infozion

GPT-5 PROMPTING GUIDE

Aug 20, 2025

BY INFOZION TECHNOLOGIES

www.infozion.in

GPT-5 prompting guide

GPT-5, our newest flagship model, represents a substantial leap forward in agentic task performance, coding, raw intelligence, and steerability.

While we trust it will perform excellently “out of the box” across a wide range of domains, in this guide we’ll cover prompting tips to maximize the quality of model outputs, derived from our experience training and applying the model to real-world tasks. We discuss concepts like improving agentic task performance, ensuring instruction adherence, making use of newly API features, and optimizing coding for frontend and software engineering tasks - with key insights into AI code editor Cursor’s prompt tuning work with GPT-5.

We’ve seen significant gains from applying these best practices and adopting our canonical tools whenever possible, and we hope that this guide, along with the prompt optimizer tool we’ve built, will serve as a launchpad for your use of GPT-5. But, as always, remember that prompting is not a one-size-fits-all exercise - we encourage you to run experiments and iterate on the foundation offered here to find the best solution for your problem.

Agentic workflow predictability

We trained GPT-5 with developers in mind: we've focused on improving tool calling, instruction following, and long-context understanding to serve Agentic workflow predictability as the best foundation model for agentic applications. If adopting GPT-5 for agentic and tool calling flows, we recommend upgrading to the Responses API, where reasoning is persisted between tool calls, leading to more efficient and intelligent outputs

Controlling agentic eagerness

Agentic scaffolds can span a wide spectrum of control—some systems delegate the vast majority of decision-making to the underlying model, while others keep the model on a tight leash with heavy programmatic logical branching. GPT-5 is trained to operate anywhere along this spectrum, from making high-level decisions under ambiguous circumstances to handling focused, well-defined tasks. In this section we cover how to best calibrate GPT-5's agentic eagerness: in other words, its balance between proactivity and awaiting explicit guidance.

Prompting for less eagerness

GPT-5 is, by default, thorough and comprehensive when trying to gather context in an agentic environment to ensure it will produce a correct answer. To reduce the scope of GPT-5's agentic behavior—including limiting tangential tool-calling action and minimizing latency to reach a final answer—try the following:

- Switch to a lower reasoning_effort . This reduces exploration depth but improves efficiency and latency. Many workflows can be accomplished with consistent results at medium or even low reasoning_effort .

- Define clear criteria in your prompt for how you want the model to explore the problem space. This reduces the model's need to explore and reason about too many ideas:

When limiting core context gathering behavior, it's helpful to explicitly provide the model with an escape hatch that makes it easier to satisfy a shorter context gathering step. Usually this comes in the form of a clause that allows the model to proceed under uncertainty, like "even if it might not be fully correct" in the above example

If you're willing to be maximally prescriptive, you can even set fixed tool call budgets, like the one below. The budget can naturally vary based on your desired search depth.

Prompting for more eagerness

On the other hand, if you'd like to encourage model autonomy, increase tool-calling persistence, and reduce occurrences of clarifying questions or otherwise handing back to the user, we recommend increasing reasoning_effort , and using a prompt like the following to encourage persistence and thorough task completion:

Generally, it can be helpful to clearly state the stop conditions of the agentic tasks, outline safe versus unsafe actions, and define when, if ever, it's acceptable for the model to hand back to the user. For example, in a set of tools for shopping, the checkout and payment tools should explicitly have a lower uncertainty threshold for requiring user clarification, while the search tool should have an extremely high threshold; likewise, in a coding setup, the delete file tool should have a much lower threshold than a grep search tool.

Tool preambles

We recognize that on agentic trajectories monitored by users, intermittent model updates on what it's doing with its tool calls and why can provide for a much better interactive user experience - the longer the rollout, the bigger the difference these updates make. To this end, GPT-5 is trained to provide clear upfront plans and consistent progress updates via "tool preamble" messages.

You can steer the frequency, style, and content of tool preambles in your prompt—from detailed explanations of every single tool call to a brief upfront plan and everything in between. This is an example of a highquality preamble prompt:

Here’s an example of a tool preamble that might be emitted in response to such a prompt—such preambles can drastically improve the user’s ability to follow along with your agent’s work as it grows more complicated:

Reasoning effort

We provide a reasoning_effort parameter to control how hard the model thinks and how willingly it calls tools; the default is medium , but you should scale up or down depending on the difficulty of your task. For complex, multi-step tasks, we recommend higher reasoning to ensure the best possible outputs. Moreover, we observe peak performance when distinct, separable tasks are broken up across multiple agent turns, with one turn for each task.

Reusing reasoning context with the responses API

We strongly recommend using the Responses API when using GPT-5 to unlock improved agentic flows, lower costs, and more efficient token usage in your applications.

We’ve seen statistically significant improvements in evaluations when using the Responses API over Chat Completions—for example, we observed Tau-Bench Retail score increases from 73.9% to 78.2% just by switching to the Responses API and

and including previous_response_id to pass back previous reasoning items into subsequent requests. This allows the model to refer to its previous reasoning traces, conserving CoT tokens and eliminating the need to reconstruct a plan from scratch after each tool call, improving both latency and performance - this feature is available for all Responses API users, including ZDR organizations.

