

Product names: Teslacool, FridgeX, Neuradrink

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

Product names: WatchX, iSpace, The Watch Company

Product description: A home milkshake maker

Product names: ShakeX, The Milkshake Company, HyperShake

Output:

ShoeX, MarsFit, The Shoe Company

There are also some rules or best practices you would do well to follow, which could be included in the prompt as context, to guide the AI towards a name that works. This technique is sometimes referred to as *pre-warming* or *internal retrieval* ([Liu 2021](#)), and it is simple but effective. Starting the conversation asking for best practice advice, then asking it to follow its own advice, can help a lot. In effect you are using it to generate its own direction.

Input:

Please give me 5 tips for naming products based on expert industry advice.

Output:

Certainly! Naming a product is crucial as it often gives the first impression to potential customers. Here are five expert tips to guide you in naming products:

1. **Simplicity & Memorability**: Opt for names that are short, easy to spell, and memorable. Long, complicated names can be difficult for customers to recall. Remember, the easier the name is to remember, the more likely it is that consumers will think of your product when making a purchase

decision.

...

5. **Positive Connotations**: Names that evoke positive emotions or imagery can create a favorable perception of your product. Avoid names that have negative or controversial associations in any of the primary languages or cultures where you intend to do business. It's often worthwhile to conduct linguistic checks and market testing to ensure there aren't unintended meanings.

Lastly, once you've shortlisted a few names, it's a good idea to run them by diverse focus groups or conduct surveys to gauge reactions. Also, ensure that the name you choose is available for trademarking and doesn't infringe on existing trademarks.

Then within the same chat window, where the model has the context of the past advice it gave, you ask your initial prompt for the task you wanted to complete.

Input:

Using this advice, can I have a list of product names for a pair of shoes that can fit any foot size?

Output:

Certainly! Based on the criteria given:

1. **Simplicity & Memorability**
2. **Relevance & Descriptiveness**
3. **Differentiation & Uniqueness**
4. **Adaptable & Scalable**
5. **Positive Connotations**

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

1. ****FitAll Walkers**:** A simple and memorable name that instantly suggests a universal fit.

...

10. ****TrueFit Universals**:** Emphasizing genuine comfort and fit for all.

Before finalizing any name, remember to check for trademarks to ensure there's no conflict with existing products or brands.

Another fruitful strategy in our experience is to take the best advice out there for the task you want to accomplish, and insert that context into the prompt. For example, you could take [Brandwatch's 5 Golden Rules for naming a product](#) or another trusted external resource you find, and insert that as context into the prompt. This will increase the length of the prompt significantly, which costs more money (when using the API as a developer), but may be worth the tradeoff if the quality of the response improves.

Input:

Can I have a list of product names for a pair of shoes that can fit any foot size? Please follow the five rules below:

1. It should be readable and writable

If your product name is hard to pronounce, people won't talk about it and if they can't write it down (and spell it correctly!) when they hear it, how do you expect them to Google it?

Keep it simple and don't go with any wacky spellings just for the sake of it.

...

5. It should evoke an emotion, feeling or idea

Your product name should tie back into what your product is, what the feeling you want people to have when experiencing

your product is, and/or what idea are you trying to get across.

It should be emotive and inspiring.

Output:

1. FlexFit Footwear
2. OmniSize Sneakers
3. AdaptStride Shoes
- ...
18. OmniComfort Kicks
19. FlexSize Footwear
20. Boundless Soles

There are other myriad ways of providing direction. In the image generation example, direction was given by specifying that the business meeting is taking place around a glass top table. If you change only that detail, you can get a completely different image, as detailed in [Figure 1-5](#).

Input:

```
https://s.mj.run/TKAsyhNiKmc stock photo of business meeting  
of four people gathered around a campfire outdoors in the  
woods, Panasonic, DC-GH5
```

The output is shown in [Figure 1-5](#).



Figure 1-5. Stock photo of business meeting in the woods

Role-playing is also important for image generation, and one of the quite powerful ways you can give Midjourney direction is to supply the name of an artist or art style to emulate. One artist that features heavily in the AI art world is Van Gogh, known for his bold, dramatic brush strokes and vivid use of colors. Watch what happens when you include his name in the prompt, as shown in [Figure 1-6](#).

Input:

```
people in a business meeting, by van gogh
```

The output is shown in [Figure 1-6](#).



Figure 1-6. People in a business meeting, by Van Gogh

In order to get that last prompt to work, you need to strip back a lot of the other direction. For example, losing the base image and the words `stock photo` as well as the camera `Panasonic, DC-GH5` helps bring in Van Gogh's style. The problem you often run into is often that with too much direction, the model can quickly get to a conflicting combination that it can't resolve. If your prompt is overly specific, there might not be enough samples in the training data to generate an image that's consistent with all of your criteria. In cases like these you should choose which element is more important (in this case, Van Gogh), and defer to that.

Direction is one of the most commonly used and broadest principles. It can take the form of simply using the right descriptive words to clarify your intent, or channeling the personas of relevant business celebrities. While too much direction can narrow the creativity of the model, too little direction is the more common problem.

2. Specify Format

AI models are universal translators. Not only does that mean translating from French to English, or Urdu to Klingon, but also between data structures like JSON to YAML, or natural language to Python code. These models are capable of returning a response in almost any format, so an important part of prompt engineering is finding ways to specify what format you want the response to be in.

Every now and again you'll find that the same prompt will return a different format, for example, a numbered list instead of comma separated. This isn't a big deal most of the time, because most prompts are one-off and typed into ChatGPT or Midjourney. However when you're incorporating AI tools into production software, occasional flips in format can cause all kinds of errors.

Just like when working with a human, you can avoid wasted effort by specifying up front the format you expect the response to be in. For text generation models, it can often be helpful to output JSON instead of a simple ordered list, because that's the universal format for API responses, which can make it simpler to parse and spot errors, as well as to use to render the frontend HTML of an application. YAML is also another popular choice because it enforces a parseable structure while still being simple and human-readable.

In the original prompt you gave direction through both the examples provided, and the colon at the end of the prompt indicated it should complete the list inline. To swap the format to JSON you need to update both and leave the JSON uncompleted, so GPT-4 knows to complete it.

Input:

```
Return a comma-separated list of product names in JSON for
"A pair of shoes that can fit any foot size.".
Return only JSON.
```

Examples:

```
[ {  
    "Product description": "A home milkshake maker.",  
    "Product names": [ "HomeShaker", "Fit Shaker",  
                      "QuickShake", "Shake Maker" ]  
},  
{  
    "Product description": "A watch that can tell  
accurate time in space.",  
    "Product names": [ "AstroTime", "SpaceGuard",  
                      "Orbit-Accurate", "EliptoTime" ]  
]  
]
```

Output:

```
[  
{  
    "Product description": "A pair of shoes that can \\  
fit any foot size.",  
    "Product names": [ "FlexFit Footwear", "OneSize Step",  
                      "Adapt-a-Shoe", "Universal Walker" ]  
}  
]
```

The output we get back is the completed JSON containing the product names. This can then be parsed and used programmatically, in an application or local script. It's also easy from this point to check if there's an error in the formatting using a JSON parser like Python's standard `json` library, because broken JSON will result in a parsing error, which can act as a trigger to retry the prompt or investigate before continuing. If you're still not getting the right format back, it can help to specify at the beginning or end of the prompt, or in the system message if using a chat model: `You are a helpful assistant that only responds in JSON`, or specify [JSON output](#) in the model parameters where available (this is called *grammars* with [LLaMA models](#).