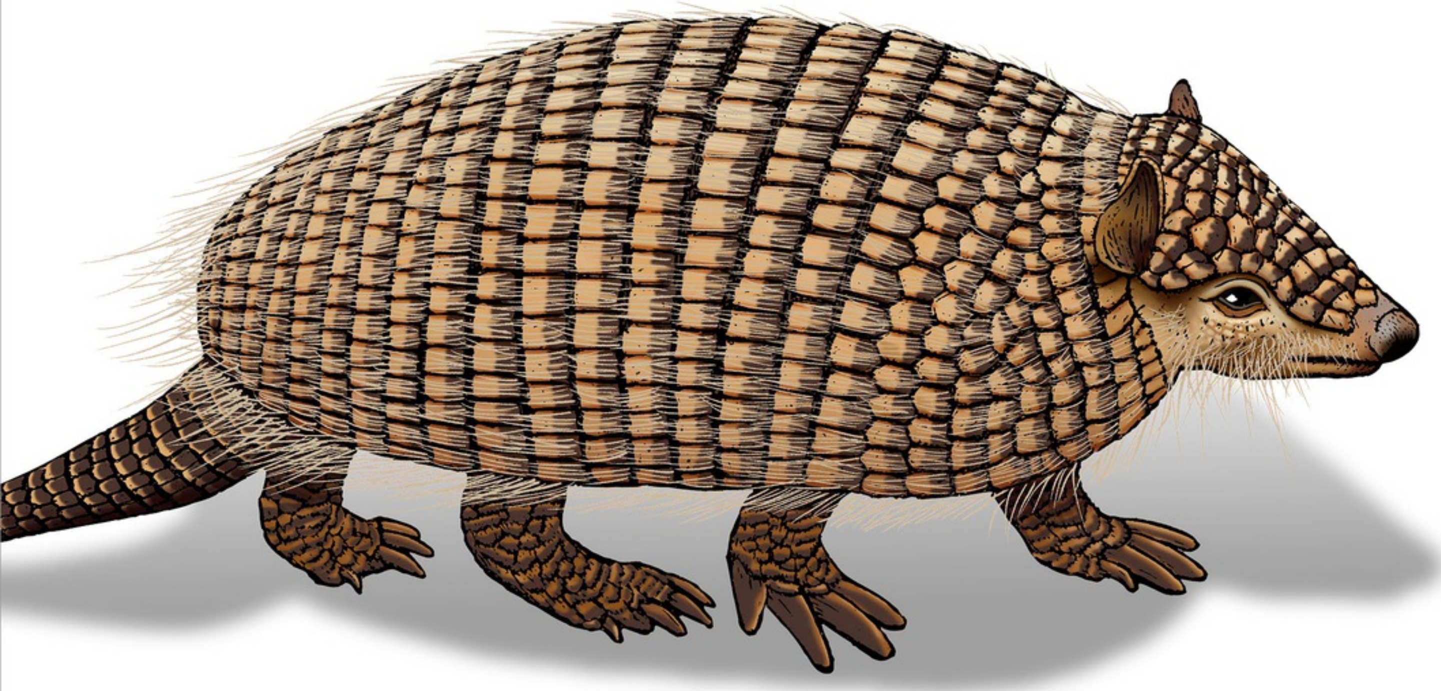
Prompt Engineering for Generative AI

Future-Proof Inputs for Reliable AI Outputs



James Phoenix
& Mike Taylor

Praise for *Prompt Engineering for Generative AI*

The absolute best book-length resource I've read on prompt engineering. Mike and James are masters of their craft.

—Dan Shipper, cofounder and CEO, Every

This book is a solid introduction to the fundamentals of prompt engineering and generative AI. The authors cover a wide range of useful techniques for all skill levels from beginner to advanced in a simple, practical, and easy-to-understand way. If you're looking to improve the accuracy and reliability of your AI systems, this book should be on your shelf.

—Mayo Oshin, founder and CEO, Siennai Analytics, early LangChain contributor

Phoenix and Taylor's guide is a lighthouse amidst the vast ocean of generative AI. Their book became a cornerstone for my team at Phiture AI Labs, as we learned to harness LLMs and diffusion models for creating marketing assets that resonate with the essence of our clients' apps and games. Through prompt engineering, we've been able to generate bespoke, on-brand content at scale. This isn't just theory; it's a practical masterclass in transforming AI's raw potential into tailored solutions, making it an essential read for developers looking to elevate their AI integration to new heights of creativity and efficiency.

