Guidelines for Prompting

In this lesson, you'll practice two prompting principles and their related tactics in order to write effective prompts for large language models.

Setup

Load the API key and relevant Python libaries.

In this course, we've provided some code that loads the OpenAI API key for you.

In [5]:
```
import openai
import os

from dotenv import load_dotenv, find_dotenv
_ = load_dotenv(find_dotenv())

openai.api_key  = os.getenv('OPENAI_API_KEY')
```

helper function

Throughout this course, we will use OpenAI's `gpt-3.5-turbo` model and the chat completions endpoint (https://platform.openai.com/docs/guides/chat).

This helper function will make it easier to use prompts and look at the generated outputs.
Note: In June 2023, OpenAI updated gpt-3.5-turbo. The results you see in the notebook may be slightly different than those in the video. Some of the prompts have also been slightly modified to product the desired results.

In [6]:
```
def get_completion(prompt, model="gpt-3.5-turbo"):
    messages = [{"role": "user", "content": prompt}]
    response = openai.ChatCompletion.create(
        model=model,
        messages=messages,
        temperature=0, # this is the degree of randomness of the mod
    )
    return response.choices[0].message["content"]
```

Note: This and all other lab notebooks of this course use OpenAI library version `0.27.0`.

In order to use the OpenAI library version `1.0.0`, here is the code that you would use instead for the `get_completion` function:

```
client = openai.OpenAI()

def get_completion(prompt, model="gpt-3.5-turbo"):
    messages = [{"role": "user", "content": prompt}]
    response = client.chat.completions.create(
        model=model,
        messages=messages,
        temperature=0
    )
    return response.choices[0].message.content
```

Prompting Principles

- **Principle 1: Write clear and specific instructions**
- **Principle 2: Give the model time to "think"**

Tactics

Tactic 1: Use delimiters to clearly indicate distinct parts of the input

- Delimiters can be anything like: ```, """, < >, `<tag> </tag>`, `:`

In [7]:
```
text = f"""
You should express what you want a model to do by \
providing instructions that are as clear and \
specific as you can possibly make them. \
This will guide the model towards the desired output, \
and reduce the chances of receiving irrelevant \
or incorrect responses. Don't confuse writing a \
clear prompt with writing a short prompt. \
In many cases, longer prompts provide more clarity \
and context for the model, which can lead to \
more detailed and relevant outputs.
"""
prompt = f"""
Summarize the text delimited by triple backticks \
into a single sentence.
```{text}```
"""
response = get_completion(prompt)
print(response)
```

```
It is important to provide clear and specific instructions to a model
in order to guide it towards the desired output and reduce the chances
of receiving irrelevant or incorrect responses, with longer prompts of
ten providing more clarity and context for the model.
```

Tactic 2: Ask for a structured output

- JSON, HTML
```

In [8]:
```
prompt = f"""
Generate a list of three made-up book titles along \
with their authors and genres.
Provide them in JSON format with the following keys:
book_id, title, author, genre.
"""
response = get_completion(prompt)
print(response)
```

```
[
    {
        "book_id": 1,
        "title": "The Midnight Garden",
        "author": "Elena Blackwood",
        "genre": "Fantasy"
    },
    {
        "book_id": 2,
        "title": "Echoes of the Past",
        "author": "Nathan Rivers",
        "genre": "Mystery"
    },
    {
        "book_id": 3,
        "title": "Whispers in the Wind",
        "author": "Aria Nightingale",
        "genre": "Romance"
    }
]
```

Tactic 3: Ask the model to check whether conditions are satisfied

In [9]:
```
text_1 = f"""
Making a cup of tea is easy! First, you need to get some \
water boiling. While that's happening, \
grab a cup and put a tea bag in it. Once the water is \
hot enough, just pour it over the tea bag. \
Let it sit for a bit so the tea can steep. After a \
few minutes, take out the tea bag. If you \
like, you can add some sugar or milk to taste. \
And that's it! You've got yourself a delicious \
cup of tea to enjoy.
"""
prompt = f"""
You will be provided with text delimited by triple quotes.
If it contains a sequence of instructions, \
re-write those instructions in the following format:

Step 1 - ...
Step 2 - …
…
Step N - …

If the text does not contain a sequence of instructions, \
then simply write \"No steps provided.\"

\"\"\"{text_1}\"\"\"
"""
response = get_completion(prompt)
print("Completion for Text 1:")
print(response)
```

```
Completion for Text 1:
Step 1 - Get some water boiling.
Step 2 - Grab a cup and put a tea bag in it.
Step 3 - Once the water is hot enough, pour it over the tea bag.
Step 4 - Let it sit for a bit so the tea can steep.
Step 5 - After a few minutes, take out the tea bag.
Step 6 - Add some sugar or milk to taste.
Step 7 - Enjoy your delicious cup of tea.
```

