Prompting and Prompt Engineering

Prompt structure (system/user/assistant roles), zero-shot and few-shot prompting, effective prompt-writing practices, and an introduction to common techniques like chain-of-thought.

Intermediate 20 min

Last updated: August 20, 2026
Read online: https://www.learnforgex.com/en/topics/prompting-and-prompt-engineering

---

Prompting and Prompt Engineering

The last two lessons established the mechanism: an LLM's behavior at inference time is shaped entirely by its context (see "A First Look at Context" in "How Large Language Models Work"), and that context has real, finite limits (see "Tokens and Context Windows"). This lesson covers the practical skill of deliberately constructing that context to get a reliable, useful response -- commonly called **prompting**, and, when done systematically, **prompt engineering**. Since a prompt is nothing more than text placed into a model's context, everything here is really a direct, practical application of "In-Context Learning: How a Model Uses What You Give It" -- there's no new mechanism to learn, only a more deliberate way of using the one you already know.

What Is a Prompt?

A **prompt** is the text you give an LLM to produce a response -- an instruction, a question, a piece of text to transform, or some combination of these. Recall from "A First Look at Context" that context is the *only* channel for shaping an LLM's output at inference time; a prompt is simply the part of that context that you, the person or system using the model, directly compose and control (as opposed to background information a system might also insert, which "Tools & MCP" covers later in this course). A well-written prompt isn't a magic phrase -- it's a clear, complete specification of the task, written in a way that gives the model's in-context learning (see "How Large Language Models Work") enough to work with.

The Structure of a Prompt: Roles

Most modern LLM interactions aren't a single undifferentiated block of text -- they're structured into distinct **roles**, each labeled so the model can tell them apart:

- **System prompt:** instructions that set the model's overall behavior, persona, or constraints for the whole conversation -- for example, "You are a helpful assistant that only answers questions about cooking." Typically set once, by the application, not by the end user.
- **User prompt:** the actual request or question from whoever is interacting with the model in a given turn.
- **Assistant (model) response:** the model's own previous replies, which, in a multi-turn conversation, become part of the context for the next turn (a direct consequence of "In-Context Learning: How a Model Uses What You Give It" -- the model re-reads its own prior output as part of the conversation history).

Separating these roles lets a system establish behavior once (system) that persists across many different user requests (user), which is far more reliable than trying to repeat the same instructions inside every single user message.

Zero-Shot, Few-Shot, and Examples in Prompts

A **zero-shot** prompt asks the model to perform a task with instructions alone, no examples -- "Translate this sentence to French." A **few-shot** prompt includes a small number of worked examples of the task before the actual request, letting the model's in-context learning pick up the desired pattern, format, or style directly from those examples rather than from a description of them. Few-shot prompting tends to help most when the desired output has a specific format or style that's hard to fully describe in words but easy to demonstrate -- for example, showing two or three examples of the exact JSON structure you want back is often more reliable than describing that structure in a paragraph of instructions.

> If a zero-shot prompt keeps producing output in the wrong format or style, reach for a couple of well-chosen examples before writing longer and longer instructions -- showing is frequently more reliable than describing, precisely because of how in-context learning actually works.

Writing Effective Prompts

A few concrete practices consistently improve how reliably a model does what you actually want:

- **Be specific about the task and the desired output.** "Summarize this" is far more likely to produce an inconsistent result than "Summarize this in exactly three bullet points, focused on financial figures."
- **Give the model the information it actually needs.** Recall from "How Large Language Models Work" that anything not in pretraining or context is simply unavailable to the model -- if a task depends on specific facts, include them, don't assume the model already has them.

- **Specify the output format explicitly** when a particular structure matters (a list, a table description in prose since this platform avoids markdown tables, a specific JSON shape) -- don't leave format to chance if your downstream code depends on it.
- **Break a complex task into smaller, explicit steps** inside the prompt itself, rather than asking for a complicated multi-part result all at once -- this tends to produce more reliable results than one large, ambiguous instruction.