—Moritz Daan, Founder/Partner, Phiture Mobile Growth Consultancy

Prompt Engineering for Generative AI is probably the most future-proof way of future-proofing your tech career. This is without a doubt the best resource for anyone working in practical applications of AI. The rich, refined principles in here will help both new and seasoned AI engineers stay on top of this very competitive game for the foreseeable future.

—Ellis Crosby, CTO and cofounder, Incremento

This is an essential guide for agency and service professionals. Integrating AI with service and client delivery, using automation management, and speeding up solutions will set new industry standards. You'll find useful, practical information and tactics in the book, allowing you to understand and utilize AI to its full potential.

—Byron Tassoni-Resch, CEO and cofounder, WeDiscover

A really interesting and informative read, mixing practical tips and tricks with some solid foundational information. The world of GenAI is developing at breakneck speed, and having a toolset that can deliver results, regardless of the foundational model being used, is worth its weight in gold!

—Riaan Dreyer, chief digital and data officer, Bank of Iceland

The authors expertly translate prompt engineering intricacies into a practical toolkit for text and image generation. This guide, spanning standard practices to cutting-edge techniques, empowers readers with practical tips to maximize generative AI model capabilities.

—Aditya Goel, generative AI consultant

Prompt Engineering for Generative AI

Future-Proof Inputs for Reliable AI Outputs at Scale

James Phoenix and Mike Taylor

O'REILLY®

Beijing • Boston • Farnham • Sebastopol • Tokyo

Prompt Engineering for Generative AI

by James Phoenix and Mike Taylor

Copyright © 2024 Saxifrage, LLC and Just Understanding Data LTD. All rights reserved.

Printed in the United States of America.

Published by O’Reilly Media, Inc., 1005 Gravenstein Highway North, Sebastopol, CA 95472.

O’Reilly books may be purchased for educational, business, or sales promotional use. Online editions are also available for most titles (http://oreilly.com). For more information, contact our corporate/institutional sales department: 800-998-9938 or *corporate@oreilly.com*.

- Acquisitions Editor: Nicole Butterfield
- Development Editor: Corbin Collins
- Copyeditor: Piper Editorial Consulting, LLC
- Proofreader: Kim Wimpsett
- Indexer: nSight, Inc.
- Interior Designer: David Futato
- Cover Designer: Karen Montgomery
- Illustrator: Kate Dullea

- May 2024: First Edition

Revision History for the First Edition

- 2024-05-15: First Release

See http://oreilly.com/catalog/errata.csp?isbn=9781098153434 for release details.

The O’Reilly logo is a registered trademark of O’Reilly Media, Inc. *Prompt Engineering for Generative AI*, the cover image, and related trade dress are trademarks of O’Reilly Media, Inc.

The views expressed in this work are those of the authors and do not represent the publisher’s views. While the publisher and the authors have used good faith efforts to ensure that the information and instructions contained in this work are accurate, the publisher and the authors disclaim all responsibility for errors or omissions, including without limitation responsibility for damages resulting from the use of or reliance on this work. Use of the information and instructions contained in this work is at your own risk. If any code samples or other technology this work contains or describes is subject to open source licenses or the intellectual property rights of others, it is your responsibility to ensure that your use thereof complies with such licenses and/or rights.

978-1-098-15343-4

[LSI]

Preface

The rapid pace of innovation in generative AI promises to change how we live and work, but it's getting increasingly difficult to keep up. The number of AI papers published on arXiv is growing exponentially, Stable Diffusion has been among the fastest growing open source projects in history, and AI art tool Midjourney's Discord server has tens of millions of members, surpassing even the largest gaming communities. What most captured the public's imagination was OpenAI's release of ChatGPT, which reached 100 million users in two months, making it the fastest-growing consumer app in history. Learning to work with AI has quickly become one of the most in-demand skills.

