Introduction to prompt engineering

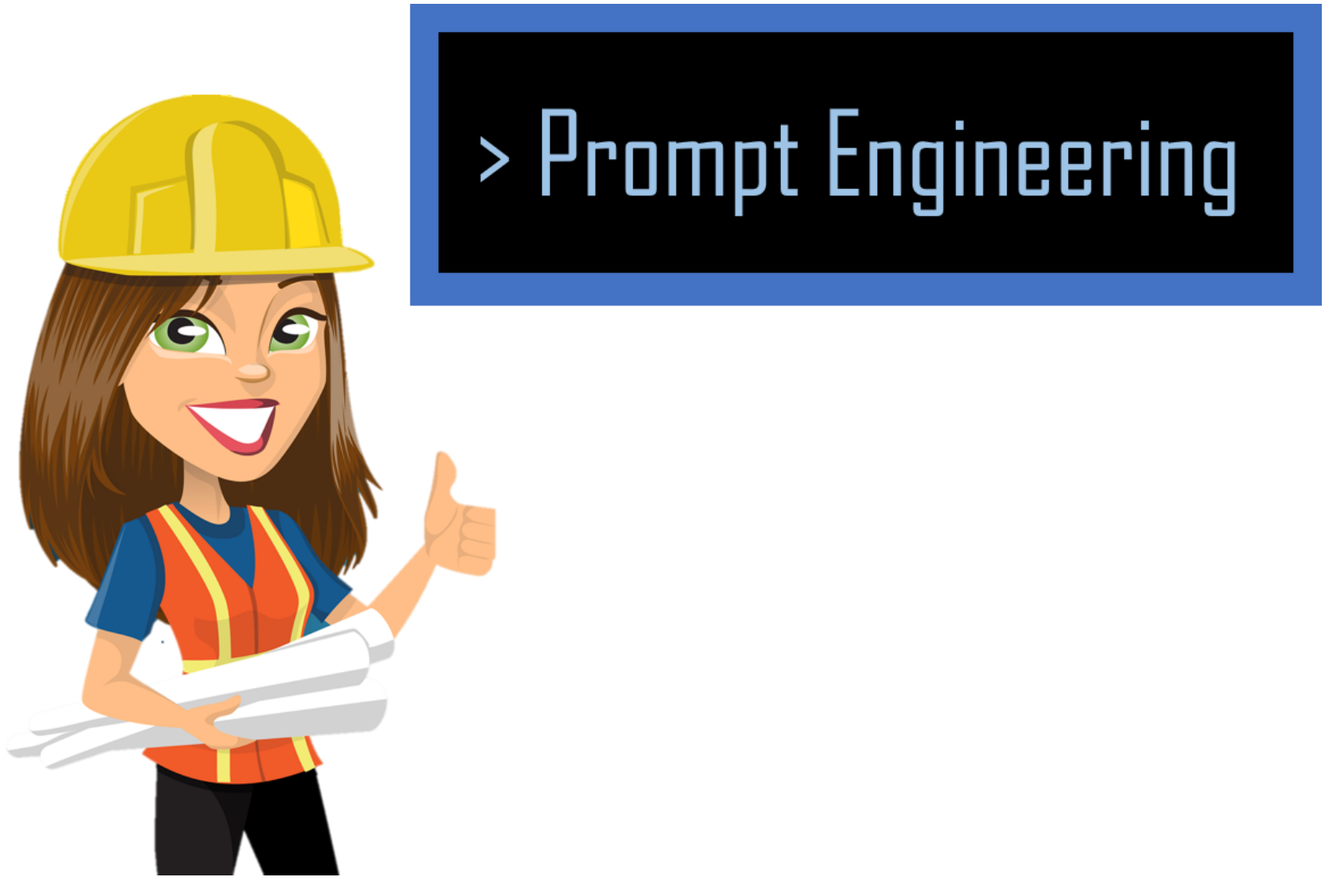
PROMPT ENGINEERING WITH THE OPENAI API

Fouad Trad
Machine Learning Engineer

datacamp

What is prompt engineering?

Crafting prompts or instructions given to LLMs to get desired responses



Prompt engineering is like crafting a recipe

Why prompt engineering?

Why prompt engineering?

