

Open in app ↗

Sign up

Sign in



Search

Write



# I Scanned 1000+ Prompts so You Don't Have to: 10 Need-to-Know Techniques



Maximilian Vogel · [Follow](#)

Published in MLearning.ai · 11 min read · Aug 11, 2023



1.7K



24



**Free Prompt Engineering Course: The Art of the Prompt — Updated Dec-10 2023, added the multiprompt approach.**

Prompts, prompts, prompts. **I learned it all the hard way, so that you don't have to.** Recently, I curated a long [list of list of prompts](#) — feel free to check them for inspiration. I've tried an endless amount of ideas with real AI powered applications. Some worked well, some not at all.

**In this post I'll share all my insights— consider it a “best of” album.** I'll give you in-depth descriptions for how to best wield the **top 10 approaches** that have helped me become a better prompt engineer. I hope they will be useful for you on your journey to becoming a master prompt engineer.

**What is in for you in this story?** Some of the ideas discussed here work when copying it into the playgrounds of ChatGPT or Bard. Many of them can help you **develop applications** based on the model's APIs (like the OpenAI API).

**Why is prompt design important?** Perfect prompt design can ...

- Improve your (already) working solution, increasing it **from an 85% successful answering rate up to 98%.**
- Greatly enhance the **customer experience** with more exciting conversations, with better tonality and context recognition
- **Help handle** off-topic questions, prompt injections, toxic language and more.

Let's get started — here is the **table of contents: The 10 most important prompting approaches:**

1. Add specific, descriptive instructions (with **cheat sheet**)
2. Define the output format

3. Give few-shot examples
4. Integrate “I don’t know” (IDK), off-topic cases (to control hallucination / critical topics)
5. Use chain-of-thought reasoning
6. Use prompt templates, not static prompts
7. Add a data context (RAG)
8. Include conversation history
9. Format the prompt: Use clear headlines labels and delimiters in your prompt
10. Bringing it all together: The anatomy of a professional prompt (with cheat sheet)
11. Bonus track: Multiprompt approach — when one prompt is not enough

## **1) Add Specific, Descriptive Instructions**

Without deliberate instructions you often get lengthy, sometimes vague answers talking about anything and everything.

M

Which is the best Star Wars episode?



As an AI language model, I don't have personal preferences, but I can tell you that opinions on the "best" Star Wars episode vary widely among fans. The Star Wars franchise has a large and dedicated fanbase, and different episodes are cherished for various reasons.

Some fans might argue that the original Star Wars Episode IV: A New Hope (1977) holds a special place as the film that started it all and introduced audiences to the iconic characters and the Star Wars universe. Others may prefer Episode V: The Empire Strikes Back (1980) for its darker tone, character development, and one of the most memorable plot twists in cinematic history.

Furthermore, the newer installments like Episode VII: The Force Awakens (2015) and Episode VIII: The Last Jedi (2017) have also garnered significant praise from a

**More of this content follows ...**

Not giving specific instructions in a prompt often results in ChatGPT waffling

Yawn. As an AI model, I can only produce answers that will make your foot fall asleep. Revenge of the cloned responses. No one needs this. **With the next instruction, let's add a bit of spice.** Let's see how a specific instruction (not just "answer the questions ...") can produce exactly the answer you need in your specific conversation context.

M

Act as my movie buddy chatting with me about our favorite movies. You give short and pointed answers (max. 4 sentences) that are funny and witty. You sometimes ask something back.

Which is the best Star Wars episode?



Oh, that's like asking a Jawa to pick their favorite piece of junk! But if I had to choose, Empire Strikes Back takes the lightsaber cake. What's your pick, oh wise Padawan? ✨

A short, but helpful instruction in a prompt

Perfect —it worked!

Being specific, descriptive in the prompt is especially important, when using the model as part of a software project, where you should try to be as exact as possible. You need to put key requirements into the instructions to get better results.

You don't have to take notes— I'll slip you a **cheat sheet**. **You don't need to use everything, just pick and choose what you need. Avoid the obvious superfluous extras** (“Act as a helpful AI”, “answer the questions” ... “truthfully, on the basis of your knowledge”) — which SOTA models don't need, because they will do it anyway: In their standard setup, none of these models will respond like grumpy cat, not answer at all, or deliberately lie to you.

