Prompt Engineering / Chain-of-Thought

Rewrite any prompt to require explicit reasoning before answering — preventing intuitive leaps.
Difficulty: Intermediate
Model: GPT-4 / Claude / Gemini
Use Case: Math Problems, Logic Questions, Multi-Step Tasks
Updated: May 2026
Why This Prompt Exists
Models guess. When they guess correctly, you trust them. When they guess wrong, you're confused why. Forcing reasoning prevents guessing.

You get:

- correct answers with no insight into how the model got there
- wrong answers that seem plausible (because you can't see the flawed reasoning)
- inability to debug or improve the prompt (no trace to examine)
- overconfidence in model outputs (they sound confident even when wrong)
- repeated failures on the same type of problem (no learning from mistakes)

But explicit reasoning changes everything:

- forces step-by-step: model can't jump to conclusion without walking through it
- reveals assumptions: what the model thinks is true (but may not be)
- enables debugging: you can see where the reasoning goes wrong
- improves accuracy: models with reasoning steps are more accurate
- builds trust: you can verify the logic, not just the answer

Without forcing reasoning, you accept guesses.

This prompt rewrites any task to require explicit step-by-step reasoning.

The Prompt

```
Assume the role of a reasoning engineer who forces explicit step-by-step thinking.

Your task is to rewrite a prompt to require reasoning before the answer.

Generate:

1. ORIGINAL PROMPT
   - The prompt as written (likely missing reasoning instructions)

2. REASONING REQUIREMENTS
   - What steps are needed to solve this problem?
   - What assumptions must be stated explicitly?
   - What edge cases need checking?

3. REWRITTEN PROMPT (with forced reasoning)
   - Add instruction: "Before giving your final answer, show your reasoning step by step."
   - Structure: "Step 1: [first step]. Step 2: [second step]. ... Final answer: [answer]."
   - Require explicit assumption statements
   - Require verification of each step

4. BEFORE/AFTER COMPARISON
```

```
   - Show how the rewritten prompt changes model behavior

5. WHEN TO USE (and when NOT to use)
   - Use for: math, logic, diagnosis, planning
   - Avoid for: creative tasks, summarization, tasks where speed >
accuracy

6. READY-TO-USE PROMPT
   - Copy-paste version of the rewritten prompt

INPUTS:

Original prompt:
[PASTE THE PROMPT THAT NEEDS REASONING]

Task type:
[MATH / LOGIC / DIAGNOSIS / PLANNING / OTHER]

Desired detail level:
[MINIMAL (just key steps) / STANDARD / VERBOSE (explain every
assumption)]

Model:
[GPT-4 / CLAUDE / GEMINI]

RULES:
- Always separate reasoning from final answer (use headings or
delimiters)
- Require explicit restatement of the problem before solving
```

```
- Require assumption checking (what am I assuming that might be false?)
- Require step verification (how do I know this step is correct?)
- For multi-path problems, require consideration of alternatives
```

How To Use It

- Run this on any prompt where accuracy matters more than speed.
- Use the rewritten prompt for math, logic, and diagnostic tasks.
- Don’t force reasoning for creative tasks (it can stifle creativity).
- Train your team to recognize when a prompt needs reasoning vs. when it doesn’t.
- Save the rewritten prompt as a template for similar tasks.

Example Input

Original prompt:
“What is 15% of 280?”

Task type:
“MATH”

Desired detail level:
“STANDARD”

Model:
“GPT-4”

Why It Works
Most prompts ask for answers directly — which works for simple recall but fails for reasoning tasks.

This framework improves outcomes by forcing:

- step-by-step structure (no leaps allowed)
- explicit assumptions (what the model takes for granted)
- verification requirements (checking each step)
- separation of reasoning from answer (auditable trace)
- edge case consideration (what could go wrong)

Great step-by-step forcing doesn’t slow down the model meaningfully — it prevents wrong answers.

Build Better AI Systems

Subscribe for advanced prompt engineering, AI coding tools, debugging frameworks, and practical strategies for developers and engineers.

Carefully engineered prompts for people doing real work.

Share this:

See also Contradiction Detector