Recap: OpenAI API

- Allows interaction with OpenAI models
- Already set up in this course
- Access to Chat Completions endpoint

Recap: message roles

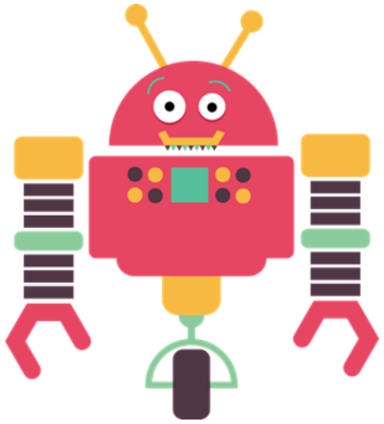
Every message has one of **three** roles

User

Recap: message roles

Every message has one of **three** roles

User

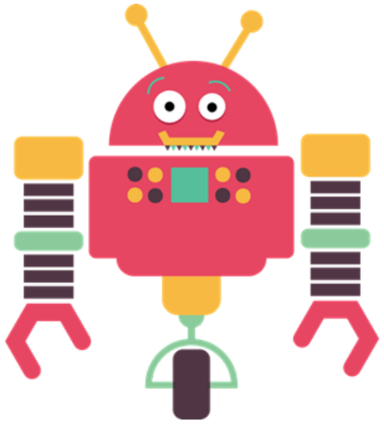


Assistant

Recap: message roles

Every message has one of **three** roles

User



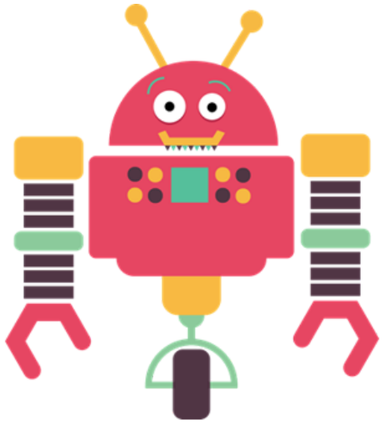
Assistant

System

Recap: message roles

Every message has one of **three** roles

User



Assistant

