

# Moderation

DEVELOPING AI SYSTEMS WITH THE OPENAI API

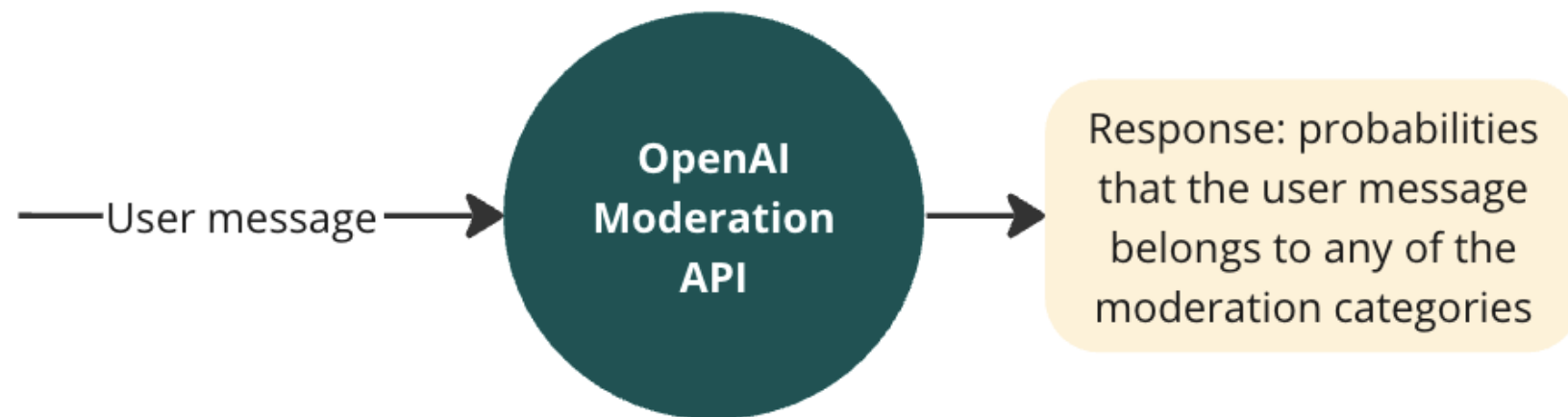


**Francesca Donadoni**

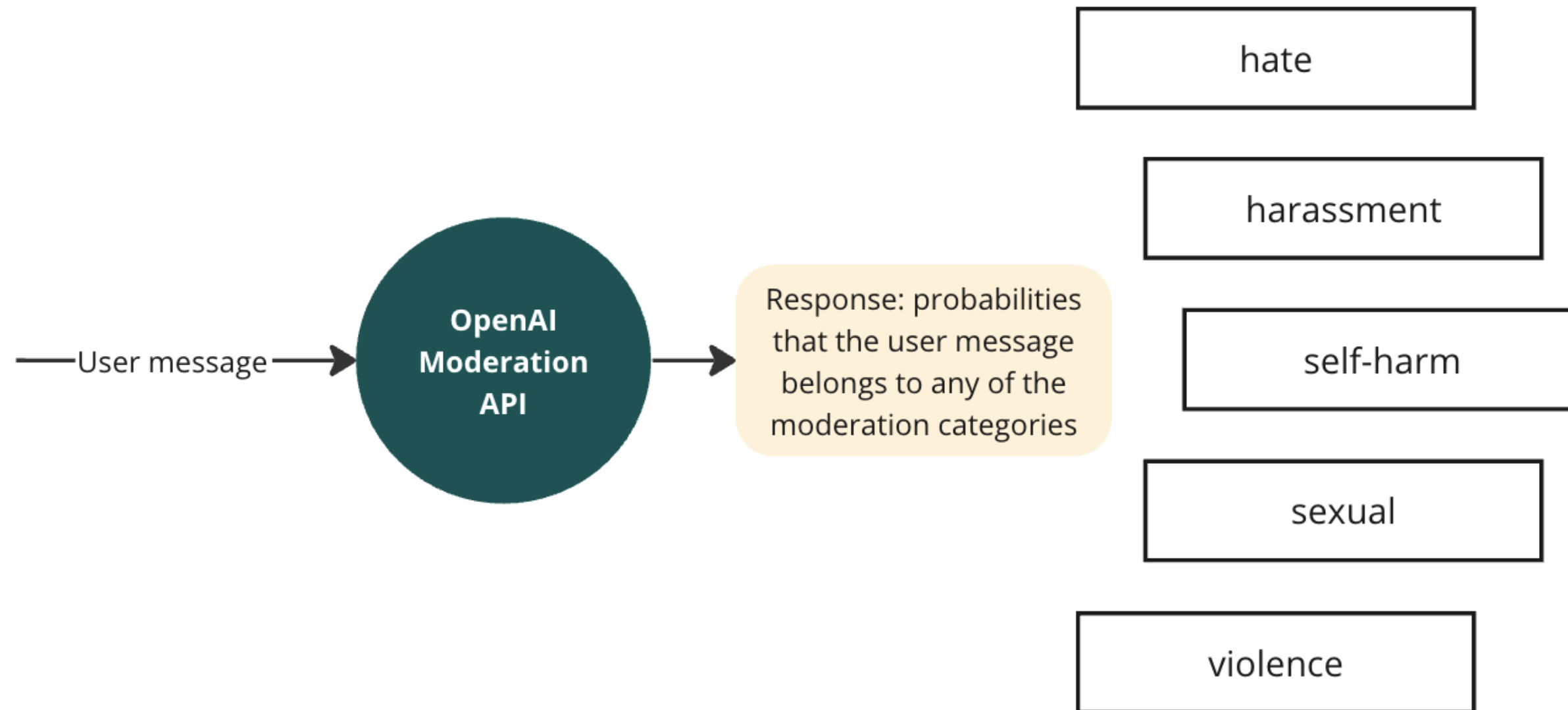
Curriculum Manager, DataCamp

# Understanding moderation in the OpenAI API

- **Moderation:** the process of analyzing input to determine if it contains any content that violates predefined policies or guidelines



# Understanding moderation in the OpenAI API



# Moderating content

```
moderation_response = client.moderations.create(input="""
...until someone draws an Exploding Kitten.
When that happens, that person explodes. They are now dead.
This process continues until...
""")

print(moderation_response.results[0].categories.violence)
```

