<https://www.promptingguide.ai/introduction/tips>

1. **Elements of a Prompt**

As we cover more and more examples and applications with prompt engineering, you will notice that certain elements make up a prompt.

A prompt contains any of the following elements:

**Instruction** - a specific task or instruction you want the model to perform

**Context** - external information or additional context that can steer the model to better responses

**Input Data** - the input or question that we are interested to find a response for

**Output Indicator** - the type or format of the output.

You do not need all the four elements for a prompt and the format depends on the task at hand. We will touch on more concrete examples in upcoming guides.

### BEGIN INSTRUCTION ###

### END INSTRUCTION ###

### BEGIN EXAMPLE ###

### END EXAMPLE ###

There are a few different parts of a prompt that you will see over and over again. They are roughly:

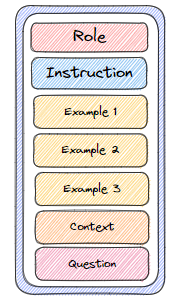
* A role
* An instruction/task
* A question
* Context
* Examples (few-shot)

Translate the text below to Spanish:

Text: "hello!"

* **Formalizing Prompts**

RoleExample 1InstructionExample 2Example 3ContextQuestion



TAKEAWAYS

* Understand the basic parts of a prompt

We have now covered multiple types of prompts, as well as ways to combine them. This lesson will provide you with terms to explain different types of prompts. Although there have been approaches to formalize discourse around prompt engineering[1](https://learnprompting.org/docs/basics/formalizing#fn-1), the field is ever changing, so we will provide you with just enough information to get started.

* **Parts of a Prompt**[**​**](https://learnprompting.org/docs/basics/formalizing#parts-of-a-prompt)

There are a few different parts of a prompt that you will see over and over again. They are roughly:

* A role
* An instruction/task
* A question
* Context
* Examples (few-shot)

Understanding the structure and components of a prompt is crucial in effectively utilizing language models. The key parts of a prompt include a role, an instruction or task, a question, context, and examples. Not all prompts will contain all these elements, and their order can vary

<https://docs.cloud.deepset.ai/docs/prompt-engineering-guidelines>

# General Tips

* Prompts can contain instructions or questions to pass to the model. They can also contain additional information, such as context or examples.
* Prompt engineering is an iterative process. Be ready to experiment.
* Version your prompts to compare the performance of different versions and choose the one that worked best.
* Start with simple, zero-shot prompts and then keep making them more complex to reach the required accuracy of answers. You can add more context and examples to each iteration of your prompt. If you're not seeing any improvements, fine-tune the model.
* Always use the latest models available.

# Prompt Format

Remember that the format of the prompt depends on the task you want the model to perform. Experiment and test what works best.

Prompts with the format that follows these guidelines seem to achieve better results than prompts with random formatting:

* The prompt has instructions, context, input data, and output format. Not all these components are necessary, and they may vary depending on the model's task.
* You can try experimenting with specific prompt structures, for example:
* Question:
* Context:
* Answer:

or for question answering without a given context:

Q:

A:

## Instructions

* Put them at the beginning of the prompt.
* Use commands, for example, "write", "summarize", "translate". Try using different keywords and see which yield the best results.
* Separate instructions from the context with a delimiter, for example, quotation marks (""") or hashes (###).
* Separate the instructions, examples, context, and input data with one or more line breaks.

Examples:

### Instruction ###

Give a title to the article. The title must have 5 words.

Article: Online retail giant Amazon has said it plans to shutdown three warehouses in the UK putting 1200 jobs at risk.

Title:

Instructions: Detect the language of the text. Answer with the name of the language.

Text: Professionelle Beratung und ein Top-Service sind für uns selbstverständlich. Daher bringen unsere Mitarbeiter eine große Portion Kompetenz und Erfahrung mit.

## Context

* Specify any information you want the model to use to generate the answer. The context should help the model arrive at better responses.

Example:

Answer the question using the context. If you are not sure about the answer, answer with "I don't know".

Context: Contrails are a manmade type of cirrus cloud formed when water vapor from the exhaust of a jet engine condenses on particles, which come from either the surrounding air or the exhaust itself, and freezes, leaving behind a visible trail. The exhaust can also trigger the formation of cirrus by providing ice nuclei when there is an insufficient naturally-occurring supply in the atmosphere. One of the environmental impacts of aviation is that persistent contrails can form into large mats of cirrus, and increased air traffic has been implicated as one possible cause of the increasing frequency and amount of cirrus in Earth's atmosphere.

Question: Why do airplanes leave contrails in the sky?

## Output Format

* Be specific about the format you want the model to use for the answer. If needed, specify the length, style, and so on.
* It may help to give the model a couple of concrete examples.

Example:

Classify the text into neutral, negative, or positive.

Text: I love Berlin!

Answer:

By giving the model examples, you're showing exactly what answer you expect.

## Language

* Be direct, specific, and detailed.
* Be concise, as there's usually a limit on the number of tokens in a prompt and longer prompts are computationally more expensive than shorter ones.
* Focus on what you want the model to do rather than what you don't want it to do.

Example:

Using specific languageUsing imprecise language

Explain the concept nuclear sampling in 3 to 5 sentences to a 10-year old.

# Few-Shot Prompts

Few-shot prompts contain a couple of examples for the model. By giving the model examples, you demonstrate what you expect it to do and enable it to perform in-context learning to improve its performance.

Use few-shot prompts if you require the model to return an answer in a specific format or if a simple prompt didn't yield the results you needed.

* Use consistent formatting.
* Provide examples in random order. For example, don't put all negative examples first and positive ones second. Mix them up as the order might bias the model.
* Make sure you use labels.

Examples:

This example is from the [Brown et al. 2020](https://arxiv.org/abs/2005.14165)paper:

A "whatpu" is a small, furry animal native to Tanzania. An example of a sentence that uses

the word whatpu is:

We were traveling in Africa and we saw these very cute whatpus.

To do a "farduddle" means to jump up and down really fast. An example of a sentence that uses

the word farduddle is:

Model answer:

One day when I was playing tag with my little sister, she got really excited and she

started doing these crazy farduddles.

Or for sentiment analysis:

Classify the text into neutral, negative, or positive.

Text: I love Berlin!

Answer: positive

Text: I hate Paris.

Answer: negative

Text: I've never been to Bangalore.

Answer: neutral

Text: I don't like Tokio.

Answer:

<https://dataplatform.cloud.ibm.com/docs/content/wsj/analyze-data/fm-prompt-tips.html?context=wx>

1. **Tip 2: Include all the needed prompt components**

Effective prompts usually have one or more of the following components: instruction, context, examples, and cue.

1. **Instruction**

An instruction is an imperative statement that tells the model what to do. For example, if you want the model to list ideas for a dog-walking business, your instruction could be: "List ideas for starting a dog-walking business:"

1. **Context**

Including background or contextual information in your prompt can nudge the model output in a desired direction. Specifically, (tokenized) words that appear in your prompt text are more likely to be included in the generated output.

1. **Examples**

To indicate the format or shape that you want the model response to be, include one or more pairs of example input and corresponding desired output showing the pattern you want the generated text to follow. (Including one example in your prompt is called *one-shot prompting*, including two or more examples in your prompt is called *few-shot* prompting, and when your prompt has no examples, that's called *zero-shot* prompting.)

Note that when you are prompting models that have been fine-tuned, you might not need examples.

1. **Cue**

A cue is text at the end of the prompt that is likely to start the generated output on a desired path. (Remember, as much as it seems like the model is *responding to your prompt*, the model is really *appending text to your prompt* or *continuing your prompt*.)

<https://medium.com/@kalimick120/the-essential-fundamentals-to-prompt-engineering-for-better-results-with-large-language-models-29546152b8c2>

|  |  |
| --- | --- |
| 1. **The Essential Fundamentals to Prompt Engineering for Better Results with Large Language Models**   [Val Trevor](https://medium.com/@kalimick120?source=post_page-----29546152b8c2--------------------------------)  ·  Follow  5 min read  ·  Aug 17  This article shares the basics of prompt engineering, so you can get the best outcomes when using large language models. I’ve spent a lot of time learning and figuring things out myself, but instead of giving you a list of the best GPT prompts, I’ve put together a simple guide that focuses on teaching you the fundamental concepts of prompt engineering. These fundamentals guide me in getting the best responses possible. So by the end of this read, you’ll have a good understanding of what you can achieve with prompts and how keeping these concepts in mind can greatly improve your results.   1. **Elements of a Prompt**   Let’s start by examining the different elements that make up a prompt. A prompt can consist of the following five elements:   1. **Input or Context**: This sets the stage and provides the necessary background information for the model to generate an appropriate response. 2. **Instructions**: Clear instructions guide the model on what action to take. Instructions can be in the form of one or multiple sentences or even questions. 3. **Examples**: Including examples, also known as “few-shot learning,” can provide the model with a reference for the desired output or conversation. 4. **Desired Output Format**: By specifying the desired output format, such as a yes-or-no answer or a short explanation, you can influence the way the model responds.   Not all of these elements need to be present in a prompt. In fact, you can even start a sentence and rely on the model’s completion to generate a response. However, for a well-constructed prompt, it is recommended to have at least one instruction or question present. Additionally, you can mix and match these elements to create prompts tailored to your needs.   1. **Use Cases for Prompts**   Now that we understand the elements of a prompt, let’s look at some of the common use cases where prompts can be employed effectively with large language models. While this list isn’t exhaustive, it covers the most prevalent use cases:   1. **Summarization**: Prompt the model to summarize a given text or passage. 2. **Classification**: Instruct the model to classify a text into predefined categories, such as sports, finance, or education. 3. **Translation**: Utilize prompts to translate text from one language to another. 4. **Text Generation or Completion**: Explore the model’s creative capabilities by generating or completing text based on a prompt. 5. **Question Answering**: Ask the model general questions or provide additional context to obtain specific answers. 6. **Coaching**: Seek suggestions or improvements for a given script or scenario. 7. **Image Generation** (with certain models): Prompt the model to generate images, such as requesting an image of a cute puppy.   These use cases offer a glimpse into the diverse applications of prompts. They highlight the versatility of language models and the potential they hold for various tasks.   1. **Effective Prompting Tips**   To achieve better results with prompts, here are some easy-to-follow guidelines to keep in mind:   1. **Be Clear and Concise**: Use direct instructions or questions that leave no room for ambiguity. 2. **Provide Relevant Context**: Include any additional information or data that can aid the model in generating accurate responses. 3. **Offer Examples**: Consider using few-shot learning by including one or more examples to guide the model. 4. **Specify Desired Output Format**: Define the format in which you want the model to present its response, whether it’s a short answer, bullet points, or something else. 5. **Encourage Factual Responses**: Ask the model to rely on reliable sources or provide specific quotations to support its answers and minimize potential hallucinations. 6. **Align with the Task**: Ensure that your prompt instructions are in line with the desired task or outcome. For example, if you need customer support assistance, set the scene accordingly. 7. **Experiment with Different Personas**: Explore using different personas to obtain specific styles or voices in the model’s responses.   By following these tips, you can greatly enhance your prompts and obtain more accurate and desired outputs from the language model.   1. **Controlling Output with Prompting Techniques**   Here are some specific prompting techniques you can utilize to exert control over the model’s output:   1. **Length Controls**: Specify the desired length of the generated response to ensure it aligns with your requirements. 2. **Tone Controls**: Direct the model to respond with a specific tone, such as politeness or formality. 3. **Style Controls**: Guide the model to present the response in a particular style, such as bullet points or a paragraph. 4. **Audience Controls**: Instruct the model to explain the topic in a way suitable for a specific audience, like a five-year-old child. 5. **Context Controls**: Determine the level of context you want the model to consider while generating the response. 6. **Scenario-Based Guiding**: Set the scene or scenario to provide a context for the model’s response, making it more relevant to the desired outcome. 7. **Chain of Thought Prompting**: Break down complex questions or tasks into step-by-step instructions, helping the model arrive at the accurate answer. 8. **Avoid Hallucinations**: Explicitly instruct the model to avoid making things up or to support its responses with relevant quotations.   By employing these techniques, you can exercise greater control over the model’s output and steer it towards more desired and accurate responses.   1. **Effective Iteration and Best Practices**   Finding the most effective prompt often requires some trial and error and iterative refinement. Here are some tips to guide you during the iteration process:   1. **Experiment with Different Prompts**: Try out different prompt variations to discover what works best for your specific task. 2. **Combine Direct Instructions with Examples**: Incorporate direct instructions alongside examples to provide more clarity and guidance to the model. 3. **Rephrase Instructions**: Vary the level of conciseness in your instructions, experimenting with more or less specificity. 4. **Explore Different Personas**: Test how using different personas affects the style and tone of the model’s responses. 5. **Gradually Increase or Decrease Examples**: Fine-tune the number of examples included in your prompts to determine the optimal balance for your needs.   By iterating and refining your prompts using these best practices, you can maximize the accuracy and relevance of the model’s responses.   1. **In Conclusion**   Prompt engineering is a vital skill when working with large language models. By understanding the elements of a prompt, exploring various use cases, implementing effective prompting techniques, and iterating based on best practices, you can harness the full potential of these models and achieve superior results.  A well-crafted prompt can significantly enhance the performance of language models and enable them to provide accurate, informative, and contextually appropriate responses.  I hope you found this essential guide to prompt engineering helpful. | 1. **Các nguyên tắc cơ bản cần thiết để thúc đẩy kỹ thuật để có kết quả tốt hơn với các mô hình ngôn ngữ lớn**   [Val Trevor](https://medium.com/@kalimick120?source=post_page-----29546152b8c2--------------------------------)  ·  Theo  đọc 5 phút  ·  ngày 17 tháng 8  Bài viết này chia sẻ những kiến ​​thức cơ bản về kỹ thuật nhắc nhở để bạn có thể đạt được kết quả tốt nhất khi sử dụng mô hình ngôn ngữ lớn. Tôi đã dành nhiều thời gian để tìm hiểu và tự mình tìm hiểu mọi thứ, nhưng thay vì đưa cho bạn danh sách các lời nhắc GPT hay nhất, tôi đã tổng hợp một hướng dẫn đơn giản tập trung vào việc dạy cho bạn các khái niệm cơ bản về kỹ thuật nhắc nhở. Những nguyên tắc cơ bản này hướng dẫn tôi nhận được phản hồi tốt nhất có thể. Vì vậy, khi kết thúc bài đọc này, bạn sẽ hiểu rõ về những gì bạn có thể đạt được khi có lời nhắc và việc ghi nhớ những khái niệm này có thể cải thiện đáng kể kết quả của bạn như thế nào.   1. **Các yếu tố của một lời nhắc**   Hãy bắt đầu bằng cách kiểm tra các yếu tố khác nhau tạo nên lời nhắc. Một lời nhắc có thể bao gồm năm thành phần sau:   1. **Đầu vào hoặc bối cảnh** : Điều này đặt ra giai đoạn và cung cấp thông tin cơ bản cần thiết để mô hình tạo ra phản hồi thích hợp. 2. **Hướng dẫn** : Hướng dẫn rõ ràng hướng dẫn mô hình về hành động cần thực hiện. Hướng dẫn có thể ở dạng một hoặc nhiều câu hoặc thậm chí là câu hỏi. 3. **Ví dụ** : Bao gồm các ví dụ, còn được gọi là "học vài lần", có thể cung cấp cho mô hình tham chiếu về đầu ra hoặc cuộc trò chuyện mong muốn. 4. **Định dạng đầu ra mong muốn** : Bằng cách chỉ định định dạng đầu ra mong muốn, chẳng hạn như câu trả lời có hoặc không hoặc một lời giải thích ngắn gọn, bạn có thể tác động đến cách mô hình phản hồi.   Không phải tất cả các yếu tố này đều cần phải có trong lời nhắc. Trên thực tế, bạn thậm chí có thể bắt đầu một câu và dựa vào khả năng hoàn thành của mô hình để tạo ra câu trả lời. Tuy nhiên, để có lời nhắc được xây dựng tốt, bạn nên có ít nhất một hướng dẫn hoặc câu hỏi. Ngoài ra, bạn có thể trộn và kết hợp các yếu tố này để tạo lời nhắc phù hợp với nhu cầu của mình.   1. **Các trường hợp sử dụng lời nhắc**   Bây giờ chúng ta đã hiểu các thành phần của lời nhắc, hãy xem xét một số trường hợp sử dụng phổ biến trong đó lời nhắc có thể được sử dụng hiệu quả với các mô hình ngôn ngữ lớn. Mặc dù danh sách này không đầy đủ nhưng nó bao gồm các trường hợp sử dụng phổ biến nhất:   1. **Tóm tắt** : Nhắc model tóm tắt một văn bản hoặc đoạn văn nhất định. 2. **Phân loại** : Hướng dẫn mô hình phân loại văn bản thành các danh mục được xác định trước, chẳng hạn như thể thao, tài chính hoặc giáo dục. 3. **Dịch** : Sử dụng lời nhắc để dịch văn bản từ ngôn ngữ này sang ngôn ngữ khác. 4. **Tạo hoặc hoàn thành văn bản** : Khám phá khả năng sáng tạo của mô hình bằng cách tạo hoặc hoàn thành văn bản dựa trên lời nhắc. 5. **Trả lời câu hỏi** : Đặt những câu hỏi chung cho mô hình hoặc cung cấp ngữ cảnh bổ sung để có được câu trả lời cụ thể. 6. **Huấn luyện** : Tìm kiếm đề xuất hoặc cải tiến cho một kịch bản hoặc kịch bản nhất định. 7. **Tạo hình ảnh** (với một số mô hình nhất định): Nhắc mô hình tạo hình ảnh, chẳng hạn như yêu cầu hình ảnh của một chú chó con dễ thương.   Những trường hợp sử dụng này cung cấp cái nhìn sơ lược về các ứng dụng đa dạng của lời nhắc. Chúng nêu bật tính linh hoạt của các mô hình ngôn ngữ và tiềm năng mà chúng nắm giữ cho các nhiệm vụ khác nhau.   1. **Mẹo nhắc nhở hiệu quả**   Để đạt được kết quả tốt hơn với lời nhắc, dưới đây là một số nguyên tắc dễ thực hiện cần ghi nhớ:   1. **Hãy rõ ràng và ngắn gọn** : Sử dụng những hướng dẫn trực tiếp hoặc những câu hỏi không có chỗ cho sự mơ hồ. 2. **Cung cấp bối cảnh liên quan** : Bao gồm mọi thông tin hoặc dữ liệu bổ sung có thể hỗ trợ mô hình tạo ra phản hồi chính xác. 3. **Đưa ra ví dụ** : Hãy cân nhắc sử dụng phương pháp học vài lần bằng cách đưa vào một hoặc nhiều ví dụ để hướng dẫn mô hình. 4. **Chỉ định định dạng đầu ra mong muốn** : Xác định định dạng mà bạn muốn mô hình trình bày phản hồi của nó, cho dù đó là một câu trả lời ngắn, dấu đầu dòng hay thứ gì khác. 5. **Khuyến khích các câu trả lời thực tế** : Yêu cầu mô hình dựa vào các nguồn đáng tin cậy hoặc cung cấp các trích dẫn cụ thể để hỗ trợ cho câu trả lời của mình và giảm thiểu ảo giác tiềm ẩn. 6. **Căn chỉnh với Nhiệm vụ** : Đảm bảo rằng các hướng dẫn nhắc nhở của bạn phù hợp với nhiệm vụ hoặc kết quả mong muốn. Ví dụ: nếu bạn cần hỗ trợ hỗ trợ khách hàng, hãy đặt bối cảnh phù hợp. 7. **Thử nghiệm với các cá tính khác nhau** : Khám phá việc sử dụng các cá tính khác nhau để có được phong cách hoặc giọng nói cụ thể trong phản hồi của mô hình.   Bằng cách làm theo những mẹo này, bạn có thể cải thiện đáng kể lời nhắc của mình và thu được kết quả đầu ra chính xác và mong muốn hơn từ mô hình ngôn ngữ.   1. **Kiểm soát đầu ra bằng kỹ thuật nhắc nhở**   Dưới đây là một số kỹ thuật nhắc nhở cụ thể mà bạn có thể sử dụng để kiểm soát đầu ra của mô hình:   1. **Kiểm soát độ dài** : Chỉ định độ dài mong muốn của phản hồi được tạo để đảm bảo nó phù hợp với yêu cầu của bạn. 2. **Điều khiển giọng điệu** : Hướng dẫn model phản hồi bằng một giọng điệu cụ thể, chẳng hạn như sự lịch sự hoặc trang trọng. 3. **Điều khiển kiểu** : Hướng dẫn mô hình trình bày phản hồi theo một kiểu cụ thể, chẳng hạn như dấu đầu dòng hoặc đoạn văn. 4. **Kiểm soát đối tượng** : Hướng dẫn model giải thích chủ đề theo cách phù hợp với đối tượng cụ thể, giống như một đứa trẻ năm tuổi. 5. **Kiểm soát bối cảnh** : Xác định mức độ bối cảnh bạn muốn mô hình xem xét trong khi tạo phản hồi. 6. **Hướng dẫn dựa trên kịch bản** : Đặt cảnh hoặc kịch bản để cung cấp bối cảnh cho phản hồi của mô hình, làm cho nó phù hợp hơn với kết quả mong muốn. 7. **Chuỗi nhắc nhở suy nghĩ** : Chia nhỏ các câu hỏi hoặc nhiệm vụ phức tạp thành các hướng dẫn từng bước, giúp mô hình đi đến câu trả lời chính xác. 8. **Tránh ảo giác** : Hướng dẫn rõ ràng cho model tránh bịa đặt mọi thứ hoặc hỗ trợ phản hồi của họ bằng những trích dẫn có liên quan.   Bằng cách sử dụng các kỹ thuật này, bạn có thể kiểm soát tốt hơn đầu ra của mô hình và điều khiển nó theo hướng phản hồi chính xác và mong muốn hơn.   1. **Lặp lại hiệu quả và thực tiễn tốt nhất**   Việc tìm kiếm lời nhắc hiệu quả nhất thường đòi hỏi một số thử nghiệm, sai sót và sàng lọc lặp đi lặp lại. Dưới đây là một số mẹo hướng dẫn bạn trong quá trình lặp lại:   1. **Thử nghiệm với các lời nhắc khác nhau** : Hãy thử các biến thể lời nhắc khác nhau để khám phá xem cách nào phù hợp nhất với nhiệm vụ cụ thể của bạn. 2. **Kết hợp hướng dẫn trực tiếp với ví dụ** : Kết hợp hướng dẫn trực tiếp bên cạnh các ví dụ để mang lại sự rõ ràng và hướng dẫn hơn cho mô hình. 3. **Hướng dẫn viết lại** : Thay đổi mức độ ngắn gọn trong hướng dẫn của bạn, thử nghiệm mức độ cụ thể ít nhiều. 4. **Khám phá các cá tính khác nhau** : Kiểm tra xem việc sử dụng các cá tính khác nhau ảnh hưởng như thế nào đến phong cách và giọng điệu phản hồi của model. 5. **Tăng hoặc giảm dần các ví dụ** : Tinh chỉnh số lượng ví dụ có trong lời nhắc của bạn để xác định số dư tối ưu cho nhu cầu của bạn.   Bằng cách lặp lại và tinh chỉnh lời nhắc bằng các phương pháp hay nhất này, bạn có thể tối đa hóa độ chính xác và mức độ liên quan của các phản hồi của mô hình.   1. **Tóm lại là**   Kỹ thuật prompt là một kỹ năng quan trọng khi làm việc với các mô hình ngôn ngữ lớn. Bằng cách hiểu các yếu tố của lời nhắc, khám phá các trường hợp sử dụng khác nhau, triển khai các kỹ thuật nhắc nhở hiệu quả và lặp lại dựa trên các phương pháp hay nhất, bạn có thể khai thác toàn bộ tiềm năng của các mô hình này và đạt được kết quả vượt trội.  Lời nhắc được xây dựng tốt có thể nâng cao đáng kể hiệu suất của các mô hình ngôn ngữ và cho phép chúng đưa ra phản hồi chính xác, giàu thông tin và phù hợp với ngữ cảnh.  Tôi hy vọng bạn thấy hướng dẫn cần thiết này giúp ích cho kỹ thuật prompt. |