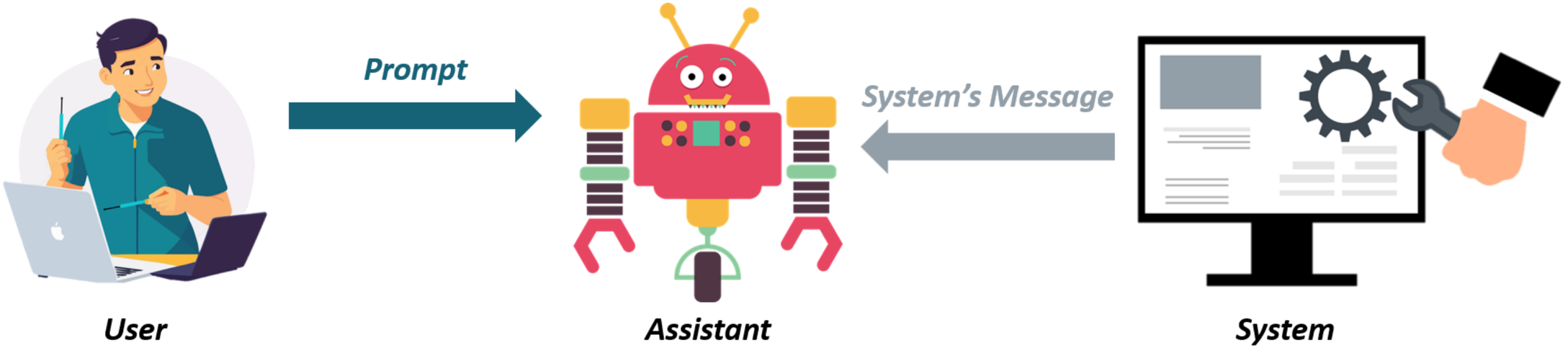
System's Message

System

- **System message:** guides model behavior

Recap: message roles

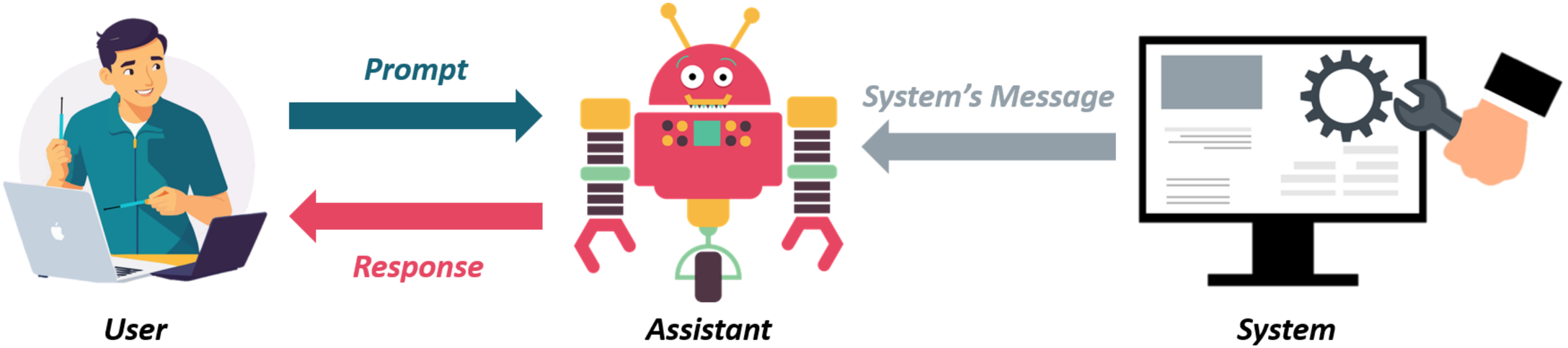
Every message has one of **three** roles



- **System message:** guides model behavior
- **User message:** prompt from the user

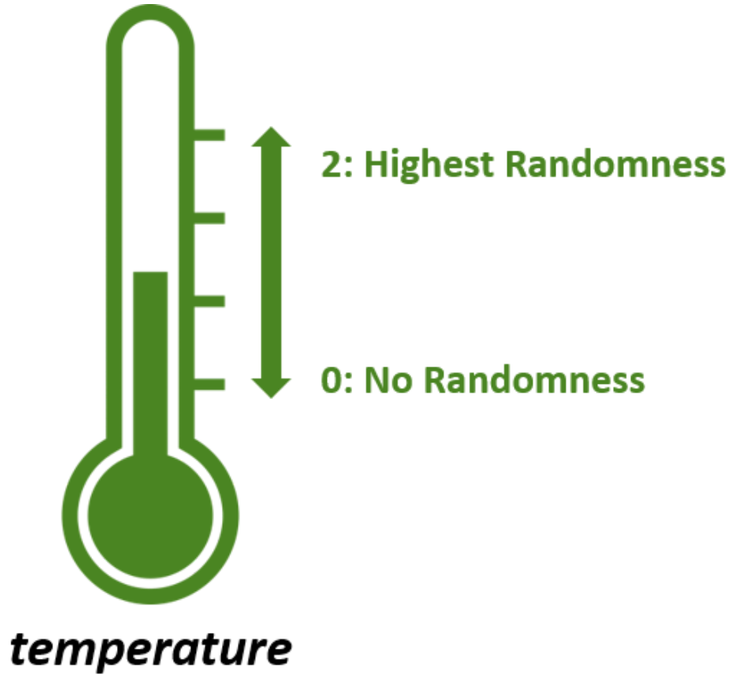
Recap: message roles

Every message has one of **three** roles



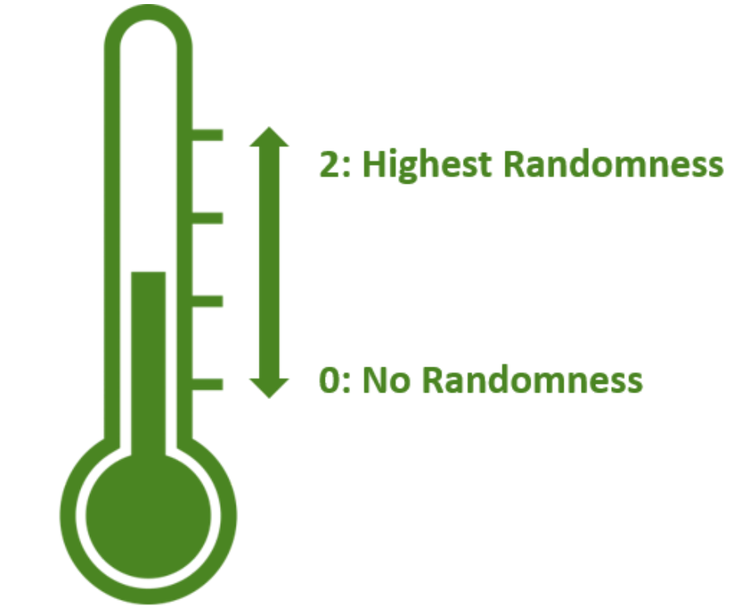
- **System message:** guides model behavior
- **User message:** prompt from the user
- **Assistant message:** response to user prompt

