Veo 3 is now available in the Gemini API!

**Learn more** (https://developers.googleblog.com/en/veo-3-now-available-gemini-api/)

# Prompt design strategies

*Prompt design* is the process of creating prompts, or natural language requests, that elicit accurate, high quality responses from a language model.

This page introduces basic concepts, strategies, and best practices to get you started designing prompts to get the most out of Gemini AI models.

## Topic-specific prompt guides

Looking for more specific prompt strategies? Check out our other prompting guides on:

- Prompting with media files (/gemini-api/docs/files#prompt-guide)

- Prompting for image generation (/gemini-api/docs/image-generation#imagen-prompt-guide)

- Prompting for video generation (/gemini-api/docs/video#prompt-guide)

Google AI Studio also hosts a prompt gallery (https://aistudio.google.com/app/gallery) meant to interactively showcase many of the concepts shared in this guide.

## Clear and specific instructions

An effective and efficient way to customize model behavior is to provide it with clear and specific instructions. Instructions can be in the form of a question, step-by-step tasks, or as complex as mapping out a user's experience and mindset.

### Input

Input is the required text in the prompt that you want the model to provide a response to. Inputs can be a question that the model answers (question input), a task the model performs (task input), an entity the model operates on (entity input), or partial input that the model completes or continues (completion input).

| Input type | Prompt | Generated output |
|---|---|---|
| Question | What's a good name for a flower shop that specializes in selling bouquets of<br>dried flowers? Create a list of 5 options with just the names. | Here are 10 names for a flower shop specializing in dried flowers:<br><br>1. Everlasting Blooms<br>2. Dried & Delightful<br>3. The Petal Preserve<br>4. Whispers of Wildflowers<br>5. Forever Blooms |
| Task | Give me a simple list of just the things that I must bring on a camping trip. The list should have 5 items. | 1. **Tent:** Your shelter from the elements.<br>2. **Sleeping Bag:** Stay warm and comfortable.<br>3. **Sleeping Pad:** Insulation and cushioning for a good night's sleep.<br>4. **Headlamp or Flashlight:** Light for nighttime activities.<br>5. **First Aid Kit:** Be prepared for minor injuries. |
| Entity | Classify the following items as [large, small]:<br>Elephant<br>Mouse<br>Snail | Here's the classification:<br><br>* **Elephant:** Large<br>* **Mouse:** Small<br>* **Snail:** Small |

## Partial input completion

Generative language models work like an advanced auto completion tool. When you provide partial content, the model can provide the rest of the content or what it thinks is a continuation of that content as a response. When doing so, if you include any examples or context, the model can take those examples or context into account.

The following example provides a prompt with an instruction and an entity input:

:

```
he given order, return a JSON object that has the fields cheeseburger, hamburger, fri
:, with the value being the quantity.

': A burger and a drink.
```

```
se:
```

```
seburger": 0,
urger": 1,
s": 0,
k": 1
```

(gemini-2.

While the model did as prompted, writing out the instructions in natural language can sometimes be challenging and it leaves a lot to the model's interpretation. For example, a restaurants menu might contain many items. To reduce the size of the JSON response, you probably want to omit the items that weren't ordered. In this case, you can give an example and a response prefix (#prefixes) and let the model complete it:

```
:
```

```
I fields are cheeseburger, hamburger, fries, and drink.
': Give me a cheeseburger and fries
it:
```

```
seburger": 1,
s": 1
```

```
': I want two burgers, a drink, and fries.
it:
```

```
se:
```

```
urger": 2,
k": 1,
s": 1
```

(gemini-2.

Notice how "cheeseburger" was excluded from the output because it wasn't a part of the order.