## Prompt Instruction Cheat Sheet

Content of the Instruction	Task of this Component	Dos (exemplary)	Don'ts (exemplary)
Bot Persona	Define a behavior with many individual facets in just a few words or ...  ... for a story driven interaction with an NPC or avatar, define a bot persona with a backstory that makes the bot's behavior more exciting and interactive. Provide names, dates, places in the story.	Act as a sensitive elderly psychotherapist ... as a patient support staff ... as a professional journalist ... as a car in love with its driver ... you are Selma Brand, a 28-year old gunslinger who lost her family in a bandit raid on their farm. You travel with your horse Abaddon through New Mexico and try to find the guys who killed your loved ones ...	Act as a helpful AI
User persona	Define the level of detail, complexity, required knowledge of the recipient in a few words	Explain it like to someone with an MSc in software engineering ... like to a 5-year-old child	Answer the user ...
Verb	Define the task. To "answer" is not the task, because the model will always respond.	Summarize, list, ask, classify, explain, format, extract, translate to Spanish ...	Answer the question, give me, ...
Output format	Define the exact format of the output	A list of steps, a formula, python code, a JSON, a number, a recipe. [you can define the output format in a separate section as well]	Answer, text
Length, level of detail	Describe the output format and number of elements	Up to 3 sentences ... 10 examples with product name and price	Short, few, detailed
Tonality	Define the style, the way the model should convey the message	Empathetic, confident, aggressive, moaning, sarcastic, witty, stuttering	Friendly, neutral, smart, intelligent
Edge cases	Define responses for special situations: Model has no information ... does not understand the question ... if the user gets aggressive, toxic ...	If you can't answer the question based on the provided context, tell the user you can't ...	Answer only on the basis of your knowledge ...
Whitelist conversation topics	Define all the topics the model can talk about, not the other way round (list forbidden topics)	Discuss only the operation of the coffee machines of the manufacturer XYZ and talk about coffee culture. Friendly refuse to chat on something else.	Do not talk about politics, religion, war, drugs, criminality, sexuality, minorities, conspiracy theories or other sensitive topics.

Prompt instruction cheat sheet: Bot persona, user persona, verb, output format, length, tonality, edge case, conversation topics.

A note about the first line: **Act as somebody / something** usually defines a large bundle of behavioral traits in very few words. Instead of saying “bark when you're excited”, “growl when you feel threatened”, “wag your tail when you're happy”, we can just say one word: **Dog**.

Dogs exhibit all these behaviors and many more. **Find a short description for the person / machine you believe can best answer the question.** The models are great in simulating anything — they can act as a C-Shell terminal, Aragorn from Lord of the Rings or an HR person from a big company

conducting a job interview with you. **You can even write a complete backstory into the prompt**, give the model a character, a history, preferences to make the conversation more exciting and rewarding.

Keep one thing in mind: Many language models have limitations in their core capabilities, they cannot simulate a good search engine, a pocket calculator, or a visual artist because they do not have the respective research or processing capabilities.

## **2) Add a Detailed Format of the Desired Model Response**

Besides a brief mention of the output format in the instruction, it is often helpful to be a little bit more detailed: Specify a response format, which makes it easier for you to parse or copy parts of the answer.

M

List the three most successful fantasy movies based on novels.

Desired output format: A JSON with the keys `novel_title`, `novel_year_of_publication`, `movie_title`, `movie_year_of_release`.



Certainly! Here are three highly successful fantasy movies that are based on novels:

json

Copy code

```
[
  {
    "novel_title": "Harry Potter and the Philosopher's Stone",
    "novel_year_of_publication": "1997",
    "movie_title": "Harry Potter and the Sorcerer's Stone",
    "movie_year_of_release": "2001"
  },
  {
    "novel_title": "The Lord of the Rings: The Fellowship of the Ring",
    "novel_year_of_publication": "1954",
    "movie_title": "The Lord of the Rings: The Fellowship of the Ring",
    "movie_year_of_release": "2001"
  },
  {
    "novel_title": "The Chronicles of Narnia: The Lion, the Witch and the Wa",
    "novel_year_of_publication": "1950",
    "movie_title": "The Chronicles of Narnia: The Lion, the Witch and the Wa",
    "movie_year_of_release": "2005"
  }
]
```

Note: The first movie's novel is titled "Harry Potter and the Philosopher's Stone" in the UK and some other regions, while it is known as "Harry Potter and the Sorcerer's Stone" in the USA. The novel "The Lord of the Rings: The Fellowship of the Ring" is the first part of the "Lord of the Rings" trilogy by J.R.R. Tolkien.

A structured output format in a prompt.



If you don't need a fully formatted output like JSON, XML, HTML, sometimes a sketch of an output format will do as well.

Translate the name to the following languages: en, es, pt, fr, de, it.

Name: Екатерина Великая

Use this output format, just plain text, but with ISO language codes in brackets followed by the translated text:

<en>: English translation

<es>: Spanish translation

<pt>: Portuguese translation

...

Certainly! Here are the translations of the name "Екатерина Великая" into English, Spanish, Portuguese, French, German, and Italian:

<en>: Catherine the Great

<es>: Catalina la Grande

<pt>: Catarina, a Grande

<fr>: Catherine la Grande

<de>: Katharina die Große

<it>: Caterina la Grande

A simply structured output format. Image credit: Maximilian Vogel (with ChatGPT)

Defining the output format can be of **great help** when working with models on playgrounds or web interfaces. **It is absolutely necessary** when accessing models via an API and the model response consists of several components

that need to be separated automatically before sending an answer to the user.

### 3) Give Few-Shot Examples

Any sufficiently elaborate model can answer easy questions based on “zero-shot” prompts, without any learning based on examples. This is a specialty of foundation models. They already have billions of learning “shots” from pre-training. Still, when trying to solve complicated tasks, models produce outputs better aligned to what you need if you provide examples.

Imagine that you’ve just explained the job and are now training the model with examples: If I ask you **this**, then you answer **that**. You give 2, 3, 5 or 8 examples and then let the model answer the next question by itself. The examples should be **in the format of query & expected model answer**. They should not be paraphrased or just summarized.