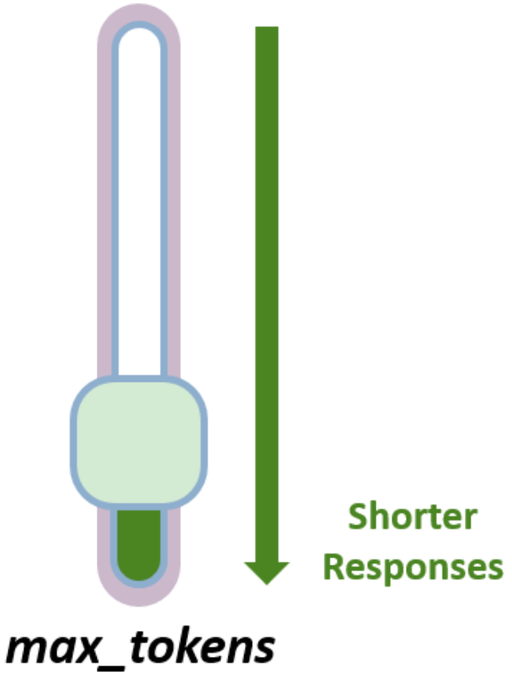
Recap: control parameters



- `temperature` : controls answer's randomness

Recap: control parameters





- `temperature` : controls answer's randomness
- `max_tokens` : controls response length

Recap: communicating with the OpenAI API

```
prompt = "What is prompt engineering?"
client = OpenAI(api_key="api_key")
response = client.chat.completions.create(
  model = "gpt-3.5-turbo",
  messages = [{"role": "user",
               "content": prompt}],
  temperature = 0
)
print(response.choices[0].message.content)
```

```
Prompt engineering refers to the process of designing and refining prompts or
instructions given to a language model like ChatGPT in order to elicit desired
responses or behaviors. It involves formulating specific guidelines or hints to
guide the model's output towards a desired outcome.
```

Creating get_response() function

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

Usage

```
response = get_response("What is prompt engineering?")
print(response)
```

```
Prompt engineering refers to the process of designing and refining prompts or instructions given to a language model
like ChatGPT in order to elicit desired responses or behaviors. It involves formulating specific guidelines or hints
to guide the model's output towards a desired outcome.
```

Prompt improvement

```
prompt = "What is prompt engineering? Explain it in terms that can be understood by a 5-year-old"
response = get_response(prompt)
print(response)
```

```
Imagine you have a very smart friend who can understand and answer lots of questions. But sometimes, they might not understand exactly what you want or give the wrong answer. So, prompt engineering is like giving your friend really clear instructions or hints to help them give you the best answer possible.
```

Let's practice!

