

# Chapter 1. The Five Principles of Prompting

---

## A NOTE FOR EARLY RELEASE READERS

With Early Release ebooks, you get books in their earliest form—the authors' raw and unedited content as they write—so you can take advantage of these technologies long before the official release of these titles.

This will be the 1st chapter of the final book. Please note that the GitHub repo will be made active later on.

If you have comments about how we might improve the content and/or examples in this book, or if you notice missing material within this chapter, please reach out to the author at [ccollins@oreilly.com](mailto:ccollins@oreilly.com).

---

*Prompt engineering* is the process of discovering prompts which reliably yield useful or desired results.

A *prompt* is the input you provide, typically text, when interfacing with an AI model like ChatGPT or Midjourney. The prompt serves as a set of instructions the model uses to predict the desired response: text from *Large Language Models* (LLMs) like [ChatGPT](#), or images from *diffusion models* like [Midjourney](#).

Here is a simple example of a prompt input for a product name generator (inspired by one of [OpenAI's examples](#)), and the resulting output from ChatGPT.

Input:

Can I have a list of product names for a pair of shoes that can fit any foot size?

Output:

Certainly! Here's a list of product names for a pair of shoes that can fit any foot size:

```
UniFit SoleStrides  
FlexiSize All-Fit Shoes  
OmniFit FootFlex  
...  
AnyStep AdaptiWalkers  
FlexiRange UniversalSteps  
PerfectGlide MultiFit Shoes
```

This is already a remarkable response for a naive prompt, which feels like magic because we got here with very little effort. As the state-of-the-art models improve, the likelihood you will get *good enough* results on your first try goes up. For any throwaway interactions with an AI, where you don't plan to do the same task again, the naive approach is all you need.

However, if you planned to put this prompt into production, you'd benefit from investing more work into getting it right. Mistakes cost you money in terms of the fees OpenAI charges based on the length of the prompt and response, as well as the time spent fixing mistakes. If you were building a product name generator with thousands of users, there are some obvious issues you'd want attempt to fix:

- *Vague Direction*: You're not briefing the AI on what style of name you want, or what attributes it should have. Do you want a single word or a concatenation? Can the words be made up or is it important that they're in real english? Do you want the AI to emulate somebody you admire who is famous for great product names?
- *Unformatted Output*: You're getting back a list of separated names line-by-line, of unspecified length. When you run this prompt multi-

ple times you'll see sometimes it comes back with a numbered list, and often it has text at the beginning which makes it hard to parse programmatically.

- *Missing Examples:* You haven't given the AI any examples of what *good* names look like. It's autocompleting using an average of its training data, i.e. the entire internet (with all its inherent bias), but is that what you want? Ideally you'd feed it examples of successful names, common names in an industry, or even just other names you like.
- *Limited Evaluation:* You have no consistent or scalable way to define which names are good or bad, so you have to manually review each response. If you can institute a rating system or other form of measurement, you can optimize the prompt to get better results, and identify how many times it fails.
- *No Task Division:* You're asking a lot of a single prompt here: there are lots of factors that go into product naming, and this important task is being naively outsourced to the AI all in one go, with no task specialization or visibility into how it's handling this task for you.

Addressing these problems is the basis for the core principles we use throughout this book. There are many different ways to ask an AI model to do the same task, and even slight changes can make a big difference. LLMs work by continuously predicting the next token (approximately 3/4 of a word), starting from what was in your prompt. Each new token is selected based on its probability of appearing next, with an element of randomness (controlled by the *temperature* parameter). As demonstrated in [Figure 1-1](#), the word *kick* had a lower probability of coming after the start of the name *OneSize* (0.02%), where a more predictable response would be *Shoes* (88.91%).

