness with a goal and a loop

An **agent** is a harness allowed to keep going: given an objective, it decides the next action, takes it, looks at the result, and decides again, until it thinks the job is done or it runs out of room.

That loop is where usefulness and danger both come from. It can complete multi-step work that a single prompt cannot. It can also go confidently in the wrong direction for twenty steps, or take an action you would not have approved. This is why serious agent software asks before irreversible things, and why you should be wary of agent tools that never ask.

Why "a good model" does not mean "correct output"

These are separate claims:

CLAIM	WHAT IT MEANS	HOW TO CHECK
"It uses a strong model"	The writing component is capable	Nothing about correctness
"It checks its work"	There is a verification step	Ask what it checks, and what happens when the check fails
"It produced the file"	Output exists	Open it and check the figures yourself

A capable model with no checking will produce confident mistakes faster than a weaker one, because fluent wrong answers are harder to spot than clumsy ones. Verification is a design decision, not a property of model size.

How to read an AI product's claims

Four questions that cut through most marketing:

1. **What does it actually touch?** Reading your files is different from writing to them.
2. **Where does the thinking happen?** On your machine, or on a server that receives your data?
3. **What happens when it is wrong?** Silence, or a visible failure you can correct?
4. **What is it doing that a model alone could not?** If the answer is "nothing", you are paying for a chat wrapper.

> **Keep this idea:** the model is the engine; the harness is the vehicle. Most of what makes AI useful or useless in practice happens in the part nobody advertises.

TRY THIS

Take an AI product you have seen advertised this month. Answer the four questions above from its own website. Notice how often the site does not answer question 3 at all — and treat that silence as information.

MODULE 4

Cloud or local: the real choice

There are two places AI can run, and the difference matters more than any feature list.

Cloud means someone else's computer. You send text over the internet, their machine does the work, the answer comes back. Every assistant you have used works this way.

Local means your own computer. The model is a file on your disk. It runs without a connection, without an account, and nothing leaves the machine.

Neither is better. They solve different problems.

The honest comparison

	CLOUD	LOCAL
Capability at the top end	Highest — the biggest models live here	Good, not equal to the largest
Speed	Fast, whatever your hardware	Depends entirely on your machine
Cost	A subscription, forever	A one-off download, then free to run
Privacy	Your text reaches a company	Nothing leaves your computer
Works offline	No	Yes
Works without an account	No	Yes
Available in your country	Not always	Always — it is just a file
Maintenance	Theirs	Yours

The privacy point, stated precisely

Cloud assistants receive your prompts. That is not a scandal; it is how the service works. What varies is what happens next: whether your text is stored, for how long, whether it is used to improve their models, and whether a human might read it. Those answers live in each company's terms, they change, and they differ by plan.

Local removes the question entirely, because there is nobody to send it to. That is why people handling client records, medical details or anything confidential tend to care about it.

Two honest caveats:

- **Local is not automatically safe.** Whoever installed the software can log it. A model running on a shared or work computer is not private from that computer's owner.
- **Cloud is not automatically unsafe.** Most providers handle data carefully. The point is that you are trusting a policy rather than a physical fact.

The practical rule: **never put anything in a cloud chat that you would not email to a stranger.** Client lists, identity documents, medical records, passwords, unpublished financial figures. Not because something terrible will happen, but because you cannot un-send it.

Availability is a real constraint

Some AI services do not operate in some countries. This is usually a commercial decision by the company rather than a law in your country — Hong Kong, Macau and mainland China are common examples for the largest assistants.

This matters practically: if a service does not operate where you are, using workarounds breaks its terms and puts the account at risk. This course does not explain how to do that, because there is no version of it that is compliant.

A local model sidesteps the issue completely: it is a file. It does not check where you are, does not need an account, and does not stop working because a company changed its policy.

What local actually requires

Less than people expect, and more than they hope:

- **Memory (RAM) is the main constraint**, not the processor. 8 GB runs small models; 16 GB is comfortable for everyday ones; 32 GB opens up larger models.
- **A graphics card helps a lot but is not required.** Without one, models run on the processor and are slower — often a few words per second rather than dozens.
- **Two memory sticks instead of one can double the speed**, because processor-based AI is limited by how fast memory can feed it. Many laptops ship with one stick.
- **Disk space** for the model files, typically 2 to 20 GB each depending on size.

You can have a genuinely useful local assistant on an ordinary laptop. You cannot have the largest, fastest model on one. That is the trade.