Maximizing coding performance,from planning to execution

GPT-5 leads all frontier models in coding capabilities: it can work in large codebases to fix bugs, handle large diffs, and implement multi-file refactors or large new features. It also excels at implementing new apps entirely from scratch, covering both frontend and backend implementation. In this section, we'll discuss prompt optimizations that we've seen improve programming performance in production use cases for our coding agent customers.

Frontend app development

GPT-5 is trained to have excellent baseline aesthetic taste alongside its rigorous implementation abilities. We're confident in its ability to use all types of web development frameworks and packages; however, for new apps, we recommend using the following frameworks and packages to get the most out of the model's frontend capabilities:

- **Frameworks:** Next.js (TypeScript), React, HTML
- **Styling / UI:** Tailwind CSS, shadcn/ui, Radix Themes
- **Icons:** Material Symbols, Heroicons, Lucide
- **Animation:** Motion
- **Fonts:** San Serif, Inter, Geist, Mona Sans, IBM Plex Sans, Manrope

Zero-to-one app generation

GPT-5 is excellent at building applications in one shot. In early Maximizing coding performance,from planning to execution Frontend app development Zero-to-one app generation experimentation with the model, users have found that prompts like the one below—asking the model to iteratively execute against selfconstructed excellence rubrics—improve output quality by using GPT-5's thorough planning and self-reflection capabilities.

Matching codebase design standard

When implementing incremental changes and refactors in existing apps, model-written code should adhere to existing style and design standards, and "blend in" to the codebase as neatly as possible. Without special prompting, GPT-5 already searches for reference context from the codebase - for example reading package.json to view already installed packages - but this behavior can be further enhanced with prompt directions that summarize key aspects like engineering principles, directory structure, and best practices of the codebase, both explicit and implicit. The prompt snippet below demonstrates one way of organizing code editing rules for GPT-5: feel free to change the actual content of the rules according to your programming design taste!

Collaborative coding in production: Cursor's GPT-5 prompt tuning

We're proud to have had AI code editor Cursor as a trusted alpha tester for GPT-5: below, we show a peek into how Cursor tuned their prompts to get the most out of the model's capabilities. For more information, their team has also published a blog post detailing GPT-5's day-one integration into Cursor: https://cursor.com/blog/gpt-5

System prompt and parameter tuning

Cursor's system prompt focuses on reliable tool calling, balancing verbosity and autonomous behavior while giving users the ability to configure custom instructions. Cursor's goal for their system prompt is to allow the Agent to operate relatively autonomously during long horizon tasks, while still faithfully following user-provided instructions.

The team initially found that the model produced verbose outputs, often including status updates and post-task summaries that, while technically relevant, disrupted the natural flow of the user; at the same time, the code outputted in tool calls was high quality, but sometimes hard to read due to terseness, with single-letter variable names dominant. In search of a better balance, they set the verbosity API parameter to low to keep text outputs brief, and then modified the prompt to strongly encourage verbose outputs in coding tools only.

This dual usage of parameter and prompt resulted in a balanced format combining efficient, concise status updates and final work summary with much more readable code diffs.

Cursor also found that the model occasionally deferred to the user for clarification or next steps before taking action, which created unnecessary friction in the flow of longer tasks. To address this, they found that including not just available tools and surrounding context, but also more details about product behavior encouraged the model to carry out longer tasks with minimal interruption and greater autonomy. Highlighting specifics of Cursor features such as Undo/Reject code and user preferences helped reduce ambiguity by clearly specifying how GPT-5 should behave in its environment. For longer horizon tasks, they found this prompt improved performance:

Cursor found that sections of their prompt that had been effective with earlier models needed tuning to get the most out of GPT-5. Here is one example below:

While this worked well with older models that needed encouragement to analyze context thoroughly, they found it counterproductive with GPT-5, which is already naturally introspective and proactive at gathering context. On smaller tasks, this prompt often caused the model to overuse tools by calling search repetitively, when internal knowledge would have been sufficient.

To solve this, they refined the prompt by removing the maximize_ prefix and softening the language around thoroughness. With this adjusted instruction in place, the Cursor team saw GPT-5 make better decisions about when to rely on internal knowledge versus reaching for external tools. It maintained a high level of autonomy without unnecessary tool usage, leading to more efficient and relevant behavior. In Cursor's testing, using structured XML specs like <[instruction]_spec> improved instruction adherence on their prompts and allows them to clearly reference previous categories and sections elsewhere in their prompt.

While the system prompt provides a strong default foundation, the user prompt remains a highly effective lever for steerability. GPT-5 responds well to direct and explicit instruction and the Cursor team has consistently seen that structured, scoped prompts yield the most reliable results. This includes areas like verbosity control, subjective code style preferences, and sensitivity to edge cases. Cursor found allowing users to configure their own custom Cursor rules to be particularly impactful with GPT-5's improved steerability, giving their users a more customized experience.

