Claude Docs

Prompt engineering > Extended thinking tips

Extended thinking tips

Copy page

This guide provides advanced strategies and techniques for getting the most out of Claude's extended thinking features. Extended thinking allows Claude to work through complex problems step-by-step, improving performance on difficult tasks.

See Extended thinking models for guidance on deciding when to use extended thinking.

Before diving in

This guide presumes that you have already decided to use extended thinking mode and have reviewed our basic steps on how to get started with extended thinking as well as our extended thinking implementation guide.

Technical considerations for extended thinking

- Thinking tokens have a minimum budget of 1024 tokens. We recommend that you start with the minimum thinking budget and incrementally increase to adjust based on your needs and task complexity.
- For workloads where the optimal thinking budget is above 32K, we recommend that you use batch processing to avoid networking issues. Requests pushing the model to think above 32K tokens causes long running requests that might run up against system timeouts and open connection limits.
- Extended thinking performs best in English, though final outputs can be in any language Claude supports.
- If you need thinking below the minimum budget, we recommend using standard mode, with thinking turned off, with traditional chain-of-thought prompting with XML tags (like `<thinking>`). See chain of thought prompting.

Prompting techniques for extended thinking

step instructions

Claude often performs better with high level instructions to just think deeply about a task rather than step-by-step prescriptive guidance. The model's creativity in approaching problems may exceed a human's ability to prescribe the optimal thinking process.

For example, instead of:

User

```
Think through this math problem step by step:
1. First, identify the variables
2. Then, set up the equation
3. Next, solve for x
...
```

Consider:

User

```
Please think about this math problem thoroughly and in great detail.
Consider multiple approaches and show your complete reasoning.
Try different methods if your first approach doesn't work.
```

Try in Console

That said, Claude can still effectively follow complex structured execution steps when needed. The model can handle even longer lists with more complex instructions than previous versions. We recommend that you start with more generalized instructions, then read Claude's thinking output and iterate to provide more specific instructions to steer its thinking from there.

Multishot prompting with extended thinking

Multishot prompting works well with extended thinking. When you provide Claude examples of how to think through problems, it will follow similar reasoning patterns within its extended thinking blocks.

You can include few-shot examples in your prompt in extended thinking scenarios by using XML tags like `<thinking>` or `<scratchpad>` to indicate canonical patterns of extended thinking in those examples.

best.

Example:

User

```
I'm going to show you how to solve a math problem, then I want you to solve a

Problem 1: What is 15% of 80?

<thinking>
To find 15% of 80:
1. Convert 15% to a decimal: 15% = 0.15
2. Multiply: 0.15 × 80 = 12
</thinking>

The answer is 12.

Now solve this one:
Problem 2: What is 35% of 240?
```

Try in Console

Maximizing instruction following with extended thinking

Claude shows significantly improved instruction following when extended thinking is enabled. The model typically:

1. Reasons about instructions inside the extended thinking block
2. Executes those instructions in the response

To maximize instruction following:

- Be clear and specific about what you want
- For complex instructions, consider breaking them into numbered steps that Claude should work through methodically
- Allow Claude enough budget to process the instructions fully in its extended thinking

Using extended thinking to debug and steer Claude's behavior

You can use Claude's thinking output to debug Claude's logic, although this method is not always perfectly reliable.

- We don't recommend passing Claude's extended thinking back in the user text block, as this doesn't improve performance and may actually degrade results.
- Prefilling extended thinking is explicitly not allowed, and manually changing the model's output text that follows its thinking block is likely going to degrade results due to model confusion.

When extended thinking is turned off, standard `assistant` response text prefill is still allowed.

ⓘ Sometimes Claude may repeat its extended thinking in the assistant output text. If you want a clean response, instruct Claude not to repeat its extended thinking and to only output the answer.

Making the best of long outputs and longform thinking

For dataset generation use cases, try prompts such as "Please create an extremely detailed table of..." for generating comprehensive datasets.

For use cases such as detailed content generation where you may want to generate longer extended thinking blocks and more detailed responses, try these tips:

- Increase both the maximum extended thinking length AND explicitly ask for longer outputs
- For very long outputs (20,000+ words), request a detailed outline with word counts down to the paragraph level. Then ask Claude to index its paragraphs to the outline and maintain the specified word counts

⚠ We do not recommend that you push Claude to output more tokens for outputting tokens' sake. Rather, we encourage you to start with a small thinking budget and increase as needed to find the optimal settings for your use case.

Here are example use cases where Claude excels due to longer extended thinking:

› **Complex STEM problems**

› **Constraint optimization problems**

Have Claude reflect on and check its work for improved consistency and error handling

You can use simple natural language prompting to improve consistency and reduce errors:

1. Ask Claude to verify its work with a simple test before declaring a task complete
2. Instruct the model to analyze whether its previous step achieved the expected result
3. For coding tasks, ask Claude to run through test cases in its extended thinking

Example:

User

```
Write a function to calculate the factorial of a number.
Before you finish, please verify your solution with test cases for:
- n=0
- n=1
- n=5
- n=10
And fix any issues you find.
```

Try in Console

Next steps

Extended thinking cookbook ↗

Explore practical examples of extended thinking in our cookbook.

</>

Extended thinking guide

See complete technical documentation for implementing extended thinking.

Claude Docs

Solutions

AI agents

Code modernization

Coding

Customer support

Education

Financial services

Government

Life sciences

Partners

Amazon Bedrock

Google Cloud's Vertex AI

Learn

Blog

Catalog

Courses

Use cases

Connectors

Customer stories

Engineering at Anthropic

Events

Powered by Claude

Service partners

Startups program

Company

Anthropic

Careers

Economic Futures

Research

News

Transparency

Help and security

Availability

Status

Support

Discord

Terms and policies

Privacy policy

Responsible disclosure policy

Terms of service: Commercial

Terms of service: Consumer

Usage policy