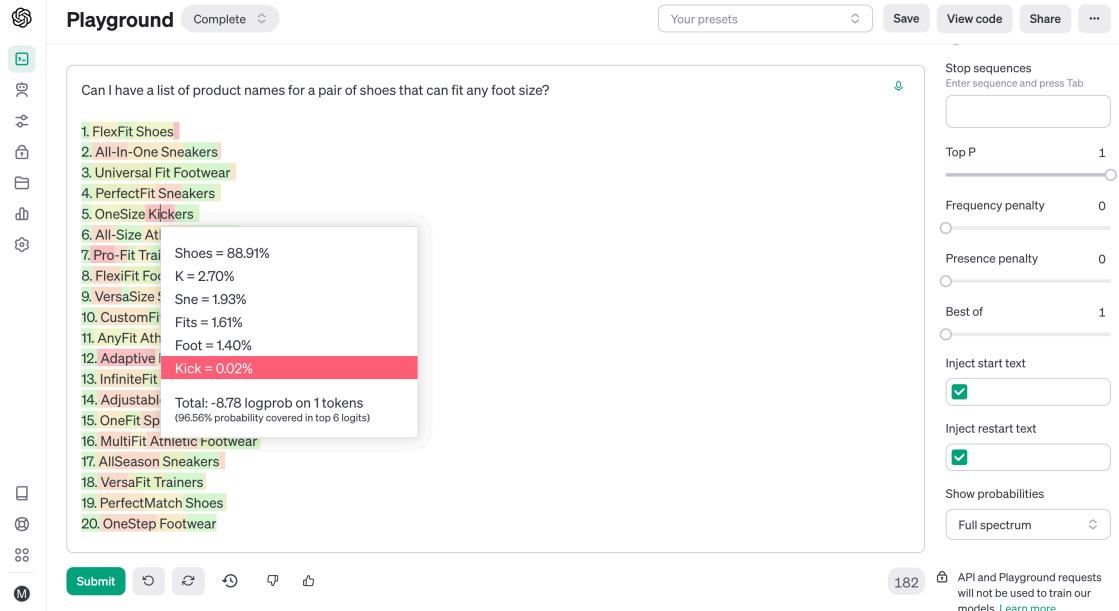


Figure 1-1. How the response breaks down into tokens

LLMs are trained on essentially the entire text of the internet, and are then further fine-tuned to give helpful responses. Average prompts will return average responses, leading some to be underwhelmed when their results don't live up to the hype. What you put in your prompt changes the probability of every word generated, so it matters a great deal to the results you'll get. These models have seen the best and worst of what humans have produced, and are capable of emulating almost anything if you know the right way to ask. OpenAI charges based on the number of tokens used in the prompt and the response, and so prompt engineers need to make these tokens count, by optimizing prompts for cost, quality, and reliability.

Here's the same example with the application of several prompt engineering techniques. We give ask for names in the style of Steve Jobs, state that we want a comma separated list, and supply examples of the task done well:

Input:

Brainstorm a list of product names for a shoe that fits any foot size, in the style of Steve Jobs.

Return the results as a comma separated list, in this format:  
Product description: A shoe that fits any foot size

```
Product names: [list of 3 product names]
```

```
## Examples
```

```
Product description: A refrigerator that dispenses beer
```

```
Product names: iBarFridge, iFridgeBeer, iDrinkBeerFridge
```

```
Product description: A watch that can tell accurate time in space
```

```
Product names: iNaut, iSpace, iTime
```

```
Product description: A home milkshake maker
```

```
Product names: iShake, iSmoothie, iShake Mini
```

Output:

```
Product description: A shoe that fits any foot size
```

```
Product names: iFitFoot, iPerfectFit, iShoeSize
```

While no prompt is ever perfect, this prompt is optimized to reliably deliver solid product names in the right format. The user of your product name generator can choose somebody other than Steve Jobs to get the types of names they like, they can change the response format if needed, and the output of this prompt can become the input of another. Finally, you could periodically update the examples you use in the prompt based on user feedback, making your system smarter over time.

## Overview of the Five Principles of Prompting

The process for optimizing this prompt follows the *Five Principles of Prompting*, which we will dissect using this example in the remainder of this chapter, and recall throughout the book. They map exactly to the five issues we raised when discussing the naive text prompt. You'll find references back to these principles throughout the rest of the book, to help you connect the dots to how they're used in practice. The Five Principles of Prompting are as follows:

1. *Give Direction*: Describe the desired style in detail, or reference a relevant persona.
2. *Specify Format*: Define what rules to follow, and the required structure of the response.
3. *Provide Examples*: Insert a diverse set of test cases where the task was done correctly.
4. *Evaluate Quality*: Identify errors and rate responses, testing what drives performance.
5. *Divide Labor*: Split tasks into multiple steps, chained together for complex goals.