<https://www.linkedin.com/pulse/elements-prompt-engineering-data-ins-technology-llc/>

Prompt engineering is the process of crafting effective and precise prompts or instructions when working with natural language models like GPT-3.5. Well-constructed prompts can help you achieve the desired results and control the model's output. Here are some key elements of prompt engineering:

1. **Clarity and Specificity**: Your prompt should be clear and specific about what you want the model to do. Ambiguity can lead to unpredictable results. Use explicit language to convey your instructions.
2. **Context Establishment**: Provide relevant context if necessary. This might include setting the scene, defining terms, or explaining the background information needed for the model to understand the task.
3. **Desired Format or Output**: Clearly state the desired format or structure of the answer. For example, if you want a list, specify that in your prompt. If you need a summary or a detailed explanation, make it explicit.
4. **Examples and Constraints**: You can use example inputs and outputs to guide the model's behavior. Showing the model what you expect can be more effective than describing it. Additionally, you can use constraints to restrict certain types of responses.
5. **Length and Complexity**: Indicate the desired length and complexity of the response. If you want a concise answer, specify that. If you need a detailed and comprehensive explanation, mention it in your prompt.
6. **Positive and Negative Reinforcement**: You can use positive or negative reinforcement to guide the model's behavior. For instance, you can say, "Please provide a fact-based answer" to discourage speculative responses.
7. **Multiple Prompts**: Experiment with different prompts to see which one yields the best results. Sometimes, slight variations in wording can lead to significant changes in output.
8. **Prompt Testing**: Before relying on the model for important tasks, test your prompts to ensure they produce the desired results. This iterative process can help you refine your prompts over time.
9. **Adaptability**: Be prepared to adapt your prompts as you interact with the model. If the initial results are not satisfactory, refine your prompts based on the model's responses.

Remember that prompt engineering can be a creative and iterative process. It often requires experimentation and fine-tuning to achieve the desired results with natural language models.

<https://help.openai.com/en/articles/6654000-best-practices-for-prompt-engineering-with-openai-api>

Best practices for prompt engineering with OpenAI API

How to give clear and effective instructions to GPT-3 and Codex

**J**

Written by Jessica Shieh

Updated over a week ago

💡 If you're just getting started with OpenAI API, we recommend reading the [Introduction](https://beta.openai.com/docs/introduction/introduction) and [Quickstart](https://beta.openai.com/docs/quickstart) tutorials first.

1. **How prompt engineering works**

Due to the way the instruction-following [models](https://beta.openai.com/docs/models) are trained or the data they are trained on, there are specific prompt formats that work particularly well and align better with the tasks at hand. Below we present a number of prompt formats we find work reliably well, but feel free to explore different formats, which may fit your task best.

1. **Rules of Thumb and Examples**

**Note**: the "*{text input here}*" is a placeholder for actual text/context

1. **1. Use the latest model**

For best results, we generally recommend using the latest, most capable models. As of November 2022, the best options are the **“text-davinci-003”** [model](https://beta.openai.com/docs/models) for text generation, and the **“code-davinci-002”** model for code generation.

1. **2. Put instructions at the beginning of the prompt and use ### or """ to separate the instruction and context**

Less effective ❌:

Summarize the text below as a bullet point list of the most important points.  
  
{text input here}

Better ✅:

Summarize the text below as a bullet point list of the most important points.  
  
Text: """  
{text input here}  
"""

1. **3. Be specific, descriptive and as detailed as possible about the desired context, outcome, length, format, style, etc**

Be specific about the context, outcome, length, format, style, etc

Less effective ❌:

Write a poem about OpenAI.

Better ✅:

Write a short inspiring poem about OpenAI, focusing on the recent DALL-E product launch (DALL-E is a text to image ML model) in the style of a {famous poet}

1. **4. Articulate the desired output format through examples (**[**example 1**](https://beta.openai.com/playground/p/DoMbgEMmkXJ5xOyunwFZDHdg)**,** [**example 2**](https://beta.openai.com/playground/p/3U5Wx7RTIdNNC9Fg8fc44omi)**).**

Less effective ❌:

Extract the entities mentioned in the text below. Extract the following 4 entity types: company names, people names, specific topics and themes.  
  
Text: {text}

Show, and tell - the models respond better when shown specific format requirements. This also makes it easier to programmatically parse out multiple outputs reliably.

Better ✅:

Extract the important entities mentioned in the text below. First extract all company names, then extract all people names, then extract specific topics which fit the content and finally extract general overarching themes  
  
Desired format:  
Company names: <comma\_separated\_list\_of\_company\_names>  
People names: -||-  
Specific topics: -||-  
General themes: -||-  
  
Text: {text}

1. **5. Start with zero-shot, then few-shot (**[**example**](https://beta.openai.com/playground/p/Ts5kvNWlp7wtdgWEkIAbP1hJ)**), neither of them worked, then fine-tune**

✅ Zero-shot

Extract keywords from the below text.  
  
Text: {text}  
  
Keywords:

✅ Few-shot - provide a couple of examples

Extract keywords from the corresponding texts below.  
  
Text 1: Stripe provides APIs that web developers can use to integrate payment processing into their websites and mobile applications.  
Keywords 1: Stripe, payment processing, APIs, web developers, websites, mobile applications  
##  
Text 2: OpenAI has trained cutting-edge language models that are very good at understanding and generating text. Our API provides access to these models and can be used to solve virtually any task that involves processing language.  
Keywords 2: OpenAI, language models, text processing, API.  
##  
Text 3: {text}  
Keywords 3:

✅Fine-tune: see fine-tune best practices [here](https://docs.google.com/document/d/1h-GTjNDDKPKU_Rsd0t1lXCAnHltaXTAzQ8K2HRhQf9U/edit).

1. **6. Reduce “fluffy” and imprecise descriptions**

Less effective ❌:

The description for this product should be fairly short, a few sentences only, and not too much more.

Better ✅:

Use a 3 to 5 sentence paragraph to describe this product.

1. **7. Instead of just saying what not to do, say what to do instead**

Less effective ❌:

The following is a conversation between an Agent and a Customer. DO NOT ASK USERNAME OR PASSWORD. DO NOT REPEAT.  
  
Customer: I can’t log in to my account.  
Agent:

Better ✅:

The following is a conversation between an Agent and a Customer. The agent will attempt to diagnose the problem and suggest a solution, whilst refraining from asking any questions related to PII. Instead of asking for PII, such as username or password, refer the user to the help article www.samplewebsite.com/help/faq  
  
Customer: I can’t log in to my account.  
Agent:

1. **8. Code Generation Specific - Use “leading words” to nudge the model toward a particular pattern**

Less effective ❌:

# Write a simple python function that  
# 1. Ask me for a number in mile  
# 2. It converts miles to kilometers

In this code example below, adding “*import*” hints to the model that it should start writing in Python. (Similarly “SELECT” is a good hint for the start of a SQL statement.)

Better ✅:

# Write a simple python function that  
# 1. Ask me for a number in mile  
# 2. It converts miles to kilometers  
   
import

1. **Parameters**

Generally, we find that **model** and **temperature** are the most commonly used parameters to alter the model output.

1. **model -** Higher performance [models](https://beta.openai.com/docs/models) are more expensive and have higher latency.
2. **temperature -** A measure of how often the model outputs a less likely token. The higher the temperature, the more random (and usually creative) the output. This, however, is not the same as “truthfulness”. For most factual use cases such as data extraction, and truthful Q&A, the temperature of 0 is best.
3. **max\_tokens** (**maximum length)** - Does not control the length of the output, but a hard cutoff limit for token generation. Ideally you won’t hit this limit often, as your model will stop either when it thinks it’s finished, or when it hits a stop sequence you defined.
4. **stop (stop sequences)** - A set of characters (tokens) that, when generated, will cause the text generation to stop.

For other parameter descriptions see the [API reference](https://beta.openai.com/docs/api-reference/completions/create).

1. **Additional Resources**

If you're interested in additional resources, we recommend:

* Guides
  + [Text completion](https://beta.openai.com/docs/guides/completion/text-completion) - learn how to generate or edit text using our models
  + [Code completion](https://beta.openai.com/docs/guides/code/code-completion-private-beta) - explore prompt engineering for Codex
  + [Fine-tuning](https://beta.openai.com/docs/guides/fine-tuning/fine-tuning) - Learn how to train a custom model for your use case
  + [Embeddings](https://beta.openai.com/docs/guides/embeddings/embeddings) - learn how to search, classify, and compare text
  + [Moderation](https://beta.openai.com/docs/guides/moderation/moderation)
* [OpenAI cookbook repo](https://github.com/openai/openai-cookbook/tree/main/examples) - contains example code and prompts for accomplishing common tasks with the API, including Question-answering with Embeddings
* [Community Forum](https://community.openai.com/)

<https://www.promptingguide.ai/techniques/zeroshot>

|  |  |
| --- | --- |
| 1. **Zero-Shot Prompting**   Large LLMs today, such as GPT-3, are tuned to follow instructions and are trained on large amounts of data; so they are capable of performing some tasks "zero-shot."  We tried a few zero-shot examples in the previous section. Here is one of the examples we used:  *Prompt:*  Classify the text into neutral, negative or positive. Text: I think the vacation is okay.Sentiment:  *Output:*  Neutral  Note that in the prompt above we didn't provide the model with any examples of text alongside their classifications, the LLM already understands "sentiment" -- that's the zero-shot capabilities at work.  Instruction tuning has shown to improve zero-shot learning [Wei et al. (2022)(opens in a new tab)](https://arxiv.org/pdf/2109.01652.pdf). Instruction tuning is essentially the concept of finetuning models on datasets described via instructions. Furthermore, [RLHF(opens in a new tab)](https://arxiv.org/abs/1706.03741) (reinforcement learning from human feedback) has been adopted to scale instruction tuning wherein the model is aligned to better fit human preferences. This recent development powers models like ChatGPT. We will discuss all these approaches and methods in upcoming sections.  When zero-shot doesn't work, it's recommended to provide demonstrations or examples in the prompt which leads to few-shot prompting. In the next section, we demonstrate few-shot prompting. | 1. **Nhắc nhở không bắn**   Các LLM lớn ngày nay, chẳng hạn như GPT-3, được điều chỉnh để làm theo hướng dẫn và được đào tạo về lượng lớn dữ liệu; vì vậy chúng có khả năng thực hiện một số nhiệm vụ "không bắn".  Chúng tôi đã thử một số ví dụ về zero-shot trong phần trước. Đây là một trong những ví dụ chúng tôi đã sử dụng:  *Lời nhắc:*  Classify the text into neutral, negative or positive. Text: I think the vacation is okay.Sentiment:  *Đầu ra:*  Neutral  Lưu ý rằng trong lời nhắc ở trên, chúng tôi không cung cấp cho mô hình bất kỳ ví dụ nào về văn bản cùng với các phân loại của chúng, LLM đã hiểu "tình cảm" -- đó là khả năng không bắn vào nơi làm việc.  Điều chỉnh hướng dẫn đã cho thấy cải thiện việc học không cần bắn [Wei et al. (2022)(mở trong tab mới)](https://arxiv.org/pdf/2109.01652.pdf). Điều chỉnh lệnh về cơ bản là khái niệm về tinh chỉnh mô hình trên các tập dữ liệu được mô tả thông qua hướng dẫn. Hơn nữa, [RLHF(mở trong tab mới)](https://arxiv.org/abs/1706.03741)(học tăng cường từ phản hồi của con người) đã được áp dụng để điều chỉnh quy mô hướng dẫn trong đó mô hình được căn chỉnh để phù hợp hơn với sở thích của con người. Sự phát triển gần đây này hỗ trợ các mô hình như ChatGPT. Chúng ta sẽ thảo luận về tất cả các cách tiếp cận và phương pháp này trong các phần sắp tới.  Khi tính năng nhắc không bắn không hiệu quả, bạn nên cung cấp minh họa hoặc ví dụ trong lời nhắc để dẫn đến nhắc nhở ít lần bắn. Trong phần tiếp theo, chúng tôi trình bày cách nhắc nhở vài lần bắn. |

-> Correct the English paragraph below to improve its fluency and coherence.

A good IELTS English passage should have the following characteristics:

1. **Clear Structure**: A well-organized passage often starts with a strong topic sentence, followed by supporting sentences with specific information, examples, or arguments.
2. **Rich Grammar and Vocabulary**: Using a variety of grammatical structures and vocabulary can make your passage more interesting and compelling. It can also help you get a higher score.
3. **Clarity and Coherence**: The meaning of the passage should be clear and easy to understand. Sentences should be well-constructed and ideas should flow logically from one to the next.
4. **Inclusion of Opinion or Argument**: In IELTS essays, especially in Opinion Essays, a good paragraph usually contains a clear argument or a personal viewpoint.
5. **Supporting Evidence and Examples**: Use concrete evidence and examples to support your point of view. This adds credibility and depth to your arguments.
6. **Summary Sentence**: The paragraph may end with a summary sentence or a transition sentence that leads to the next paragraph.
7. **No Spelling or Grammar Errors**: This seems basic but is crucial. Spelling or grammar mistakes can lead to a reduction in your score.

A well-written English paragraph should include the following elements:

1. **Topic Sentence**: A strong topic sentence that establishes the main idea of the paragraph. This sentence sets the tone and focus for the rest of the paragraph.
2. **Supporting Sentences**: These sentences provide evidence, examples, or explanations to support the topic sentence. They help to develop the main idea by adding detail and depth.
3. **Transitional Phrases**: Transitions are essential for guiding the reader through your ideas. Words like "however," "moreover," "for instance," and "as a result" can help to clarify the connections between your thoughts.
4. **Concluding Sentence**: This is a sentence that sums up the paragraph's main points and often provides a segue into the next paragraph.
5. **Vocabulary and Grammar**: A good paragraph will employ a diverse range of vocabulary and grammatical structures, but without sacrificing clarity.
6. **Cohesion and Coherence**: Your paragraph should be easy to read, with sentences that flow logically from one to the next.
7. **Purposeful Structure**: Every sentence should serve a purpose. Whether it's establishing a fact, providing an example, or concluding an argument, each sentence should contribute to the paragraph's overall point.
8. **Clarity and Precision**: The meaning of the paragraph should be clear, avoiding unnecessary jargon or overly complex sentences.
9. **Tone and Style**: The tone and style should be appropriate to the audience and purpose of the writing.

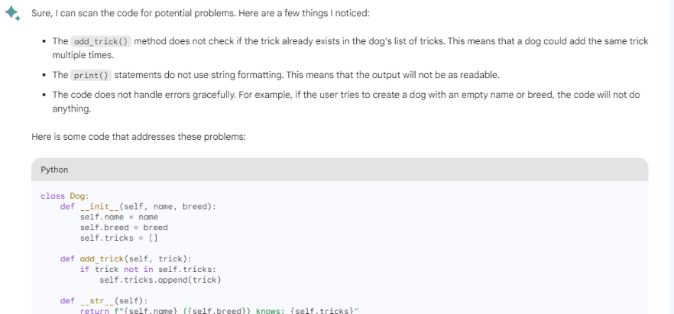
|  |  |
| --- | --- |
| Bad | Good |
| Modify the below Makefile so that emp\_ip\_addr will be assigned the value from script scan-ip.sh. In other words, migrate the code from Bash script scan-ip.sh to Makefile  # BEGIN Makefile  run-sim:  subnet=192.168.81;\  emp\_ip\_addr=$$subnet".2";\  # END Makefile  # BEGIN scan-ip.sh  subnet="192.168.81"  emp\_ip\_addr=""  for ((i = 2; i <= 255; i++)); do  ip="$subnet.$i"  if ! ping -c 1 -w 1 "$ip" &>/dev/null; then  emp\_ip\_addr=$ip  echo "Available IP Address: $emp\_ip\_addr"  break  fi  done  # END scan-ip.sh | Migrate the code from the scan-ip.sh script to the Makefile |
| Output: | Output: |
| run-sim:  subnet=192.168.81; \  emp\_ip\_addr=$$(./scan-ip.sh "$$subnet"); \  echo "Emp IP Address: $$emp\_ip\_addr" | run-sim:  subnet=192.168.81; \  emp\_ip\_addr=""; \  for ((i = 2; i <= 255; i++)); do \  ip="$$subnet.$$i"; \  if ! ping -c 1 -w 1 "$$ip" &>/dev/null; then \  emp\_ip\_addr=$$ip; \  break; \  fi; \  done; \  echo "Emp IP Address: $$emp\_ip\_addr" |

Write CSS code to style h1 headings with Open Sans font, dark blue, size 24.

* **1. Debug Code**

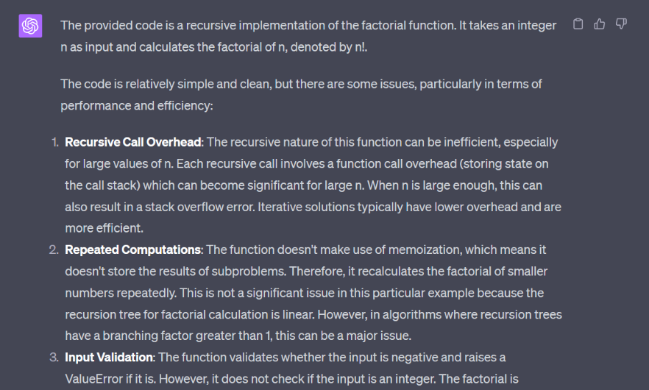
*"Scan the following code for potential problems"*

If you give the AI this problem and insert the source code afterwards, you can find potential problems with the code, even if the code executes successfully. This works in ChatGPT, Bard, GitHub Copilot, and TabNine.



* **2. Improve Performance**

*"Evaluate the following code and look for performance issues" followed by the code in question.*



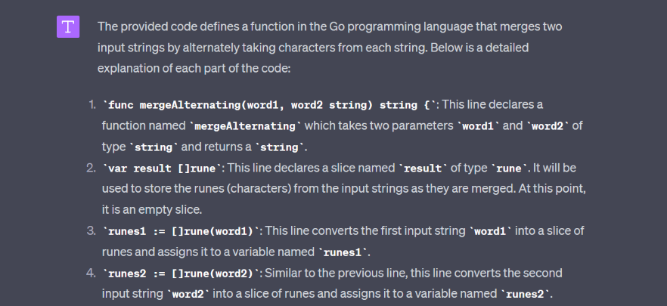
* **3. Generate Tests**

*"Write a test for the following {language} code" followed by the code in question.*



* **4. Explain This Code**

*"Explain how {something} works in {language}."*



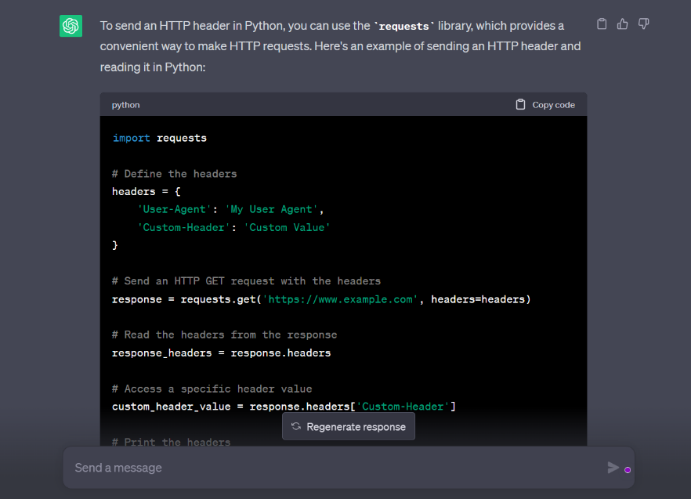
* **5. Translate Code**

*"Translate the following {first language} code into {second language}."*



* **6. Correct Syntax**

*"What is the correct syntax to {do something} in {language}?"*



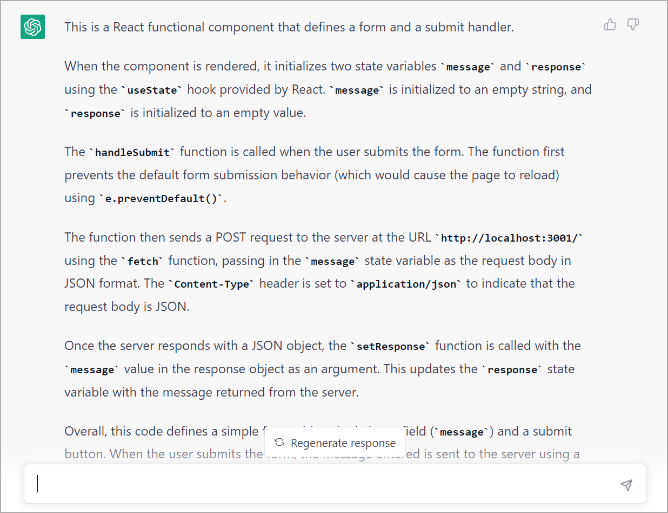
* **7. Write a function**

*"Write a function to {do something} in {language}."*



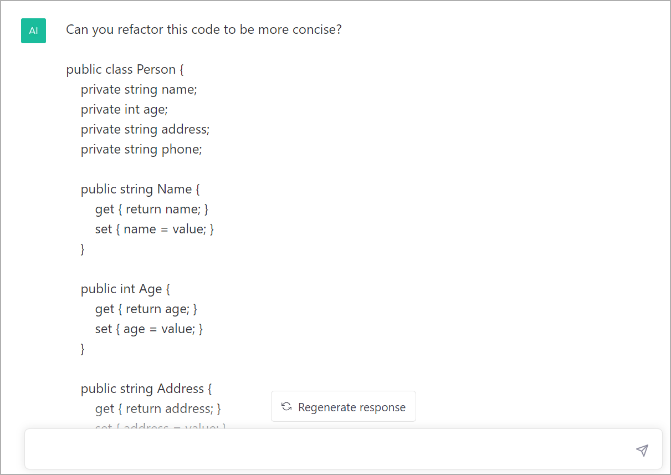
Let’s ask ChatGPT to help.

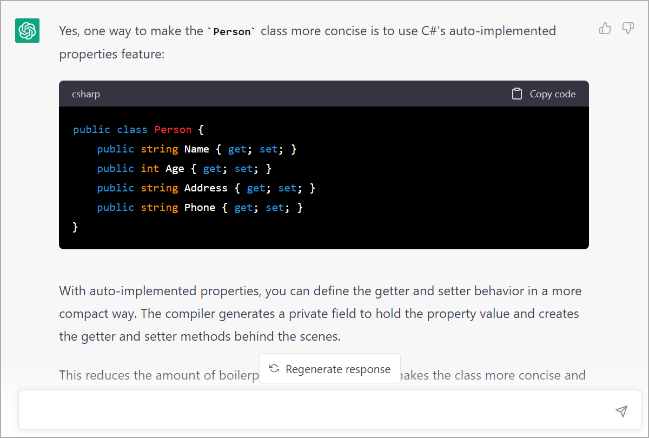




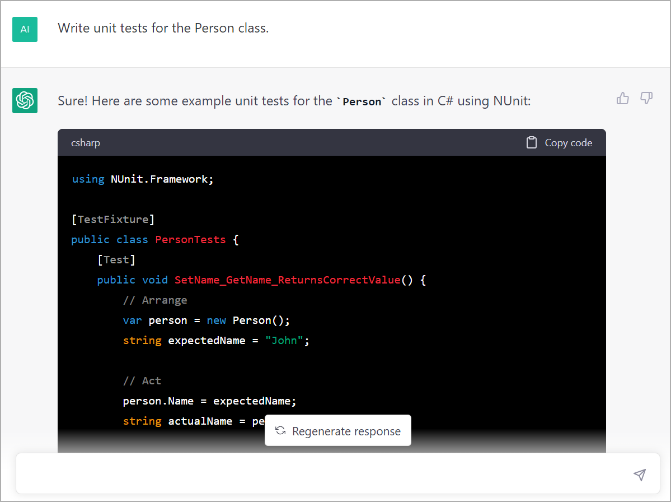
And voila!  An explanation that can help us better understand the code.

How about getting some help refactoring and cleaning up our code?





And let’s see one more example of how to get help with unit testing.  We’ll ask ChatGPT to generate unit tests for the Person class referenced above.



<https://www.promptingguide.ai/introduction/settings>

**Temperature** - In short, the lower the temperature, the more deterministic the results in the sense that the highest probable next token is always picked. Increasing temperature could lead to more randomness, which encourages more diverse or creative outputs. You are essentially increasing the weights of the other possible tokens. In terms of application, you might want to use a lower temperature value for tasks like fact-based QA to encourage more factual and concise responses. For poem generation or other creative tasks, it might be beneficial to increase the temperature value.

1. **Basics of Prompting**
2. **Basic Prompts**

You can achieve a lot with simple prompts, but the quality of results depends on how much information you provide it and how well-crafted it is. A prompt can contain information like the *instruction* or *question* you are passing to the model and include other details such as *context*, *inputs*, or *examples*. You can use these elements to instruct the model better and as a result get better results.

Let's get started by going over a basic example of a simple prompt:

*Prompt*

The sky is

*Output:*

blue The sky is blue on a clear day. On a cloudy day, the sky may be gray or white.

As you can see, the language model outputs a continuation of strings that make sense given the context "The sky is". The output might be unexpected or far from the task you want to accomplish.

This basic example also highlights the necessity to provide more context or instructions on what specifically you want to achieve.

Let's try to improve it a bit:

*Prompt:*

Complete the sentence: The sky is

*Output:*

so beautiful today.

Is that better? Well, you told the model to complete the sentence so the result looks a lot better as it follows exactly what you told it to do ("complete the sentence"). This approach of designing optimal prompts to instruct the model to perform a task is what's referred to as **prompt engineering**.

The example above is a basic illustration of what's possible with LLMs today. Today's LLMs are able to perform all kinds of advanced tasks that range from text summarization to mathematical reasoning to code generation.

1. **Prompt Formatting**

You have tried a very simple prompt above. A standard prompt has the following format:

<Question>?

or

<Instruction>

You can format this into a question answering (QA) format, which is standard in a lot of QA datasets, as follows:

Q: <Question>?

A:

When prompting like the above, it's also referred to as *zero-shot prompting*, i.e., you are directly prompting the model for a response without any examples or demonstrations about the task you want it to achieve. Some large language models do have the ability to perform zero-shot prompting but it depends on the complexity and knowledge of the task at hand.

Given the standard format above, one popular and effective technique to prompting is referred to as *few-shot prompting* where you provide exemplars (i.e., demonstrations). You can format few-shot prompts as follows:

<Question>?

<Answer>

<Question>?

<Answer>

<Question>?

<Answer>

<Question>?

The QA format version would look like this:

Q: <Question>?

A: <Answer>

Q: <Question>?

A: <Answer>

Q: <Question>?

A: <Answer>

Q: <Question>?

A:

Keep in mind that it's not required to use QA format. The prompt format depends on the task at hand. For instance, you can perform a simple classification task and give exemplars that demonstrate the task as follows:

*Prompt:*

This is awesome! // Positive

This is bad! // Negative

Wow that movie was rad! // Positive

What a horrible show! //

*Output:*

Negative

Few-shot prompts enable in-context learning, which is the ability of language models to learn tasks given a few demonstrations.

1. **Elements of a Prompt**

As we cover more and more examples and applications with prompt engineering, you will notice that certain elements make up a prompt.

A prompt contains any of the following elements:

**Instruction** - a specific task or instruction you want the model to perform

**Context** - external information or additional context that can steer the model to better responses

**Input Data** - the input or question that we are interested to find a response for

**Output Indicator** - the type or format of the output.

You do not need all the four elements for a prompt and the format depends on the task at hand. We will touch on more concrete examples in upcoming guides.

|  |  |
| --- | --- |
| 1. **General Tips for Designing Prompts**   Here are some tips to keep in mind while you are designing your prompts:   1. **Start Simple**   As you get started with designing prompts, you should keep in mind that it is really an iterative process that requires a lot of experimentation to get optimal results. Using a simple playground from OpenAI or Cohere is a good starting point.  You can start with simple prompts and keep adding more elements and context as you aim for better results. Iterating your prompt along the way is vital for this reason. As you read the guide, you will see many examples where specificity, simplicity, and conciseness will often give you better results.  When you have a big task that involves many different subtasks, you can try to break down the task into simpler subtasks and keep building up as you get better results. This avoids adding too much complexity to the prompt design process at the beginning.   1. **The Instruction**   You can design effective prompts for various simple tasks by using commands to instruct the model what you want to achieve, such as "Write", "Classify", "Summarize", "Translate", "Order", etc.  Keep in mind that you also need to experiment a lot to see what works best. Try different instructions with different keywords, contexts, and data and see what works best for your particular use case and task. Usually, the more specific and relevant the context is to the task you are trying to perform, the better. We will touch on the importance of sampling and adding more context in the upcoming guides.  Others recommend that you place instructions at the beginning of the prompt. Another recommendation is to use some clear separator like "###" to separate the instruction and context.  For instance:  *Prompt:*  ### Instruction ###  Translate the text below to Spanish:  Text: "hello!"  *Output:*  ¡Hola!   1. **Specificity**   Be very specific about the instruction and task you want the model to perform. The more descriptive and detailed the prompt is, the better the results. This is particularly important when you have a desired outcome or style of generation you are seeking. There aren't specific tokens or keywords that lead to better results. It's more important to have a good format and descriptive prompt. In fact, providing examples in the prompt is very effective to get desired output in specific formats.  When designing prompts, you should also keep in mind the length of the prompt as there are limitations regarding how long the prompt can be. Thinking about how specific and detailed you should be. Including too many unnecessary details is not necessarily a good approach. The details should be relevant and contribute to the task at hand. This is something you will need to experiment with a lot. We encourage a lot of experimentation and iteration to optimize prompts for your applications.  As an example, let's try a simple prompt to extract specific information from a piece of text.  *Prompt:*  Extract the name of places in the following text.  Desired format:  Place: <comma\_separated\_list\_of\_company\_names>  Input: "Although these developments are encouraging to researchers, much is still a mystery. “We often have a black box between the brain and the effect we see in the periphery,” says Henrique Veiga-Fernandes, a neuroimmunologist at the Champalimaud Centre for the Unknown in Lisbon. “If we want to use it in the therapeutic context, we actually need to understand the mechanism.""  *Output:*  Place: Champalimaud Centre for the Unknown, Lisbon  Input text is obtained from [this Nature article(opens in a new tab)](https://www.nature.com/articles/d41586-023-00509-z).   1. **Avoid Impreciseness**   Given the tips above about being detailed and improving format, it's easy to fall into the trap of wanting to be too clever about prompts and potentially creating imprecise descriptions. It's often better to be specific and direct. The analogy here is very similar to effective communication -- the more direct, the more effective the message gets across.  For example, you might be interested in learning the concept of prompt engineering. You might try something like:  Explain the concept prompt engineering. Keep the explanation short, only a few sentences, and don't be too descriptive.  It's not clear from the prompt above how many sentences to use and what style. You might still somewhat get good responses with the above prompts but the better prompt would be one that is very specific, concise, and to the point. Something like:  Use 2-3 sentences to explain the concept of prompt engineering to a high school student.   1. **To do or not to do?**   Another common tip when designing prompts is to avoid saying what not to do but say what to do instead. This encourages more specificity and focuses on the details that lead to good responses from the model.  Here is an example of a movie recommendation chatbot failing at exactly what I don't want it to do because of how I wrote the instruction -- focusing on what not to do.  *Prompt:*  The following is an agent that recommends movies to a customer. DO NOT ASK FOR INTERESTS. DO NOT ASK FOR PERSONAL INFORMATION.Customer: Please recommend a movie based on my interests.Agent:  *Output:*  Sure, I can recommend a movie based on your interests. What kind of movie would you like to watch? Do you prefer action, comedy, romance, or something else?  Here is a better prompt:  *Prompt:*  The following is an agent that recommends movies to a customer. The agent is responsible to recommend a movie from the top global trending movies. It should refrain from asking users for their preferences and avoid asking for personal information. If the agent doesn't have a movie to recommend, it should respond "Sorry, couldn't find a movie to recommend today.".Customer: Please recommend a movie based on my interests.Agent:  *Output:*  Sorry, I don't have any information about your interests. However, here's a list of the top global trending movies right now: [list of movies]. I hope you find something you like!  Some of the examples above were adopted from the ["Best practices for prompt engineering with OpenAI API" article.](https://help.openai.com/en/articles/6654000-best-practices-for-prompt-engineering-with-openai-api) | 1. **Lời khuyên chung cho việc thiết kế lời nhắc**   Dưới đây là một số mẹo cần ghi nhớ khi bạn thiết kế lời nhắc của mình:   1. **Bắt đầu đơn giản**   Khi bắt đầu thiết kế lời nhắc, bạn nên nhớ rằng đây thực sự là một quá trình lặp đi lặp lại, đòi hỏi nhiều thử nghiệm để có được kết quả tối ưu. Sử dụng một sân chơi đơn giản từ OpenAI hoặc Cohere là điểm khởi đầu tốt.  Bạn có thể bắt đầu với những lời nhắc đơn giản và tiếp tục bổ sung thêm nhiều yếu tố cũng như bối cảnh hơn khi bạn hướng tới kết quả tốt hơn. Việc lặp lại lời nhắc của bạn trong suốt quá trình là rất quan trọng vì lý do này. Khi đọc hướng dẫn, bạn sẽ thấy nhiều ví dụ trong đó tính cụ thể, đơn giản và ngắn gọn thường mang lại cho bạn kết quả tốt hơn.  Khi bạn có một nhiệm vụ lớn bao gồm nhiều nhiệm vụ phụ khác nhau, bạn có thể thử chia nhỏ nhiệm vụ đó thành những nhiệm vụ phụ đơn giản hơn và tiếp tục xây dựng khi bạn đạt được kết quả tốt hơn. Điều này tránh tạo thêm quá nhiều sự phức tạp cho quá trình thiết kế ngay từ đầu.   1. **Hướng dẫn**   Bạn có thể thiết kế các lời nhắc hiệu quả cho các tác vụ đơn giản khác nhau bằng cách sử dụng các lệnh để hướng dẫn mô hình những gì bạn muốn đạt được, chẳng hạn như "Write", "Classify", "Summarize", "Translate", "Order", v.v.  Hãy nhớ rằng bạn cũng cần thử nghiệm nhiều để xem điều gì hiệu quả nhất. Hãy thử các hướng dẫn khác nhau với các từ khóa, ngữ cảnh và dữ liệu khác nhau và xem hướng dẫn nào phù hợp nhất cho trường hợp sử dụng và nhiệm vụ cụ thể của bạn. Thông thường, bối cảnh càng cụ thể và phù hợp với nhiệm vụ bạn đang cố gắng thực hiện thì càng tốt. Chúng tôi sẽ đề cập đến tầm quan trọng của việc lấy mẫu và bổ sung thêm ngữ cảnh trong các hướng dẫn sắp tới.  Những người khác khuyên bạn nên đặt hướng dẫn ở đầu lời nhắc. Một khuyến nghị khác là sử dụng một số dấu phân cách rõ ràng như "###" để phân tách hướng dẫn và ngữ cảnh.  Ví dụ:  *Lời nhắc:*  ### Instruction ###Translate the text below to Spanish:Text: "hello!"  *Đầu ra:*  ¡Hola!   1. **Tính đặc hiệu**   Hãy thật cụ thể về hướng dẫn và nhiệm vụ mà bạn muốn mô hình thực hiện. Lời nhắc càng mô tả và chi tiết thì kết quả càng tốt. Điều này đặc biệt quan trọng khi bạn có được kết quả mong muốn hoặc phong cách phát triển mà bạn đang tìm kiếm. Không có mã thông báo hoặc từ khóa cụ thể nào dẫn đến kết quả tốt hơn. Điều quan trọng hơn là phải có định dạng tốt và lời nhắc mang tính mô tả. Trên thực tế, việc cung cấp các ví dụ trong lời nhắc rất hiệu quả để có được kết quả mong muốn ở các định dạng cụ thể.  Khi thiết kế lời nhắc, bạn cũng nên lưu ý độ dài của lời nhắc vì có những hạn chế về độ dài của lời nhắc. Suy nghĩ về việc bạn nên cụ thể và chi tiết như thế nào. Bao gồm quá nhiều chi tiết không cần thiết không hẳn là một cách tiếp cận tốt. Các chi tiết phải phù hợp và đóng góp cho nhiệm vụ hiện tại. Đây là điều bạn sẽ cần phải thử nghiệm rất nhiều. Chúng tôi khuyến khích thực hiện nhiều thử nghiệm và lặp lại để tối ưu hóa lời nhắc cho ứng dụng của bạn.  Ví dụ: hãy thử một lời nhắc đơn giản để trích xuất thông tin cụ thể từ một đoạn văn bản.  *Lời nhắc:*  Extract the name of places in the following text. Desired format:Place: <comma\_separated\_list\_of\_company\_names>Input: "Although these developments are encouraging to researchers, much is still a mystery. “We often have a black box between the brain and the effect we see in the periphery,” says Henrique Veiga-Fernandes, a neuroimmunologist at the Champalimaud Centre for the Unknown in Lisbon. “If we want to use it in the therapeutic context, we actually need to understand the mechanism.""  *Đầu ra:*  Place: Champalimaud Centre for the Unknown, Lisbon  Văn bản đầu vào được lấy từ [bài viết này(mở trong tab mới)](https://www.nature.com/articles/d41586-023-00509-z).   1. **Tránh sự thiếu chính xác**   Với các mẹo ở trên về cách trình bày chi tiết và cải tiến định dạng, bạn rất dễ rơi vào cái bẫy muốn quá thông minh về lời nhắc và có khả năng tạo ra các mô tả không chính xác. Thường thì tốt hơn là nên cụ thể và trực tiếp. Sự tương tự ở đây rất giống với giao tiếp hiệu quả -- thông điệp càng trực tiếp thì càng hiệu quả.  Ví dụ: bạn có thể muốn tìm hiểu khái niệm về kỹ thuật prompt. Bạn có thể thử một cái gì đó như:  Explain the concept prompt engineering. Keep the explanation short, only a few sentences, and don't be too descriptive.  Không rõ gợi ý ở trên nên sử dụng bao nhiêu câu và văn phong như thế nào. Bạn vẫn có thể nhận được phản hồi tốt phần nào với những lời nhắc trên nhưng lời nhắc tốt hơn sẽ là lời nhắc rất cụ thể, ngắn gọn và đi vào trọng tâm. Cái gì đó như:  Use 2-3 sentences to explain the concept of prompt engineering to a high school student.   1. **Làm hay không làm?**   Một mẹo phổ biến khác khi thiết kế lời nhắc là tránh nói những gì không nên làm mà thay vào đó hãy nói những gì cần làm. Điều này khuyến khích tính cụ thể hơn và tập trung vào các chi tiết dẫn đến phản hồi tốt từ mô hình.  Dưới đây là ví dụ về chatbot đề xuất phim không thực hiện được chính xác những gì tôi không muốn nó làm do cách tôi viết hướng dẫn -- tập trung vào những điều không nên làm.  *Lời nhắc:*  The following is an agent that recommends movies to a customer. DO NOT ASK FOR INTERESTS. DO NOT ASK FOR PERSONAL INFORMATION.Customer: Please recommend a movie based on my interests.Agent:  *Đầu ra:*  Sure, I can recommend a movie based on your interests. What kind of movie would you like to watch? Do you prefer action, comedy, romance, or something else?  Đây là một lời nhắc tốt hơn:  *Lời nhắc:*  The following is an agent that recommends movies to a customer. The agent is responsible to recommend a movie from the top global trending movies. It should refrain from asking users for their preferences and avoid asking for personal information. If the agent doesn't have a movie to recommend, it should respond "Sorry, couldn't find a movie to recommend today.".Customer: Please recommend a movie based on my interests.Agent:  *Đầu ra:*  Sorry, I don't have any information about your interests. However, here's a list of the top global trending movies right now: [list of movies]. I hope you find something you like!  Một số ví dụ ở trên được áp dụng từ [bài viết "Các phương pháp thực hành tốt nhất để xử lý prompt với API OpenAI".](https://help.openai.com/en/articles/6654000-best-practices-for-prompt-engineering-with-openai-api) |

<https://www.promptingguide.ai/introduction/examples>

* **Ví dụ về lời nhắc**

Phần trước đã giới thiệu một ví dụ cơ bản về cách nhắc LLM.

Phần này sẽ cung cấp thêm ví dụ về cách sử dụng lời nhắc để hoàn thành các nhiệm vụ khác nhau và giới thiệu các khái niệm chính trong quá trình thực hiện. Thông thường, cách tốt nhất để học các khái niệm là xem qua các ví dụ. Một số ví dụ dưới đây minh họa cách bạn có thể sử dụng các lời nhắc được soạn thảo kỹ lưỡng để thực hiện các loại nhiệm vụ khác nhau.

chủ đề:

* [Tóm tắt văn bản](https://www.promptingguide.ai/introduction/examples#text-summarization)
* [Khai thác thông tin](https://www.promptingguide.ai/introduction/examples#information-extraction)
* [Trả lời câu hỏi](https://www.promptingguide.ai/introduction/examples#question-answering)
* [Phân loại văn bản](https://www.promptingguide.ai/introduction/examples#text-classification)
* [Cuộc hội thoại](https://www.promptingguide.ai/introduction/examples#conversation)
* [Tạo mã](https://www.promptingguide.ai/introduction/examples#code-generation)
* [Lý luận](https://www.promptingguide.ai/introduction/examples#reasoning)
* **Tóm tắt văn bản**

Một trong những nhiệm vụ tiêu chuẩn trong việc tạo ngôn ngữ tự nhiên là tóm tắt văn bản. Tóm tắt văn bản có thể bao gồm nhiều hương vị và lĩnh vực khác nhau. Trên thực tế, một trong những ứng dụng hứa hẹn nhất của mô hình ngôn ngữ là khả năng tóm tắt các bài viết, khái niệm thành những bản tóm tắt prompt và dễ đọc. Hãy thử thực hiện một tác vụ tóm tắt cơ bản bằng lời nhắc.

Giả sử bạn muốn tìm hiểu về thuốc kháng sinh, bạn có thể thử lời nhắc như sau:

*Lời nhắc:*

Explain antibioticsA:

*Đầu ra:*

Antibiotics are a type of medication used to treat bacterial infections. They work by either killing the bacteria or preventing them from reproducing, allowing the body’s immune system to fight off the infection. Antibiotics are usually taken orally in the form of pills, capsules, or liquid solutions, or sometimes administered intravenously. They are not effective against viral infections, and using them inappropriately can lead to antibiotic resistance.

"A:" là định dạng nhắc nhở rõ ràng mà bạn sử dụng khi trả lời câu hỏi. Bạn đã sử dụng nó ở đây để nói với mô hình rằng có một câu trả lời cần được giải đáp thêm. Trong ví dụ này, không rõ điều này hữu ích như thế nào so với việc không sử dụng nó nhưng chúng tôi sẽ để nó cho các ví dụ sau. Giả sử rằng đây là quá nhiều thông tin và bạn muốn tóm tắt thêm. Trên thực tế, bạn có thể hướng dẫn mô hình tóm tắt thành một câu như sau:

*Lời nhắc:*

Antibiotics are a type of medication used to treat bacterial infections. They work by either killing the bacteria or preventing them from reproducing, allowing the body’s immune system to fight off the infection. Antibiotics are usually taken orally in the form of pills, capsules, or liquid solutions, or sometimes administered intravenously. They are not effective against viral infections, and using them inappropriately can lead to antibiotic resistance.Explain the above in one sentence:

*Đầu ra:*

Antibiotics are medications used to treat bacterial infections by either killing the bacteria or stopping them from reproducing, but they are not effective against viruses and overuse can lead to antibiotic resistance.

Không chú ý quá nhiều đến tính chính xác của kết quả đầu ra ở trên, đó là điều chúng tôi sẽ đề cập đến trong hướng dẫn sau, mô hình đã cố gắng tóm tắt đoạn văn trong một câu. Bạn có thể trở nên thông minh hơn với các hướng dẫn nhưng chúng tôi sẽ để việc đó ở chương sau. Hãy tạm dừng ở đây và thử nghiệm để xem liệu bạn có nhận được kết quả tốt hơn không.

* **Khai thác thông tin**

Mặc dù các mô hình ngôn ngữ được đào tạo để thực hiện việc tạo ngôn ngữ tự nhiên và các nhiệm vụ liên quan, nhưng nó cũng có khả năng thực hiện phân loại và một loạt các nhiệm vụ xử lý ngôn ngữ tự nhiên (NLP) khác.

Dưới đây là ví dụ về lời nhắc trích xuất thông tin từ một đoạn văn nhất định.

*Lời nhắc:*

Author-contribution statements and acknowledgements in research papers should state clearly and specifically whether, and to what extent, the authors used AI technologies such as ChatGPT in the preparation of their manuscript and analysis. They should also indicate which LLMs were used. This will alert editors and reviewers to scrutinize manuscripts more carefully for potential biases, inaccuracies and improper source crediting. Likewise, scientific journals should be transparent about their use of LLMs, for example when selecting submitted manuscripts.Mention the large language model based product mentioned in the paragraph above:

*Đầu ra:*

The large language model based product mentioned in the paragraph above is ChatGPT.

Có nhiều cách để bạn có thể cải thiện kết quả trên nhưng cách này đã rất hữu ích rồi.

Đến bây giờ, rõ ràng là bạn có thể yêu cầu mô hình thực hiện các nhiệm vụ khác nhau bằng cách chỉ dẫn nó phải làm gì. Đó là khả năng mạnh mẽ mà các nhà phát triển sản phẩm AI đang sử dụng để xây dựng các sản phẩm và trải nghiệm mạnh mẽ.

Nguồn đoạn: [ChatGPT: năm ưu tiên nghiên cứu(mở trong tab mới)](https://www.nature.com/articles/d41586-023-00288-7)

* **Trả lời câu hỏi**

Một trong những cách tốt nhất để mô hình phản hồi các câu trả lời cụ thể là cải thiện định dạng của lời nhắc. Như đã đề cập trước đó, lời nhắc có thể kết hợp các chỉ dẫn, ngữ cảnh, đầu vào và đầu ra để có được kết quả được cải thiện. Mặc dù những thành phần này không bắt buộc nhưng nó sẽ trở thành một phương pháp thực hành tốt vì bạn càng hướng dẫn cụ thể thì bạn sẽ nhận được kết quả càng tốt. Dưới đây là ví dụ về cách hiển thị sau lời nhắc có cấu trúc chặt chẽ hơn.

*Lời nhắc:*

Answer the question based on the context below. Keep the answer short and concise. Respond "Unsure about answer" if not sure about the answer.Context: Teplizumab traces its roots to a New Jersey drug company called Ortho Pharmaceutical. There, scientists generated an early version of the antibody, dubbed OKT3. Originally sourced from mice, the molecule was able to bind to the surface of T cells and limit their cell-killing potential. In 1986, it was approved to help prevent organ rejection after kidney transplants, making it the first therapeutic antibody allowed for human use.Question: What was OKT3 originally sourced from?Answer:

*Đầu ra:*

Mice.

Bối cảnh thu được từ [Thiên nhiên(mở trong tab mới)](https://www.nature.com/articles/d41586-023-00400-x).

* **Phân loại văn bản**

Cho đến nay, bạn đã sử dụng các hướng dẫn đơn giản để thực hiện một tác vụ. Là một kỹ sư nhanh nhẹn, bạn cần phải giỏi hơn trong việc đưa ra những hướng dẫn tốt hơn. Nhưng đó không phải là tất cả! Bạn cũng sẽ thấy rằng đối với những trường hợp sử dụng khó hơn, chỉ cung cấp hướng dẫn là không đủ. Đây là lúc bạn cần suy nghĩ nhiều hơn về ngữ cảnh và các yếu tố khác nhau mà bạn có thể sử dụng trong lời nhắc. Các phần tử khác bạn có thể cung cấp là input datahoặc examples.

Hãy thử chứng minh điều này bằng cách cung cấp một ví dụ về phân loại văn bản.

*Lời nhắc:*

Classify the text into neutral, negative or positive. Text: I think the food was okay. Sentiment:

*Đầu ra:*

Neutral

Bạn đã đưa ra hướng dẫn phân loại văn bản và mô hình đã phản hồi bằng 'Neutral', điều này đúng. Điều này không có gì sai nhưng giả sử rằng điều bạn thực sự cần là mô hình cung cấp nhãn theo định dạng chính xác mà bạn muốn. Vì vậy, thay vì Neutral, bạn muốn nó trả về neutral. Làm thế nào để bạn đạt được điều này? Có nhiều cách khác nhau để làm điều này. Bạn quan tâm đến tính cụ thể ở đây, vì vậy bạn càng cung cấp nhiều thông tin prompt thì kết quả càng tốt. Bạn có thể thử cung cấp các ví dụ để chỉ định hành vi đúng. Hãy thử lại lần nữa:

*Lời nhắc:*

Classify the text into neutral, negative or positive. Text: I think the vacation is okay.Sentiment: neutral Text: I think the food was okay. Sentiment:

*Đầu ra:*

neutral

Hoàn hảo! Lần này mô hình trả về neutralchính là nhãn cụ thể mà bạn đang tìm kiếm. Có vẻ như ví dụ được cung cấp trong lời nhắc đã giúp mô hình có kết quả cụ thể.

Để làm nổi bật lý do tại sao đôi khi việc cụ thể lại quan trọng, hãy xem ví dụ bên dưới và phát hiện ra vấn đề:

*Lời nhắc:*

Classify the text into nutral, negative or positive. Text: I think the vacation is okay.Sentiment:

*Đầu ra:*

Neutral

vấn đề ở đây là gì? Như một gợi ý, nhãn hiệu trang điểm nutralhoàn toàn bị model bỏ qua. Thay vào đó, mô hình xuất ra Neutralvì nó có một số sai lệch đối với nhãn đó. Nhưng hãy giả sử rằng điều bạn thực sự muốn là nutral. Bạn sẽ khắc phục điều này như thế nào? Có lẽ bạn có thể thử thêm mô tả vào nhãn hoặc thêm nhiều ví dụ khác vào lời nhắc? Nếu bạn không chắc chắn, chúng ta sẽ thảo luận một vài ý tưởng trong các phần sắp tới.

* **Cuộc hội thoại**

Có lẽ một trong những điều thú vị hơn mà bạn có thể đạt được với kỹ thuật prompt là hướng dẫn hệ thống LLM về cách ứng xử, mục đích và danh tính của nó. Điều này đặc biệt hữu ích khi bạn đang xây dựng các hệ thống đàm thoại như chatbot dịch vụ khách hàng.

Ví dụ: hãy tạo một hệ thống đàm thoại có thể tạo ra các câu trả lời mang tính kỹ thuật và khoa học hơn cho các câu hỏi. Lưu ý cách bạn nói rõ ràng với nó cách ứng xử thông qua hướng dẫn. Điều này đôi khi được gọi là *nhắc nhở vai trò* .

*Lời nhắc:*

The following is a conversation with an AI research assistant. The assistant tone is technical and scientific.Human: Hello, who are you?AI: Greeting! I am an AI research assistant. How can I help you today?Human: Can you tell me about the creation of blackholes?AI:

*Đầu ra:*

Sure! Black holes are regions of spacetime where the gravitational force is so strong that nothing, not even light, can escape from it. They are created when a very massive star dies and its core collapses in on itself, forming a singularity of infinite density. The intense gravity of the singularity pulls in all the matter and radiation around it, creating the black hole.

Trợ lý nghiên cứu AI của chúng tôi nghe có vẻ hơi quá kỹ thuật phải không? Được rồi, hãy thay đổi hành vi này và hướng dẫn hệ thống đưa ra câu trả lời dễ tiếp cận hơn.

*Lời nhắc:*

The following is a conversation with an AI research assistant. The assistant answers should be easy to understand even by primary school students.Human: Hello, who are you?AI: Greeting! I am an AI research assistant. How can I help you today?Human: Can you tell me about the creation of black holes?AI:

*Đầu ra:*

Sure! Black holes are created when a very massive star runs out of fuel and collapses in on itself. The gravity of the star is so strong that nothing, not even light, can escape its pull. This creates a point in space-time with an intense gravitational force called a black hole.

Bạn đã đạt được một số tiến bộ. Bạn có thể tiếp tục cải thiện nó. Nếu bạn thêm nhiều ví dụ hơn, bạn có thể nhận được kết quả tốt hơn nữa.

* **Tạo mã**

Một ứng dụng mà LLM khá hiệu quả là tạo mã. Phi công phụ là một ví dụ tuyệt vời về điều này. Có rất nhiều tác vụ tạo mã mà bạn có thể thực hiện với những lời nhắc thông minh. Chúng ta hãy xem một vài ví dụ dưới đây.

Đầu tiên, hãy thử một chương trình đơn giản chào đón người dùng.

*Lời nhắc:*

/\*Ask the user for their name and say "Hello"\*/

*Đầu ra:*

let name = prompt("What is your name?");console.log(`Hello, ${name}!`);

Lưu ý rằng bạn thậm chí không cần chỉ định ngôn ngữ sẽ sử dụng.

Hãy chuyển đổi cấp độ một chút. Ví dụ bên dưới cho thấy LLM có thể mạnh mẽ như thế nào nếu bạn nỗ lực hơn một chút trong việc thiết kế lời nhắc.

*Lời nhắc:*

"""Table departments, columns = [DepartmentId, DepartmentName]Table students, columns = [DepartmentId, StudentId, StudentName]Create a MySQL query for all students in the Computer Science Department"""

*Đầu ra:*

SELECT StudentId, StudentName FROM students WHERE DepartmentId IN (SELECT DepartmentId FROM departments WHERE DepartmentName = 'Computer Science');

Điều này rất ấn tượng. Trong trường hợp này, bạn đã cung cấp dữ liệu về lược đồ cơ sở dữ liệu và yêu cầu nó tạo một truy vấn MySQL hợp lệ.

* **Lý luận**

Có lẽ một trong những nhiệm vụ khó khăn nhất đối với LLM ngày nay là nhiệm vụ đòi hỏi một số hình thức lý luận. Lý luận là một trong những lĩnh vực thú vị nhất do các loại ứng dụng phức tạp có thể xuất hiện từ LLM.

Đã có một số cải tiến trong các nhiệm vụ liên quan đến khả năng toán học. Điều đó nói lên rằng, điều quan trọng cần lưu ý là các LLM hiện tại gặp khó khăn trong việc thực hiện các nhiệm vụ lý luận nên điều này đòi hỏi các kỹ thuật kỹ thuật prompt thậm chí còn tiên tiến hơn. Chúng tôi sẽ đề cập đến các kỹ thuật nâng cao này trong hướng dẫn tiếp theo. Hiện tại, chúng tôi sẽ đề cập đến một số ví dụ cơ bản để thể hiện khả năng số học.

*Lời nhắc:*

What is 9,000 \* 9,000?

*Đầu ra:*

81,000,000

Hãy thử điều gì đó khó khăn hơn.

*Lời nhắc:*

The odd numbers in this group add up to an even number: 15, 32, 5, 13, 82, 7, 1. A:

*đầu ra*

No, the odd numbers in this group add up to an odd number: 119.