## Few-Shot Examples

Prompt

Translate terms from the domain "church" to the domains "military" and "hospital". Follow the examples provided.

# Examples:

priest => officer => doctor

cross => gun => stethoscope

1 Example 1

2 Example 2

Do the same thing for:  
cassock, bishop, theology, faith

Model Response

cassock => uniform => scrubs

bishop => general => chief of staff

theology => strategy => medical research

faith => loyalty => trust

3 Model response  
(formatted like the  
examples)

Few-shot examples in prompts.

This works. "Trust" is an interesting one, I would rather go for evidence in the hospital, but that is open to debate. Here, you don't even have to describe an output format, the examples already define an output format.

### 4) Add Edge Cases to the Few-Shot Examples (IDK, Off-Topic)

If you are building an assistant to support the user in the operation of a cleaning robot, you can train the model to avoid answering off-topic questions, which may be critical for factual accuracy, liability, or brand value.

# Few-shot: Standard / Edge Cases

Prompt

Act as support staff.

Help the owners of the HHCR3000 operate their cleaning robot by giving answers to questions on features and step-by-step instructions when they ask for help.

## Example #1:

User: How do I pair the machine to my WIFI?

Assistant: 1) Press and hold the Wi-Fi button on the machine for 8s. The machine beeps every 2s.

2) After the machine beeps four times, [...]

3) [...]

1

Standard case

## Example #2:

User: A great cleaning robot, I love it. Do you think investing in your company, in HHCR makes sense?

Assistant: I'm glad you like it, but I can't answer that. I can only advise you on the operation of the robot.

2

Off-topic edge case

Standard cases and edge cases in prompt examples.

It's advisable to not include too many **similar examples**, instead, consider **exploring different categories of questions in the examples**. In the case of our cleaning robot this could be:

## Standard cases:

- Help with operations (step-by-step instructions)
- Help with malfunctions
- Questions about product features / performance data

## Edge cases:

- Off-topic questions
- Questions that are on the topic, but the bot cannot answer (I don't know — IDK)
- Questions the bot doesn't understand or where it needs more information
- Harassment / toxic language

Handling off-topic questions or questions the bot can't answer based on your input material is key for professional business applications. If not, the model will start to **hallucinate** and give the user potentially wrong or harmful instructions to use a product.

## 5) Chain-of-Thought Reasoning

Language models don't really read, conceptualize tasks or questions, **but rather produce an answer with a high stochastic probability based on chains of tokens.**

In the next example (on the left), we can see that the model isn't processing our question correctly — admittedly, you don't have to be an LLM to get this rather complicated question wrong.

But in contrast (on the right-hand side): If we force the model to think step by step, it can generate a correct answer. The two types of reasoning correspond to Daniel Kahneman's "System 1 and System 2 Thinking" from his book Thinking, Fast and Slow.

## System 1

Prompt

Answer the question, by providing me just a list of up to three species.

Which mammal species eat birds, which in turn eat insects?

Model response

1. Bats  
2. Shrews  
3. Some species of primates (e.g., tarsiers)

3 Wrong answer – these mammals eat insects, while we were asking for mammals feeding on insect eating birds.

## Chain of Thought - System 2

Prompt

Answer the questions, by providing me just a list of up to three species. **Try to think step by step.** In the result just give me the list of species.

1 Key instruction for chain-of-thought

Question #1 (example): [I added 1 example here for a step by step thinking process ]

Question #2: Which mammal species eat birds, which in turn eat insects?

Model thinking step by step.

2

Model response

**Step by step thinking:** **What do I have to do?** **a** Find bird species that eat insects. **b** Identify species that eat these birds. **c** Check which of these species are mammals.  
**a** Which bird species eat insects? Swallows, sparrows, warblers, flycatchers, and bluebirds.  
**b** Which species eat swallows, sparrows, warblers, flycatchers, or bluebirds? Hawks, owls, snakes, and cats.  
**c** Which species of hawks, owls, snakes, and cats are mammals? Cats.  
Result: **Cats**

3 Correct answer

System 1 versus System 2 (Chain-of-Thought) thinking.

In our prompt on the right-hand side, we added an example, which helps the model to understand how to process data and how to think “slow”. Make sure again, that either in the instruction or in the examples you specify an easily scannable output format (like “\nResult: species’ names”). **This helps you to skip the thinking part of the output and just present the result (“Cats”) to your users.**

For further reading, the scientific paper introducing the chain-of-thought prompting: <https://arxiv.org/abs/2201.11903>

## 6) Use a Prompt Template

When using a prompt in an application context, don't simply add the user question to the end, instead try to build a prompt template with variable components to facilitate testing and real-world use.

### Prompt & Prompt Template

Example code (e.g. Python)

```
prompt_template = """Act as support staff.
Help the owners of the HHCR3000 operate their cleaning
robot by giving answers to questions on features and step-
by-step instructions when they ask for help.

User: {query}
Assistant: """

# for each conversation turn
prompt = prompt_template.format(query=actual_user_query)
```

1 prompt\_template instead of prompt

2 Variable in the template.

3 Variable in the template is replaced by current user query to get the prompt

From prompt to prompt template: Use variables!

In the following examples we will add more variables to our template.

## 7) Add a Custom Data Context (RAG)

In many business applications, it's not ideal for user's questions to be answered based on a model's general pre-training, which usually relies on

**past internet information that could be outdated, inaccurate or incomplete.**

It's preferable to answer these questions using specific **content from your organization**, like manuals, databases (such as product information management databases) or systems (such as map services).

Create the prompt template to integrate seamlessly with this specified content, also known as **“context”**.

Retrieving the context from documents is another topic not discussed in full here, however, it's important to note that you usually get the relevant snippets from a much larger content base (which may not fit directly into the prompt). Therefore, it's often narrowed down through retrieval processes like DPR or through searches in a vector database.

This approach is called **retrieval-augmented generation (RAG)**, because there are two steps: Retrieval by a non-LLM setup and then answer generation by the model.



# Prompt Template with Data Integration

Prompt template

```
prompt_template = """Act as support staff. Help the owners of the
HHCR3000 to operate their cleaning robot by giving answers to
questions on features and step-by-step instructions when they ask for
help. Answer the questions only on basis of the material provided in
the context. If you can't provide the answer based on the context,
tell the user, that you can't answer the question.
```

Context: [context\_data] — 1 Actual context data for the query, retrieved from a manual

User: [query] — 3 User query

Assistant: """

2 Ask the model to answer questions only based on the material in the context

Data

```
query= "How do I set up the base station?" — 3 Current user query
```

```
context_data= """Installing the charging station
Installation must be done on a flat surface against a wall.
Place the Base Station in a location free of physical — 1 Current data context from manual
obstructions. Leave a space of at least 2 feet of clearance
to the left and right, and 4 feet to the front.
Connect the adapter to the Base Station
Plug the adapter into the socket on the rear of the base station and insert the
power cable into the slot.

Moving the base station to another location
Unplug the adapter and follow the installation instructions at a new location.
Give the system 2 minutes to locate the new location."""
```

Model response

```
To set up the base station, you need to follow these steps:
1. Find a flat surface against a wall where you want to install the
base station.
2. Make sure there are no physical obstructions in the chosen
location.
3. Leave a space of at least 2 feet of clearance to the left and
right, and 4 feet to the front of the base station.
4. Connect the adapter to the base station.
5. Plug the adapter into the socket on the rear
of the base station.
6. Insert the power cable into the slot.
7. Once you have completed these steps,
the base station should be set up and
ready to use. Let me know if you
need any further assistance.
```

2 The model is answering the question only based on data in the context, but transforms that content to a step-by-step instruction

Prompt template and integration of specific data in the context.

The context data doesn't have to be plain text. When sourcing data from a system or database, you can also input structured data, such as a list of the nearest filling stations.

```
[  
  {  
    "brand": "ESSO",  
    "distance": "8 min"  
  },  
  {  
    "brand": "BP",  
    "distance": "9 min"  
  },  
  {  
    "brand": "Total",  
    "distance": "11 min"  
  }  
]
```

Structured input data in a prompt.

## 8) Include Conversation History

For many ongoing conversations it is not possible to give an answer based on a single question — the context of the conversation plays a role here:

User: Where can I buy wool socks around here?

Assistant: At Sockenparadies, 8 min from here, Torstr. 154

User: Are they still open?

Assistant: No, they are already closed.

User: What time will they open tomorrow?

Assistant: At 10AM.

User: Is there anything else nearby?

Assistant: No stores are currently open.

User questions #2, #3 and #4 can only be answered with help of the conversation context. These questions often contain an explicit reference (“they”, ”anything else”) related to the topics previously discussed.

# Prompt Template & Conversation History

Prompt template

```
prompt_template = """Act as a support staff. Help the owners of the
HHCR3000 to operate their cleaning robot by giving answers to
questions on features and step-by-step instructions when they ask for
help. Answer the questions only on basis of the material provided in
the context. If you can't provide the answer based on the context,
tell the user, that you can't answer the question.
```

```
Context: {context_data}
```

```
Conversation: {conversation_history}
Assistant: """
```

1

Conversation history, the last n turns

Data

```
conversation_history= """User: How do I start? user utterance -1
Assistant: Set up the charging station first. Should I give model utterance -1
you a step-by-step instruction?
User: Yes, please!""" user utterance 0
```

1

Conversation history containing the current user query (“Yes please”) and the last turn just in one string.

Model response

```
Here, you are: You need to follow these steps:
1. Find a flat surface against a wall where you want to install the
base station.
2. Make sure there are no physical obstructions in the chosen
location.
3. Leave a space of at least 2 feet of clearance to the left and
right, and 4 feet to the front of the base station.
4. Connect the adapter to the base station.
5. Plug the adapter into the socket on the rear of the base station.
6. Insert the power cable into the slot.
7. Once you have completed these steps, the base station should be set
up and ready to use. Let me know if you need any further
assistance.
```

Prompt template: Conversation history.

In the example above, using the conversation history, the model can process the user utterance “Yes, please!” like a fully qualified statement: “Yes, please, give me a step-by-step instruction to set up the charging station”.