A decision list

Choose **cloud** when: you need the best possible answer, you are working with nothing confidential, you want zero setup, and a monthly fee is acceptable.

Choose **local** when: the material is confidential, you need it to work without internet, you object to a subscription, a service is unavailable where you live, or you want the thing to keep working in five years without anyone's permission.

Choose **both** when: you want the convenience of cloud for ordinary tasks and local for anything that matters. That is what most informed people end up doing.

> **Keep this idea:** cloud rents you capability you do not own; local gives you capability you control. The right answer depends on what you are putting into it.

TRY THIS

Write down three things you would like AI to help with this month. Mark each as "fine to send to a company" or "should stay on my machine". You have just decided your own cloud/local split.

MODULE 5

What you are actually paying for

Most people can name their subscription price but not what it buys. That gap is where money quietly disappears, because AI pricing is designed around usage and usage is invisible.

Free tiers are real, and limited

Free access is genuinely usable, and most people never hit the ceiling. The usual limits:

- **A smaller or older model** — often the previous generation rather than the newest
- **A message cap**, per hour or per day
- **A context limit** — less text can be held in view at once
- **Slower responses** at busy times
- **Fewer extras**: file reading, image understanding, longer outputs

If your use is occasional writing and reading, a free tier may be all you ever need. The honest test is not the feature table; it is whether you hit a wall doing your actual work.

What a paid tier actually buys

It buys **more capacity**, not ownership:

YOU GET	YOU DO NOT GET
Bigger models and higher limits	The model itself
Faster responses at peak times	Anything you keep if you stop paying
More context, file handling, features	Protection from a price change
Priority during outages	A say in what happens to the service

That distinction matters at the moment you cancel: everything you built *inside* their product stops working, while everything you exported remains yours.

> **Rule worth internalising:** if you cannot export it, you do not own it. Keep your documents as files on your own disk, not only inside somebody's assistant.

The arithmetic nobody does

A plan at a modest monthly price, compounded over the years you will plausibly use it:

MONTHLY	1 YEAR	3 YEARS	5 YEARS
$10	$120	$360	$600
$20	$240	$720	$1,200
$30	$360	$1,080	$1,800

Three years of a mid-tier subscription is more than a mid-range second-hand laptop. That is neither an argument for nor against — it is simply the number, and it is rarely shown next to the monthly price.

How usage-based pricing works

Many services charge per **token** rather than per month: you pay for the text going in (prompt) and the text coming out (completion), at different rates.

Three consequences that surprise people:

1. **Long documents cost more to send than short questions.** Pasting a 40-page report is expensive in a way that typing a paragraph is not.
2. **Output usually costs more than input**, often several times more per token. Asking for "a detailed 3,000-word analysis" is the expensive direction.
3. **Conversations accumulate.** Every follow-up resends the history, so long chats get more expensive per message as they grow.

For heavy or predictable use, a flat subscription is usually cheaper. For occasional or bursty use, pay-per-use is usually cheaper. Guessing wrong in either direction is the common mistake.

The costs that are not on the invoice

- **Lock-in.** Work created inside a tool you cannot export has a switching cost you discover later.
- **Price changes.** Subscriptions are repriced. A habit built on a cheap tier becomes a decision later.
- **Data.** "Free" is sometimes paid for with usage data. Read what the terms say about training on your content, and decide whether that is a fair trade for you.
- **Your attention.** Tools that interrupt to upsell cost you something real, if unmeasured.

A two-minute estimate

Do this before subscribing to anything:

1. How many days a month would you actually use it? (Be honest; the answer is usually under ten.)
2. Does a free tier cover that volume?
3. If not, what is the cheapest paid tier that does?
4. Monthly × 36 — is that number still comfortable for three years?
5. Could a local model do the 80% of your use that matters?

If a subscription survives those five questions, it is probably worth it. Most do not survive question four once it is written down.

> **Keep this idea:** you are renting capacity, not buying capability. Rent when you need scale; own when you need the thing to keep working without permission.

TRY THIS

Add up what you currently pay monthly for AI tools. Multiply by 36. Then look up the price of a used laptop with 16 GB of memory. The comparison is the entire argument, and now you have the number for your own situation.

MODULE 6

Asking properly

Most disappointing results come from the request, not the model. A vague instruction gets a vague answer, and then people conclude the tool is weak.

The good news: there is no secret vocabulary. Five habits cover almost everything.