Điều đó không đúng! Hãy cố gắng cải thiện điều này bằng cách cải thiện lời nhắc.

*Lời nhắc:*

The odd numbers in this group add up to an even number: 15, 32, 5, 13, 82, 7, 1. Solve by breaking the problem into steps. First, identify the odd numbers, add them, and indicate whether the result is odd or even.

*Đầu ra:*

Odd numbers: 15, 5, 13, 7, 1Sum: 41 41 is an odd number.

Tốt hơn nhiều, phải không? Nhân tiện, tôi đã thử điều này một vài lần và hệ thống đôi khi bị lỗi. Nếu bạn cung cấp hướng dẫn tốt hơn kết hợp với ví dụ, điều đó có thể giúp nhận được kết quả chính xác hơn.

Chúng tôi sẽ tiếp tục đưa thêm nhiều ví dụ về các ứng dụng phổ biến vào phần hướng dẫn này.

Trong phần sắp tới, chúng tôi sẽ đề cập đến các khái niệm và kỹ thuật kỹ thuật prompt nâng cao hơn nữa để cải thiện hiệu suất trên tất cả những nhiệm vụ này và những nhiệm vụ khó khăn hơn.

<https://www.pluralsight.com/blog/software-development/how-use-chatgpt-programming-coding>

|  |  |
| --- | --- |
| * **How to use ChatGPT to write code** * By Amber Israelsen    |    March 22, 2023   **Greetings, developers!** Have you ever had a case of “coder’s block” where you need a little help or inspiration to write code?  Or need some help understanding code that you inherited from someone else?  Well, you’re in luck!  From generating boilerplate code to debugging or explaining existing code, ChatGPT is a no-brainer way to be a faster, more efficient software engineer.  OpenAI’s ChatGPT falls into the category of [generative AI](https://www.pluralsight.com/courses/chatgpt-generative-ai-big-picture), which is a type of artificial intelligence that generates new or novel content based on data it’s been trained on.  That means ChatGPT can not only write essays, jokes and poems for you; it can also be your pair programmer.  So let’s dive in and see how you can use it to work smarter, not harder.   * **Table of contents** * [**Can ChatGPT write code?**](https://www.pluralsight.com/blog/software-development/how-use-chatgpt-programming-coding#can-it-code) * [**Should you code with ChatGPT?**](https://www.pluralsight.com/blog/software-development/how-use-chatgpt-programming-coding#should-you-code) * [**Is ChatGPT coding trustworthy?**](https://www.pluralsight.com/blog/software-development/how-use-chatgpt-programming-coding#is-it-trustworthy) * [**How does ChatGPT help to grow your coding skills?**](https://www.pluralsight.com/blog/software-development/how-use-chatgpt-programming-coding#will-my-skills-grow) * [**How else can I use ChatGPT as a developer?**](https://www.pluralsight.com/blog/software-development/how-use-chatgpt-programming-coding#how-else-use) * [**How to use ChatGPT as a programmer: ChatGPT Prompt Examples**](https://www.pluralsight.com/blog/software-development/how-use-chatgpt-programming-coding#prompt-examples) * [**Will ChatGPT replace programmers? Can I use ChatGPT to do my job for me?**](https://www.pluralsight.com/blog/software-development/how-use-chatgpt-programming-coding#will-chatgpt-replace-me) * [**Conclusion**](https://www.pluralsight.com/blog/software-development/how-use-chatgpt-programming-coding#conclusion) * **Can ChatGPT write code?**   As a large language model, [ChatGPT](https://openai.com/blog/chatgpt) is all about text.  And that text includes—lucky for you!—code.  It has the ability to write code snippets in popular programming languages such as JavaScript, Python, C#, PHP and Java.  Or generate sample CSV input for a data science problem.  Or create bash scripts or infrastructure-as-code templates in JSON or YAML.  And a lot more.  Let’s see an example.  Maybe you need some boilerplate code for a simple index.html page that says “Hello World.”  Using a natural language request, you can turn ChatGPT loose to solve this problem.  ChatGPT generating hello world  Not only does ChatGPT generate some (copy-pasteable) code, but it also explains what the code is doing, which can be especially helpful if you’re new to the language.  Need to style this webpage using some CSS?  Let’s ask!  Write CSS code to style h1 headings with Open Sans font, dark blue, size 24.  Certainly! Here's an example of CSS code that will style all `h1` headings with the Open Sans font, dark blue color, and font size of 24:  ChatGPT generating CSS  I know what you’re thinking: “I can use AI to write CSS?  YESSS!  If you tell me it can write regular expressions too, then I’m totally sold.”  (Hint: it can!)   * **Should you code with ChatGPT?**   So.  Just because you*can* write code with ChatGPT, does that mean you *should*?  There are a variety of opinions on this.  On one side, people argue that using an AI tool to write code makes you lazy and unable to think for yourself.  On the other side, folks say that leveraging ChatGPT can make you a faster programmer, freeing up your time to focus on the more important things like architecture, best practices and business requirements.  I can see both sides of the argument.  Let’s take a look at some pros and cons.   * **The pros** * It’s quick and easy to generate code using ChatGPT.  If you were going to StackOverflow or Google to find a code snippet anyway, why not shorten the amount of time it takes to find it? * ChatGPT generally gets syntax right, potentially saving you time chasing compile-time or runtime errors that we humans can introduce if we’re coding from scratch. * By using ChatGPT for coding, you might get introduced to some alternative ways of doing things that you wouldn’t have thought about otherwise. * ChatGPT makes it faster and easier to learn new languages and concepts, with explanations and code all in the same interface. * **The cons** * While artificial intelligence has come a long way, and ChatGPT is quite frankly mind-blowing, it’s not 100% correct all the time.  It makes mistakes just like us.  And sometimes it’s really confident when it makes those mistakes.  So you still need to check it, test it and debug it-–just like you’ve done in the past. * ChatGPT lacks the overall context for what you’re building and why.  Sure, it can give you a code snippet, or even an entire code file.  But it won’t understand conventions or best practices for your company or project.  It doesn’t know how that code will interact with other code.  It doesn’t understand requirements around performance, security, privacy, accessibility and so on.  Again, as a human, you’re ultimately responsible for the code that ships. * **Is ChatGPT coding trustworthy?**   Let’s see what ChatGPT has to say about its own trustworthiness.  ChatGPT talking about if ChatGPT code is safe  Even ChatGPT is aware of its own limitations!  Just as you wouldn’t rely on your phone’s autocorrect functionality to send an important text message, you shouldn’t rely on ChatGPT for writing perfect code.  Take secure coding, for example.  Remember that ChatGPT works by “learning” a huge set of existing data.  In the case of ChatGPT version 3, it was trained on data through the end of 2021.  Because the cybersecurity landscape is constantly changing, and new vulnerabilities appear every day, ChatGPT won’t have the latest intelligence to inform the code it writes.  ChatGPT has no knowledge of your specific requirements for authentication, secrets management, third-party components, vulnerability scanning and so on.  It might give you boilerplate code with a placeholder for hard-coded credentials.  If you don’t know that those should be stored in an environment variable or in a secrets management service, then this is a huge security risk for your application.  Ultimately, while ChatGPT is good for writing code snippets and simple applications, human developers have the critical job of putting the pieces together and applying best practices.   * **How does ChatGPT help to grow your coding skills?**   When you need to learn a new programming language or feature, what do you do today?  You might search for tutorials, watch courses, try writing and running code to see what it does.  This likely involves jumping around to several different resources, which can be time-consuming.  And how many times do you encounter errors trying to run code you found online?  For me, it’s almost always!  ChatGPT can reduce the amount of time it takes to do all of this.  It effectively aggregates information from the many sources it was trained on, and presents them to you in a single interface.  It also explains what the code does, and sometimes even gives you alternative ways of doing things.  It’s like having a real-time tutor with you as you’re writing and reading the code.  Another huge bonus of learning through ChatGPT is the fact that it uses natural language processing for input and output.  Meaning, just ask questions in plain English, as if you were asking a mentor or a friend.  You don’t have to know exact technical terms or syntax to ask the question.  You can also refine and build out your conversation over time.  If you didn’t quite “get” a concept that it explained, you can ask something like, “Can you explain that in a simpler way?”  Or, “Can you explain that for a 5-year-old?”  Being able to tailor responses to your current level of understanding is a pretty remarkable benefit.   * **How else can I use ChatGPT as a developer?**   So far, we’ve focused on using ChatGPT to write code, but there are several other ways it can make you faster at your job.  For example, it can explain code for you, help you refactor, write test cases, and even fix bugs.  Let’s say you found some code on StackOverflow, but you don’t quite understand what it does.  Let’s ask ChatGPT to help.  Asking ChatGPT about code snippets  Asking ChatGPT about code snippets answer  And voila!  An explanation that can help us better understand the code.  How about getting some help refactoring and cleaning up our code?  More refactoring with ChatGPT  ChatGPT refactoring code  And let’s see one more example of how to get help with unit testing.  We’ll ask ChatGPT to generate unit tests for the Person class referenced above.  ChatGPT writing unit tests  As you can see, there are tons of ways you can use ChatGPT to speed up everyday tasks.  As always, you’ll want to validate the output it gives you, but as a starting point, this is pretty hard to beat.  Check out [this short video](https://www.youtube.com/watch?v=PsIhcpzwf7U) for additional ways to use ChatGPT as a developer.   * **How to use ChatGPT as a programmer: ChatGPT Prompt Examples**   As AI tools become more and more prevalent in our lives, the way to get an “edge” is to know how to use them effectively.  For ChatGPT specifically, that means you’ll need to be able to write effective prompts to get the answers you want, and quickly.  Here are some templates you can use.   * **To write new code**   **Prompt template:**Act as a *[technical role]*.  Write *[a description of the thing you want to write, using as much detail as you can.]*  **Example:**Act as a front-end developer.  Write code to validate a contact form that includes name, email address and message.  The email address is required and should be in the format of a valid email.  If the form is successfully submitted, display a success message.   * **To write new code for a specific tech stack**   **Prompt template:** Act as a *[technical role]*.  The tech stack is *[describe the different technologies you’re using]*.  Write *[a description of the thing(s) you’re building]*.  **Example:**Act as a full-stack developer.  The tech stack is AWS Lambda, DynamoDB, API Gateway and HTML.  Write a Lambda function to calculate the sum of two numbers input by the user and save them to DynamoDB.  Write an HTML file with input boxes for two numbers.  When the user clicks submit, pass the two numbers to API Gateway.   * **To debug existing code**   **Prompt template:** I am getting the following error in my code *[paste in the error]*.  How can I fix it?  **Example:**I am getting the following error in my code: Module not found: Error: Can't resolve './app'.  How can I fix it?   * **To evaluate different technologies**   **Prompt template:**Act as a *[technical role]*.  Describe two different technologies to *[accomplish some task]*.  Give me pros and cons, along with examples.  **Example:**Act as a data analyst.  Describe two different technologies to do exploratory data analysis.  Give me pros and cons, along with examples.   * **Will ChatGPT replace programmers? Can I use ChatGPT to do my job for me?**   ChatGPT is very impressive, and it launched the “AI revolution” into overdrive, with new tools and features coming out every day.  And because it’s continuously learning through user feedback and new data, it will continue to get better and better.  Which begs the question: Will ChatGPT take my job?  As a programmer, remember that your ultimate job is to translate business requirements into code. That requires a lot of analysis, creativity, critical thinking, weighing different options, architecting, optimization and so on.  Humans are much better at those skills than robots are, and I don’t think ChatGPT threatens that.  Also, as we’ve seen from the examples above, you need to understand how a technology works in order to ask the right questions and validate the answers that ChatGPT gives you.  In other words, you need to be a skilled developer to effectively use ChatGPT for developer-related things.  But let it do the tedious work that you likely have to search for anyway.  In short, I think ChatGPT will be used to accelerate work, just like IntelliSense, static code analysis, autocomplete, compile-time error checking, automated testing, IDEs and other developer tools have for years.  It’s another tool in the toolkit to make you a better, faster programmer.   * **Conclusion**   And that’s a wrap!  Hopefully by now, you have a good idea of how ChatGPT can be used to boost your day-to-day software development skills.  Whether you’re a seasoned developer who wants to do less typing, or a newbie wanting to learn a new programming language, ChatGPT is a powerful tool to make it happen.  It’s like having an expert pair programmer at your side (though don’t forget you still need to validate the expert!).  To learn more about artificial intelligence and ChatGPT, check out these additional resources:   * [ChatGPT and Generative AI: The Big Picture](https://www.pluralsight.com/courses/chatgpt-generative-ai-big-picture) * [Artificial Intelligence Explained](https://app.pluralsight.com/library/courses/artificial-intelligence-explained) * [Artificial Intelligence: The Big Picture of AI](https://app.pluralsight.com/library/courses/artificial-intelligence-big-picture) | * **Cách dùng ChatGPT để viết code** * Bởi Amber Israelsen | Ngày 22 tháng 3 năm 2023   **Xin chào, các nhà phát triển!**Bạn đã bao giờ gặp trường hợp “ngăn cản lập trình viên” khi bạn cần một chút trợ giúp hoặc nguồn cảm hứng để viết code chưa? Hoặc cần một số trợ giúp để hiểu code mà bạn đã thừa hưởng từ người khác? Chà, bạn thật may mắn! Từ việc tạo code soạn sẵn cho đến gỡ lỗi hoặc giải thích code hiện có, ChatGPT là một cách dễ dàng để trở thành một kỹ sư phần mềm nhanh hơn, hiệu quả hơn.  ChatGPT của OpenAI thuộc danh mục [AI tổng quát](https://www.pluralsight.com/courses/chatgpt-generative-ai-big-picture) , là một loại trí tuệ nhân tạo tạo ra nội dung mới hoặc nội dung mới lạ dựa trên dữ liệu mà nó đã được đào tạo. Điều đó có nghĩa là ChatGPT không chỉ có thể viết tiểu luận, truyện cười và thơ cho bạn; nó cũng có thể là lập trình viên cặp của bạn.  Vì vậy, hãy đi sâu vào và xem cách bạn có thể sử dụng nó để làm việc thông minh hơn chứ không phải chăm chỉ hơn.   * **Mục lục** * [**ChatGPT có thể viết code không?**](https://www.pluralsight.com/blog/software-development/how-use-chatgpt-programming-coding#can-it-code) * [**Bạn có nên viết code bằng ChatGPT không?**](https://www.pluralsight.com/blog/software-development/how-use-chatgpt-programming-coding#should-you-code) * [**Code ChatGPT có đáng tin cậy không?**](https://www.pluralsight.com/blog/software-development/how-use-chatgpt-programming-coding#is-it-trustworthy) * [**ChatGPT giúp phát triển kỹ năng code của bạn như thế nào?**](https://www.pluralsight.com/blog/software-development/how-use-chatgpt-programming-coding#will-my-skills-grow) * [**Làm cách nào khác để tôi có thể sử dụng ChatGPT với tư cách là nhà phát triển?**](https://www.pluralsight.com/blog/software-development/how-use-chatgpt-programming-coding#how-else-use) * [**Cách sử dụng ChatGPT với tư cách là một lập trình viên: Ví dụ nhắc về ChatGPT**](https://www.pluralsight.com/blog/software-development/how-use-chatgpt-programming-coding#prompt-examples) * [**ChatGPT sẽ thay thế các lập trình viên? Tôi có thể sử dụng ChatGPT để thực hiện công việc của mình không?**](https://www.pluralsight.com/blog/software-development/how-use-chatgpt-programming-coding#will-chatgpt-replace-me) * [**Phần kết luận**](https://www.pluralsight.com/blog/software-development/how-use-chatgpt-programming-coding#conclusion) * **ChatGPT có thể viết code không?**   Là một mô hình ngôn ngữ lớn, [ChatGPT](https://openai.com/blog/chatgpt) hoàn toàn là về văn bản. Và văn bản đó bao gồm—may cho bạn!—code. Nó có khả năng viết các đoạn code bằng các ngôn ngữ lập trình phổ biến như JavaScript, Python, C#, PHP và Java. Hoặc tạo đầu vào CSV mẫu cho một vấn đề về khoa học dữ liệu. Hoặc tạo tập lệnh bash hoặc mẫu cơ sở hạ tầng dưới dạng code trong JSON hoặc YAML. Và còn nhiều nữa.  Hãy xem một ví dụ. Có thể bạn cần một số code soạn sẵn cho một trang index.html đơn giản có nội dung “Xin chào thế giới”. Sử dụng yêu cầu ngôn ngữ tự nhiên, bạn có thể biến ChatGPT lỏng lẻo để giải quyết vấn đề này.  ChatGPT không chỉ tạo ra một số code (có thể sao chép và dán được) mà còn giải thích chức năng của code, điều này có thể đặc biệt hữu ích nếu bạn là người mới sử dụng ngôn ngữ này.  Bạn cần tạo kiểu cho trang web này bằng cách sử dụng một số CSS? Thử hỏi xem sao!  Tôi biết bạn đang nghĩ gì: “Tôi có thể sử dụng AI để viết CSS? CÓ! Nếu bạn nói với tôi rằng nó cũng có thể viết các biểu thức chính quy, thì tôi hoàn toàn bị thuyết phục.” (Gợi ý: có thể!)   * **Bạn có nên viết code bằng ChatGPT không?**   Chỉ vì bạn *có thể* viết code bằng ChatGPT, điều đó có nghĩa là bạn *nên làm vậy* ? Có nhiều ý kiến ​​khác nhau về điều này. **Một bên, mọi người lập luận rằng việc sử dụng công cụ AI để viết code khiến bạn trở nên lười biếng và không thể tự suy nghĩ. Mặt khác, mọi người nói rằng việc tận dụng ChatGPT có thể giúp bạn trở thành một lập trình viên nhanh hơn, giải phóng thời gian của bạn để tập trung vào những thứ quan trọng hơn như kiến ​​trúc, các phương pháp hay nhất và yêu cầu kinh doanh**.  Tôi có thể thấy cả hai mặt của cuộc tranh luận. Chúng ta hãy xem xét một số ưu và nhược điểm.   * **những ưu điểm** * Thật prompt và dễ dàng để tạo code bằng ChatGPT. Nếu bạn vẫn truy cập StackOverflow hoặc Google để tìm một đoạn code, tại sao không rút ngắn thời gian cần thiết để tìm đoạn code đó? * ChatGPT thường có cú pháp đúng, có khả năng giúp bạn tiết kiệm thời gian tìm kiếm các lỗi thời gian biên dịch hoặc thời gian chạy mà con người chúng ta có thể mắc phải nếu chúng ta viết code từ đầu. * Bằng cách sử dụng ChatGPT để viết code, bạn có thể được giới thiệu một số cách khác để thực hiện những việc mà bạn chưa từng nghĩ tới. * ChatGPT giúp việc học các ngôn ngữ và khái niệm mới nhanh hơn và dễ dàng hơn, với tất cả các giải thích và code trong cùng một giao diện. * **khuyết điểm** * Mặc dù trí tuệ nhân tạo đã đi một chặng đường dài và ChatGPT khá thẳng thắn, nhưng không phải lúc nào nó cũng chính xác 100%. Nó phạm sai lầm giống như chúng ta. Và đôi khi nó thực sự tự tin khi mắc phải những lỗi đó. Vì vậy, bạn vẫn cần kiểm tra, chạy thử và gỡ lỗi-–giống như bạn đã làm trong quá khứ. * ChatGPT thiếu bối cảnh tổng thể cho những gì bạn đang xây dựng và lý do. Chắc chắn, nó có thể cung cấp cho bạn một đoạn code hoặc thậm chí toàn bộ tệp code. Nhưng nó sẽ không hiểu các quy ước hoặc phương pháp hay nhất cho công ty hoặc dự án của bạn. Nó không biết code đó sẽ tương tác với code khác như thế nào. Nó không hiểu các yêu cầu xung quanh hiệu suất, bảo mật, quyền riêng tư, khả năng truy cập, v.v. Một lần nữa, với tư cách là một con người, bạn chịu trách nhiệm cuối cùng về code được vận chuyển. * **Code ChatGPT có đáng tin cậy không?**   Hãy xem ChatGPT nói gì về độ tin cậy của chính nó.  Ngay cả ChatGPT cũng nhận thức được những hạn chế của chính nó! Giống như việc bạn không dựa vào chức năng tự động sửa lỗi của điện thoại để gửi một tin nhắn văn bản quan trọng, bạn cũng không nên dựa vào ChatGPT để viết code hoàn hảo.  Lấy code an toàn, ví dụ. Hãy nhớ rằng ChatGPT hoạt động bằng cách “học” một bộ dữ liệu khổng lồ hiện có. Đối với ChatGPT phiên bản 3, ChatGPT đã được đào tạo dựa trên dữ liệu cho đến cuối năm 2021. Do bối cảnh an ninh mạng liên tục thay đổi và các lỗ hổng mới xuất hiện hàng ngày nên ChatGPT sẽ không có thông tin mới nhất để cung cấp thông tin cho code mà nó viết.  ChatGPT không biết gì về các yêu cầu cụ thể của bạn đối với xác thực, quản lý bí mật, các thành phần của bên thứ ba, quét lỗ hổng, v.v. Nó có thể cung cấp cho bạn code soạn sẵn với trình giữ chỗ cho thông tin đăng nhập được code cứng. Nếu bạn không biết rằng chúng nên được lưu trữ trong một biến môi trường hoặc trong một dịch vụ quản lý bí mật, thì đây là một rủi ro bảo mật lớn cho ứng dụng của bạn.  Cuối cùng, trong khi ChatGPT phù hợp để viết các đoạn code và ứng dụng đơn giản, thì các nhà phát triển con người có nhiệm vụ quan trọng là ghép các phần lại với nhau và áp dụng các phương pháp hay nhất.   * **ChatGPT giúp phát triển kỹ năng code của bạn như thế nào?**   Khi bạn cần học một ngôn ngữ lập trình hoặc tính năng mới, bạn sẽ làm gì hôm nay? Bạn có thể tìm kiếm các hướng dẫn, xem các khóa học, thử viết và chạy code để xem những gì nó làm được. Điều này có thể liên quan đến việc nhảy xung quanh một số tài nguyên khác nhau, có thể tốn thời gian. Và bao nhiêu lần bạn gặp lỗi khi cố chạy code bạn tìm thấy trực tuyến? Đối với tôi, nó gần như luôn luôn!  ChatGPT có thể giảm lượng thời gian cần thiết để thực hiện tất cả những điều này. Nó tổng hợp thông tin một cách hiệu quả từ nhiều nguồn mà nó đã được đào tạo và trình bày chúng cho bạn trong một giao diện duy nhất. Nó cũng giải thích chức năng của code và đôi khi còn cung cấp cho bạn những cách khác để thực hiện công việc. Nó giống như có một gia sư thời gian thực đồng hành cùng bạn khi bạn viết và đọc code.  Một phần thưởng lớn khác của việc học qua ChatGPT là nó sử dụng xử lý ngôn ngữ tự nhiên cho đầu vào và đầu ra. Có nghĩa là, chỉ đặt câu hỏi bằng tiếng Anh đơn giản, như thể bạn đang hỏi một người cố vấn hoặc một người bạn. Bạn không cần phải biết chính xác các thuật ngữ kỹ thuật hoặc cú pháp để đặt câu hỏi.  Bạn cũng có thể tinh chỉnh và xây dựng cuộc trò chuyện của mình theo thời gian. Nếu bạn không hoàn toàn “hiểu” một khái niệm mà nó giải thích, bạn có thể hỏi những câu như: “Bạn có thể giải thích điều đó theo cách đơn giản hơn không?” Hoặc, "Bạn có thể giải thích điều đó cho một đứa trẻ 5 tuổi không?" Có thể điều chỉnh các câu trả lời phù hợp với mức độ hiểu biết hiện tại của bạn là một lợi ích khá đáng chú ý.   * **Làm cách nào khác để tôi có thể sử dụng ChatGPT với tư cách là nhà phát triển?**   Cho đến nay, chúng tôi đã tập trung vào việc sử dụng ChatGPT để viết code, nhưng có một số cách khác có thể giúp bạn thực hiện công việc nhanh hơn. Ví dụ: nó có thể giải thích code cho bạn, giúp bạn cấu trúc lại, viết các trường hợp thử nghiệm và thậm chí sửa lỗi.  Giả sử bạn đã tìm thấy một số code trên StackOverflow, nhưng bạn không hiểu lắm về chức năng của nó. Hãy nhờ ChatGPT giúp đỡ.  Và Voila! Một lời giải thích có thể giúp chúng tôi hiểu rõ hơn về code.  Làm thế nào về việc nhận một số trợ giúp tái cấu trúc và làm sạch code của chúng tôi?  Và hãy xem thêm một ví dụ về cách nhận trợ giúp với thử nghiệm đơn vị. Chúng tôi sẽ yêu cầu ChatGPT tạo các bài kiểm tra đơn vị cho lớp Person được tham chiếu ở trên.  Như bạn có thể thấy, có rất nhiều cách bạn có thể sử dụng ChatGPT để tăng tốc các tác vụ hàng ngày. Như mọi khi, bạn sẽ muốn xác thực đầu ra mà nó mang lại cho bạn, nhưng ngay từ đầu, điều này khá khó để đánh bại.  Hãy xem [video ngắn này](https://www.youtube.com/watch?v=PsIhcpzwf7U) để biết thêm các cách sử dụng ChatGPT với tư cách là nhà phát triển.   * **Cách sử dụng ChatGPT với tư cách là một lập trình viên: Ví dụ nhắc về ChatGPT**   Khi các công cụ AI ngày càng trở nên phổ biến trong cuộc sống của chúng ta, cách để đạt được “lợi thế” là biết cách sử dụng chúng một cách hiệu quả. Đối với ChatGPT cụ thể, điều đó có nghĩa là bạn sẽ cần có khả năng viết lời nhắc hiệu quả để nhận được câu trả lời bạn muốn một cách prompt. Dưới đây là một số mẫu bạn có thể sử dụng.   * **Để viết code mới**   **Mẫu lời nhắc:** Đóng vai trò *[vai trò kỹ thuật]* . Viết *[mô tả về điều bạn muốn viết, càng chi tiết càng tốt.]*  **Ví dụ:** Đóng vai trò là nhà phát triển front-end. Viết code để xác thực biểu mẫu liên hệ bao gồm tên, địa chỉ email và tin nhắn. Địa chỉ email là bắt buộc và phải ở định dạng email hợp lệ. Nếu biểu mẫu được gửi thành công, hiển thị thông báo thành công.   * **Để viết code mới cho một ngăn xếp công nghệ cụ thể**   **Mẫu lời nhắc:** Đóng vai trò *[vai trò kỹ thuật]* . Ngăn xếp công nghệ là *[mô tả các công nghệ khác nhau mà bạn đang sử dụng]* . Viết *[mô tả về (những) thứ bạn đang xây dựng]* .  **Ví dụ:** Đóng vai trò là nhà phát triển full-stack. Ngăn xếp công nghệ là AWS Lambda, DynamoDB, API Gateway và HTML. Viết hàm Lambda để tính tổng của hai số do người dùng nhập và lưu chúng vào DynamoDB. Viết một tệp HTML có các hộp nhập cho hai số. Khi người dùng nhấp vào gửi, hãy chuyển hai số tới API Gateway.   * **Để gỡ lỗi code hiện có**   **Mẫu lời nhắc:** Tôi gặp lỗi sau trong code của mình *[dán vào lỗi]* . Làm thế nào tôi có thể sửa chữa nó?  **Ví dụ:** Tôi gặp phải lỗi sau trong code của mình: Không tìm thấy mô-đun: Lỗi: Không thể giải quyết './app'. Làm thế nào tôi có thể sửa chữa nó?   * **Để đánh giá các công nghệ khác nhau**   **Mẫu lời nhắc:** Đóng vai trò *[vai trò kỹ thuật]* . Mô tả hai công nghệ khác nhau để *[hoàn thành một số nhiệm vụ]* . Hãy cho tôi những ưu và nhược điểm, cùng với các ví dụ.  **Ví dụ:** Đóng vai trò là nhà phân tích dữ liệu. Mô tả hai công nghệ khác nhau để thực hiện phân tích dữ liệu khám phá. Hãy cho tôi những ưu và nhược điểm, cùng với các ví dụ.   * **ChatGPT sẽ thay thế các lập trình viên? Tôi có thể sử dụng ChatGPT để thực hiện công việc của mình không?**   ChatGPT rất ấn tượng và nó đã khởi động “cuộc cách mạng AI” với các công cụ và tính năng mới ra mắt mỗi ngày. Và bởi vì nó liên tục học hỏi thông qua phản hồi của người dùng và dữ liệu mới, nó sẽ tiếp tục ngày càng tốt hơn. Điều này đặt ra câu hỏi: Liệu ChatGPT có đảm nhận công việc của tôi không?  Là một lập trình viên, hãy nhớ rằng công việc cuối cùng của bạn là dịch các yêu cầu kinh doanh thành code. Điều đó đòi hỏi rất nhiều phân tích, sáng tạo, tư duy phản biện, cân nhắc các lựa chọn khác nhau, kiến ​​trúc, tối ưu hóa, v.v. Con người giỏi hơn nhiều ở những kỹ năng đó so với rô-bốt và tôi không nghĩ ChatGPT đe dọa điều đó.  Ngoài ra, như chúng ta đã thấy từ các ví dụ trên, bạn cần hiểu cách thức hoạt động của một công nghệ để đặt câu hỏi phù hợp và xác thực câu trả lời mà ChatGPT cung cấp cho bạn. Nói cách khác, bạn cần phải là một nhà phát triển lành nghề để sử dụng ChatGPT một cách hiệu quả cho những việc liên quan đến nhà phát triển. Nhưng hãy để nó làm công việc tẻ nhạt mà bạn có thể phải tìm kiếm.  Tóm lại, tôi nghĩ ChatGPT sẽ được sử dụng để tăng tốc công việc, giống như IntelliSense, phân tích code tĩnh, tự động hoàn thành, kiểm tra lỗi thời gian biên dịch, kiểm tra tự động, IDE và các công cụ dành cho nhà phát triển khác đã có trong nhiều năm. Đó là một công cụ khác trong bộ công cụ giúp bạn trở thành một lập trình viên giỏi hơn, nhanh hơn.   * **Phần kết luận**   Và đó là một bọc! Hy vọng rằng đến bây giờ, bạn đã hiểu rõ về cách sử dụng ChatGPT để nâng cao kỹ năng phát triển phần mềm hàng ngày của mình. Cho dù bạn là một nhà phát triển dày dạn kinh nghiệm muốn gõ ít hơn hay một người mới muốn học một ngôn ngữ lập trình mới, ChatGPT là một công cụ mạnh mẽ để biến điều đó thành hiện thực. Nó giống như có một lập trình viên cặp chuyên gia ở bên cạnh bạn (mặc dù đừng quên rằng bạn vẫn cần xác thực chuyên gia đó!).  Để tìm hiểu thêm về trí tuệ nhân tạo và ChatGPT, hãy xem các tài nguyên bổ sung sau:   * [ChatGPT và AI sáng tạo: Bức tranh toàn cảnh](https://www.pluralsight.com/courses/chatgpt-generative-ai-big-picture) * [Trí tuệ nhân tạo giải thích](https://app.pluralsight.com/library/courses/artificial-intelligence-explained) * [Trí tuệ nhân tạo: Bức tranh lớn về AI](https://app.pluralsight.com/library/courses/artificial-intelligence-big-picture) |