## Constraints

Specify any constraints on reading the prompt or generating a response. You can tell the model what to do and not to do. For example, you can specify a constraint in the prompt on how long you want a summary to be:

```
:
```

```
rize this text in one sentence:
  A quantum computer exploits quantum mechanical phenomena to perform calculations
entially
r than any modern traditional computer. At very tiny scales, physical matter acts as
cles and as waves, and quantum computing uses specialized hardware to leverage this
ior.
perating principles of quantum devices is beyond the scope of classical physics. When
yed
ale, quantum computers could be used in a wide variety of applications such as: in
security to break existing encryption methods while helping researchers create new or

rology to develop better weather forecasting etc. However, the current state of the a
um
ters are still largely experimental and impractical.
```

```
se:
```

```
iting quantum mechanical phenomena, quantum computers can perform calculations
entially faster than traditional computers for potential applications like cybersecur
```

eteorology, although they are currently largely experimental and impractical.

(gemini-2.

## Response format

You can give instructions that specify the format of the response. For example, you can ask for the response to be formatted as a table, bulleted list, elevator pitch, keywords, sentence, or paragraph. The following system instruction tells the model to be more conversational in its response:

instruction

juestions should be answered comprehensively with details, unless the user requests a
.se response specifically.

is a smart way to make a business that sells DVD's in 2025?

se:

:ing a business selling DVDs in 2025 presents significant challenges due to the
/helming dominance of digital streaming and on-demand services; a "smart" approach wou
:fore require a highly specialized niche, a strong value proposition beyond just the c
.ikely a focus on community or collector markets rather than mainstream consumption.

s a breakdown of what a smart DVD business in 2025 would likely entail:

:treme Niche Specialization: You cannot compete with streaming libraries. Your busines
 focus on what digital doesn't offer easily or at all. This could include:

:-of-Print & Rare Titles: Sourcing films or TV shows that have never been released
:ally or are currently unavailable on major streaming platforms.

(gemini-2.

## Format responses with the completion strategy

The completion strategy (#completion) can also help format the response. The following example
prompts the model to create an essay outline:

:

:e an outline for an essay about hummingbirds.

se:


is a possible outline for an essay about hummingbirds:

itroduction
ok: Start with a captivating fact or description about hummingbirds (e.g., their spee
scent colors, or tiny size).
ickground: Briefly introduce hummingbirds — where they are found (Americas), their
ation as unique birds.
iesis Statement: State the main point of the essay, which could be about the remarkabl
ations, behaviors, and ecological importance of hummingbirds.

ppearance and Identification


(gemini-2.



The prompt didn't specify the format for the outline and the model chose a format for you. To get the
model to return an outline in a specific format, you can add text that represents the start of the
outline and let the model complete it based on the pattern that you initiated.


:



e an outline for an essay about hummingbirds.
itroduction



se:


 here is an outline for an essay about hummingbirds, starting from the introduction y
ded:

itroduction
k: Start with a captivating fact or description about hummingbirds (e.g., their speec
scent colors, or tiny size).
kground: Briefly introduce hummingbirds — where they are found (Americas), their
ation as unique birds, and their general characteristics.
sis Statement: State the main point of the essay, focusing on the remarkable adaptati

iors, and ecological significance that make hummingbirds extraordinary.

(gemini-2.

# Zero-shot vs few-shot prompts

You can include examples in the prompt that show the model what getting it right looks like. The model attempts to identify patterns and relationships from the examples and applies them when generating a response. Prompts that contain a few examples are called *few-shot* prompts, while prompts that provide no examples are called *zero-shot* prompts. Few-shot prompts are often used to regulate the formatting, phrasing, scoping, or general patterning of model responses. Use specific and varied examples to help the model narrow its focus and generate more accurate results.

We recommend to always include few-shot examples in your prompts. Prompts without few-shot examples are likely to be less effective. In fact, you can remove instructions from your prompt if your examples are clear enough in showing the task at hand.

The following zero-shot prompt asks the model to choose the best explanation.

```
:
```

```
;e choose the best explanation to the question:

:ion: How is snow formed?
mation1: Snow is formed when water vapor in the air freezes into ice
als in the atmosphere, which can combine and grow into snowflakes as they
 through the atmosphere and accumulate on the ground.
mation2: Water vapor freezes into ice crystals forming snow.
·r:
```

```
se:
```

```
mation1 is the better explanation because it provides more detail on the
·ss, including how ice crystals combine and grow into snowflakes as they
 through the atmosphere.
```

(gemini-2.

