

Prompt Engineering Patterns Cheat Sheet

The Four Pillars of a Prompt

1 • Context What the AI needs to know before it can help you. Sets the scene, domain, and persona. <i>"You are reviewing a Python REST API for a fintech startup."</i>	2 • Instruction The specific action you want performed. Be direct and unambiguous. <i>"Identify all security vulnerabilities and explain each one."</i>	3 • Constraints Boundaries on format, length, tone, and scope. Prevents over-generation. <i>"Limit to 5 issues. Use plain English, no jargon."</i>	4 • Format How the output should be structured for downstream use or readability. <i>"Return a numbered list with: Issue, Risk Level, Fix."</i>
---	--	---	--

Five Core Patterns

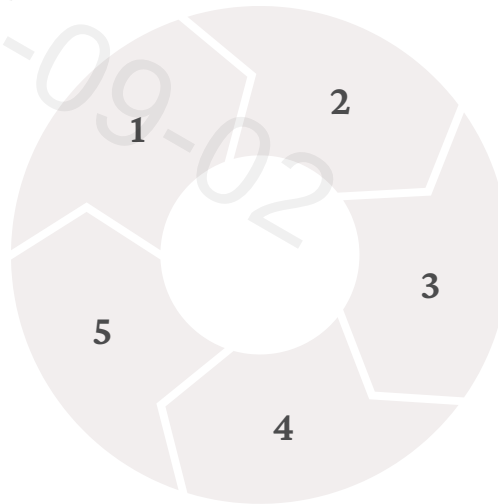
Pattern	When to Use	Example Phrase
Role-Based	Always — sets expertise and tone	"You are a senior {role}..."
Task Decomposition	Complex, multi-step tasks	Break into sequential sub-prompts
Structured Outputs	When results must be parseable	"Return JSON with keys: ..."
Few-Shot	Stylistic or nuanced tasks	Include 2–5 examples before the ask
Chain-of-Thought	Debugging or analysis	"Think step by step..."

 Prompts are code. Template them, version them, share them.

The Prompt Engineering Loop

1. Draft
Write your initial prompt using the Four Pillars as a checklist.

5. Templatize
Once stable, extract into a reusable template. Version and share it.



2. Run

Execute against your model. Capture the full output before judging.

3. Evaluate

Does the output meet your spec? Check format, accuracy, and completeness.

4. Refine

Adjust one variable at a time — context, constraint, or format.

Prompt Debugging Table

Symptom	Likely Fix
Output is too long / rambling	Add a Constraint: word limit or item count
Wrong tone or expertise level	Strengthen the Role: specify seniority and domain
Output not parseable / wrong structure	Use Structured Output pattern with explicit schema
Misses nuance or style	Add Few-Shot examples that demonstrate the target style
Reasoning errors or wrong conclusions	Invoke Chain-of-Thought: "Think step by step before answering"
Ignores part of the task	Use Task Decomposition: split into sequential sub-prompts

Anti-Patterns to Avoid

→ **✗ One giant prompt for everything** — Monolithic prompts degrade quality. Decompose complex tasks into focused sub-prompts with clear handoffs.

→ **✗ Assuming AI knows your context** — The model has no memory of your codebase, team conventions, or prior sessions. Always supply context explicitly.

→ **✗ Accepting first output without refinement** — First outputs are drafts. Run the Prompt Engineering Loop — evaluate and refine before treating output as final.

→ **✗ Copy-pasting AI output without review** — You own the output. Review for correctness, security implications, and alignment with your actual requirements.