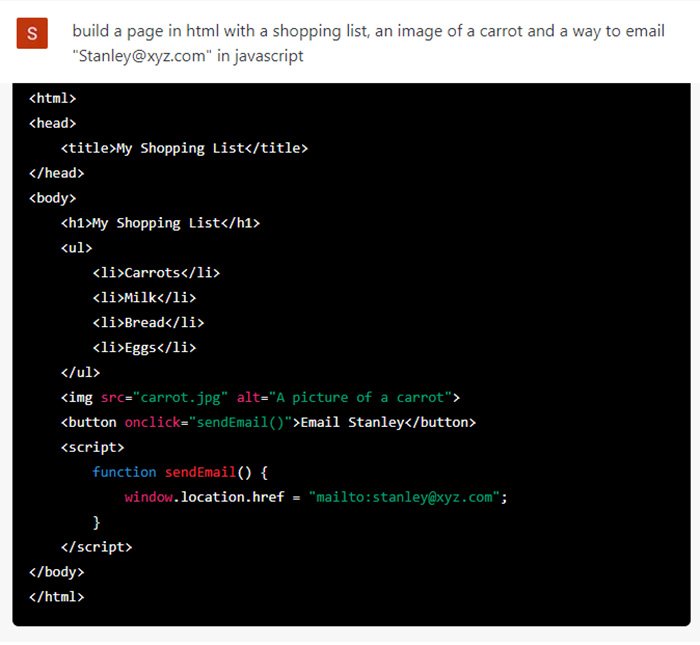
|  |  |
| --- | --- |
| **Example: Getting it to spin up and test Python code**  Write a Python function called add\_numbers that takes a list of numbers as an input and returns their sum. If the input is not a list or if it contains elements that are not numbers, the function should raise a ValueError with an appropriate error message.  Giving ChatGPT a request to create a python program  A python program made by ChatGPT  Of course it didn’t test it, because I didn’t specify. That's on me, and a prime example of poor [prompt engineering](https://www.pluralsight.com/resources/blog/software-development/prompt-engineering-for-developers)! So I specifically asked it to run the test, and it did.  ChatGPT testing out some python code and giving an output  Much better! Keep in mind that even if it’s testing its own code, you should treat it just like a human and not just push directly to production. Whether it’s coding or any other business process, Gen AI is strongest when you build business processes around it to mitigate any potential risks from using it.  This means code reviewing what ChatGPT produces, testing it yourself, limiting the scope of deployment, sandboxing the generated code, and having a rollback strategy. It’s the same for tools like [self-fixing and auto-code AI](https://www.pluralsight.com/blog/machine-learning/regenerative-auto-code-ai-gpt): make sure you’re not just blindly accepting the code that ChatGPT is feeding you.  **3. Math equations**  If you hate (nay, [omniloathe](https://www.urbandictionary.com/define.php?term=omniloathe)) math like me, you probably want to handball it to a computer. ChatGPT has been famously bad at math because it’s simply guessing the next word in a sentence. However, now it can build Python code to solve problems, and it does this fairly well.  **Example: Solving simple, text-based math**  A prompt for a simple math question  ChatGPT's answer to a simple math question  It’s correct, but as any student knows, you can’t just have the answer - you’ve got to show how you got there. So let’s get ChatGPT to show us their logic.  ChatGPT showing how it worked out some math using python  Looks sound. That was a pretty easy question, though. Could we give it something more qualitative? | **Ví dụ: Làm cho nó quay lên và kiểm tra code Python**  Tất nhiên nó không kiểm tra nó, vì tôi không chỉ định. Đó là tại tôi, và là một ví dụ điển hình về  [kỹ thuật kém prompt](https://www.pluralsight.com/resources/blog/software-development/prompt-engineering-for-developers) ! Vì vậy, tôi đã đặc biệt yêu cầu nó chạy thử nghiệm, và nó đã làm được.  Tốt hơn nhiều! Hãy nhớ rằng ngay cả khi nó đang kiểm tra code của chính nó, bạn nên đối xử với nó như một con người và không chỉ đẩy trực tiếp vào sản xuất. Cho dù đó là code hay bất kỳ quy trình kinh doanh nào khác, Gen AI sẽ mạnh nhất khi bạn xây dựng các quy trình kinh doanh xung quanh nó để giảm thiểu mọi rủi ro tiềm ẩn khi sử dụng nó.  Điều này có nghĩa là code xem xét những gì ChatGPT tạo ra, tự kiểm tra code đó, giới hạn phạm vi triển khai, hộp cát code được tạo và có chiến lược khôi phục. Điều này cũng tương tự đối với các công cụ như [AI tự sửa lỗi và code tự động](https://www.pluralsight.com/blog/machine-learning/regenerative-auto-code-ai-gpt) : hãy đảm bảo rằng bạn không mù quáng chấp nhận code mà ChatGPT đang cung cấp cho bạn.  **3. Phương trình toán học**  Nếu bạn ghét toán học (không,  [không phải ai cũng biết](https://www.urbandictionary.com/define.php?term=omniloathe) ) như tôi, bạn có thể muốn ném nó vào máy tính. ChatGPT nổi tiếng là kém môn toán vì nó chỉ đơn giản là đoán từ tiếp theo trong một câu. Tuy nhiên, bây giờ nó có thể xây dựng code Python để giải quyết các vấn đề và nó làm điều này khá tốt.  **Ví dụ: Giải toán đơn giản có lời văn**  Điều đó đúng, nhưng như bất kỳ sinh viên nào cũng biết, bạn không thể chỉ có câu trả lời - bạn phải chỉ ra cách bạn đạt được điều đó. Vì vậy, hãy để ChatGPT cho chúng tôi thấy logic của họ.  Có vẻ đúng. Đó là một câu hỏi khá dễ dàng, mặc dù. Chúng ta có thể cung cấp cho nó một cái gì đó chất lượng hơn? |