Everyone using AI professionally quickly learns that the quality of the output depends heavily on what you provide as input. The discipline of *prompt engineering* has arisen as a set of best practices for improving the reliability, efficiency, and accuracy of AI models. "In ten years, half of the world's jobs will be in prompt engineering," claims Robin Li, the cofounder and CEO of Chinese tech giant Baidu. However, we expect prompting to be a skill required of many jobs, akin to proficiency in Microsoft Excel, rather than a popular job title in itself. This new wave of disruption is changing everything we thought we knew about computers. We're used to writing algorithms that return the same result every time—not so for AI, where the responses are non-deterministic. Cost and latency are real factors again, after decades of Moore's law making us complacent in expecting real-time computation at negligible cost. The biggest hurdle is the tendency of these models to confidently make things up, dubbed *hallucination*, causing us to rethink the way we evaluate the accuracy of our work.

We've been working with generative AI since the GPT-3 beta in 2020, and as we saw the models progress, many early prompting tricks and hacks became no longer necessary. Over time a consistent set of principles emerged that were still useful with the newer models, and worked across both text and image generation. We have written this book based on these timeless principles, helping you learn transferable skills that will continue to be useful no matter what happens with AI over the next five years. The key to working with AI isn't "figuring out how to hack the prompt by adding one magic word to the end that changes everything else," as OpenAI cofounder Sam Altman asserts, but what will always matter is the "quality of ideas and the understanding of what you want." While we don't know if we'll call it "prompt engineering" in five years, working effectively with generative AI will only become more important.

Software Requirements for This Book

All of the code in this book is in Python and was designed to be run in a Jupyter Notebook or Google Colab notebook. The concepts taught in the book are transferable to JavaScript or any other coding language if preferred, though the primary focus of this book is on prompting techniques rather than traditional coding skills. The code can all be found on GitHub, and we will link to the relevant notebooks throughout. It's highly recommended that you utilize the GitHub repository and run the provided examples while reading the book.

For non-notebook examples, you can run the script with the format `python content/chapter_x/script.py` in your terminal, where `x` is the chapter number and `script.py` is the name of the script. In some instances, API keys need to be set as environment variables, and we will make that clear. The packages used update frequently, so install our *requirements.txt* in a virtual environment before running code examples.

The *requirements.txt* file is generated for Python 3.9. If you want to use a different version of Python, you can generate a new *requirements.txt* from this *requirements.in* file found within the GitHub repository, by running these commands:

```
`pip install pip-tools`
`pip-compile requirements.in`
```

For Mac users:

1. Open Terminal: You can find the Terminal application in your Applications folder, under Utilities, or use Spotlight to search for it.
2. Navigate to your project folder: Use the `cd` command to change the directory to your project

folder. For example: `cd path/to/your/project`.

3. Create the virtual environment: Use the following command to create a virtual environment named `venv` (you can name it anything): `python3 -m venv venv`.
4. Activate the virtual environment: Before you install packages, you need to activate the virtual environment. Do this with the command `source venv/bin/activate`.
5. Install packages: Now that your virtual environment is active, you can install packages using `pip`. To install packages from the *requirements.txt* file, use `pip install -r requirements.txt`.
6. Deactivate virtual environment: When you’re done, you can deactivate the virtual environment by typing **`deactivate`**.

For Windows users:

1. Open Command Prompt: You can search for `cmd` in the Start menu.
2. Navigate to your project folder: Use the `cd` command to change the directory to your project folder. For example: `cd path\to\your\project`.
3. Create the virtual environment: Use the following command to create a virtual environment named `venv`: `python -m venv venv`.
4. Activate the virtual environment: To activate the virtual environment on Windows, use `.\venv\Scripts\activate`.
5. Install packages: With the virtual environment active, install the required packages: `pip install -r requirements.txt`.
6. Deactivate the virtual environment: To exit the virtual environment, simply type: `deactivate`.

Here are some additional tips on setup:

- Always ensure your Python is up-to-date to avoid compatibility issues.
- Remember to activate your virtual environment whenever you work on the project.
- The *requirements.txt* file should be in the same directory where you create your virtual environment, or you should specify the path to it when using `pip install -r`.

Access to an OpenAI developer account is assumed, as your `OPENAI_API_KEY` must be set as an environment variable in any examples importing the OpenAI library, for which we use version 1.0. Quick-start instructions for setting up your development environment can be found in OpenAI's documentation on their website.

You must also ensure that *billing is enabled* on your OpenAI account and that a valid payment method is attached to run some of the code within the book. The examples in the book use GPT-4 where not stated, though we do briefly cover Anthropic's competing Claude 3 model, as well as Meta's open source Llama 3 and Google Gemini.

For image generation we use Midjourney, for which you need a Discord account to sign up, though these principles apply equally to DALL-E 3 (available with a ChatGPT Plus subscription or via the API) or Stable Diffusion (available as an API or it can run locally on your computer if it has a GPU). The image generation examples in this book use Midjourney v6, Stable Diffusion v1.5 (as many extensions are still only compatible with this version), or Stable Diffusion XL, and we specify the differences when this is important.

We provide examples using open source libraries wherever possible, though we do include commercial vendors where appropriate—for example, Chapter 5 on vector databases demonstrates both FAISS (an open source library) and Pinecone (a paid vendor). The examples demonstrated in the book should be easily modifiable for alternative models and vendors, and the skills taught are transferable. Chapter 4 on advanced text generation is focused on the LLM framework LangChain, and Chapter 9 on advanced image generation is built on AUTOMATIC1111's open source Stable Diffusion Web UI.

Conventions Used in This Book

The following typographical conventions are used in this book:

Italic

Indicates new terms, URLs, email addresses, filenames, and file extensions.

`Constant width`

Used for program listings, as well as within paragraphs to refer to program elements such as variable or function names, databases, data types, environment variables, statements, and keywords.

`Constant width bold`

Shows commands or other text that should be typed literally by the user.

`Constant width italic`

Shows text that should be replaced with user-supplied values or by values determined by context.

TIP

This element signifies a tip or suggestion.

NOTE

This element signifies a general note.

WARNING

This element indicates a warning or caution.

Throughout the book we reinforce what we call the Five Principles of Prompting, identifying which principle is most applicable to the example at hand. You may want to refer to Chapter 1, which describes the principles in detail.

PRINCIPLE NAME

This will explain how the principle is applied to the current example or section of text.

Using Code Examples

Supplemental material (code examples, exercises, etc.) is available for download at *https://oreil.ly/prompt-engineering-for-generative-ai*.

If you have a technical question or a problem using the code examples, please send email to *bookquestions@oreilly.com*.

This book is here to help you get your job done. In general, if example code is offered with this book, you may use it in your programs and documentation. You do not need to contact us for permission unless you’re reproducing a significant portion of the code. For example, writing a program that uses several chunks of code from this book does not require permission. Selling or distributing examples from O’Reilly books does require permission. Answering a question by citing this book and quoting example code does not require permission. Incorporating a significant amount of example code from this book into your product’s documentation does require permission.

We appreciate, but generally do not require, attribution. An attribution usually includes the title, author, publisher, and ISBN. For example: “*Prompt Engineering for Generative AI* by James Phoenix and Mike Taylor (O’Reilly). Copyright 2024 Saxifrage, LLC and Just Understanding Data LTD, 978-1-098-15343-4.”

If you feel your use of code examples falls outside fair use or the permission given above, feel free to contact us at *permissions@oreilly.com*.

O’Reilly Online Learning

NOTE

For more than 40 years, *O’Reilly Media* has provided technology and business training, knowledge, and insight to help companies succeed.

Our unique network of experts and innovators share their knowledge and expertise through books, articles, and our online learning platform. O'Reilly's online learning platform gives you on-demand access to live training courses, in-depth learning paths, interactive coding environments, and a vast collection of text and video from O'Reilly and 200+ other publishers. For more information, visit *https://oreilly.com*.

How to Contact Us

Please address comments and questions concerning this book to the publisher:

- O'Reilly Media, Inc.
- 1005 Gravenstein Highway North
- Sebastopol, CA 95472
- 800-889-8969 (in the United States or Canada)
- 707-827-7019 (international or local)
- 707-829-0104 (fax)
- *support@oreilly.com*
- *https://www.oreilly.com/about/contact.html*

We have a web page for this book, where we list errata, examples, and any additional information. You can access this page at *https://oreil.ly/prompt-engineering-generativeAI*.

For news and information about our books and courses, visit *https://oreilly.com*.

Find us on LinkedIn: *https://linkedin.com/company/oreilly-media*.

Watch us on YouTube: *https://youtube.com/oreillymedia*.

Acknowledgments

We’d like to thank the following people for their contribution in conducting a technical review of the book and their patience in correcting a fast-moving target:

- Mayo Oshin, early LangChain contributor and founder at SeinnAI Analytics
- Ellis Crosby, founder at Scarlett Panda and AI agency Incremen.to
- Dave Pawson, O’Reilly author of *XSL-FO*
- Mark Phoenix, a senior software engineer
- Aditya Goel, GenAI consultant

We are also grateful to our families for their patience and understanding and would like to reassure them that we still prefer talking to them over ChatGPT.