1. Say who it is for and what shape it should take

Compare:

> **Weak:** "Write something about our new pricing."
>
> **Better:** "Write a 150-word email to existing customers explaining that our prices rise from March. Plain, no marketing language, no exclamation marks. They should feel informed, not sold to."

The second one names the audience, the length, the tone and what to avoid. It will produce something you can use rather than something you must rewrite.

2. Give the material instead of describing it

Models work far better with the actual text than with your summary of it.

> **Weak:** "Summarise a report about our Q3 sales and the problems we had."
>
> **Better:** *[paste the report]* "Summarise this in five bullets, each under 20 words, aimed at someone who has thirty seconds."

If the document is long, give the section that matters. If you cannot paste it, say so and ask what information would help — then supply that.

3. Ask for a file, not a conversation

If you want something you can use, say so:

- "As a table with four columns: date, item, amount, note."
- "As a checklist I can print."
- "As a plain-text outline I can paste into Word."
- "As a single paragraph with no preamble."

Without this you get chat, which you then have to convert by hand. The format request is usually worth more than any other improvement to the prompt.

4. Ask it to check, and know what that is worth

Adding "check your arithmetic and show the working" genuinely helps with some failures and does nothing for others. A model asked to verify its own invented citation will happily confirm it.

What self-checking *is* good for: internal consistency, unit conversions, spotting that a total does not match its parts, noticing that a list has eleven items where you asked for ten.

What it cannot do: know whether a fact is true. For that, use a source outside the model.

A better instruction than "check your work":

> "List every number you used and where each came from. Flag anything you assumed."

That produces something you can audit in ten seconds.

5. Work in steps instead of one giant request

One enormous prompt asking for planning, drafting, formatting and proofreading at once tends to produce something mediocre at every stage. Split it:

1. "Here is my material. Propose an outline of five sections."
2. *[pick one]* "Write section 2 in 120 words, plain tone."
3. "Now tighten it: remove any sentence that does not add information."

Each step is checkable. Errors get caught before they compound.

Five rewrites, side by side

VAGUE	USEFUL
"Write a letter to my landlord"	"Write a 120-word letter to my landlord asking him to fix the leaking tap in the bathroom. Polite, firm, mention that I reported it on 3 March. No legal threats."
"Make a budget"	"Make a monthly household budget as a table with columns: category, planned, actual. Categories: rent, utilities, groceries, transport, other. Leave the actual column blank."
"Explain quantum computing"	"Explain how a quantum computer differs from a normal one, in under 150 words, for someone with no physics. Use one analogy, and say clearly where the analogy breaks."

VAGUE	USEFUL
"Summarise this"	"Summarise this contract clause in two sentences, and separately list any obligation it places on me with a deadline."
"Fix my CV"	"Rewrite these three bullet points so each starts with a verb and includes a number. Keep each under 18 words. Do not invent achievements."

One habit that prevents most bad outcomes

Tell it what not to do. "Do not invent figures." "Do not add enthusiasm." "Do not include a summary at the end." Models are helpful to a fault; they will add things you did not want unless you say otherwise.

> **Keep this idea:** you are not casting a spell, you are briefing a capable assistant with no context. Say who it is for, what shape you want, what to avoid, and check anything factual somewhere else.

TRY THIS

Take the last thing you asked an AI for and rewrite it using habits 1, 3 and 5. Compare the two answers. The improvement is usually larger than people expect, and it costs one extra sentence.

MODULE 7

Settings that actually change the answer

Every assistant has a settings panel. Most of it does not matter. Five settings do, and changing them deliberately is the difference between fighting the tool and using it.

1. Temperature — how much variation

The single most useful dial. It controls how much randomness is allowed when the model picks each next token.

TEMPERATURE	BEHAVIOUR	USE IT FOR
Low (0–0.3)	Same question gives near-identical answers; sticks to the safest continuation	Facts, code, calculations, extracting from documents, anything you will check
Medium (0.6–0.8)	Balanced; the usual default	Drafting, explaining, general writing
High (1.0+)	More varied, more surprising, more likely to wander	Brainstorming, naming, alternatives you would not have thought of

If you find yourself re-asking the same question hoping for a better answer, the temperature is the reason. And if you are using a tool for factual work and cannot find this setting, that is itself worth knowing.

2. Context length — what it can see at once

How much text the model holds in view: your message, the conversation, any documents.