Optimizing intelligence and instruction-following

Steering

As our most steerable model yet, GPT-5 is extraordinarily receptive to prompt instructions surrounding verbosity, tone, and tool calling behavior.

Verbosity

In addition to being able to control the reasoning_effort as in previous reasoning models, in GPT-5 we introduce a new API parameter called verbosity, which influences the length of the model's final answer, as opposed to the length of its thinking. Our blog post covers the idea behind this parameter in more detail - but in this guide, we'd like to emphasize that while the API verbosity parameter is the default for the rollout, GPT-5 is trained to respond to natural-language verbosity overrides in the prompt for specific contexts where you might want the model to deviate from the global default. Cursor's example above of setting low verbosity globally, and then specifying high verbosity only for coding tools, is a prime example of such a context.

Instruction following

Like GPT-4.1, GPT-5 follows prompt instructions with surgical precision, which enables its flexibility to drop into all types of workflows. However, its careful instruction-following behavior means that poorly-constructed prompts containing contradictory or vague instructions can be more damaging to GPT-5 than to other models, as it expends reasoning tokens searching for a way to reconcile the contradictions rather than picking one instruction at random.

Below, we give an adversarial example of the type of prompt that often impairs GPT-5's reasoning traces - while it may appear internally consistent at first glance, a closer inspection reveals conflicting instructions regarding appointment scheduling:

- Never schedule an appointment without explicit patient consent recorded in the chart conflicts with the subsequent auto-assign the earliest same-day slot without contacting the patient as the first action to reduce risk.

- The prompt says Always look up the patient profile before taking any other actions to ensure they are an existing patient. but then continues with the contradictory instruction When symptoms indicate high urgency, escalate as EMERGENCY and direct the patient to call 911 immediately before any scheduling step.

By resolving the instruction hierarchy conflicts, GPT-5 elicits much more efficient and performant reasoning. We fixed the contradictions by:

- Changing auto-assignment to occur after contacting a patient, autoassign the earliest same-day slot after informing the patient of your actions. to be consistent with only scheduling with consent.

- Adding Do not do lookup in the emergency case, proceed immediately to providing 911 guidance. to let the model know it is ok to not look up in case of emergency

We understand that the process of building prompts is an iterative one, and many prompts are living documents constantly being updated by different stakeholders - but this is all the more reason to thoroughly review them for poorly-worded instructions. Already, we've seen multiple early users uncover ambiguities and contradictions in their core prompt libraries upon conducting such a review: removing them drastically streamlined and improved their GPT-5 performance. We recommend testing your prompts in our prompt optimizer tool to help identify these types of issues.

Minimal reasoning

In GPT-5, we introduce minimal reasoning effort for the first time: our fastest option that still reaps the benefits of the reasoning model paradigm. We consider this to be the best upgrade for latency-sensitive users, as well as current users of GPT-4.1.

Perhaps unsurprisingly, we recommend prompting patterns that are similar to GPT-4.1 for best results. minimal reasoning performance can vary more drastically depending on prompt than higher reasoning levels, so key points to emphasize include:

1.Prompting the model to give a brief explanation summarizing its thought process at the start of the final answer, for example via a bullet point list, improves performance on tasks requiring higher intelligence.

2. Requesting thorough and descriptive tool-calling preambles that - For high-acuity Red and Orange cases, auto-assign the earliest same-day slot Minimal reasoning continually update the user on task progress improves performance in agentic workflows.

3. Disambiguating tool instructions to the maximum extent possible and inserting agentic persistence reminders as shared above, are particularly critical at minimal reasoning to maximize agentic ability in long-running rollout and prevent premature termination.

4. Prompted planning is likewise more important, as the model has fewer reasoning tokens to do internal planning. Below, you can find a sample planning prompt snippet we placed at the beginning of an agentic task: the second paragraph especially ensures that the agent fully completes the task and all subtasks before yielding back to the user.

Markdown formatting

By default, GPT-5 in the API does not format its final answers in Markdown, in order to preserve maximum compatibility with developers whose applications may not support Markdown rendering. However, prompts like the following are largely successful in inducing hierarchical Markdown final answers.

Occasionally, adherence to Markdown instructions specified in the system prompt can degrade over the course of a long conversation. In the event that you experience this, we've seen consistent adherence from appending a Markdown instruction every 3-5 user messages.

Meta prompting

Finally, to close with a meta-point, early testers have found great success using GPT-5 as a meta-prompter for itself. Already, several users have deployed prompt revisions to production that were generated simply by asking GPT-5 what elements could be added to an unsuccessful prompt to elicit a desired behavior, or removed to prevent an undesired one.

Appendix

SWE-Bench verified developer instructions

Agentic coding tool definitions

As shared in the GPT-4.1 prompting guide, here is our most updated apply_patch implementation: we highly recommend using apply_patch for file edits to match the training distribution. The newest implementation should match the GPT-4.1 implementation in the overwhelming majority of cases.

Taubench-Retail minimal reasoning instructions

Terminal-Bench prompt

Infozion Technologies LLP

infoziontechnologies

Infozion Technologies

www.infozion.in

+91-8604-37-4831

hi@infozion.in