<https://www.pluralsight.com/resources/blog/software-development/prompt-engineering-for-developers>

|  |  |
| --- | --- |
| **Prompt engineering 101 for developers**  How you can get generative AI (like ChatGPT and Google Bard) to be better at helping you debug and translate code, generate tests, write functions, and more.  **By Jeremy Morgan**  Jul 11, 2023 • 7 Minute Read   * **Software Development** * **AI & Machine Learning** * **Learning & Development**   [Subscribe to the newsletter](https://www.pluralsight.com/subscribe)  Prompt engineering is the latest term thrown around when talking about [generative AI systems](https://www.pluralsight.com/blog/machine-learning/what-is-chatgpt-generative-ai). Far from a buzzword — like, say, the metaverse — prompt engineering is a crucial method for you to fine tune and optimize the responses you get from language models.  Prompt engineering can really be summed up as this: if you want better results, you need to ask better questions. This article will help you do just that.  Before we get started, the usual coding caveat applies: don't use code without validating it! You can and will get incorrect results, so make sure you understand what is being generated. Think of these tools as something to help your efficiency, rather than write all the code for you.   * **A bit more on Prompt Engineering**   A lazy example of a prompt  Have you ever got a vague brief from a client or coworker, with no scope or details (like the screenshot above)? It’s a recipe for disaster, right? You don’t know what they want, and if you proceed anyway, you’ll likely get it wrong.  Normally, you’d ask some follow up questions and get some clarity around the brief. However, AI models aren’t that intuitive. Like any machine, they do exactly what you tell them to: garbage in, garbage out. So the onus is on the user to make sure they provide the right ‘brief’.  That’s what prompt engineering is in a nutshell: providing effective prompts or instructions to an AI model to get accurate, effective responses. As a developer, this allows you to get output suitable for specific applications and tasks.  That said, no matter how good your prompt engineering skills are, you are still limited by the capabilities of the model itself. The quality and diversity of the model’s training data is crucial.   * **Wait, is prompt engineering just another word for Google-Fu?**   For those familiar with the earlier days of the internet, this might sound an awful lot like “Google-Fu”, a term for your skill at creating a successful search engine query. These days, that’s pretty much second nature to everyone who uses the internet.  There are a lot of similarities between the two: you’re putting in a prompt, and trying to get a certain result. However, one of the key differences between prompt engineering and Google-Fu is that an AI model will not give the same response every time, whereas a search engine will — it’s a bit of a lottery. Engineers can guide the response from the model with prompt engineering, and trial and error is required.  Just like with Google-Fu, it’s likely one day we’ll all intuitively be prompt engineers. However, since generative AI is in its early stages, the level of technical literacy isn’t there yet.   * **The basics of prompt engineering**   Right, now let’s get down to actually creating a great prompt. To maximize the effectiveness of prompt engineering, you should focus on the**structure of the prompt**, your **phrasing**, and **context**. We'll show examples of these principles in this article.   * **What your starting prompt should include**   Introductions are important. While you don’t need to say “Hi, I’m Joe McHuman, nice to meet you”, your first prompt with the AI should contain the following structure:   1. **Introduction:**Set up the context for which you're chatting in. It helps to give the AI an imaginary ‘role’ to think of themselves in. e.g. “Act as a software engineer. You're an expert in Python and …” 2. **Task:**e.g. “I want you to develop software to manage my record collection.” 3. **Contextual Information:**e.g. “I want it to be a web based application written in Python.” 4. **Instructions:**e.g. “I want you to generate the code to write the program.” 5. **Closing:**e.g. “I want to host it as an AWS Lambda function.”   So, putting all of the above together, our example prompt would look like this:  *"Act as a software engineer. You're an expert in Python and AWS technologies. I want you to develop software to manage my record collection. Make it a web based application written in Python. Generate the code for the program, and I want to host it as a Lambda function."*  And with a tool like ChatGPT, it immediately gets to work:  Writing an opening prompt for an AI model  Now you have a conversation started about how to develop your application (and you didn’t even need to make any small talk!).  Now, while it’s important to structure your prompts, remember **you don’t need to stick to the above structure faithfully**. If you don’t need a portion of information, don’t fill it in just to tick a box. Sometimes, less is more!   * **General prompt engineering guidelines:**   Whether you’re crafting an opening prompt or five exchanges into a conversation, consider the following guidelines to get the best results.   * **1. Keep your prompts clear and concise**   Don’t confuse the poor AI. Clear and concise prompts help the AI model understand the intended task more easily. Avoid ambiguity and unnecessary complexity so it’s easily able to figure out what you’re after.   * **2. Consider context and specificity**   Context is important. If you ask your friend if saying “I’m sorry” and “My bad” mean the same thing, they might say yes. However, they might say no if you add the context of being at a funeral.  By giving an AI model the right contextual information, it’s able to give you a more relevant and accurate answer (e.g. “Actually, Joe McHuman, I wouldn’t recommend saying that at the wake.”) Incorporate relevant contextual cues within your prompt, and make sure the prompt is specific to the intended task, avoiding generic or vague instructions.   * **3. Use a conversational style**   This might sound a bit odd when you’re talking to a robot, but keep it natural and conversational. Pretend you’re talking to a really smart person who’s helping you out, and keep the language simple and accessible.  Why? Remember that what you put in is what you get out. This conversational approach helps create a more human-like interaction and makes the AI’s output easier to understand.   * **4. Leverage active voice and rhetorical questions**   For those not familiar with the term, active voice is not giving an AI an instruction while you go for a jog. It’s structuring a sentence like this:   * **Active Voice:** Can you write a persuasive essay on the benefits of **exercise?**(Subject + Verb + Object) * **Passive Voice:** An essay on the benefits of exercise is to be written by you. (Object + Verb + Subject)   In the above example, “You” is the subject, “Write” is the verb, and “A persuasive essay” is the object.  *Why is any of this important?* Active voice in prompts imparts a sense of action and clarity to the AI model.It emphasizes what you're asking for and encourages a focused response.  Additionally, incorporating rhetorical questions within prompts can stimulate the AI model's "thinking process" and guide it towards generating more insightful and meaningful outputs.   * **Example prompts for coders**   Here are some great example prompts to use when you’re writing code. These will help you in the development process to speed up development time and help find errors.   * **1. Debug Code**   *"Scan the following code for potential problems"*  If you give the AI this problem and insert the source code afterwards, you can find potential problems with the code, even if the code executes successfully. This works in ChatGPT, Bard, GitHub Copilot, and TabNine.  Debugging code with GenAI   * **2. Improve Performance**   *"Evaluate the following code and look for performance issues" followed by the code in question.*  Improving performance of code with AI   * **3. Generate Tests**   *"Write a test for the following {language} code" followed by the code in question.*  Generating coding tests using ai models   * **4. Explain This Code**   *"Explain how {something} works in {language}."*  Explaining how code works in a set language   * **5. Translate Code**   *"Translate the following {first language} code into {second language}."*  Translating code from one language to another with AI   * **6. Correct Syntax**   *"What is the correct syntax to {do something} in {language}?"*  Getting the correct syntax for code with generative AI   * **7. Write a function**   *"Write a function to {do something} in {language}."*  Write a function to do something in a set language.   * **Conclusion**   These are a few examples of prompt engineering. It's an art that takes a lot of trial and error, but it's worth it in the long run! Continue to experiment and test until you get the best results.  To see some more examples, or learn more about generative AI, check out the following articles:   * [How to use ChatGPT to write code](https://www.pluralsight.com/blog/software-development/how-use-chatgpt-programming-coding) * [ChatGPT Vs Bard: Which is better for coding?](https://www.pluralsight.com/blog/software-development/chatgpt-vs-bard-coding) * [ChatGPT plugins: How to make ChatGPT smarter](https://www.pluralsight.com/blog/machine-learning/chat-gpt-plugins-guide) * [Self-Fixing and Auto-Code with AI: Debating the Pros and Cons](https://www.pluralsight.com/blog/machine-learning/regenerative-auto-code-ai-gpt) * [Pigeon VS AI: Is a wild bird smarter than artificial intelligence?](https://www.pluralsight.com/blog/machine-learning/pigeon-vs-ai-associated-learning) | **Đưa lời nhắc prompt kỹ thuật 101 cho developer**  Cách bạn có thể làm cho AI tổng quát (như ChatGPT và Google Bard) trở nên tốt hơn trong việc giúp bạn gỡ lỗi và dịch code, tạo các bài kiểm tra, viết hàm, v.v.  **Bởi Jeremy Morgan**  Ngày 11 tháng 7 năm 2023 • Đọc 7 phút   * **Phát triển phần mềm** * **Trí tuệ nhân tạo & Máy học** * **Học hỏi & Phát triển**   [Đăng ký nhận bản tin](https://www.pluralsight.com/subscribe)  Kỹ thuật đưa lời nhắc prompt là thuật ngữ mới nhất được sử dụng khi nói về [các hệ thống AI tổng quát](https://www.pluralsight.com/blog/machine-learning/what-is-chatgpt-generative-ai) . Khác xa với một từ thông dụng — chẳng hạn như metaverse — kỹ thuật đưa lời nhắc prompt là một phương pháp quan trọng để bạn tinh chỉnh và tối ưu hóa các phản hồi bạn nhận được từ các mô hình ngôn ngữ.  Kỹ thuật đưa lời nhắc prompt thực sự có thể được tóm tắt như sau: nếu bạn muốn có kết quả tốt hơn, bạn cần đặt những câu hỏi hay hơn. Bài viết này sẽ giúp bạn làm điều đó.  Trước khi chúng tôi bắt đầu, hãy áp dụng cảnh báo code thông thường: không sử dụng code mà không xác thực code đó! Bạn có thể và sẽ nhận được kết quả không chính xác, vì vậy hãy đảm bảo rằng bạn hiểu những gì đang được tạo ra. Hãy nghĩ về những công cụ này như một thứ giúp ích cho hiệu quả của bạn, thay vì viết tất cả code cho bạn.   * **Thêm một chút về Kỹ thuật đưa lời nhắc prompt**   **Bạn đã bao giờ nhận được một bản tóm tắt mơ hồ từ khách hàng hoặc đồng nghiệp, không có phạm vi hoặc chi tiết** (như ảnh chụp màn hình ở trên) chưa? Đó là một công thức cho thảm họa, phải không? Bạn không biết họ muốn gì và nếu bạn vẫn tiếp tục, rất có thể bạn sẽ hiểu sai.  Thông thường, bạn sẽ hỏi một số câu hỏi tiếp theo và hiểu rõ hơn về bản tóm tắt. Tuy nhiên, các mô hình AI không có trực giác như vậy. Giống như bất kỳ cỗ máy nào, chúng thực hiện chính xác những gì bạn yêu cầu: rác vào, rác ra. Vì vậy, người dùng có trách nhiệm đảm bảo rằng họ cung cấp 'bản tóm tắt' phù hợp.  Tóm lại, đó chính là kỹ thuật đưa lời nhắc prompt: cung cấp lời đưa lời nhắc prompt hoặc hướng dẫn hiệu quả cho mô hình AI để nhận được phản hồi chính xác, hiệu quả. Là một developer, điều này cho phép bạn có được đầu ra phù hợp với các ứng dụng và tác vụ cụ thể.  Điều đó nói rằng, cho dù kỹ năng kỹ thuật đưa lời nhắc prompt của bạn tốt đến đâu, bạn vẫn bị giới hạn bởi khả năng của chính mô hình. Chất lượng và sự đa dạng của dữ liệu đào tạo của mô hình là rất quan trọng.   * **Đợi đã, kỹ thuật đưa lời nhắc prompt chỉ là một từ khác của Google-Fu?**   Đối với những người quen thuộc với những ngày đầu của Internet, điều này nghe có vẻ rất giống với “Google-Fu”, một thuật ngữ chỉ kỹ năng của bạn trong việc tạo truy vấn công cụ tìm kiếm thành công. Ngày nay, đó gần như là bản chất thứ hai đối với tất cả những người sử dụng internet.  Có rất nhiều điểm tương đồng giữa hai điều này: bạn đang đặt một dấu đưa lời nhắc prompt và cố gắng đạt được một kết quả nhất định. Tuy nhiên, một trong những điểm khác biệt chính giữa kỹ thuật đưa lời nhắc prompt và Google-Fu là mô hình AI sẽ không đưa ra phản hồi giống nhau mọi lúc, trong khi công cụ tìm kiếm sẽ - đó là một trò xổ số. Các kỹ sư có thể hướng dẫn phản hồi từ mô hình bằng kỹ thuật đưa lời nhắc prompt và yêu cầu thử và sai.  Cũng giống như với Google-Fu, có khả năng một ngày nào đó tất cả chúng ta sẽ trở thành những kỹ sư prompt bằng trực giác. Tuy nhiên, vì AI tổng quát đang ở giai đoạn đầu nên mức độ hiểu biết về kỹ thuật vẫn chưa có.   * **Những điều cơ bản của kỹ thuật đưa lời nhắc prompt**   Đúng vậy, bây giờ chúng ta hãy bắt tay vào việc thực sự tạo ra một dấu đưa lời nhắc prompt tuyệt vời. Để tối đa hóa hiệu quả của kỹ thuật đưa lời nhắc prompt, bạn nên tập trung vào cấu **trúc của lời đưa lời nhắc prompt** , **cách diễn đạt** và **ngữ cảnh** của bạn . Chúng tôi sẽ đưa ra các ví dụ về các nguyên tắc này trong bài viết này.   * **Lời đưa lời nhắc prompt bắt đầu của bạn nên bao gồm những gì**   Giới thiệu là quan trọng. Mặc dù bạn không cần phải nói “Xin chào, tôi là Joe McHuman, rất vui được gặp bạn”, lời đưa lời nhắc prompt đầu tiên của bạn với AI phải có cấu trúc sau:   1. **Giới thiệu:** Thiết lập ngữ cảnh mà bạn đang trò chuyện. Nó giúp cung cấp cho AI một 'vai diễn' tưởng tượng để tự nghĩ về chúng. ví dụ: “Đóng vai một kỹ sư phần mềm. Bạn là một chuyên gia về Python và…” 2. **Nhiệm vụ:** ví dụ: “Tôi muốn bạn phát triển phần mềm để quản lý bộ sưu tập hồ sơ của tôi.” 3. **Thông tin theo ngữ cảnh:** ví dụ: “Tôi muốn nó là một ứng dụng dựa trên web được viết bằng Python.” 4. **Hướng dẫn:** ví dụ: “Tôi muốn bạn generate code để viết chương trình.” 5. **Kết thúc:** ví dụ: “Tôi muốn host nó dưới dạng hàm AWS Lambda.”   Vì vậy, đặt tất cả những điều trên lại với nhau, dấu đưa lời nhắc prompt ví dụ của chúng ta sẽ như thế này:  *"Hãy hành động như một kỹ sư phần mềm. Bạn là chuyên gia về Python và công nghệ AWS. Tôi muốn bạn phát triển phần mềm để quản lý bộ sưu tập bản ghi của tôi. Biến phần mềm này thành một ứng dụng dựa trên web được viết bằng Python. Tạo code cho chương trình và tôi muốn để lưu trữ nó dưới dạng hàm Lambda."*  Và với một công cụ như ChatGPT, nó sẽ ngay lập tức hoạt động:  Bây giờ bạn đã bắt đầu cuộc trò chuyện về cách phát triển ứng dụng của mình (và bạn thậm chí không cần phải thực hiện bất kỳ cuộc trò chuyện nhỏ nào!).  Bây giờ, mặc dù việc cấu trúc các lời đưa lời nhắc prompt của bạn là rất quan trọng, nhưng hãy nhớ rằng **bạn không cần phải trung thành với cấu trúc trên** . Nếu bạn không cần một phần thông tin, đừng điền thông tin đó chỉ để đánh dấu vào một ô. Đôi khi ít lại hơn!   * **Hướng dẫn kỹ thuật đưa lời nhắc prompt chung:**   Cho dù bạn đang tạo một lời đưa lời nhắc prompt mở đầu hay năm trao đổi thành một cuộc trò chuyện, hãy xem xét các nguyên tắc sau để đạt được kết quả tốt nhất.   * **1. Giữ lời đưa lời nhắc prompt của bạn rõ ràng và ngắn gọn**   Đừng làm AI tội nghiệp bối rối. Lời đưa lời nhắc prompt rõ ràng và ngắn gọn giúp mô hình AI hiểu nhiệm vụ dự định dễ dàng hơn. Tránh sự mơ hồ và phức tạp không cần thiết để có thể dễ dàng tìm ra những gì bạn đang theo đuổi.   * **2. Xem xét bối cảnh và tính đặc thù**   Bối cảnh là quan trọng. Nếu bạn hỏi bạn của mình liệu câu nói “I'm sorry” và “My bad” có nghĩa giống nhau hay không, chúng có thể trả lời là có. Tuy nhiên, chúng có thể nói không nếu bạn thêm vào ngữ cảnh đang dự đám tang.  Bằng cách cung cấp cho một mô hình AI thông tin theo ngữ cảnh phù hợp, nó có thể cung cấp cho bạn câu trả lời chính xác và phù hợp hơn (ví dụ: “Thực ra, Joe McHuman, tôi không khuyên bạn nên nói điều đó khi thức dậy.”) Kết hợp các tín hiệu ngữ cảnh có liên quan trong lời đưa lời nhắc prompt của bạn, và đảm bảo lời đưa lời nhắc prompt dành riêng cho nhiệm vụ dự định, tránh các hướng dẫn chung chung hoặc mơ hồ.   * **3. Sử dụng phong cách đàm thoại**   Điều này nghe có vẻ hơi kỳ quặc khi bạn nói chuyện với rô-bốt, nhưng hãy giữ cho nó tự nhiên và giống như cuộc trò chuyện. Giả vờ như bạn đang nói chuyện với một người thực sự thông minh đang giúp đỡ bạn và giữ cho ngôn ngữ đơn giản và dễ tiếp cận.  Tại sao? Hãy nhớ rằng những gì bạn đưa vào là những gì bạn nhận được. Cách tiếp cận đàm thoại này giúp tạo ra sự tương tác giống con người hơn và làm cho đầu ra của AI dễ hiểu hơn.   * **4. Tận dụng giọng nói chủ động và các câu hỏi tu từ**   Đối với những người không quen thuộc với thuật ngữ này, giọng nói tích cực không đưa ra hướng dẫn cho AI khi bạn chạy bộ. Đó là cấu trúc một câu như thế này:   * **Active Voice:** Bạn có thể viết một bài luận thuyết phục về lợi ích của **việc tập thể dục không?**(Chủ ngữ + Động từ + Tân ngữ) * **Thể bị động:** Bạn sẽ viết một bài luận về lợi ích của việc tập thể dục. (Đối tượng + Động từ + Chủ đề)   Trong ví dụ trên, “Bạn” là chủ ngữ, “Viết” là động từ và “Một bài luận thuyết phục” là tân ngữ.  *Tại sao bất kỳ điều này quan trọng?*Giọng nói tích cực trong lời đưa lời nhắc prompt truyền đạt cảm giác hành động và rõ ràng cho mô hình AI. Nó nhấn mạnh những gì bạn đang yêu cầu và khuyến khích phản hồi tập trung.  Ngoài ra, việc kết hợp các câu hỏi tu từ trong lời đưa lời nhắc prompt có thể kích thích "quá trình suy nghĩ" của mô hình AI và hướng dẫn nó tạo ra các kết quả đầu ra sâu sắc và có ý nghĩa hơn.   * **Lời đưa lời nhắc prompt ví dụ cho lập trình viên**   Dưới đây là một số lời đưa lời nhắc prompt ví dụ tuyệt vời để sử dụng khi bạn viết code. Những điều này sẽ giúp bạn trong quá trình phát triển để tăng tốc thời gian phát triển và giúp tìm lỗi.   * **1. Debug Code**   *"Quét code sau để tìm các sự cố tiềm ẩn"*  Nếu bạn cung cấp cho AI vấn đề này và chèn code nguồn sau đó, bạn có thể tìm thấy các vấn đề tiềm ẩn với code, ngay cả khi code thực thi thành công. Điều này hoạt động trong ChatGPT, Bard, GitHub Copilot và TabNine.   * **2. Cải thiện hiệu suất**   *"Đánh giá code sau đây và tìm kiếm các vấn đề về hiệu suất" theo sau là code được đề cập.*  **Generate Tests**  *"Viết bài kiểm tra cho code {ngôn ngữ} sau" theo sau là code được đề cập.*  **4. Giải thích Quy tắc này**  *"Giải thích cách hoạt động của {thứ} trong {ngôn ngữ}."*   * **5. Translate Code**   *"Dịch code {ngôn ngữ thứ nhất} sau sang {ngôn ngữ thứ hai}."*   * **6. Sửa cú pháp đúng**   *"Cú pháp chính xác để {làm điều gì đó} trong {ngôn ngữ} là gì?"*   * **7. Viết hàm**   *"Viết một hàm để {làm điều gì đó} bằng {ngôn ngữ}."*   * **Phần kết luận**   Đây là một vài ví dụ về kỹ thuật đưa lời nhắc prompt. Đó là một nghệ thuật cần nhiều lần thử và sai, nhưng nó đáng giá về lâu dài! Tiếp tục thử nghiệm và kiểm tra cho đến khi bạn nhận được kết quả tốt nhất.  Để xem thêm một số ví dụ hoặc tìm hiểu thêm về AI tổng quát, hãy xem các bài viết sau:   * [Cách dùng ChatGPT để viết code](https://www.pluralsight.com/blog/software-development/how-use-chatgpt-programming-coding) * [ChatGPT Vs Bard: Cái nào tốt hơn để code?](https://www.pluralsight.com/blog/software-development/chatgpt-vs-bard-coding) * [Plugin ChatGPT: Cách làm cho ChatGPT thông minh hơn](https://www.pluralsight.com/blog/machine-learning/chat-gpt-plugins-guide) * [Tự sửa lỗi và code tự động với AI: Tranh luận về ưu và nhược điểm](https://www.pluralsight.com/blog/machine-learning/regenerative-auto-code-ai-gpt) * [Pigeon VS AI: Chim hoang dã thông minh hơn trí tuệ nhân tạo?](https://www.pluralsight.com/blog/machine-learning/pigeon-vs-ai-associated-learning) |

Easy Prompt: "Write a Python function that takes two integers as input and returns their sum."