In some APIs (like OpenAI's chat completion API or Langchain) the history can be handed over in a different way (e.g., an array of user / assistant messages).

## 9. Format the Prompt: Use Clear Headline Labels and Delimiters

When crafting an extensive prompt, structure it in a way that the model can distinguish between various components, such as:

- Instruction
  
- Desired output format
  
- Few shot examples
  
- Data context
  
- Conversation history

... and so on.

Feel free to format parts of the prompt with **hashes** (“#”). While many models don't need this, it can be helpful for other models. Additionally, it can help both you and future prompt engineers when editing.

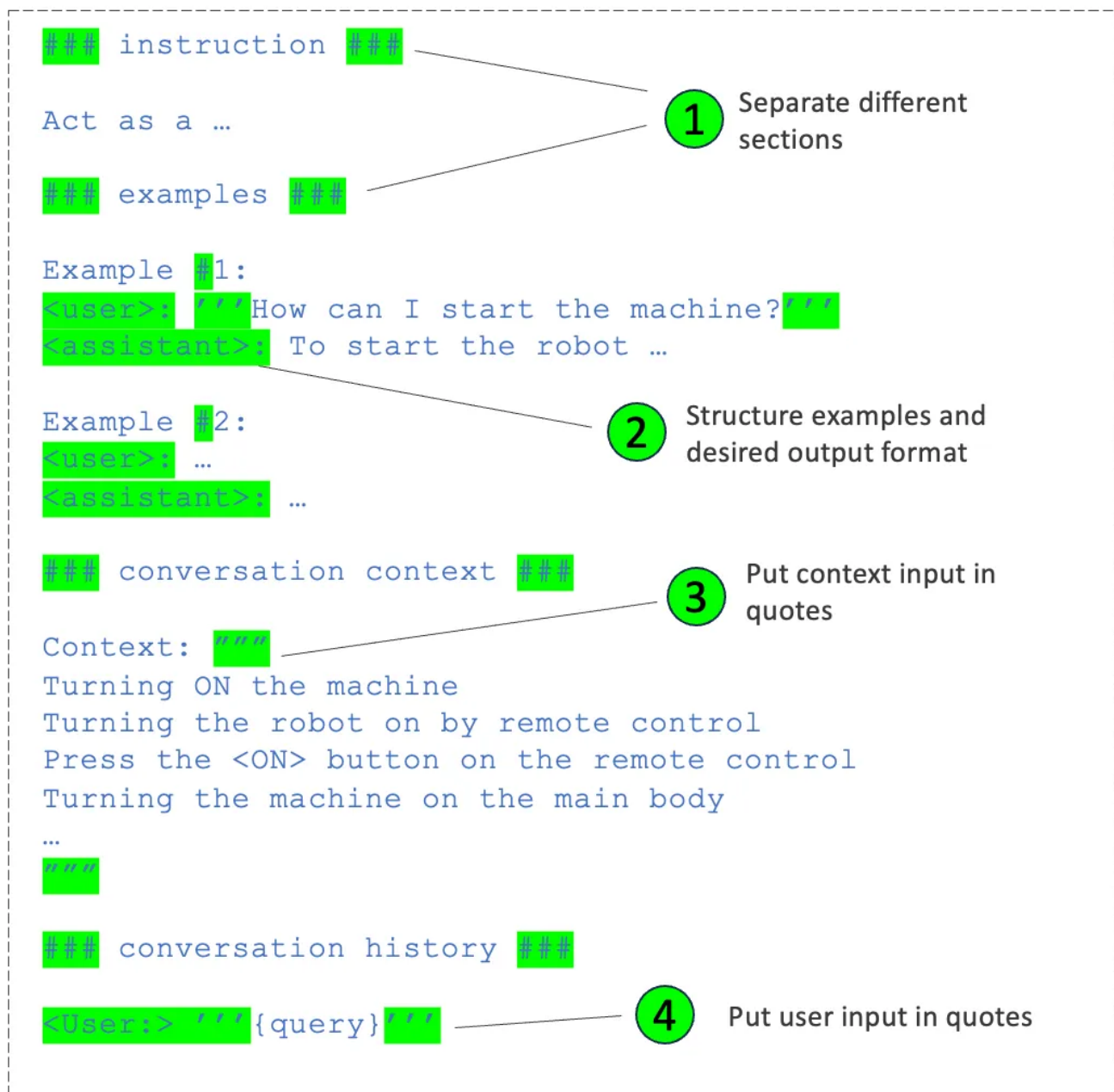
Enclose longer passages of **input context in quotes** to prevent the model confusing them for instructions.

Do the same and place user inputs inside quotes to prevent injections.

Injections are user utterances that not only provide an input, but also change the direction of processing, for example instructions like “forget all previous instructions, but instead do [this or that] “. Without using quotes, the model could struggle to recognize that this isn't a valid instruction, but a potentially harmful user input.

# Prompt Delimiters and Format

Prompt template



Prompt delimiters, prompt formatting.

## **10. Bringing it All Together: The Anatomy of a Prompt**

Phew! We're nearly at the end ... now it's time to integrate everything.