- **Too small** and it loses the beginning of the conversation, or ignores part of a document you gave it
- **Too large** costs money and memory, and on a local model it may exceed what your machine can hold

Practical habit: start a **fresh conversation** for each new task. A long thread about last week's problem quietly contaminates today's answer. New topic, new chat.

3. System instructions — your standing brief

Most tools let you set permanent instructions that apply to every conversation. This is the highest-value setting that almost nobody uses.

A good standing brief says who you are, what you want, and what to avoid:

> I run a small business in Hong Kong. Write plainly, British spelling, no marketing language, no exclamation marks. When I ask about money, show the arithmetic. If you are unsure about a fact, say so rather than guessing. Never invent figures or references.

Set it once and every answer improves, without you repeating yourself. If you use AI daily, this is worth fifteen minutes of thought.

4. Maximum output length — why answers stop mid-sentence

A cap on how many tokens can be generated. When it is reached, the answer simply stops — often mid-sentence, which people mistake for a malfunction.

If that happens: raise the limit, or ask for the answer in parts ("give me sections 1–3, then continue"). The second approach also produces better structure.

5. Which model you are talking to

Inside most products there is a model selector, and it is the biggest lever in the entire interface. The same tool with a small model and with a large one behaves like two different products.

- **Fast/small** for tidying, rewriting, extracting, routine replies
- **Careful/large** for analysis, planning, anything with several steps
- **Vision** whenever you have an image rather than text

Most people never touch this and then judge the whole tool by whichever model it defaulted to.

What is mostly noise

Worth saying plainly, because interface clutter creates the impression that expertise is needed:

- **Presence and frequency penalties** — tiny effects; ignore them
- **Top-p / nucleus sampling** — an alternative randomness control; leave it alone if temperature is set
- **Themes, avatars, response styles** — cosmetic
- **"Creativity" sliders** with no explanation — usually temperature under another name

A starting configuration worth trying

TASK	TEMPERATURE	MODEL	CONTEXT	NOTES
Extracting figures from a document	0–0.2	Careful	Large enough for the whole document	Ask it to list its sources
Writing or editing prose	0.6–0.7	Either	Medium	Give the audience and length
Brainstorming names or angles	0.9–1.1	Fast is fine	Small	Ask for twenty, then pick
Calculations and logic	0–0.2	Careful	Medium	Ask for the working
Reading an image	default	Vision	n/a	Crop to the relevant part

> **Keep this idea:** temperature for honesty, context for attention, system instructions for consistency, model choice for capability. Everything else is decoration.

TRY THIS

Set a system instruction in whatever assistant you use, using the example above as a model. Then run the same task you did yesterday. The difference in tone and reliability is the cheapest upgrade available to you.

MODULE 8

Put it together

Reading about this produces opinions. Doing one real task produces judgment. This section is one task, end to end, using everything from the earlier sections.

The capstone

Pick one. Any of them will teach the same lessons, so choose the one you would actually use.

1. **Household budget.** A table with categories, planned and actual columns, and a total that adds up. Ask for the arithmetic to be shown, then check it.
2. **Complaint letter.** Something you have genuinely been putting off. 150 words, firm but polite, with dates and facts.
3. **Study or revision plan.** Four weeks, specific topics, slots per week, realistic about your available hours.
4. **One-page business summary.** Your idea, your customer, what it costs, what it earns. Plain language, no jargon.
5. **Translate and explain a form.** Take a document you do not fully understand and get both a translation and a plain-language explanation of each field.
6. **A simple web page.** Describe what you do, ask for a single HTML file, open it in a browser.

How to do it properly

1. **Set your system instruction first** (section 7). Who you are, plain language, no invented figures.
2. **Start a fresh conversation.** Not a continuation of something else.
3. **Give the material**, not a description of the material.
4. **Ask for a file or a table**, with the columns or format you want.
5. **Ask it to show its working** wherever a number appears.
6. **Check the factual parts yourself.** Numbers, dates, names, quotes.
7. **Iterate twice.** "Tighten this" and "remove anything that does not add information" are the two instructions that improve almost any draft.
8. **Save the output as a file on your own disk.** Not only inside the tool.

Then write this down

One paragraph, for yourself:

> What did it do well? What did it get wrong? What did I have to correct? Would I trust this with something that mattered?

That paragraph is the actual outcome of the course. Everything before it was context for this judgment.

A note on why this course exists

One last thing, and it is short.

Everything useful in this course can be done on a computer most people already own. But the capable versions of these tools are, almost without exception, rented — a monthly subscription, an account, a connection, a company's permission to keep serving your country. That quietly turns a general capability into something available mainly to people who can afford to keep paying.

A small team of us are building a local-first alternative: everything on your own machine, offline, no account, and a one-off price instead of a subscription — on the order of the price of a coffee rather than the price of a new computer. It is called **AISDA ZENN**, and it is not finished.

If that sounds useful, you can follow it here: **aisda.ai**

And if you never do, this course still did its job. You now know what these systems are, what they cost, and how to get real work out of them — which is the part that stays true no matter which product you end up using.

MODULE A

Appendix

Glossary

Agent — software that gives a model a goal and lets it take several steps towards it, checking results as it goes.

Context window — how much text a model can consider at once, measured in tokens. Once something falls outside it, the model cannot see it.

Embedding — a model's representation of text as numbers, used to find things by meaning rather than exact words. The mechanism behind searching your own documents.

Fine-tuning — further training of an existing model on your own examples. Different from prompting, and rarely necessary for personal use.

GGUF — a common file format for open models you run on your own computer.

Hallucination — fluent, confident output that is factually wrong. Not a malfunction but the natural consequence of predicting plausible text.

Harness — the software wrapped around a model that provides tools, steps, memory and checking.

Inference — running a trained model to get an answer. What happens when you use it, as opposed to training it.

Mixture of experts (MoE) — a model built from several sub-models where only part runs for each token. Large memory requirement, faster than its size suggests.

Parameter — one of the learned numbers that shape a model's output. "8B" means eight billion of them.

Prompt — what you send to the model.

Quantisation — compressing a model to use fewer bits per parameter so it fits on ordinary hardware. "Q4" is roughly four bits, the usual compromise.

RAG (retrieval-augmented generation) — giving the model passages retrieved from your documents before it answers, so it works from your material rather than its memory.

System instruction — standing guidance applied to every conversation.

Temperature — the randomness setting. Low for facts, high for ideas.

Token — the unit of text a model processes, roughly three-quarters of a word.

Appendix: running one yourself, in about twenty minutes

You do not need this to understand the course. You need it to feel the difference between talking about local AI and using it.

What you need: a computer with 8 GB of memory or more, and about 5 GB of free disk space.

1. **Install a local model runner.** Desktop applications exist that bundle the engine with a chat interface — the two most widely used are LM Studio and Ollama. Choose one; both run models you download.
2. **Download a small model.** Look for something in the 3–8 billion parameter range in a Q4 version. It will be roughly 2–5 GB.
3. **Load it and ask something.** Expect a few words per second on a processor-only machine, faster if you have a graphics card.
4. **Now disconnect the internet.** Turn off Wi-Fi entirely and ask another question. It still answers. Nothing left your computer, and no account was involved.
5. **Notice the speed.** This is the honest trade: complete privacy and no ongoing cost, in exchange for being slower than a rented data centre. Whether that is acceptable depends entirely on what you are doing.

If it feels slow: check how much memory is free, use a smaller model, and read section 4 again — memory speed, not processor speed, sets the pace.

Appendix: troubleshooting the eight most common problems

"It made up a fact." Expected. Ask for sources and check them, or supply the facts yourself and ask it to work with them.

"The answer got cut off." You hit the output limit. Ask for it in parts, or raise the limit.

"It forgot what I said earlier." The conversation outgrew the context window. Start a new chat and paste the essential context in.

"It keeps giving the same wrong answer." Lower the temperature, or ask it to state its assumptions explicitly and check those.

"The arithmetic is wrong." Models predict numbers rather than compute them. Ask for the working, or use a tool that calculates.

"It refuses something harmless." Safety filters misfire on ordinary requests. Rephrase concretely and explain the legitimate purpose.

"The local model is painfully slow." Usually too little free memory, or a model too large for the machine. Use a smaller parameter count or a lower quantisation.

"It sounds like marketing." It is mirroring the average of what it has read. Say plainly: "no marketing language, no exclamation marks, short sentences."

Where to go next

The most useful next step is not another course. It is picking one recurring task of yours and doing it with AI for two weeks, keeping a note of what works. By the end you will know more about your own use case than any general guide can tell you.

If you want to go deeper on any single section, the honest advice is to search for the term from the glossary plus the word "explained" — and to be sceptical of any explanation that never mentions a limitation.