These principles are not short-lived *tips* or *hacks* but are generally accepted conventions that are useful for working with any level of intelligence, biological or artificial. These principles are model-agnostic, and should work to improve your prompt no matter which generative text or image model you're using. We first published these principles in July 2022 in the blog post ["Prompt Engineering: From Words to Art and Copy"](#), and they have stood the test of time, including mapping quite closely to OpenAI's own [Prompt engineering guide](#), which came a year later. Anyone who works closely with generative AI models is likely to converge on a similar set of strategies for solving common issues, and throughout this book you'll see hundreds of demonstrative examples of how they can be useful for improving your prompts.

We have provided downloadable one-pagers for text and image generation you can use as a checklist when applying these principles. These were created for our popular Udemy course [The Complete Prompt Engineering for AI Bootcamp](#) (50,000+ students), which was based on the same principles, but with different material to this book.

- [Text Generation One-Pager](#)
- [Image Generation One-Pager](#)

To show these principles apply equally well to prompting image models, let's use the following example, and explain how to apply each of the Five Principles of Prompting to this specific scenario. Copy and paste the entire input prompt into the Midjourney Bot in Discord, including the link

to the image at the beginning, after typing /imagine to trigger the prompt box to appear (requires a free [Discord](#) account, and a paid [Midjourney](#) account).

Input:

```
https://s.mj.run/TKAsyhNiKmc stock photo of business meeting  
of four people watching on white MacBook on top of glass-top  
table, Panasonic, DC-GH5
```

The output is shown in [Figure 1-2](#).



*Figure 1-2. Stock photo of business meeting*

This prompt takes advantage of Midjourney's ability to take a base image as an example, by uploading the image to Discord then copy and pasting

the URL into the prompt (<https://s.mj.run/TKAsyhNiKmc>), for which the royalty-free image from Unsplash is used ([Figure 1-3](#)). If you run into an error with the prompt, try uploading the image yourself and reviewing [Midjourney's documentation](#) for any formatting changes.



*Figure 1-3. Photo by Mimi Thian on Unsplash*

Let's compare this well-engineered prompt to what you get back from Midjourney if you naively asked for a stock photo in the simplest way possible. [Figure 1-4](#) shows an example of what you get without prompt engineering, an image with a darker, more stylistic take on a stock photo than you'd typically expect.

Input:

```
people in a business meeting
```

The output is shown in [Figure 1-4](#).



Figure 1-4. People in a business meeting

Although less prominent an issue in v5 of Midjourney onwards, community feedback mechanisms (when users select an image to resize to a higher resolution, that choice may be used to train the model) have reportedly biased the model towards a *fantasy* aesthetic, which is less suitable for the stock photo use case. The early adopters of Midjourney came from the digital art world and naturally gravitated towards fantasy and sci-fi styles, which can be reflected in the results from the model even when this aesthetic is not suitable.

Throughout the book the examples used will be compatible with ChatGPT Plus (GPT-4) as the text model, and Midjourney v6 or Stable Diffusion XL as the image model, though we will specify if it's important. These foundational models are the current state-of-the-art and are good at a diverse range of tasks. The principles are intended to be future-proof as much as is possible, so if you're reading this book when GPT-5,

Midjourney v7 or Stable Diffusion XXL is out, or if you're using another vendor like Google, everything you learn here should still prove useful.

## 1. Give Direction

One of the issues with the naive text prompt, discussed earlier, was that it wasn't briefing the AI on what *types* of product names you wanted. To some extent, naming a product is a subjective endeavor, and without giving the AI an idea of what names you like, it has a low probability of guessing right.

By the way, a human would also struggle to complete this task without a good *brief*, which is why creative and branding agencies require a detailed briefing on any task from their clients.

---

**TIP**

Although it's not a perfect mapping, it can be helpful to imagine what context a human might need for this task, and try including it in the prompt.

---

In the example prompt we gave direction through the use of *role-playing*, in that case emulating the style of Steve Jobs, who was famous for iconic naming products. If you change this aspect of the prompt someone else who is famous in the training data (as well as matching the examples to the right style), you'll get dramatically different results:

Input:

```
Brainstorm a list of product names for a shoe that fits any foot size, in the style of Elon Musk.
```

```
Return the results as a comma separated list, in this format:  
Product description: A shoe that fits any foot size  
Product names: [list of 3 product names]
```

```
## Examples
```

```
Product description: A refrigerator that dispenses beer
```