If your use case requires the model to produce concise responses, you can include examples in the prompt that give preference to concise responses.

The following prompt provides two examples that show preference to the shorter explanations. In the response, you can see that the examples guided the model to choose the shorter explanation (`Explanation2`) as opposed to the longer explanation (`Explanation1`) like it did previously.

(gemini-2.

:

are some examples showing a question, explanation, and answer format:

ion: Why is the sky blue?
ination1: The sky appears blue because of Rayleigh scattering, which causes
er blue wavelengths of light to be scattered more easily than longer red
engths, making the sky look blue.
ination2: Due to Rayleigh scattering effect.
r: Explanation2

ion: What is the cause of earthquakes?
ination1: Sudden release of energy in the Earth's crust.
ination2: Earthquakes happen when tectonic plates suddenly slip or break
;, causing a release of energy that creates seismic waves that can shake the
id and cause damage.
r: Explanation1

Answer the following question given the example formats above:

ion: How is snow formed?
ination1: Snow is formed when water vapor in the air freezes into ice
als in the atmosphere, which can combine and grow into snowflakes as they
through the atmosphere and accumulate on the ground.
ination2: Water vapor freezes into ice crystals forming snow.
r:

se:

r: Explanation2

(gemini-2.

## Optimal number of examples

Models like Gemini can often pick up on patterns using a few examples, though you may need to experiment with the number of examples to provide in the prompt for the best results. At the same time, if you include too many examples, the model may start to overfit (https://developers.google.com/machine-learning/glossary#overfitting) the response to the examples.

## Patterns vs anti patterns

Using examples to show the model a pattern to follow is more effective than using examples to show the model an anti pattern to avoid.

⛔ **Negative pattern:**

```
e



: end haikus with a question:
ι are fun
ιrt and simple poem
: you enjoy them?
```

✅ **Positive pattern:**

```
e



ιs end haikus with an assertion:
ι are fun
ιrt and simple poem
ι to write
```

## Consistent formatting

Make sure that the structure and formatting of few-shot examples are the same to avoid responses with undesired formats. One of the primary objectives of adding few-shot examples in prompts is to show the model the response format. Therefore, it is essential to ensure a consistent format across all examples, especially paying attention to XML tags, white spaces, newlines, and example splitters.

## Add context

You can include instructions and information in a prompt that the model needs to solve a problem, instead of assuming that the model has all of the required information. This contextual information helps the model understand the constraints and details of what you're asking for it to do.

The following example asks the model to give troubleshooting guidance for a router:

```
:


 should I do to fix my disconnected wifi? The light on my Google
 router is yellow and blinking slowly.
```

```
se:


low blinking light on your Google Wifi router usually indicates a problem
 the internet connection. Here are some steps to troubleshoot and fix the
:
start your router and modem:
lug your modem from the power outlet and wait for 30 seconds.
g your modem back in and wait for it to fully boot up.
lug your Google Wifi router and wait for 30 seconds.
g your Google Wifi router back in and wait for it to fully boot up.
```

(gemini-2.

The response looks like generic troubleshooting information that's not specific to the router or the status of the LED indicator lights.

To customize the response for the specific router, you can add to the prompt the router's troubleshooting guide as context for it to refer to when providing a response.

```
:



r the question using the text below. Respond with only the text provided.
ion: What should I do to fix my disconnected wifi? The light on my Google Wifi router
w and blinking slowly.


: Slowly pulsing yellow
```

```
it means: There is a network error.
to do:
 that the Ethernet cable is connected to both your router and your modem and both dev
urned on. You might need to unplug and plug in each device again.


: Fast blinking yellow
it means: You are holding down the reset button and are factory resetting this device
to do:
u keep holding down the reset button, after about 12 seconds, the light will turn sol
w. Once it is solid yellow, let go of the factory reset button.


: Solid yellow
it means: Router is factory resetting.
to do:
can take up to 10 minutes. When it's done, the device will reset itself and start pul
, letting you know it's ready for setup.


: Solid red
it means: Something is wrong.
to do:
cal failure. Factory reset the router. If the light stays red, contact Wifi customer
rt.



se:



 that the Ethernet cable is connected to both your router and your modem and both dev
urned on. You might need to unplug and plug in each device again.
```

(gemini-2.