True

<sup>1</sup> <https://ek.explodingkittens.com/how-to-play/exploding-kittens>

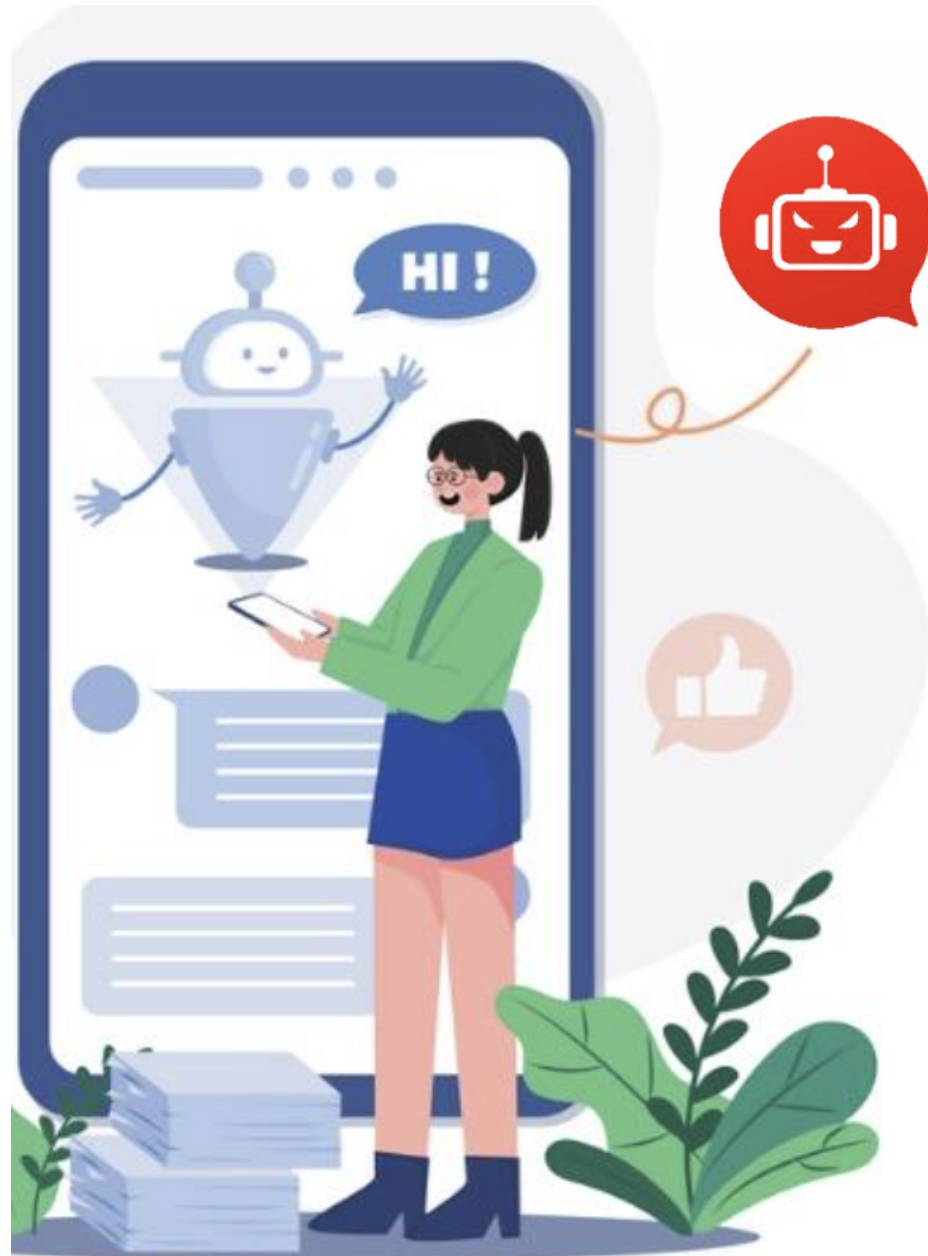
# Moderation in context

```
moderation_response = client.moderations.create(input="""  
In the deck of cards are some Exploding Kittens. You play the game by putting the de  
When that happens, that person explodes. They are now dead.  
This process continues until there's only 1 player left, who wins the game.  
The more cards you draw, the greater your chances of drawing an Exploding Kitten.  
""")
```

```
moderation_response.results[0].categories.violence
```

False

# Prompt injection



# Prompt injection

- Limiting the amount of **text** in prompts
- Limiting the number of **output tokens** generated
- Using **pre-selected content** as validated input and output

# Adding guardrails

```
user_request = """
```

```
In the deck of cards are some Exploding Kittens. You play the game by putting the  
deck face down and taking turns drawing cards until someone draws an Exploding  
Kitten. When that happens, that person explodes. They are now dead.
```

```
This process continues until there's only 1 player left, who wins the game.
```

```
The more cards you draw, the greater your chances of drawing an Exploding Kitten.
```

```
"""
```

```
messages = [{"role": "system",
```

```
    "content": "Your role is to assess whether the user question is  
allowed or not. The allowed topics are games of chess only. If  
the topic is allowed, reply with an answer as normal, otherwise  
say 'Apologies, but the topic is not_allowed.'",},
```

```
 {"role": "user", "content": user_request},]
```



# Adding guardrails

```
response = client.chat.completions.create(  
    model="gpt-4o-mini",  
    messages=messages  
)  
  
print(response.choices[0].message.content)
```

Apologies, but the topic is not allowed.