PROMPT ENGINEERING WITH THE OPENAI API

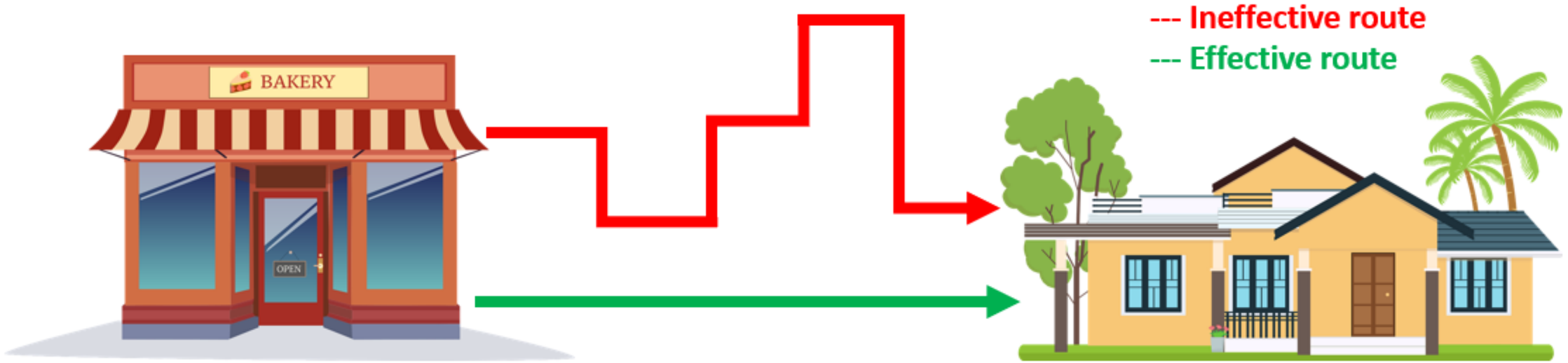
Key principles of prompt engineering

PROMPT ENGINEERING WITH THE OPENAI API

Fouad Trad
Machine Learning Engineer

datacamp

Clear and precise prompts



Key principles

- Appropriate action verbs
- Detailed and precise instructions
- Well-structured delimited prompts

Using action verbs

- Guide model what to do

Verbs to use
Write
Complete
Explain
Describe
Evaluate
...

Avoid ambiguous verbs

- Confuse model's understanding

Verbs to use	Verbs to avoid using
Write	*Understand*
Complete	*Think*
Explain	*Feel*
Describe	*Try*
Evaluate	*Know*
...	...

Effective prompt with verbs

```
prompt = "Think about the issue of deforestation."
response = get_response(prompt)
print(response)
```

```
Deforestation is a significant
environmental issue involving
permanently removing
or destroying forests and woodlands.
It has far-reaching impacts on the
environment, ecosystems, wildlife,
and human communities. [...]
```

```
prompt = "Propose strategies to reduce deforestation."
response = get_response(prompt)
print(response)
```

```
Reducing deforestation requires
a comprehensive and multi-dimensional
approach involving various stakeholders.
Here are several strategies
that can help address the issue:
Strengthen Forest Governance
Promote Sustainable Land Use [...]
```

Formulating detailed instructions

Provide **specific, descriptive, and detailed instructions** regarding:

- □ Context
- □ Output length
- □ Format and style
- □□ Audience

Effective prompt with instructions

Ineffective prompt: ~~"Tell me about dogs."~~

Effective prompt

```
prompt = "Write a descriptive paragraph about the behavior and characteristics of
Golden Retrievers, highlighting their friendly nature, intelligence,
and suitability as family pets."
print(get_response(prompt))
```

```
Golden Retrievers are beloved worldwide for their exceptional behavior, remarkable
characteristics, and friendly nature. They are highly intelligent, trainable, and
adaptable, making them great companions for families, including those with children.
```

Limiting output length

max_tokens:

- Limit on number of tokens
- Output cannot bypass it
- Might lead to incomplete or cut responses

Prompt:

- Limit on words, sentences, or paragraphs
- Output may bypass it
- Complete responses

Prompt components

- **Instructions** and **input data** to operate on
- Example: text summarization
  - **Instruction:** summarize the given text
  - **Input data:** text to summarize

Crafting a well-structured prompt with delimiters

- Start prompt with instructions
- Use **delimiters** (parentheses, brackets, backticks, etc.) to specify input parts
- Mention which delimiters are used

```
prompt = """Summarize the text delimited by triple backticks into bullet points.
            ```TEXT GOES HERE```"""
response = get_response(prompt)
```
```

Using formatted strings (f-strings)

- Include defined string into another string

```
text = "This is a sample text to summarize"
prompt = f"""Summarize the text delimited by triple backticks into bullet points.
            ```{text}```"""
print(prompt)
```

```
Summarize the text delimited by triple backticks into bullet points.
```This is a sample text to summarize```
```

Let's practice!

PROMPT ENGINEERING WITH THE OPENAI API

Structured outputs and conditional prompts

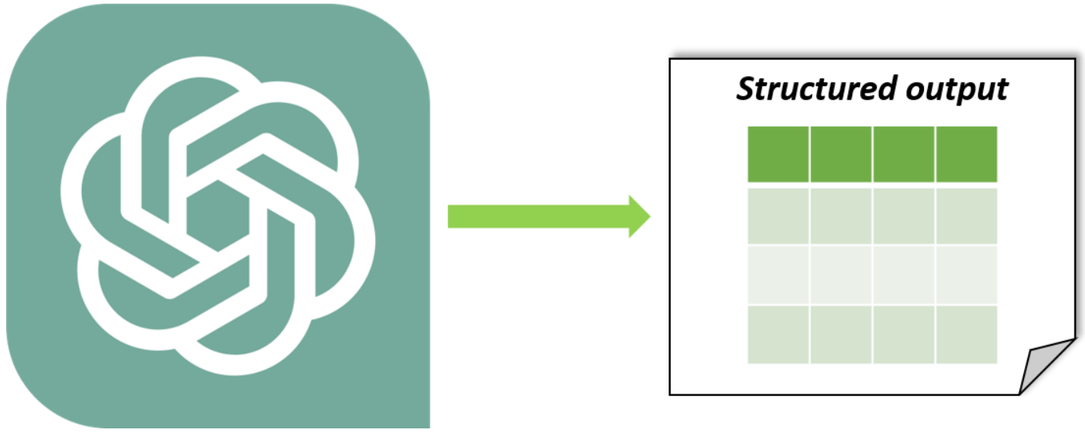
PROMPT ENGINEERING WITH THE OPENAI API

Fouad Trad
Machine Learning Engineer

datacamp

Structured outputs

- LLMs generate structured outputs only when given explicit instructions
- Output structures:
  - Table
  - List
  - Structured paragraph
  - Custom format



Tables

- Clearly mention expected columns

```
prompt = "Generate a table containing 5 movies I should watch if I am an action lover, with columns for Title and Rating."
print(get_response(prompt))
```

```
| Title                          | Rating |
|--------------------------------|--------|
| Mad Max: Fury Road             | 8.1    |
| John Wick                      | 7.4    |
| The Dark Knight                | 9.0    |
| Die Hard                       | 8.2    |
| Gladiator                      | 8.5    |
```

Lists

- Helpful for enumerations

```
prompt = "Generate a list containing the names of the top 5 cities to visit."
print(get_response(prompt))
```

```
1. Tokyo, Japan
2. Paris, France
3. Rome, Italy
4. New York City, United States
5. Sydney, Australia
```

Lists

- Requirements for numbering should be mentioned in prompt

```
prompt = "Generate an unordered list containing the names of the top 5 cities to visit."
print(get_response(prompt))
```

```
- Paris, France
- Tokyo, Japan
- Rome, Italy
- New York City, United States
- Sydney, Australia
```

Structured paragraphs

- Mention structure requirements in prompt

```
prompt = "Provide a structured paragraph with clear headings and subheadings
about the benefits of regular exercise on overall health and well-being."
print(get_response(prompt))
```

Structured paragraphs

```
I. Introduction
Regular exercise is essential for maintaining good health and overall well-being. [...]

II. Physical Health Benefits
a. Weight Management: Regular exercise is [...]
b. Reduced Risk of Chronic Diseases: Exercise has been linked to [...]
c. Increased Strength and Flexibility: Regular exercise improves muscle strength and [...]

III. Mental Health Benefits
a. Reduced Stress and Anxiety: Exercise has a profound impact on [...]
b. Improved Sleep Quality: Regular exercise is associated with improved sleep patterns [...]
c. Boosted Brain Function: Studies show that exercise enhances cognitive function [...]
```