# Add prefixes

A prefix is a word or phrase that you add to the prompt content that can serve several purposes,
depending on where you put the prefix:

- **Input prefix:** Adding a prefix to the input signals semantically meaningful parts of the input to
  the model. For example, the prefixes "English:" and "French:" demarcate two different
  languages.

- **Output prefix:** Even though the output is generated by the model, you can add a prefix for the
  output in the prompt. The output prefix gives the model information about what's expected as

a response. For example, the output prefix "JSON:" signals to the model that the output should be in JSON format.

- **Example prefix:** In few-shot prompts, adding prefixes to the examples provides labels that the model can use when generating the output, which makes it easier to parse output content.

In the following example, "Text:" is the input prefix and "The answer is:" is the output prefix.

```
:
```

```
ify the text as one of the following categories.
ge
ll
  Rhino
nswer is: large
  Mouse
nswer is: small
  Snail
nswer is: small
  Elephant
nswer is:
```

```
se:
```

```
nswer is: large
```

(gemini-2.

## Break down prompts into components

For use cases that require complex prompts, you can help the model manage this complexity by breaking things down into simpler components.

1. **Break down instructions:** Instead of having many instructions in one prompt, create one prompt per instruction. You can choose which prompt to process based on the user's input.

2. **Chain prompts:** For complex tasks that involve multiple sequential steps, make each step a prompt and chain the prompts together in a sequence. In this sequential chain of prompts, the

output of one prompt in the sequence becomes the input of the next prompt. The output of the last prompt in the sequence is the final output.

3. **Aggregate responses:** Aggregation is when you want to perform different parallel tasks on different portions of the data and aggregate the results to produce the final output. For example, you can tell the model to perform one operation on the first part of the data, perform another operation on the rest of the data and aggregate the results.

## Experiment with model parameters

Each call that you send to a model includes parameter values that control how the model generates a response. The model can generate different results for different parameter values. Experiment with different parameter values to get the best values for the task. The parameters available for different models may differ. The most common parameters are the following:

1. **Max output tokens:** Specifies the maximum number of tokens that can be generated in the response. A token is approximately four characters. 100 tokens correspond to roughly 60-80 words.

2. **Temperature:** The temperature controls the degree of randomness in token selection. The temperature is used for sampling during response generation, which occurs when `topP` and `topK` are applied. Lower temperatures are good for prompts that require a more deterministic or less open-ended response, while higher temperatures can lead to more diverse or creative results. A temperature of 0 is deterministic, meaning that the highest probability response is always selected.

3. `topK`: The `topK` parameter changes how the model selects tokens for output. A `topK` of 1 means the selected token is the most probable among all the tokens in the model's vocabulary (also called greedy decoding), while a `topK` of 3 means that the next token is selected from among the 3 most probable using the temperature. For each token selection step, the `topK` tokens with the highest probabilities are sampled. Tokens are then further filtered based on `topP` with the final token selected using temperature sampling.

4. `topP`: The `topP` parameter changes how the model selects tokens for output. Tokens are selected from the most to least probable until the sum of their probabilities equals the `topP` value. For example, if tokens A, B, and C have a probability of 0.3, 0.2, and 0.1 and the `topP` value is 0.5, then the model will select either A or B as the next token by using the temperature and exclude C as a candidate. The default `topP` value is 0.95.

5. `stop_sequences`: Set a stop sequence to tell the model to stop generating content. A stop sequence can be any sequence of characters. Try to avoid using a sequence of characters that

may appear in the generated content.