In [10]:
```
text_2 = f"""
The sun is shining brightly today, and the birds are \
singing. It's a beautiful day to go for a \
walk in the park. The flowers are blooming, and the \
trees are swaying gently in the breeze. People \
are out and about, enjoying the lovely weather. \
Some are having picnics, while others are playing \
games or simply relaxing on the grass. It's a \
perfect day to spend time outdoors and appreciate the \
beauty of nature.
"""
prompt = f"""
You will be provided with text delimited by triple quotes.
If it contains a sequence of instructions, \
re-write those instructions in the following format:

Step 1 - ...
Step 2 - …
…
Step N - …

If the text does not contain a sequence of instructions, \
then simply write \"No steps provided.\"

\"\"\"{text_2}\"\"\"
"""
response = get_completion(prompt)
print("Completion for Text 2:")
print(response)
```

```
Completion for Text 2:
No steps provided.
```

Tactic 4: "Few-shot" prompting

In [11]:
```
prompt = f"""
Your task is to answer in a consistent style.

<child>: Teach me about patience.

<grandparent>: The river that carves the deepest \
valley flows from a modest spring; the \
grandest symphony originates from a single note; \
the most intricate tapestry begins with a solitary thread.

<child>: Teach me about resilience.
"""
response = get_completion(prompt)
print(response)
```

```
<grandparent>: Resilience is like a tree that bends in the \
strongest winds but never breaks; it is the \
ability to bounce back from adversity and \
continue to grow and thrive despite challenges.
```

Principle 2: Give the model time to “think”

Tactic 1: Specify the steps required to complete a task

In [12]:
```
text = f"""
In a charming village, siblings Jack and Jill set out on \
a quest to fetch water from a hilltop \
well. As they climbed, singing joyfully, misfortune \
struck—Jack tripped on a stone and tumbled \
down the hill, with Jill following suit. \
Though slightly battered, the pair returned home to \
comforting embraces. Despite the mishap, \
their adventurous spirits remained undimmed, and they \
continued exploring with delight.
"""
# example 1
prompt_1 = f"""
Perform the following actions:
1 - Summarize the following text delimited by triple \
backticks with 1 sentence.
2 - Translate the summary into French.
3 - List each name in the French summary.
4 - Output a json object that contains the following \
keys: french_summary, num_names.

Separate your answers with line breaks.

Text:
```{text}```
"""
response = get_completion(prompt_1)
print("Completion for prompt 1:")
print(response)
```

```
Completion for prompt 1:
1 - Jack and Jill, siblings, go on a quest to fetch water from a hillt
op well, but encounter misfortune along the way.

2 - Jack et Jill, frère et sœur, partent en quête d'eau d'un puits au
sommet d'une colline, mais rencontrent des malheurs en chemin.

3 - Jack, Jill

4 -
{
  "french_summary": "Jack et Jill, frère et sœur, partent en quête d'e
au d'un puits au sommet d'une colline, mais rencontrent des malheurs e
n chemin.",
  "num_names": 2
}
```

Ask for output in a specified format
```

In [13]:
```
prompt_2 = f"""
Your task is to perform the following actions:
1 - Summarize the following text delimited by
  <> with 1 sentence.
2 - Translate the summary into French.
3 - List each name in the French summary.
4 - Output a json object that contains the
  following keys: french_summary, num_names.

Use the following format:
Text: <text to summarize>
Summary: <summary>
Translation: <summary translation>
Names: <list of names in summary>
Output JSON: <json with summary and num_names>

Text: <{text}>
"""
response = get_completion(prompt_2)
print("\nCompletion for prompt 2:")
print(response)
```

```
Completion for prompt 2:
Summary: Jack and Jill, two siblings, go on a quest to fetch water fro
m a hilltop well but encounter misfortune along the way.

Translation: Jack et Jill, deux frères et sœurs, partent en quête d'ea
u d'un puits au sommet d'une colline mais rencontrent des malheurs en
chemin.

Names: Jack, Jill

Output JSON:
{
  "french_summary": "Jack et Jill, deux frères et sœurs, partent en qu
ête d'eau d'un puits au sommet d'une colline mais rencontrent des malh
eurs en chemin.",
  "num_names": 2
}
```

Tactic 2: Instruct the model to work out its own solution before rushing to a conclusion

In [14]:
```
prompt = f"""
Determine if the student's solution is correct or not.

Question:
I'm building a solar power installation and I need \
 help working out the financials.
- Land costs $100 / square foot
- I can buy solar panels for $250 / square foot
- I negotiated a contract for maintenance that will cost \
me a flat $100k per year, and an additional $10 / square \
foot
What is the total cost for the first year of operations
as a function of the number of square feet.

Student's Solution:
Let x be the size of the installation in square feet.
Costs:
1. Land cost: 100x
2. Solar panel cost: 250x
3. Maintenance cost: 100,000 + 100x
Total cost: 100x + 250x + 100,000 + 100x = 450x + 100,000
"""
response = get_completion(prompt)
print(response)
```

```
The student's solution is correct. The total cost for the first year o
f operations as a function of the number of square feet is indeed 450x
+ 100,000.
```

Note that the student's solution is actually not correct.

We can fix this by instructing the model to work out its own solution first.