Custom output format

```
text = "Once upon a time in a quaint little village, there lived a curious young boy named
David. David was [...]"
instructions = "You will be provided with a text delimited by triple backticks. Generate a
suitable title for it. "

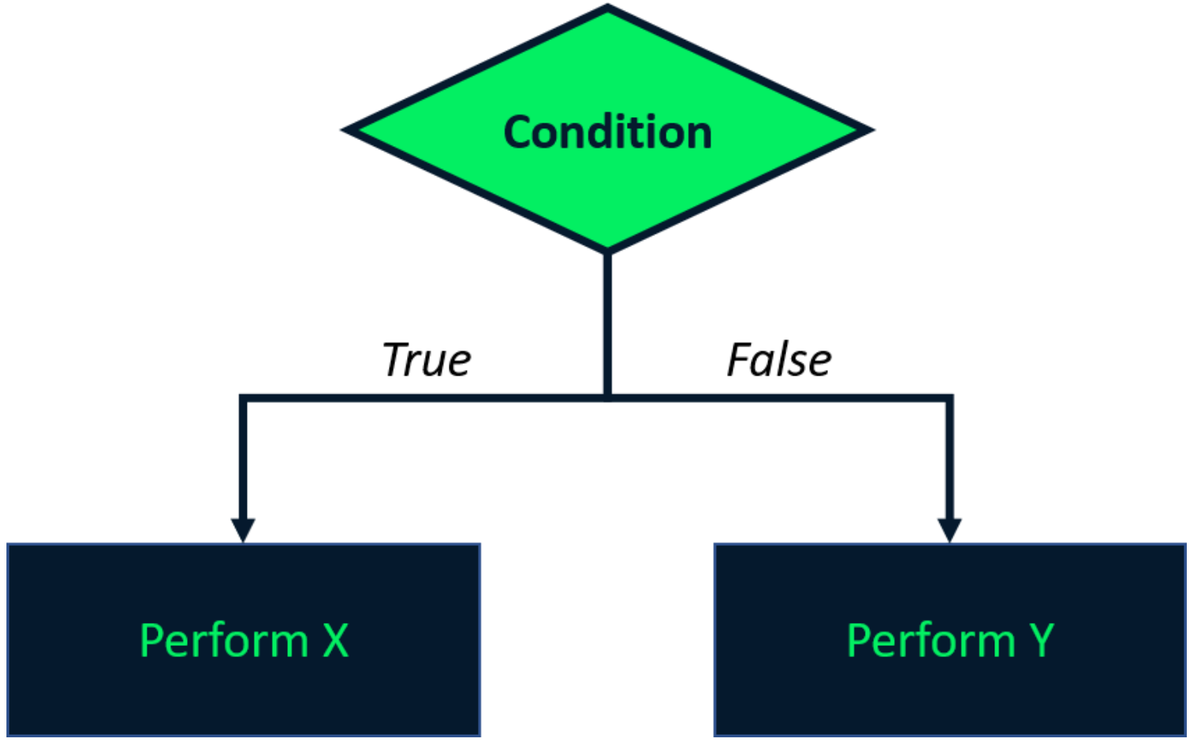
output_format = """Use the following format for the output:
        - Text: <text we want to title>
        - Title: <the generated title>"""

prompt = instructions + output_format + f"```{text}```"
print(get_response(prompt))
```

```
- Text: Once upon a time in a quaint little village, there lived a curious young boy [...]
- Title: The Extraordinary Adventure of David and the Mysterious Book
```

Conditional prompts

- Incorporate logic or conditions
- Conditional prompts follow an if-else style



Conditional prompts

```
text = "Le printemps est ma saison préférée. Quand les premières fleurs commencent
à éclore, et que les arbres se parent de feuilles vertes et tendres, je me sens
revivre [...] "
```

```
prompt = f"""You will be provided with a text delimited by triple backticks.
         If the text is written in English, suggest a suitable title for it.
         Otherwise, write 'I only understand English'.
         ```{text}```"""

print(get_response(prompt))
```

```
I only understand English
```
```

Conditional prompts

- Can incorporate multiple conditions

```
text = "In the heart of the forest, sunlight filters through the lush green canopy,
creating a tranquil atmosphere [...] "
```

```
prompt = f"""You will be provided with a text delimited by triple backticks.
         If the text is written in English, check if it contains the keyword 'technology'.



         ```{text}```"""
print(get_response(prompt))
```
```

Conditional prompts

- Can incorporate multiple conditions

```
text = "In the heart of the forest, sunlight filters through the lush green canopy,
creating a tranquil atmosphere [...] "
```

```
prompt = f"""You will be provided with a text delimited by triple backticks.
         If the text is written in English, check if it contains the keyword 'technology'.
         If it does, suggest a suitable title for it, otherwise, write 'Keyword not found'.

         ```{text}```"""
print(get_response(prompt))
```
```

Conditional prompts

- Can incorporate multiple conditions

```
text = "In the heart of the forest, sunlight filters through the lush green canopy,
creating a tranquil atmosphere [...] "
```

```
prompt = f"""You will be provided with a text delimited by triple backticks.
         If the text is written in English, check if it contains the keyword 'technology'.
         If it does, suggest a suitable title for it, otherwise, write 'Keyword not found'.
         If the text is not written in English, reply with 'I only understand English'.
         ```{text}```"""
print(get_response(prompt))
```

```
Keyword not found
```
```

Let's practice!

PROMPT ENGINEERING WITH THE OPENAI API