# Prompt iteration strategies

Prompt design can sometimes require a few iterations before you consistently get the response you're looking for. This section provides guidance on some things you can try when iterating on your prompts:

1. **Use different phrasing:** Using different words or phrasing in your prompts often yields different responses from the model even though they all mean the same thing. If you're not getting the expected results from your prompt, try rephrasing it.

   ```
   Version 1:
   How do I bake a pie?

   Version 2:
   Suggest a recipe for a pie.

   Version 3:
   What's a good pie recipe?
   ```

2. **Switch to an analogous task:** If you can't get the model to follow your instructions for a task, try giving it instructions for an analogous task that achieves the same result.

   This prompt tells the model to categorize a book by using predefined categories:

   Prompt:

   ```
   Which category does The Odyssey belong to:
   thriller
   sci-fi
   mythology
   biography
   ```

   Response:

```
The Odyssey belongs to the category of **mythology**.

Here's why:

* **Mythology:** The Odyssey tells the story of Odysseus, a hero from Greek
mythology, and his
journey home after the Trojan War. It features gods, monsters, and supernatural
events common to
Greek mythology.
.....
```

(gemini-2.5-flash)

The response is correct, but the model didn't stay within the bounds of the options. You also want to model to just respond with one of the options instead of in a full sentence. In this case, you can rephrase the instructions as a multiple choice question and ask the model to choose an option.

Prompt:

```
Multiple choice problem: Which of the following options describes the book The
Odyssey?
Options:
```

- thriller

- sci-fi

- mythology

- biography

Response:

```
The correct answer is mythology.
```

<div align="right">(gemini-2.5-flash)</div>

3. **Change the order of prompt content:** The order of the content in the prompt can sometimes affect the response. Try changing the content order and see how that affects the response.

```
Version 1:
[examples]
[context]
[input]

Version 2:
[input]
[examples]
[context]

Version 3:
[examples]
[input]
[context]
```

# Fallback responses

A fallback response is a response returned by the model when either the prompt or the response triggers a safety filter. An example of a fallback response is "I'm not able to help with that, as I'm only a language model."

If the model responds with a fallback response, try increasing the temperature.

# Things to avoid

- Avoid relying on models to generate factual information.

- Use with care on math and logic problems.

# Generative models under the hood

This section aims to answer the question - ***Is there randomness in generative models' responses, or are they deterministic?***

The short answer - yes to both. When you prompt a generative model, a text response is generated in two stages. In the first stage, the generative model processes the input prompt and generates a **probability distribution** over possible tokens (words) that are likely to come next. For example, if you prompt with the input text "The dog jumped over the ... ", the generative model will produce an array of probable next words:

```
[("fence", 0.77), ("ledge", 0.12), ("blanket", 0.03), ...]
```

This process is deterministic; a generative model will produce this same distribution every time it's input the same prompt text.

In the second stage, the generative model converts these distributions into actual text responses through one of several decoding strategies. A simple decoding strategy might select the most likely token at every timestep. This process would always be deterministic. However, you could instead choose to generate a response by *randomly sampling* over the distribution returned by the model. This process would be stochastic (random). Control the degree of randomness allowed in this decoding process by setting the temperature. A temperature of 0 means only the most likely tokens are selected, and there's no randomness. Conversely, a high temperature injects a high degree of randomness into the tokens selected by the model, leading to more unexpected, surprising model responses.

# Next steps

- Now that you have a deeper understanding of prompt design, try writing your own prompts using Google AI Studio (http://aistudio.google.com).

- To learn about multimodal prompting, see Prompting with media files (/gemini-api/docs/files#prompt-guide).

- To learn about image prompting, see the Imagen prompt guide (/gemini-api/docs/image-generation#imagen-prompt-guide)

- To learn about video prompting, see the Veo prompt guide (/gemini-api/docs/video#prompt-guide)

Last updated 2025-04-28 UTC.