Chapter 1. The Five Principles of Prompting

Prompt engineering is the process of discovering prompts that 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 multiple 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.

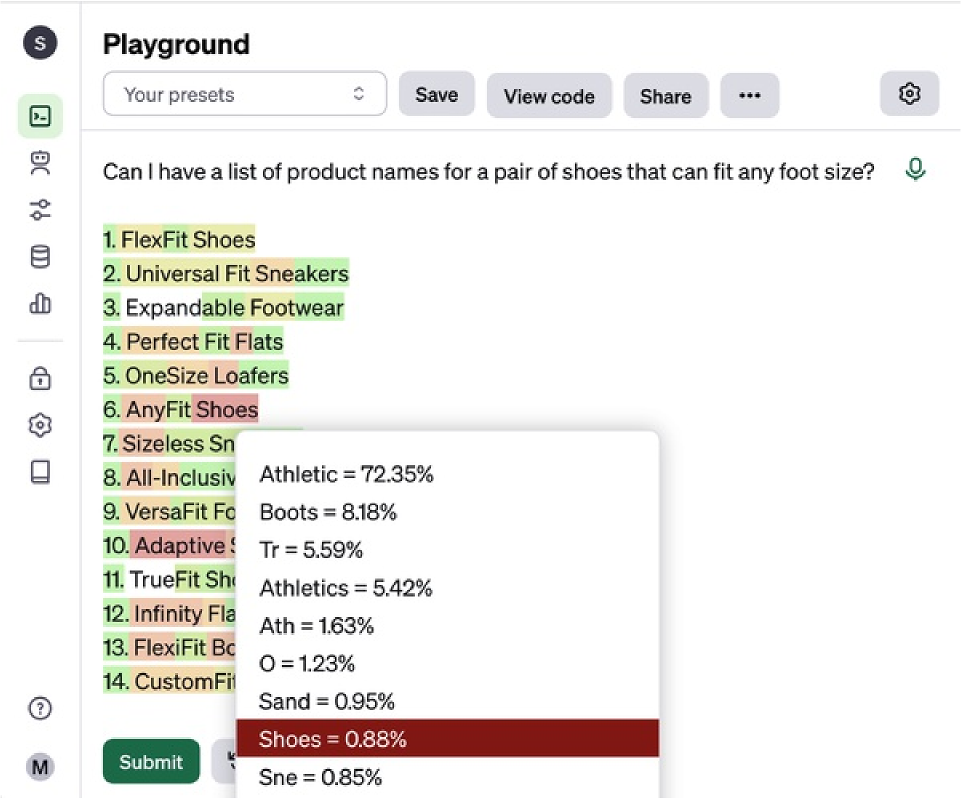
Missing examples
You haven’t given the AI any examples of what *good* names look like. It’s autocompleting using an average of its training data, i.e., the entire internet (with all its inherent bias), but is that what you want? Ideally you’d feed it examples of successful names, common names in an industry, 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 three-fourths 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 *shoes* had a lower probability of coming after the start of the name *AnyFit* (0.88%), where a more predictable response would be *Athletic* (72.35%).



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

1. FlexFit Shoes
2. Universal Fit Sneakers
3. Expandable Footwear
4. Perfect Fit Flats
5. OneSize Loafers
6. AnyFit Shoes
7. Sizeless Sn
8. All-Inclusiv
9. VersaFit Fo
10. Adaptive
11. TrueFit Sh
12. Infinity Fla
13. FlexiFit B
14. CustomFi

Athletic = 72.35%
Boots = 8.18%
Tr = 5.59%
Athletics = 5.42%
Ath = 1.63%
O = 1.23%
Sand = 0.95%
Shoes = 0.88%
Sne = 0.85%

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, 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 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:

Give Direction
Describe the desired style in detail, or reference a relevant persona

Specify Format
Define what rules to follow, and the required structure of the response

Provide Examples
Insert a diverse set of test cases where the task was done correctly

Evaluate Quality
Identify errors and rate responses, testing what drives performance.

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 (70,000+ 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 4 people watching on white MacBook on top of glass-top
table, Panasonic, DC-GH5
```

Figure 1-2 shows the output.