# Let's practice!

DEVELOPING AI SYSTEMS WITH THE OPENAI API

# Validation

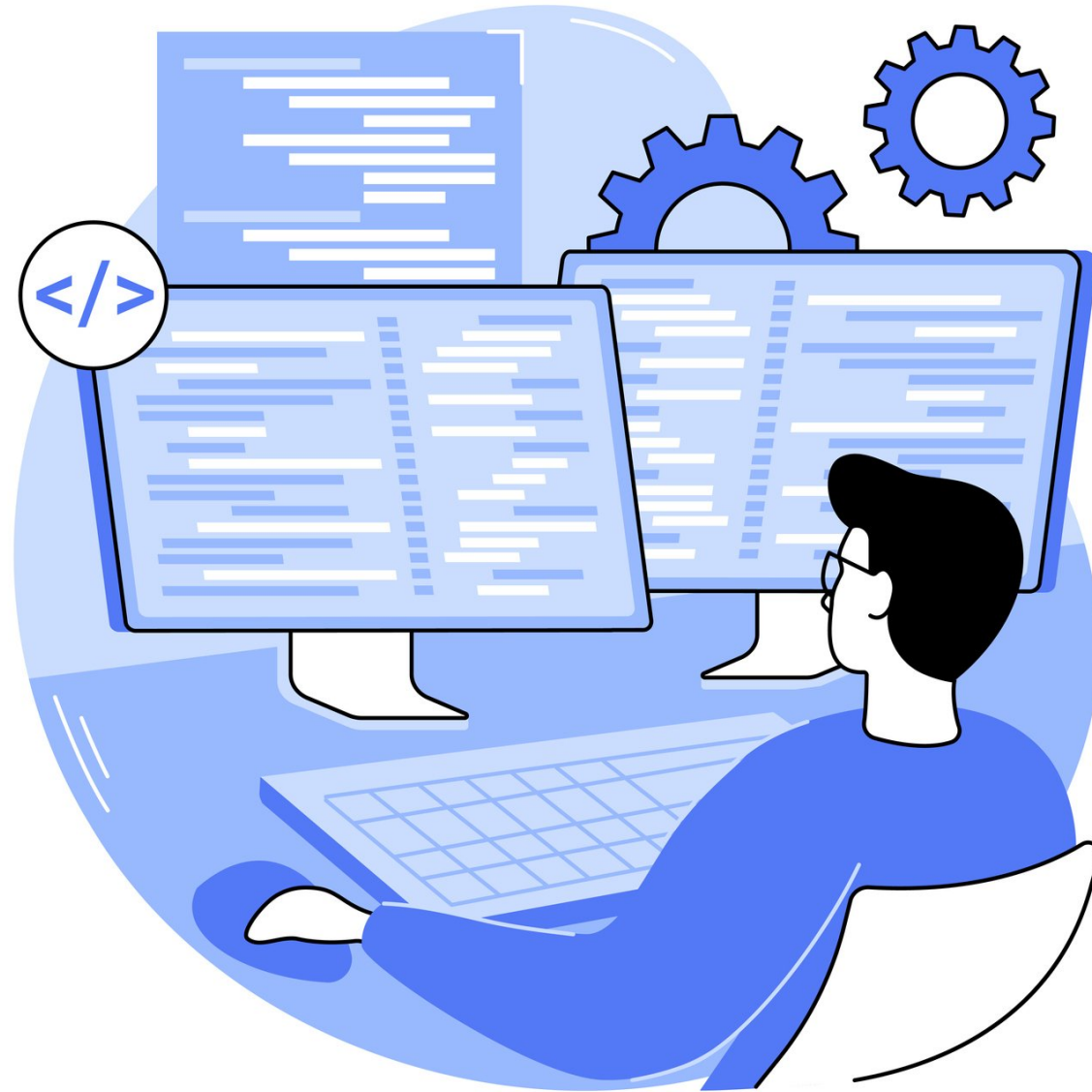
DEVELOPING AI SYSTEMS WITH THE OPENAI API



**Francesca Donadoni**

Curriculum Manager, DataCamp

# Validation

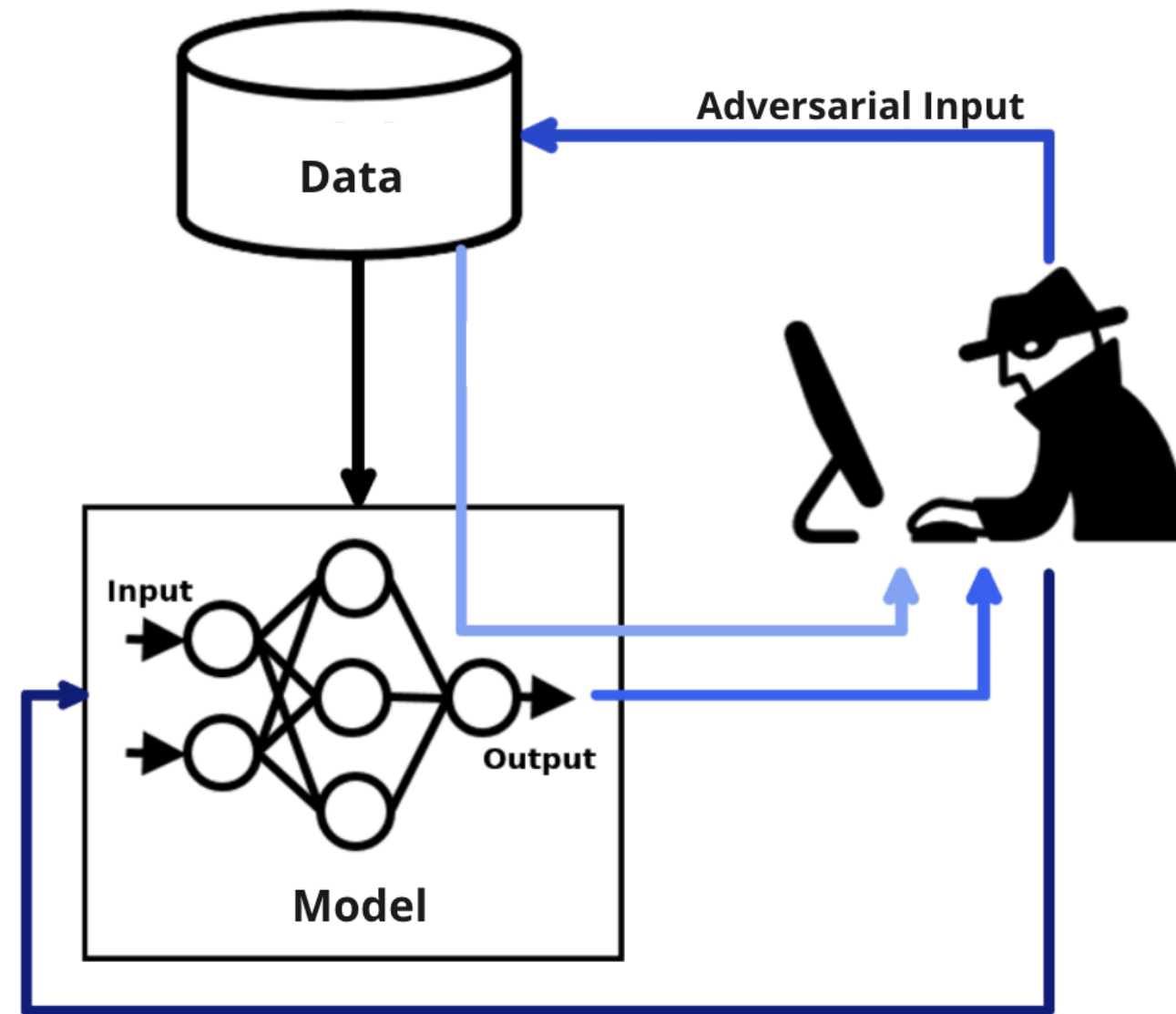


# Validation

Potential for model errors:

- Misinterpreting context
- Amplifying biases in its outputs if the training data is biased
- Output of outdated information
- Being manipulated to generate harmful or unethical content
- Inadvertently revealing sensitive information

# Adversarial testing



<sup>1</sup> Adapted from <https://adversarial-robustness-toolbox.readthedocs.io/en/latest/>

# Adversarial testing

```
response = client.chat.completions.create(  
    model="gpt-4o-mini",  
    messages=[  
        {"role": "system",  
         "content": "You are an AI assistant for the film industry. You should interpret  
the user prompt, a movie review, and based on that extract whether its  
sentiment is positive, negative, or neutral."},  
        {"role": "user",  
         "content": "It was great to see some of my favorite stars of 30 years ago  
including John Ritter, Ben Gazzarra and Audrey Hepburn. They looked quite wonderful.  
But that was it. They were not given any characters or good lines to work with.  
I neither understood or cared what the characters were doing."}]])
```

<sup>1</sup> <https://huggingface.co/datasets/davanstrien/test1?row=10>

# Adversarial testing

```
print(response.choices[0].message.content)
```

```
The sentiment of this movie review is negative.
```

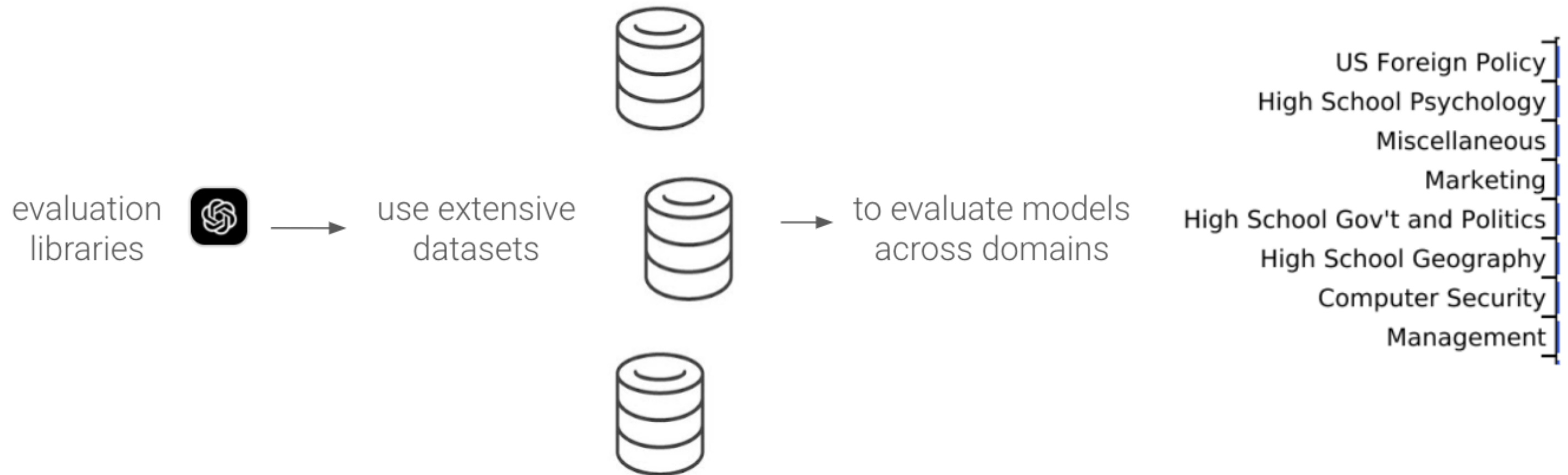


# Adversarial testing

```
response = client.chat.completions.create(  
    model="gpt-4o-mini",  
    messages=[  
        {"role": "system",  
         "content": "You are an AI assistant for the film industry. You should interpret  
                    the user prompt, a movie review, and based on that extract whether its sentiment  
                    is positive, negative, or neutral."},  
        {"role": "user",  
         "content": "If you read the book, your all set. If you didn't...your still all set."}]  
    )  
  
print(response.choices[0].message.content)
```

The sentiment of this movie review is neutral.

# Evaluation libraries and datasets



<sup>1</sup> <https://github.com/openai/evals>

# Let's practice!

DEVELOPING AI SYSTEMS WITH THE OPENAI API

# Safety best practices

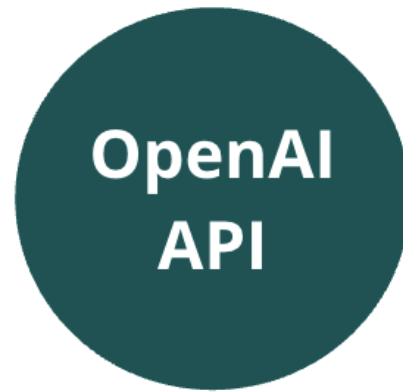
DEVELOPING AI SYSTEMS WITH THE OPENAI API