Prompt Engineering as an Iterative Process

Prompt engineering is the practice of systematically writing, testing, and refining prompts, rather than treating the first draft as final. In practice this looks like: write a prompt, run it against a handful of realistic inputs, look closely at where the output goes wrong, and revise the prompt to address that specific failure -- then repeat. This is closer to debugging or iterative software development than to one-time writing, and it's worth treating it that way: keep track of what changed between prompt versions and why, the same way you'd track changes to any other part of a system you're building.

Common Prompting Techniques

A few named patterns come up often enough to be worth knowing by name:

- **Chain-of-thought prompting:** asking the model to reason step by step before giving a final answer (for example, "think through this step by step, then give your final answer"), which tends to improve accuracy on tasks that require several logical steps -- essentially giving the model more of its own reasoning as additional context to condition its final answer on (see "In-Context Learning: How a Model Uses What You Give It").
- **Role or persona prompting:** asking the model to answer as if it were a specific kind of expert ("as an experienced security engineer, review this code") -- this shifts the *style and focus* of the response toward patterns associated with that role in the model's training data, though it does not grant the model any new knowledge it didn't already have.
- **Providing constraints explicitly:** stating length limits, things to avoid, or required elements directly in the prompt, rather than hoping the model infers them -- constraints stated in context are far more reliable than constraints left implicit.

Best Practices

- Treat prompt writing as an iterative process (see "Prompt Engineering as an Iterative Process"), not a one-shot task -- test against realistic inputs, not just the one example that happens to work.
- Give the model everything it needs directly in the prompt rather than assuming it can infer missing facts (see "Writing Effective Prompts") -- recall that context is the only channel available at inference time.

- Reach for few-shot examples (see "Zero-Shot, Few-Shot, and Examples in Prompts") when a format or style is hard to describe precisely but easy to demonstrate.

Common Mistakes

- **Assuming a longer, more elaborate prompt is always a better prompt.** Precision and relevance matter far more than length -- an unfocused prompt can bury the actual instruction in irrelevant detail, and every extra token still costs money and time (recall "Tokens and Context Windows").
- **Writing a paragraph of instructions to describe a format that could be shown with one or two examples.** As "Zero-Shot, Few-Shot, and Examples in Prompts" explained, demonstrating is often more reliable than describing.
- **Confusing role or persona prompting with actually granting new knowledge.** As "Common Prompting Techniques" noted, asking a model to respond "as an expert" changes tone and style, not the underlying facts available to it -- it still only knows what's in pretraining or context.

Summary, Cheat Sheet, and Glossary

Summary

- A **prompt** is the text a user or system composes to shape an LLM's response -- a direct, practical application of in-context learning.
- Prompts are commonly structured into **roles**: system (overall behavior), user (the request), and assistant (prior responses, which become part of context in multi-turn conversations).
- **Zero-shot** prompts give instructions alone; **few-shot** prompts include worked examples, which often communicate format and style more reliably than descriptions.
- Effective prompts are specific, supply all needed information, specify output format explicitly, and break complex tasks into explicit steps.
- **Prompt engineering** treats prompt writing as an iterative, test-and-refine process rather than a one-time draft.
- Chain-of-thought prompting, role/persona prompting, and explicit constraints are common named techniques for shaping model output.

Cheat Sheet

- Prompt = text that shapes a model's response, placed directly into context.
- System prompt = overall behavior. User prompt = the request. Assistant = prior responses (context in later turns).
- Zero-shot = instructions only. Few-shot = instructions + worked examples.
- Chain-of-thought = ask the model to reason step by step first.
- Prompt engineering = iterate: write, test, observe failures, revise.

Glossary

- **Prompt:** the text given to an LLM to produce a response.
- **Prompt engineering:** the practice of systematically writing, testing, and refining prompts.
- **System prompt / User prompt / Assistant response:** the three common roles used to structure an LLM conversation.
- **Zero-shot prompting:** giving a model instructions with no worked examples.
- **Few-shot prompting:** including a small number of worked examples in a prompt to demonstrate the desired pattern, format, or style.
- **Chain-of-thought prompting:** asking a model to reason step by step before producing a final answer.
- **Role / persona prompting:** asking a model to respond as if it were a specific kind of expert or character.