Write a Java program that takes input for name, age, gender, and student ID from the user and then prints out the entered information on the screen



What is an easy prompt and hard prompt to ask ChatGPT to execute from the user when requesting ChatGPT code a program?

ChatGPT

Asking ChatGPT to execute code poses significant challenges due to the limitations of the language model and the potential risks associated with running unverified code. Providing prompts that are too complex or open-ended can lead to unpredictable and potentially harmful outputs.

For safety reasons, it is generally advised to avoid asking ChatGPT to execute code directly. **Instead, you can use ChatGPT to help you with specific programming questions, provide code snippets, or guide you in understanding programming concepts.** This way, you can benefit from the model's language capabilities without exposing yourself to potential security risks.

Easy Prompt:

* "Can you explain the concept of loops in Python?"
* "How do I read a file in Java?"
* "What is the syntax for creating a function in JavaScript?"

Hard Prompt (Avoid asking the model to execute code directly):

* "Write a Python program to read a CSV file, perform data analysis, and visualize the results using a library."
* "Implement an algorithm in C++ to solve the traveling salesman problem with a time complexity of O(n^2)."
* "**Create a web application using HTML, CSS, and JavaScript that allows users to register, login, and submit forms with validation**."

Remember, even with simple prompts, there is no guarantee that the responses will be correct or safe to execute. Always verify and validate any code received from any source, including AI models like ChatGPT, before running it in a production environment.

What is an easy prompt and hard prompt to ask ChatGPT to write code ?

ChatGPT

**An "easy prompt" is a straightforward and clear instruction given to ChatGPT, making it easier for the model to generate the desired code.** It usually provides specific details and guidelines for the task. For example:

Easy Prompt: "Write a Python function that takes two integers as input and returns their sum."

In this prompt, the task is clearly defined, and the required input and output are specified.

On the other hand, a "hard prompt" is more challenging and may require more context or complex instructions for ChatGPT to produce the correct code. It might involve multiple steps or require the model to understand and follow a more intricate (

very complicated or detailed) problem statement. For example:

Hard Prompt: "Write a Java program that reads a CSV file, calculates the average of the numbers in the 'price' column, and outputs the result."

In this case, the prompt involves multiple steps: reading a file, parsing data, performing calculations, and displaying the output.

The distinction between easy and hard prompts depends on the complexity of the programming task and how much contextual information or guidance the model needs to generate accurate and meaningful code. **An easy prompt is usually more direct and specific, while a hard prompt may require the model to exhibit a deeper understanding of the task and its components.**

Ans2:  
Sure, here are an easy and hard prompt to ask ChatGPT to execute from the user when requesting ChatGPT code a program:

Easy prompt:

Write a Python program that prints the following:

"

Hello, world!

"

This is an easy prompt because it is a simple task that ChatGPT can easily understand and complete. The program only needs to print a single line of text, and the syntax is very straightforward.

Hard prompt:

**Write a Python program that implements a simple sorting algorithm.**

This is a harder prompt because it requires ChatGPT to understand the concept of a sorting algorithm and to implement it in Python. A sorting algorithm is a procedure for arranging data in a specific order, such as ascending or descending order. There are many different sorting algorithms, and each one has its own strengths and weaknesses.

To implement a sorting algorithm in Python, ChatGPT would need to understand the following concepts:

* Data structures such as lists and arrays
* Control flow statements such as loops and conditional statements
* Functions

**ChatGPT would also need to be able to translate the abstract concept of a sorting algorithm into the specific syntax of the Python programming language.**

This is a more challenging prompt because it requires ChatGPT to understand and apply more complex concepts. However, it is also a more rewarding prompt because it allows ChatGPT to demonstrate its ability to learn and apply new knowledge.

# Some example prompts

Correct below English paragraph

Currently, we have workaround. That is when we run Pulse Secure VPN app, we can ping and ssh access to bell server.

ChatGPT: Currently, we have a workaround. When we run the Pulse Secure VPN app, we are able to ping the Bell server and gain SSH access to it.

In the name "Gupta Sachin" which word is first name, which word is surname?

Ans1:

In the name "Gupta Sachin," "Sachin" is the first name and "Gupta" is the surname. When writing an email to him, you should address him as "Hi Sachin Gupta." The general convention is to mention the first name followed by the surname.

Ans2:  
  
In the name "Gupta Sachin", Gupta is the surname and Sachin is the first name. So, when you write an email to him, you should address him as Hi Sachin Gupta.

# ChatGPT

|  |  |
| --- | --- |
| How to make a good prompt so that ChatGPT can answer correctly ?  ChatGPT  To make a good prompt that helps ChatGPT provide accurate and relevant answers, consider the following tips:   1. Be clear and specific: Clearly state your question or provide a concise description of the information you're seeking. This helps set the context for ChatGPT and ensures it understands your intent. 2. Include relevant details: Include any relevant information or context that can assist ChatGPT in understanding the specific problem or topic. The more specific and detailed your prompt is, the better the response is likely to be. 3. Ask for clarification if needed: If you're seeking an answer to a specific question or need information on a particular aspect, explicitly ask for it in your prompt. This helps guide ChatGPT's response and ensures you get the information you're looking for. 4. Use proper formatting: Structure your prompt in a way that is easy to understand. You can use bullet points, numbered lists, or headings to organize your questions or subtopics. 5. Be polite and respectful: Maintaining a polite and respectful tone in your prompt can foster a positive interaction with ChatGPT. Remember, ChatGPT is an AI language model, and providing clear and respectful prompts helps facilitate better communication. 6. Experiment and iterate: If the initial response is not what you expected or if you need more information, you can iterate on your prompt by refining or rephrasing your question. Experimenting with different phrasings or approaches can often yield better results.   Remember that while these tips can help improve the quality of responses, ChatGPT's answers are based on patterns it has learned from training data and may not always be perfect or accurate. It's always a good idea to verify information from reliable sources and use critical thinking when evaluating responses. | Để tạo lời đưa lời nhắc prompt tốt giúp ChatGPT đưa ra câu trả lời chính xác và phù hợp, hãy xem xét các mẹo sau:   1. Rõ ràng và cụ thể: Nêu rõ câu hỏi của bạn hoặc cung cấp mô tả ngắn gọn về thông tin bạn đang tìm kiếm. Điều này giúp thiết lập ngữ cảnh cho ChatGPT và đảm bảo ChatGPT hiểu ý định của bạn. 2. Bao gồm các chi tiết có liên quan: Bao gồm mọi thông tin hoặc ngữ cảnh có liên quan có thể hỗ trợ ChatGPT hiểu vấn đề hoặc chủ đề cụ thể. **Lời đưa lời nhắc prompt của bạn càng cụ thể và chi tiết thì phản hồi càng có khả năng tốt hơn.** 3. Yêu cầu làm rõ nếu cần: Nếu bạn đang tìm kiếm câu trả lời cho một câu hỏi cụ thể hoặc cần thông tin về một khía cạnh cụ thể, hãy yêu cầu rõ ràng trong lời đưa lời nhắc prompt của bạn. Điều này giúp hướng dẫn phản hồi của ChatGPT và đảm bảo bạn nhận được thông tin mình đang tìm kiếm. 4. Sử dụng định dạng phù hợp: Cấu trúc lời đưa lời nhắc prompt của bạn theo cách dễ hiểu. Bạn có thể sử dụng các dấu đầu dòng, danh sách được đánh số hoặc tiêu đề để sắp xếp các câu hỏi hoặc chủ đề phụ của mình. 5. Lịch sự và tôn trọng: Duy trì giọng điệu lịch sự và tôn trọng trong lời đưa lời nhắc prompt của bạn có thể thúc đẩy tương tác tích cực với ChatGPT. **Hãy nhớ rằng ChatGPT là một mô hình ngôn ngữ AI và việc cung cấp lời đưa lời nhắc prompt rõ ràng và tôn trọng sẽ giúp tạo điều kiện giao tiếp tốt hơn.** 6. Thử nghiệm và lặp lại: Nếu phản hồi ban đầu không như bạn mong đợi hoặc nếu bạn cần thêm thông tin, **bạn có thể lặp lại lời đưa lời nhắc prompt của mình bằng cách tinh chỉnh hoặc diễn đạt lại câu hỏi của mình.** Thử nghiệm với các cách diễn đạt hoặc cách tiếp cận khác nhau thường có thể mang lại kết quả tốt hơn.   Hãy nhớ rằng mặc dù những mẹo này có thể giúp cải thiện chất lượng phản hồi, nhưng câu trả lời của ChatGPT dựa trên các mẫu mà ChatGPT đã học được từ dữ liệu đào tạo và có thể không phải lúc nào cũng hoàn hảo hoặc chính xác. Bạn nên xác minh thông tin từ các nguồn đáng tin cậy và sử dụng tư duy phản biện khi đánh giá các câu trả lời. |

|  |  |
| --- | --- |
| Here are some tips on how to make a good prompt so that ChatGPT can answer correctly:   * Be clear and concise. The more specific you can be about what you want ChatGPT to do, the better results you will get. For example, instead of asking "What is the capital of France?", you could ask "What is the capital of France and what is its population?" * Provide context. ChatGPT can't read your mind, so it's important to provide context for your questions. For example, if you're asking about the capital of France, you could mention that you're planning a trip to France and you want to know where the capital is. * Use examples. If you can provide examples of the desired output, ChatGPT will be more likely to understand what you're looking for. For example, if you're asking for a list of countries in Europe, you could provide a list of some countries in Europe as examples. * Use concrete language. Avoid using abstract or vague language when creating your prompt. For example, instead of asking "What is the meaning of life?", you could ask "What are some different theories about the meaning of life?" * Avoid grammatical errors. Grammatical errors can confuse ChatGPT and make it more difficult for it to understand your question. * Review and test your prompt. Before submitting your prompt to ChatGPT, be sure to review it and make sure that it is clear, concise, and provides the necessary context. You can also test your prompt by asking ChatGPT to answer it before you submit it.   Here are some additional tips:   * Use keywords. When you're creating your prompt, use keywords that are relevant to the topic you're asking about. This will help ChatGPT to focus on the right information and provide you with the most relevant answer. * Use synonyms. If you're not sure how to phrase a question, you can use synonyms to rephrase it. This will help ChatGPT to understand your question even if you don't use the exact right words. * Use multiple prompts. If you're not getting the results you want from one prompt, you can try using multiple prompts. This will give ChatGPT more information to work with and it may be able to provide you with a better answer.   I hope these tips help you to create good prompts that will allow ChatGPT to answer your questions correctly. | Dưới đây là một số mẹo về cách tạo lời đưa lời nhắc prompt tốt để ChatGPT có thể trả lời chính xác:   * Hãy rõ ràng và ngắn gọn. Bạn càng cụ thể về những gì bạn muốn ChatGPT thực hiện thì bạn càng nhận được kết quả tốt hơn. Ví dụ, thay vì hỏi "Thủ đô của Pháp là gì?", bạn có thể hỏi "Thủ đô của Pháp là gì và dân số của nó là bao nhiêu?" * Cung cấp ngữ cảnh. **ChatGPT không thể đọc được suy nghĩ của bạn, vì vậy điều quan trọng là cung cấp ngữ cảnh cho câu hỏi của bạn**. Ví dụ: nếu hỏi về thủ đô của Pháp, bạn có thể đề cập rằng bạn đang lên kế hoạch cho một chuyến đi đến Pháp và bạn muốn biết thủ đô ở đâu. * **Sử dụng các ví dụ**. Nếu bạn có thể cung cấp các ví dụ về đầu ra mong muốn, ChatGPT sẽ có nhiều khả năng hiểu những gì bạn đang tìm kiếm hơn. Ví dụ: nếu bạn đang yêu cầu danh sách các quốc gia ở Châu Âu, bạn có thể cung cấp danh sách một số quốc gia ở Châu Âu làm ví dụ. * Sử dụng ngôn ngữ cụ thể. Tránh sử dụng ngôn ngữ trừu tượng hoặc mơ hồ khi tạo lời đưa lời nhắc prompt của bạn. Ví dụ, thay vì hỏi "Ý nghĩa của cuộc sống là gì?", bạn có thể hỏi "**Một số lý thuyết khác nhau về ý nghĩa của cuộc sống là gì?"** * **Tránh các lỗi ngữ pháp.** Các lỗi ngữ pháp có thể gây nhầm lẫn cho ChatGPT và khiến ChatGPT khó hiểu câu hỏi của bạn hơn. * Xem xét và kiểm tra lời đưa lời nhắc prompt của bạn. Trước khi gửi lời đưa lời nhắc prompt của bạn tới ChatGPT, hãy nhớ xem lại lời đưa lời nhắc prompt và đảm bảo rằng lời đưa lời nhắc prompt đó rõ ràng, ngắn gọn và cung cấp ngữ cảnh cần thiết. Bạn cũng có thể kiểm tra lời đưa lời nhắc prompt của mình bằng cách yêu cầu ChatGPT trả lời trước khi bạn gửi.   Dưới đây là một số lời khuyên bổ sung:   * **Sử dụng keyword.** Khi bạn tạo lời đưa lời nhắc prompt, hãy sử dụng các từ khóa có liên quan đến chủ đề mà bạn đang hỏi. Điều này sẽ giúp ChatGPT tập trung vào đúng thông tin và cung cấp cho bạn câu trả lời phù hợp nhất. * **Sử dụng từ đồng nghĩa**. Nếu bạn không chắc cách diễn đạt câu hỏi, bạn có thể sử dụng các từ đồng nghĩa để diễn đạt lại câu hỏi đó. **Điều này sẽ giúp ChatGPT hiểu câu hỏi của bạn ngay cả khi bạn không sử dụng đúng từ.** * Sử dụng nhiều lời đưa lời nhắc prompt. Nếu bạn không nhận được kết quả mong muốn từ một lời đưa lời nhắc prompt, bạn có thể thử sử dụng nhiều lời đưa lời nhắc prompt. Điều này sẽ cung cấp cho ChatGPT thêm thông tin để làm việc và nó có thể cung cấp cho bạn câu trả lời tốt hơn.   Tôi hy vọng những mẹo này sẽ giúp bạn tạo lời đưa lời nhắc prompt tốt cho phép ChatGPT trả lời chính xác câu hỏi của bạn. |

|  |  |
| --- | --- |
| Here are some tips on how to make a good prompt so that Google Bard chatbot can provide accurate and relevant answers:   * Be clear and concise. The more specific you can be about what you want Bard to do, the better results you will get. For example, instead of asking "What is the capital of France?", you could ask "What is the capital of France and what is its population?" * Provide context. Bard can't read your mind, so it's important to provide context for your questions. For example, if you're asking about the capital of France, you could mention that you're planning a trip to France and you want to know where the capital is. * Use examples. If you can provide examples of the desired output, Bard will be more likely to understand what you're looking for. For example, if you're asking for a list of countries in Europe, you could provide a list of some countries in Europe as examples. * Use concrete language. Avoid using abstract or vague language when creating your prompt. For example, instead of asking "What is the meaning of life?", you could ask "What are some different theories about the meaning of life?" * Avoid grammatical errors. Grammatical errors can confuse Bard and make it more difficult for it to understand your question. * Review and test your prompt. Before submitting your prompt to Bard, be sure to review it and make sure that it is clear, concise, and provides the necessary context. You can also test your prompt by asking Bard to answer it before you submit it.   Here are some additional tips:   * Use keywords. When you're creating your prompt, use keywords that are relevant to the topic you're asking about. This will help Bard to focus on the right information and provide you with the most relevant answer. * Use synonyms. If you're not sure how to phrase a question, you can use synonyms to rephrase it. This will help Bard to understand your question even if you don't use the exact right words. * Use multiple prompts. If you're not getting the results you want from one prompt, you can try using multiple prompts. This will give Bard more information to work with and it may be able to provide you with a better answer. * Be patient. Bard is still under development, so it may not always be able to provide accurate and relevant answers. If you're not getting the results you want, try rephrasing your prompt or providing more context.   I hope these tips help you to create good prompts that will allow Google Bard chatbot to provide you with accurate and relevant answers. | Dưới đây là một số mẹo về cách đưa ra lời đưa lời nhắc prompt tốt để chatbot Google Bard có thể đưa ra câu trả lời chính xác và phù hợp:   * Hãy rõ ràng và ngắn gọn. Bạn càng cụ thể về những gì bạn muốn Bard làm, bạn càng nhận được kết quả tốt hơn. Ví dụ, thay vì hỏi "Thủ đô của Pháp là gì?", bạn có thể hỏi "Thủ đô của Pháp là gì và dân số của nó là bao nhiêu?" * Cung cấp ngữ cảnh. Bard không thể đọc được suy nghĩ của bạn, vì vậy điều quan trọng là cung cấp ngữ cảnh cho câu hỏi của bạn. Ví dụ: nếu hỏi về thủ đô của Pháp, bạn có thể đề cập rằng bạn đang lên kế hoạch cho một chuyến đi đến Pháp và bạn muốn biết thủ đô ở đâu. * Sử dụng các ví dụ. Nếu bạn có thể cung cấp các ví dụ về đầu ra mong muốn, Bard sẽ có nhiều khả năng hiểu những gì bạn đang tìm kiếm. Ví dụ: nếu bạn đang yêu cầu danh sách các quốc gia ở Châu Âu, bạn có thể cung cấp danh sách một số quốc gia ở Châu Âu làm ví dụ. * Sử dụng ngôn ngữ cụ thể. Tránh sử dụng ngôn ngữ trừu tượng hoặc mơ hồ khi tạo lời đưa lời nhắc prompt của bạn. Ví dụ, thay vì hỏi "Ý nghĩa của cuộc sống là gì?", bạn có thể hỏi "Một số lý thuyết khác nhau về ý nghĩa của cuộc sống là gì?" * Tránh các lỗi ngữ pháp. **Các lỗi ngữ pháp có thể khiến Bard bối rối và khiến Bard khó hiểu câu hỏi của bạn hơn.** * Xem xét và kiểm tra lời đưa lời nhắc prompt của bạn. Trước khi gửi lời đưa lời nhắc prompt của bạn tới Bard, hãy nhớ xem lại lời đưa lời nhắc prompt đó và đảm bảo rằng lời đưa lời nhắc prompt đó rõ ràng, ngắn gọn và cung cấp ngữ cảnh cần thiết. Bạn cũng có thể kiểm tra lời đưa lời nhắc prompt của mình bằng cách yêu cầu Bard trả lời trước khi bạn gửi.   Dưới đây là một số lời khuyên bổ sung:   * Sử dụng từ khóa. Khi bạn tạo lời đưa lời nhắc prompt, hãy sử dụng các từ khóa có liên quan đến chủ đề mà bạn đang hỏi. Điều này sẽ giúp Bard tập trung vào thông tin phù hợp và cung cấp cho bạn câu trả lời phù hợp nhất. * Sử dụng từ đồng nghĩa. Nếu bạn không chắc cách diễn đạt câu hỏi, bạn có thể sử dụng các từ đồng nghĩa để diễn đạt lại câu hỏi đó. Điều này sẽ giúp Bard hiểu câu hỏi của bạn ngay cả khi bạn không sử dụng đúng từ. * Sử dụng nhiều lời đưa lời nhắc prompt. Nếu bạn không nhận được kết quả mong muốn từ một lời đưa lời nhắc prompt, bạn có thể thử sử dụng nhiều lời đưa lời nhắc prompt. Điều này sẽ cung cấp cho Bard thêm thông tin để làm việc và có thể cung cấp cho bạn câu trả lời tốt hơn. * Kiên nhẫn. Bard vẫn đang được phát triển nên không phải lúc nào nó cũng có thể cung cấp câu trả lời chính xác và phù hợp. Nếu bạn không nhận được kết quả mong muốn, hãy thử diễn đạt lại lời đưa lời nhắc prompt của bạn hoặc cung cấp thêm ngữ cảnh.   Tôi hy vọng những mẹo này sẽ giúp bạn tạo lời đưa lời nhắc prompt tốt cho phép chatbot Google Bard cung cấp cho bạn câu trả lời chính xác và phù hợp. |