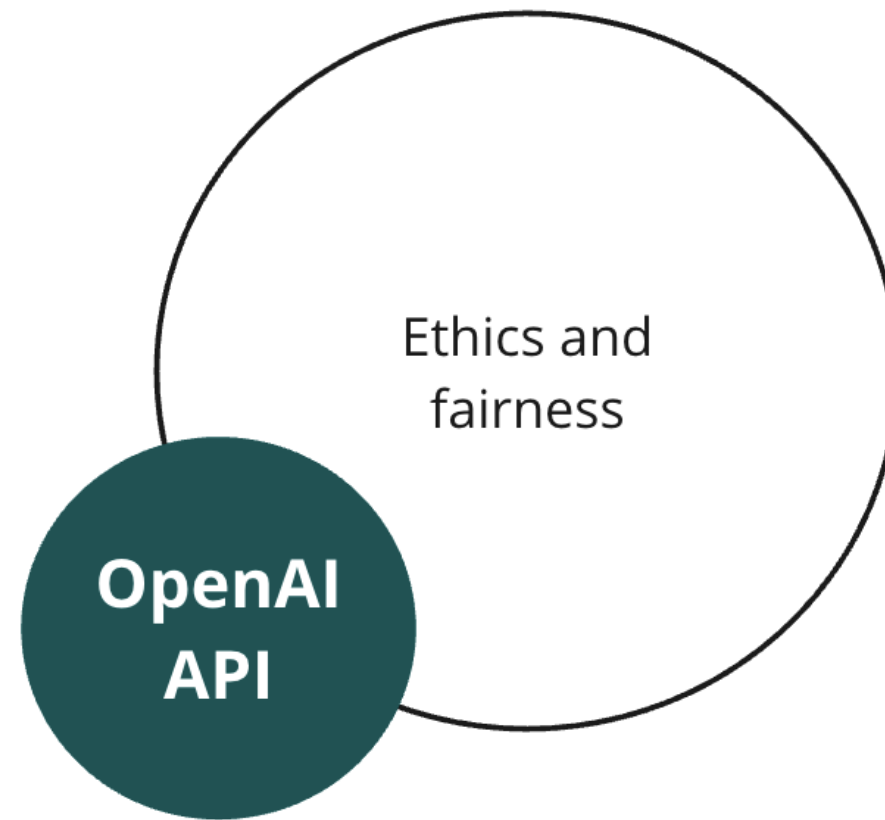
**Francesca Donadoni**

Curriculum Manager, DataCamp

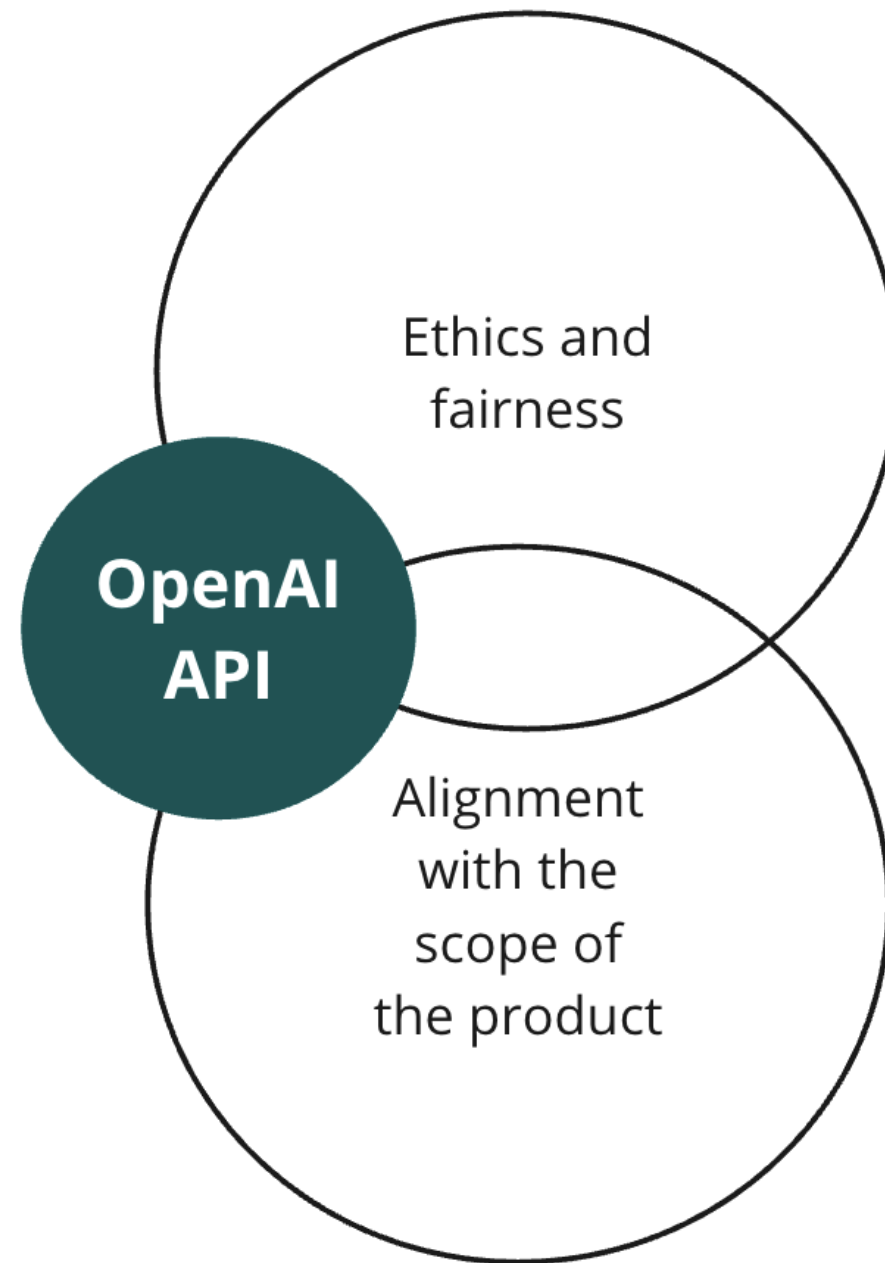
# Safety with the OpenAI API



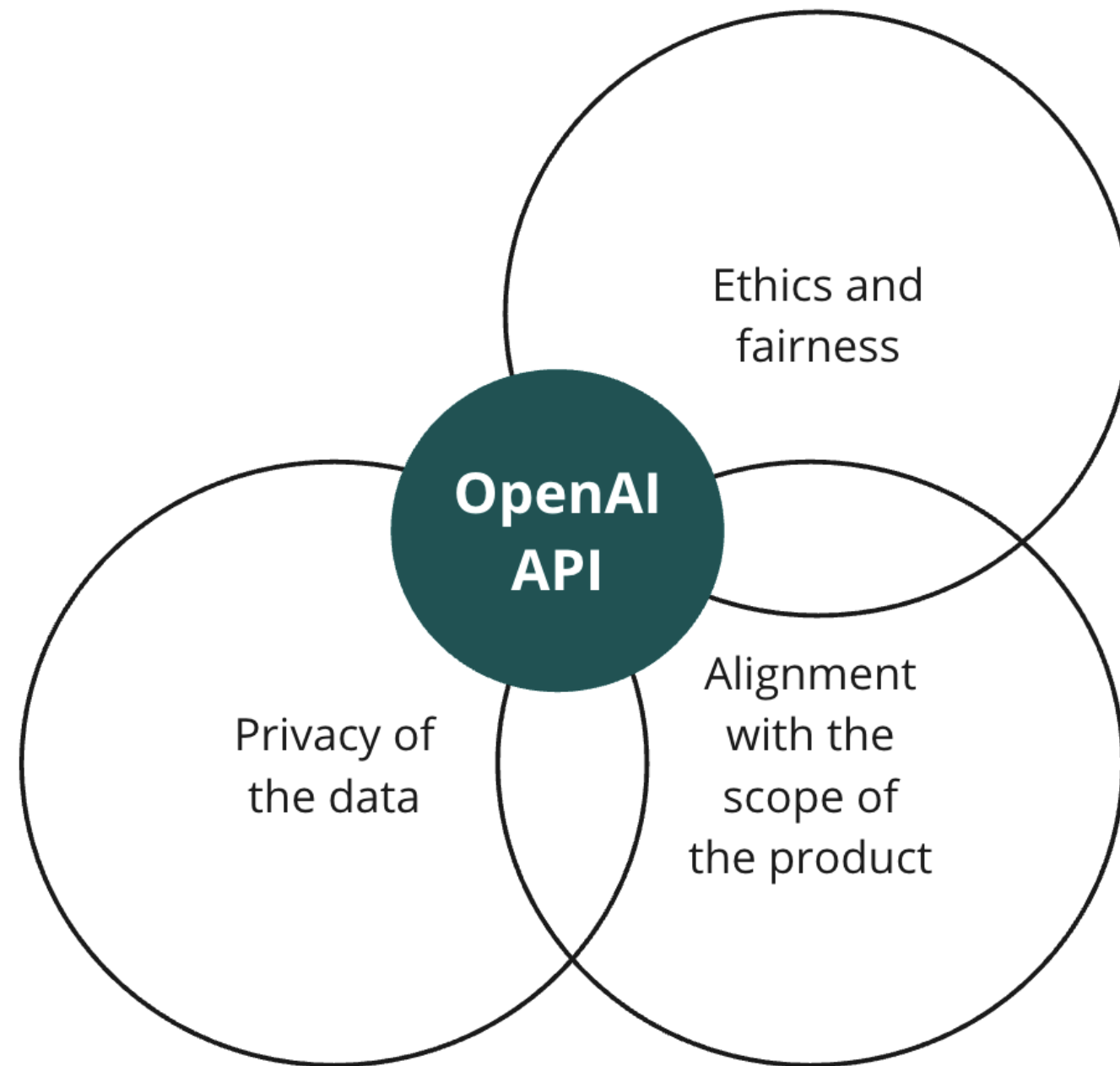
# Safety with the OpenAI API



# Safety with the OpenAI API

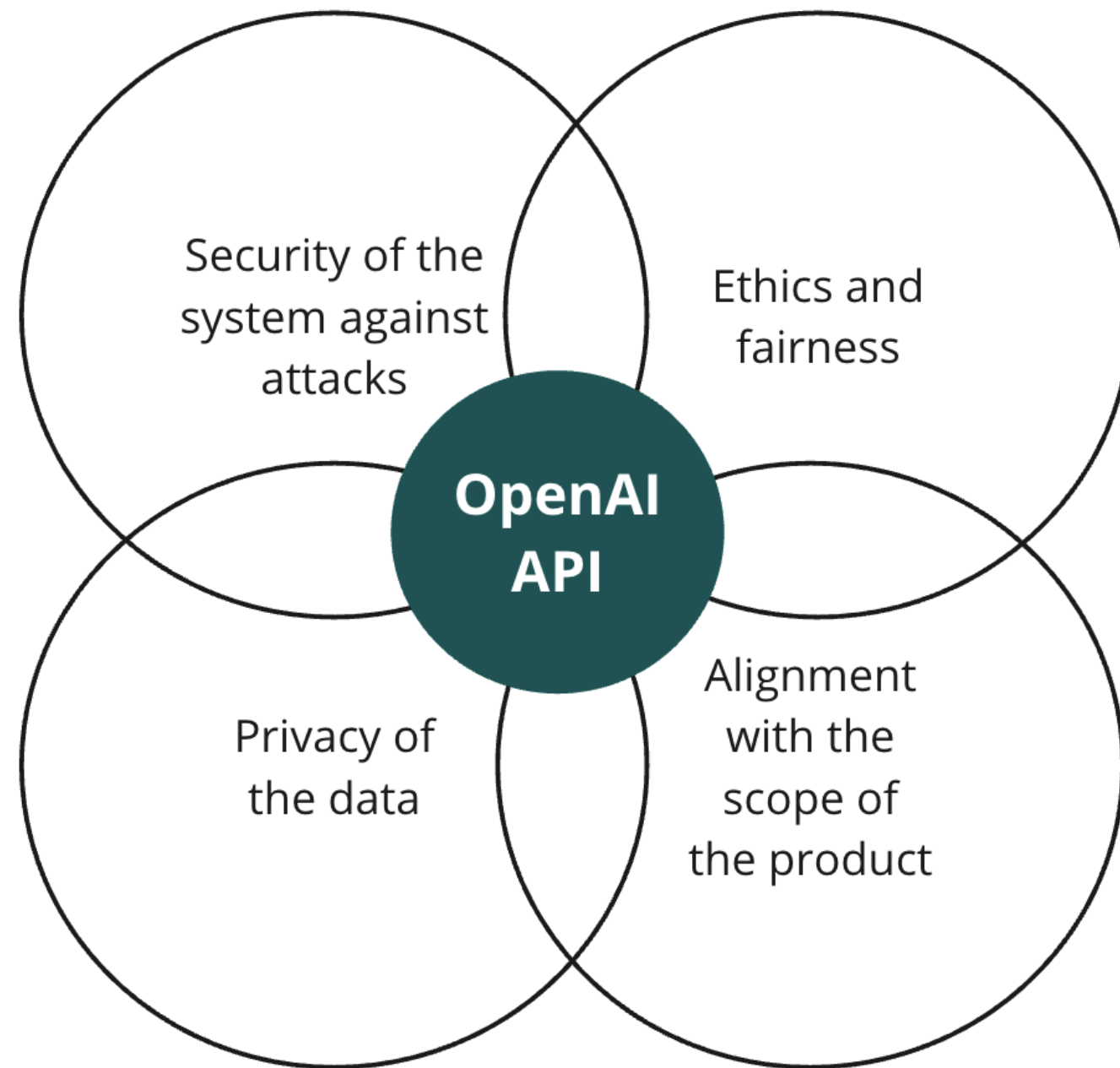


# Safety with the OpenAI API





# Safety with the OpenAI API



# Best practices

- Moderation API
- Adversarial testing
- Limit user input and output tokens
- Prompt engineering

<sup>1</sup> <https://platform.openai.com/docs/guides/safety-best-practices>

# Best practices

- Human in the loop
- "Know your customer"
- Allow users to report issues



# Best practices

- Keep your API keys safe
- Communicate limitations



# Using end-user IDs

```
import uuid
```

```
unique_id = str(uuid.uuid4())
```

```
response = client.chat.completions.create(  
    model="gpt-4o-mini",  
    messages=messages,  
    user=unique_id)
```

```
print(unique_id)
```

```
ad58af43-19b1-4fd8-9905-5ed3bce7e008
```