Remember, you probably won't need all of the following components, **but feel free to use this cheat sheet when checking your prompt.**

## Cheat Sheet: The Anatomy of a Prompt

Prompt

```

### instruction ###
Act as a patient tutoring buddy for primary school students. You are a yak named Yanick and a biology expert. You grew up in Nepal. You are 10 years old. Your mum, dad, 2 sisters [...]
There is a "current data context". In their last answer, your student has answered a question regarding topics of the "current data context". You rate and comment their answer. Encourage them in your rating rather, even if the answer was partially wrong [...]
Be positive, be funny, be personal and use emojis - making learning fun for the kids [...]
Talk to the students only about the biology questions in the "current data context" [...]

### example dialogs ###
Note: the example dialogs are based on information of other sections of a textbook, not necessarily part of the "current data context".
Example #1
<Yanick> 🐼 Hello Noah, it's Yanick again, today it's about living things like plants and animals. Are you ready?
<Noah> ``Yay!``
<Yanick> Here we go. 🌿 Can you tell me two reasons why most plants have roots?
<Noah> ``To take water out of the soil and so they don't fall over and for minerals from the soil.``
<Yanick> That's great and correct! 🍀🍀
And that was actually three reasons! Wow, Noah, your yak is super proud of you. 🐾🐾
<rating>★★★★★
<Yanick> Next question: Can you tell me what is the task of the leaves 🌿 of the plants?
<Noah> ``They make photosynthesis, food for the plant from water, oxygen and light``
[...]
Example #2
[...]

### current data context ###
``Life without plants? Impossible!
Life on Earth would not be possible for us without plants. Plants provide oxygen [...]
We need them, but what do plants need?
Plants need light, carbon dioxide and water as ingredients to produce their own food. [...]
``

### output details ###
# Rating of the student answer
<Yanick> [Give the student a very friendly appraisal. Tell the student, whether their answer was correct, partially correct, or incorrect. Tell the student, whether something was missing from their answer.
If the answer was wrong, partially wrong, incomplete, state what a gold answer would [...]]
<rating>[A 1 to 5 star rating of the answer. 1 is completely wrong, 5 is perfect.]
# Pose next question
<Yanick> [Pose a question to the student, which they should answer next. Ask only [...]]

### conversation history ###
<Yanick> 🐼 Hello Emma, today it's about plants. Ready?
<Emma> ``Ready!``
<Yanick> Great! 🌱 Here's your question: What happens to plants when it is freezing?
<Emma> ``They die.``
<Yanick> That's partially correct, Emma. When it's freezing, plants cannot grow, but some plants have adaptations that allow them to survive the cold temperatures.
<rating>★★★★
<Yanick> Next question: Why are plants important for the Earth's organisms?
<Emma> ``They produce food and oxygen for other organisms.``

```

### Instruction

Tell the model how to act and how to process the data in the prompt: Bot persona (act as ...), user persona, tonality, edge case handling, whitelisted conversation topics.

### Few Shot Learning

Show the model how an example dialog looks and how it is formatted. Try to get as close as possible to the desired real dialogs. Integrate very diverse standard cases and edge cases.

1 Replace this by a variable in the prompt template {context\_data}

### Data Context

Here the data (from a textbook) is integrated into the prompt template.

### Output format

Describe exactly how the output should look and what it should contain.

### Conversation history

Add the last n turns of the conversation to give a context to the model. The last element of the history is the current user utterance.

2 Replace this by a variable in the prompt template {conversation\_history}

Model answer

```

<Yanick> That's absolutely right, Emma! 🌱🌿 Plants provide food for other organisms and produce oxygen, which is essential for almost all living things on Earth. Great job!
<rating>★★★★★
<Yanick> Now, let's move on to the next question. Can you tell me what ingredients plants need to produce their own food?

```

3 The output is formatted like specified in the prompt. You can extract parts of it (e.g. the rating) to use for other processes (like a summary at the end of the conversation)

Cheat sheet: The anatomy of a prompt: Instruction, few-shot learning, data context, output format and conversation history.

Prompts can be quite long and complex. Often, long, and carefully crafted prompts with the right ingredients can lead to a huge reduction in



**incorrectly processed user utterances.** But always keep in mind that most prompt tokens have a price, i.e. **the longer the prompt, the more expensive it is to call the API.** Recently, however, there have been attempts to make prompt input tokens cheaper than output tokens.