In [15]:
```
 f"""
k is to determine if the student's solution \
ct or not.
 the problem do the following:
 work out your own solution to the problem including the final total.
ompare your solution to the student's solution \
uate if the student's solution is correct or not.
cide if the student's solution is correct until
 done the problem yourself.

following format:
:

 here

s solution:

s solution here
```

```
olution:

 work out the solution and your solution here

tudent's solution the same as actual solution \
culated:

o

grade:

or incorrect


:

ding a solar power installation and I need help \
out the financials.
osts $100 / square foot
buy solar panels for $250 / square foot
tiated a contract for maintenance that will cost \
t $100k per year, and an additional $10 / square \

the total cost for the first year of operations \
ction of the number of square feet.

s solution:

 the size of the installation in square feet.

cost: 100x
 panel cost: 250x
enance cost: 100,000 + 100x
st: 100x + 250x + 100,000 + 100x = 450x + 100,000

olution:

 = get_completion(prompt)
sponse)
```

```
The total cost for the first year of operations is calculated as follo
ws:
1. Land cost: $100/sq ft * x sq ft = $100x
2. Solar panel cost: $250/sq ft * x sq ft = $250x
3. Maintenance cost: $100,000 + $10/sq ft * x sq ft = $100,000 + $10x
Total cost: $100x + $250x + $100,000 + $10x = $360x + $100,000

Is the student's solution the same as actual solution just calculated:
```

```
No
```

```
Student grade:
```

```
Incorrect
```

Model Limitations: Hallucinations

- Boie is a real company, the product name is not real.

In [16]:
```
prompt = f"""
Tell me about AeroGlide UltraSlim Smart Toothbrush by Boie
"""
response = get_completion(prompt)
print(response)
```

```
The AeroGlide UltraSlim Smart Toothbrush by Boie is a high-tech toothb
rush designed to provide a superior cleaning experience. It features u
ltra-soft bristles that are gentle on the gums and teeth, while still
effectively removing plaque and debris. The toothbrush also has a slim
design that makes it easy to maneuver and reach all areas of the mout
h.

One of the standout features of the AeroGlide UltraSlim Smart Toothbru
sh is its smart technology, which includes a built-in timer and pressu
re sensor. The timer helps users brush for the recommended two minute
s, while the pressure sensor alerts users if they are brushing too har
d, helping to prevent damage to the gums and enamel.

The toothbrush is also made from antimicrobial materials, which helps
to prevent the growth of bacteria on the bristles. This makes it a hyg
ienic option for oral care.

Overall, the AeroGlide UltraSlim Smart Toothbrush by Boie is a sleek a
nd innovative toothbrush that offers a gentle yet effective cleaning e
xperience.
```

Try experimenting on your own!

In [17]:
```
prompt = f""" Tell me about AeroGlide UltraSlim Smart Toothbrush by

1. Utilize a search engine to find the latest reviews, specification
2. Focus on gathering information from credible sources, including t
3. Compile a summary that highlights key features, customer reviews,
4. Draft a brief report detailing the summarized findings, ensuring
"""
response = get_completion(prompt)
print(response)
```

```
The AeroGlide UltraSlim Smart Toothbrush by Boie is a cutting-edge den
tal health product that offers a range of innovative features to enhan
ce the brushing experience. According to the official Boie website, th
is toothbrush is designed with an ultra-slim profile for easy handling
and maneuverability, making it ideal for users of all ages.

Key features of the AeroGlide UltraSlim Smart Toothbrush include:

- Smart technology: The toothbrush is equipped with sensors that track
brushing habits and provide real-time feedback through a connected ap
p. This allows users to monitor their brushing technique and improve t
heir oral hygiene routine.
- Gentle yet effective cleaning: The toothbrush features soft, tapered
bristles that are gentle on gums while still providing thorough cleani
ng of teeth and gums.
- Long-lasting battery: The toothbrush boasts a long battery life, all
owing for extended use between charges.
- Waterproof design: The toothbrush is waterproof, making it safe for
use in the shower or while traveling.

Customer reviews of the AeroGlide UltraSlim Smart Toothbrush have been
overwhelmingly positive, with many users praising its sleek design, ea
se of use, and effectiveness in improving oral health. Reviewers have
also highlighted the convenience of the smart technology and the perso
nalized feedback provided by the connected app.

In conclusion, the AeroGlide UltraSlim Smart Toothbrush by Boie is a t
op-of-the-line dental health product that combines advanced technology
with user-friendly design. With its smart features, gentle cleaning ac
tion, and long-lasting battery, this toothbrush is a standout option f
or those looking to upgrade their oral hygiene routine.
```

Notes on using the OpenAI API outside of this classroom

To install the OpenAI Python library:

```
!pip install openai
```

The library needs to be configured with your account's secret key, which is available on the website (https://platform.openai.com/account/api-keys).

You can either set it as the `OPENAI_API_KEY` environment variable before using the library:

```
!export OPENAI_API_KEY='sk-...'
```

Or, set `openai.api_key` to its value:

```
import openai
openai.api_key = "sk-..."
```

A note about the backslash

- In the course, we are using a backslash `\` to make the text fit on the screen without inserting newline '\n' characters.
- GPT-3 isn't really affected whether you insert newline characters or not. But when working with LLMs in general, you may consider whether newline characters in your prompt may affect the model's performance.

In [ ]: