PROMPT ENGINEERING BASICS

Prompt Engineering Basics - 2-page cheat sheet

Free course companion - key patterns, failure fixes, and ready-to-paste templates.

The mental model

- **Predict, don't query:** A language model predicts text, one chunk at a time: given everything written before, what most plausibly comes next? (Lesson 1)
- **Improv actor:** Think of it as an improv actor who read the internet and never breaks character. The shape of your prompt determines the shape of the answer.
- **No hidden brain:** It has no goals, no memory of you between conversations, and no knowledge past training cutoff except what your tool feeds it.
- **Vague = average:** Vague in, vague out: with no constraints, the most likely continuation is the most average one. Bland output is probability, not laziness.

RTF - the skeleton of a solid prompt

- **Role:** Who the model should be. Concrete beats grandiose: 'a patient math tutor for 8th graders', not 'the world's best expert'. Sets voice and standard.
- **Task:** What to do, with context baked in. One task per prompt beats three - the model continues one document better than it juggles three.
- **Format:** How to deliver it. The highest-leverage line: 'reply in a table with columns X, Y, Z' or 'exactly 5 bullets, each under 20 words'.

Diagnose vagueness: 4 questions

- **Audience:** A busy hiring manager? A 12-year-old? Your future self?
- **Task:** Summarize? Compare? Draft? Rank?
- **Format:** Bullets? Table? Word cap?
- **What to avoid:** Tone, length, topics to exclude?

Context: what the model can and can't know

- **Task context:** Goal, audience, stakes - what you're doing and why.
- **Domain context:** Jargon, background, the document or data at hand.
- **Format context:** How you want the answer shaped.

Rule: the model should never have to guess anything you already know. Most failed prompts lack the first two and over-supply the third ('make it pop!').

Few-shot: teach by example

- **Show, don't describe:** Give 2-3 input/output examples before the real request. The single biggest quality jump for beginners - examples carry tone, format, length, detail level.
- **Quality:** Representative beats perfect: same format, same input kind, same quality bar. Two great examples beat five mediocre ones.
- **Most important example LAST:** The model weights recent pattern most heavily.

Constraints & negative instructions

- **Specific:** Specific and checkable: 'under 150 words', 'exactly 3 options', 'assume the reader knows Python'. Give it a ruler, not an adjective.
- **Negatives work weakly:** Mentioning 'don't be salesy' plants 'salesy' in the pattern.
- **Pair every don't with a do:** Not just 'no buzzwords' but 'no buzzwords - use plain verbs like cut, built, shipped'. The positive gives it somewhere to go.

Chain-of-thought reasoning

- **Basic:** 'Think through this step by step, then give your final answer.' Each generated step becomes context for the next.
- **Named steps beat generic:** 'First list the facts. Then identify the formula. Then compute. Then sanity-check.' Prevents skipping the hard part.
- **Two cautions:** It helps reasoning, not recall - steps won't conjure facts the model lacks. Shown work can sound rigorous and be wrong: verify load-bearing answers independently.

The 6 prompt failures - and fixes

- **1. Empty chair:** Assumes knowledge the model lacks ('well obviously I meant...'). Fix: add the missing context.
- **2. Triple task:** 'Summarize, critique, and rewrite for LinkedIn.' Fix: split into three prompts, or name an order.
- **3. Format vacuum:** No specified shape -> wall of text. Fix: name the format. The single fastest upgrade in prompting.
- **4. Example gap:** Described a style ('professional but warm') instead of showing one. Fix: add 1-2 examples.
- **5. Moving target:** Answers swing wildly; prompt contradicts itself ('be brief but thorough'). Fix: resolve it - pick one - or ask the model what's ambiguous.
- **6. Wrong tool:** Asking a text predictor for verified facts, real-time data, precise computation. Fix: use retrieval/search; check arithmetic yourself.

Don't re-roll a bad prompt hoping for luck. Name the failure, apply the fix, run once more. Debugging prompts is a skill, not a lottery.

Iterate like an experiment

- **One variable at a time:** Variables: role, task description, context, format, constraints, examples, reasoning instructions. Pick the one most likely responsible for the defect; change only it; re-run.

- **Keep a tiny log:** Three columns in a notes app: prompt version / what changed / what improved / what got worse. Prevents 'it was better two versions ago but I can't remember what changed.'

- **Build a playbook:** After a month, your log tells you which fixes work for YOUR tasks - worth more than any generic guide.

Meta-prompting: ask the model to critique your prompt

```
Critique this prompt like a prompt-engineering
instructor. List: 1) ambiguities a model would
have to guess at, 2) contradictions between
instructions, 3) missing context, 4) the single
highest-leverage improvement. Then rewrite the
prompt incorporating your fixes.
```

- **Judge the critique:** The model sometimes 'fixes' things that weren't broken, adds bloat, or misunderstands intent. Take its structural catches; keep your intent. You stay the editor.

Build a prompt library

- **Name:** What it does - 'Cold email v3'.

- **The prompt itself:** The full working text.

- **When to use:** One line - 'first outreach to warm leads'.

- **Notes:** What you learned - 'v2 added the 120-word cap after the rambling incident'. This field is the whole point.

Ten working prompts you trust beat a hundred bookmarked ones you don't. Start with 10 you actually use.

Project: research assistant workflow

- **1. Brief:** 'Write a 1-page research brief on [topic]: the 5 questions I need answered, in/out of scope, what a good answer looks like.'

- **2. Gather:** 'Find current, credible information on [question]. Prefer primary sources. For each key claim, note the source.'

- **3. Synthesize:** Paste findings: 'Synthesize into [memo/table/report]. Flag where sources disagree; mark single-source claims tentative.'

- **4. Summarize:** 'Distill into a 5-bullet executive summary plus the 3 most important caveats.'

Project: writing & editing pipeline

- **1. Draft:** 'Draft [piece] for [audience] about [topic]. Aim for [length]. Prioritize ideas - I'll handle polish later.' Deliberately low bar.

- **2. Critique:** 'Critique like a ruthless editor: weakest paragraph, claims needing evidence, boring spots, structural problems. Do NOT rewrite yet.' Diagnosis before treatment.

- **3. Revise:** 'Revise addressing critique points [2 and 4]. Keep what works.' You choose which criticisms to accept.

- **4. Polish:** 'Final pass: cut 20%, fix rhythm, check opening and closing lines. Show final only.'

Ready-to-paste templates

```
ROLE: You are [concrete role, e.g. a senior
copy editor allergic to fluff].
TASK: [one job] for [audience]. Context: [paste
the relevant background - never make it guess].
FORMAT: [exact shape, e.g. a table with columns
X, Y, Z; each cell under 15 words].
CONSTRAINTS: [checkable limits; pair each don't
with a do].
```

Next step after this course: AI Agents 101 (free) - what changes when the model takes actions, uses tools, and remembers. Agents are built from prompts; every agent failure is a prompting failure in a fancier hat.