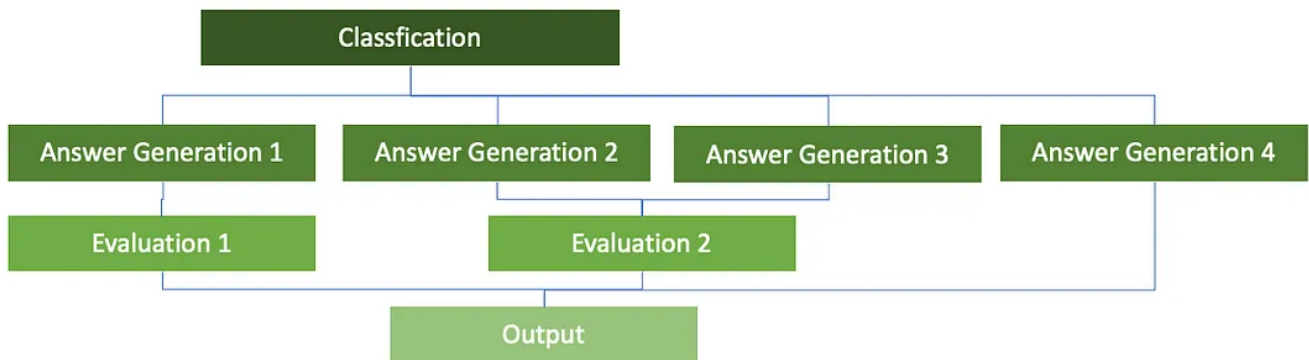
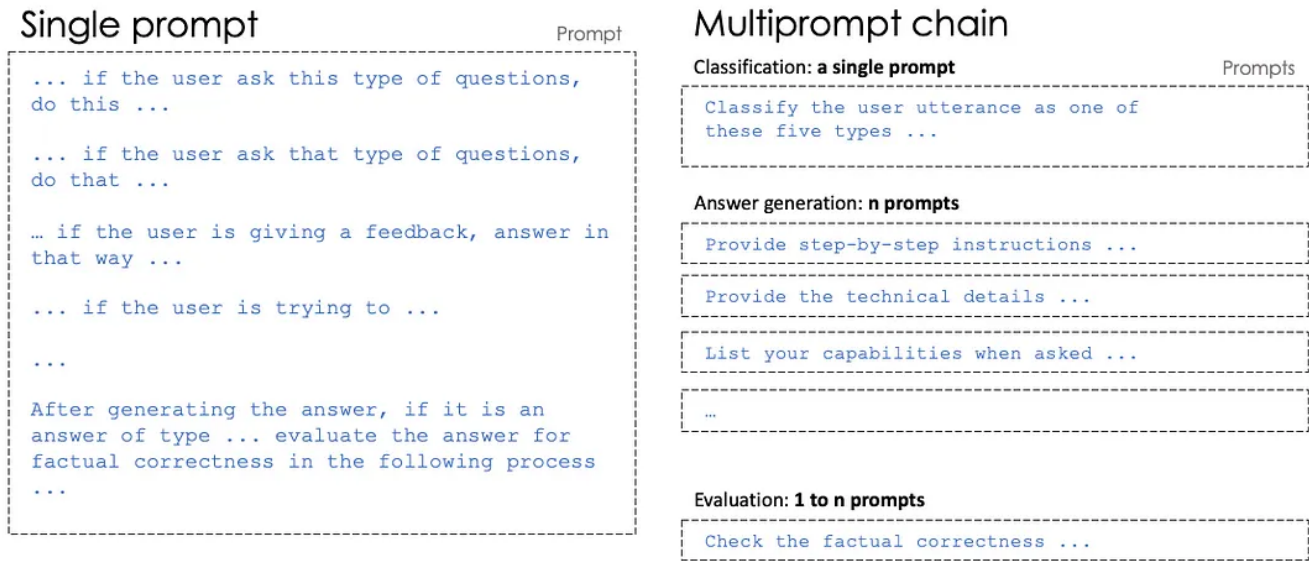
The prompt shown above is close to a little application on its own. To use it to build a basic working app that asks a child's name, asks questions and returns ratings for answers, you will only need around 50–100 lines of code in a langchain framework and a good textbook PDF. Just fyi, that creating an app or website which is really fun and allows kids to register, collect stars and badges, track learning progress will need much more development and framework.

## **11. Bonus Track: The Multiprompt Approach — When a Single Prompt is not Enough.**

If you have a large and complex application, e.g. an in-car system, a support hotline for a couple of products, an automation agent answering customer emails, etc., you may encounter problems to pack all processing rules for the model into one prompt. Length-wise it still could fit, even with few-shot-examples, but the model may get confused at some point, what to do in a specific case: Which examples are relevant for this case, which output format should be selected. The best solution is to choose a multiprompt approach, where you first classify the user utterance or the dialog history and then based on the classification, give a clear instruction and customized examples.

Implementationwise you typically send the user utterance to a model with a classification prompt forcing it to answer with an item from a list of

predefined values, like the type of user utterance (“how\_to\_question”), (“feature\_question”), (“bot\_capability\_question”), (“feedback”), ... You read the result out and open a specific chain for each type of conversation.



Multiprompt approach

This approach correctly implemented will result in a better accuracy and specificity of the answers. Splitting up the prompt can result in a reduction of input tokens, while leaving the output tokens in a similar dimension. So, it will not have a major impact on the costs.

I hope the story helps you to become a master prompt engineer or designer. Feel free to **give me feedback or ask me questions** here using the comment function.

**Good luck with your AI-powered application!**

**WRITER at MLearning.ai / AI Movie Director / imagine AI 3D Models**

<p><b>Mlearning.ai Submission Suggestions</b> How to become a writer on Mlearning.ai medium.com</p>	
---	--

- Prompt Design
- Prompt Engineering
- Llm
- Machine Learning
- MI So Good



# Written by Maximilian Vogel



7.9K Followers · Writer for MLearning.ai

Machine learning, large language models, NLP enthusiast and speaker. Co-founder BIG PICTURE.