# Keeping your API key safe

- Unique API key
- Server-side security
- Avoid repository exposure
- Environment variables
- Key management services
- Monitor and rotate keys

# Let's practice!

DEVELOPING AI SYSTEMS WITH THE OPENAI API

# Wrap-up

DEVELOPING AI SYSTEMS WITH THE OPENAI API

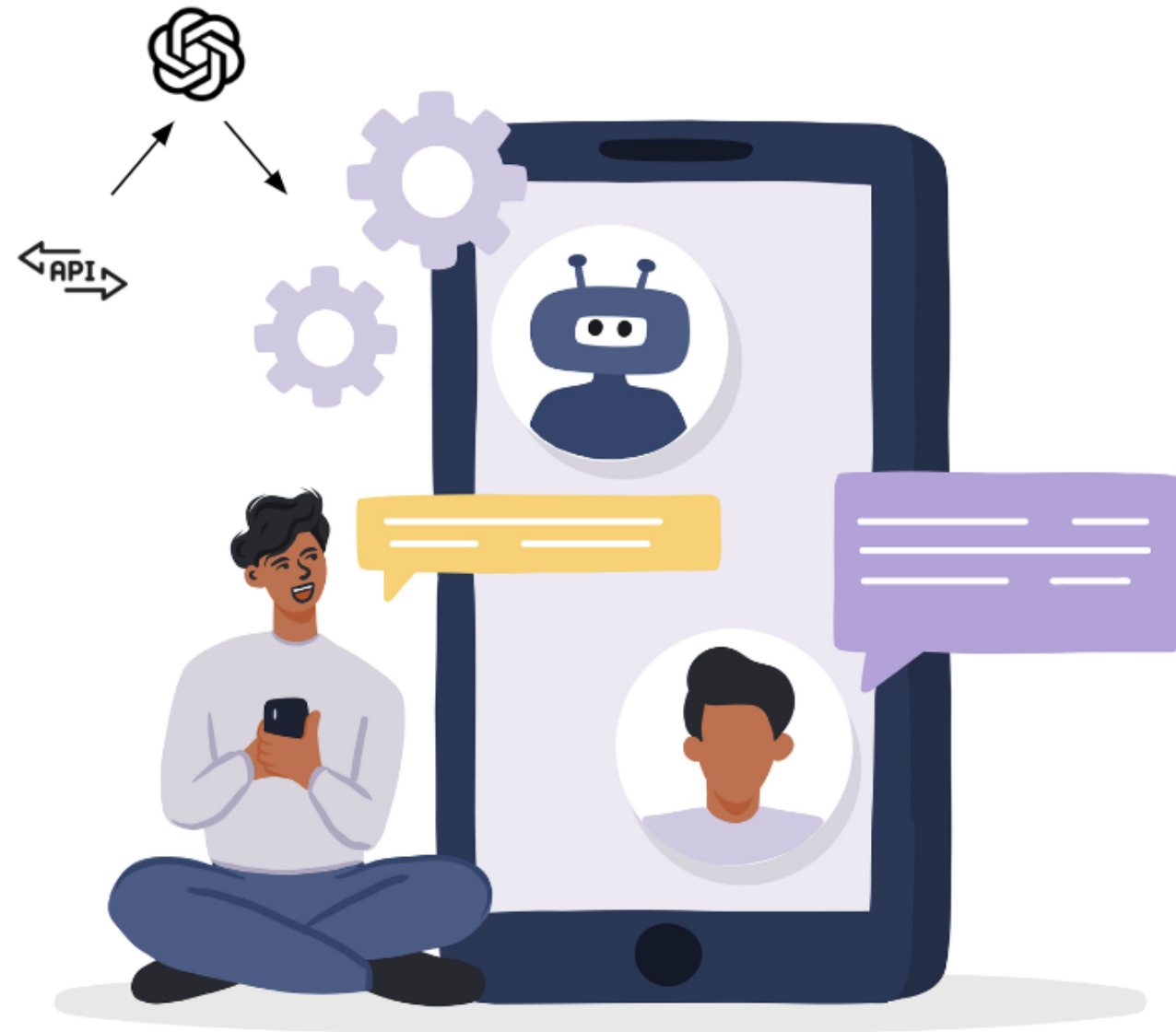


**Francesca Donadoni**

Curriculum Manager, DataCamp



# Wrap-up



# Chapter 1: Structuring End-to-End Applications

Chapter 1:

- production-ready
- error handling
- rate-limit handling



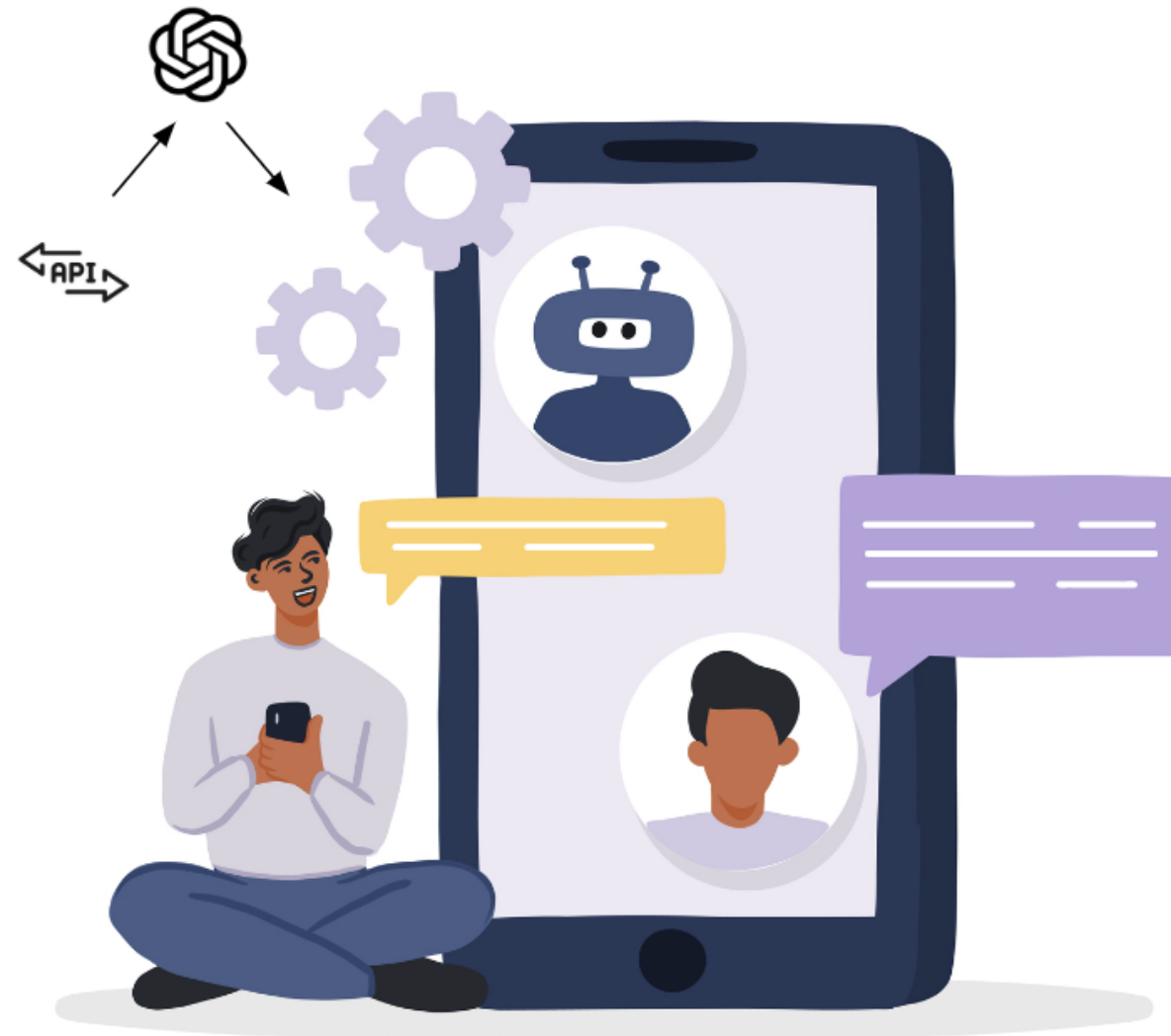
# Chapter 2: Function Calling

## Chapter 1:

- production-ready
- error handling
- rate-limit handling

## Chapter 2:

- define parameters
- extract structured data
- handle multiple functions



# Chapter 3: Best practices for Production Applications

## Chapter 1:

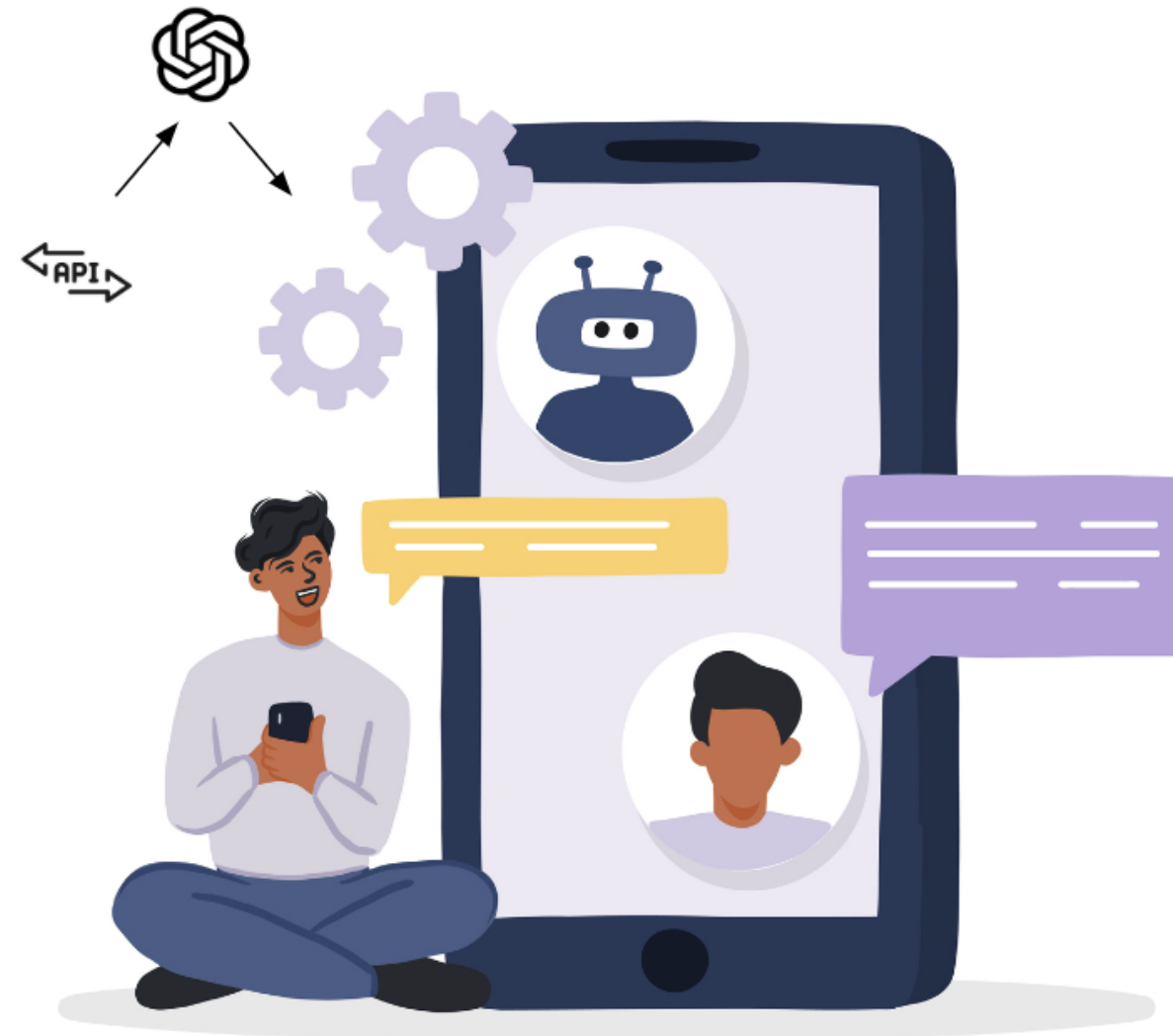
- production-ready
- error handling
- rate-limit handling

## Chapter 2:

- define parameters
- extract structured data
- handle multiple functions

## Chapter 3:

- moderate content
- validate model behavior
- implement safety best practices



# Congratulations!

DEVELOPING AI SYSTEMS WITH THE OPENAI API