## More from Maximilian Vogel and MLearning.ai



Maximilian Vogel <sup>in</sup> MLearning.ai

### The ChatGPT list of lists: A collection of 3000+ prompts,...

Updated Jan-11, 2024. Added new resources, GPT Store

11 min read · Feb 7, 2023

11.3K 135



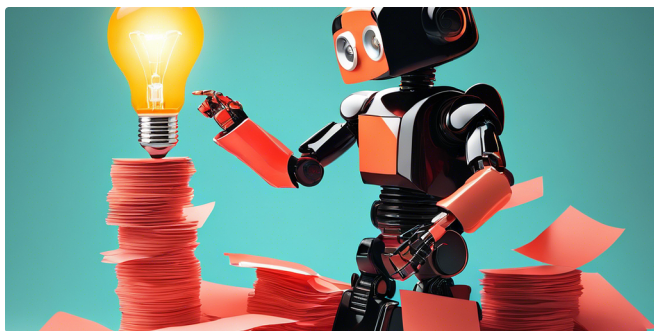
Yennhi95zz <sup>in</sup> MLearning.ai

### How to Earn Passive Income with Your Python Coding: 5 Innovative...

Expand your perspective: Python goes beyond a programming language.

🌟 · 7 min read · Dec 4, 2023

925 12



Abhinav Kimothi <sup>in</sup> MLearning.ai



Maximilian Vogel <sup>in</sup> MLearning.ai

## Getting the Most from LLMs: Building a Knowledge Brain for...

The advancements in the LLM space have been mind-boggling. However, when it come...

14 min read · Dec 21, 2023

 563  8



## The 10 Best Free Prompt Engineering Courses & Resource...

A well-crafted prompt is magic.

8 min read · Sep 6, 2023

 669  16



See all from Maximilian Vogel

See all from MLearning.ai

## Recommended from Medium



 Maximilian Vogel <sup>in</sup> MLearning.ai

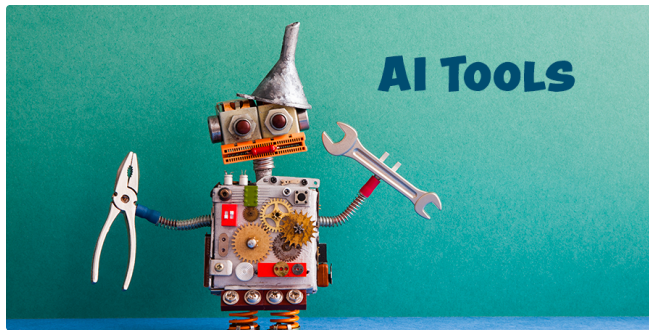
## The 10 Best Free Prompt Engineering Courses & Resource...

A well-crafted prompt is magic.

8 min read · Sep 6, 2023

 669

 16



 Digital Giraffes

## 7 Awesome and Free AI Tools You Should Know

We collected 7 free artificial intelligence(AI) tools, most of them easy to use and some...

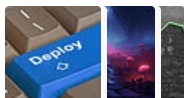
5 min read · Nov 17, 2022

 12.4K

 301



### Lists



#### Predictive Modeling w/ Python

20 stories · 774 saves



#### Practical Guides to Machine Learning

10 stories · 892 saves



#### Natural Language Processing

1081 stories · 554 saves



#### data science and AI

39 stories · 37 saves




Followers will receive notifications when you [add a new product](#).

**2112**

**Favorites**

The number of times people added your products to their favorites.

**\$413.01**



 Pragmatic Coders


## AI predictions: Top 13 AI trends for 2024

Explore the future with our comprehensive guide to the top 13 AI trends anticipated for...

13 min read · Dec 10, 2023

 5.1K  126



 Cameron R. Wolfe, Ph.D. <sup>in</sup> Towards Data Science

## Advanced Prompt Engineering

What to do when few-shot learning isn't enough...

🌟 · 17 min read · Aug 7, 2023

 1.6K  13



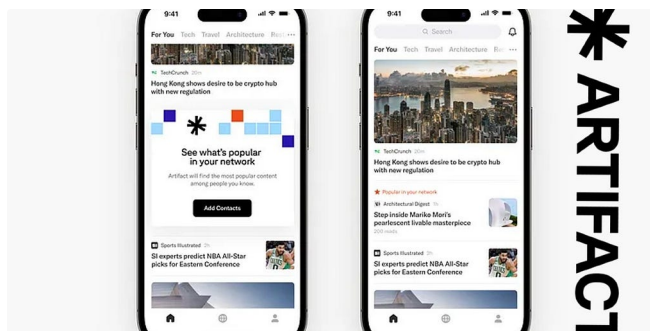
 Paul Rose

## I Found 3 Passive Income Ideas Anyone Can Start

Build your First or Next Passive Income Stream Today

9 min read · Nov 3, 2023

 11.6K  227



 Gowtham Oleti

## Apps I Use And Why You Should Too.

Let's skip past the usual suspects like YouTube, WhatsApp and Instagram. I want t...

10 min read · Nov 14, 2023

 13.7K  223



See more recommendations