

# What is the OpenAI API?

WORKING WITH THE OPENAI API



**James Chapman**

Curriculum Manager, DataCamp

# Coming up...

## Course goals

- Use the *OpenAI API* to access AI models
- Use those models to solve various tasks
- Use **Python** code to do this!

## Expected knowledge

- Subsetting Python lists and **dictionaries**
- Control flow → `if`, `elif`, `else`
- Loops → `for`, `while`

## Not expected to know

- AI or machine learning

# OpenAI, ChatGPT, and the OpenAI API

- OpenAI = AI R&D company
- ChatGPT = AI application
- OpenAI API = Interface for accessing OpenAI models



OpenAI

<sup>1</sup> Image Credit: OpenAI

# OpenAI, ChatGPT, and the OpenAI API

- OpenAI = AI R&D company
- ChatGPT = AI application
- OpenAI API = Interface for accessing OpenAI models



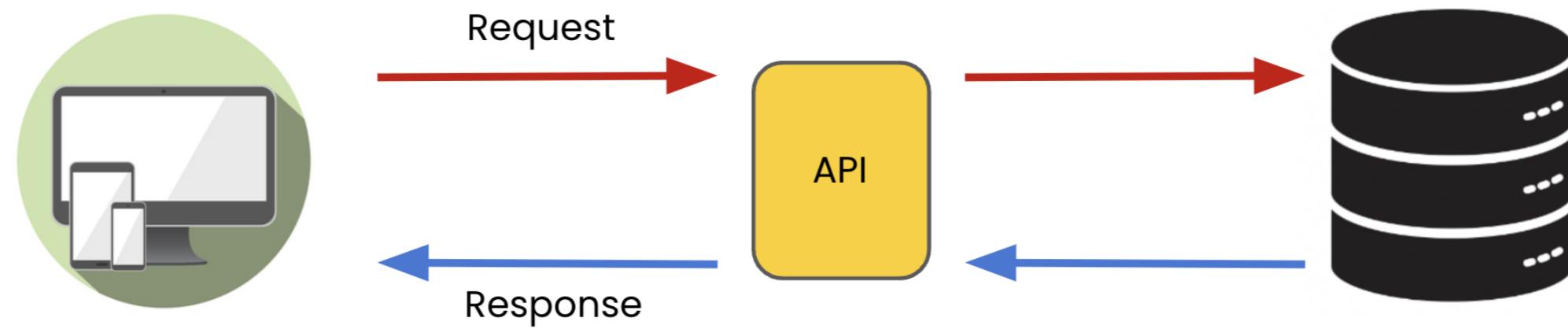
# OpenAI, ChatGPT, and the OpenAI API

- **OpenAI** = AI R&D company
- **ChatGPT** = AI application
- **OpenAI API** = Interface for accessing OpenAI models



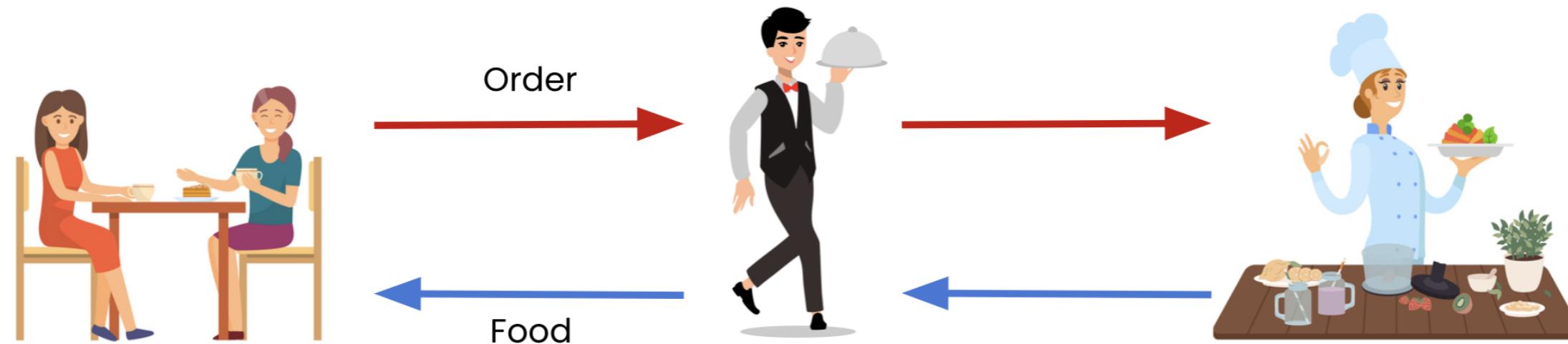
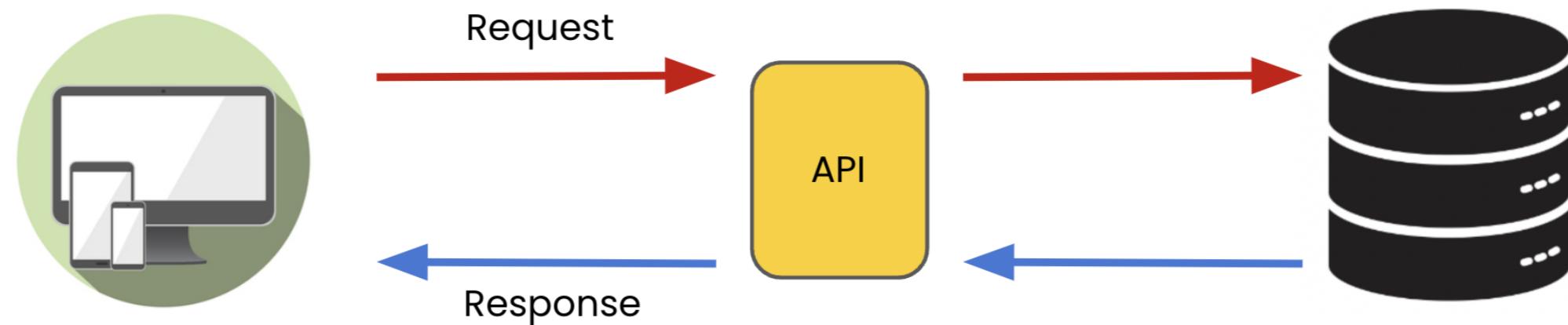
# What is an API?

- Application Programming Interface

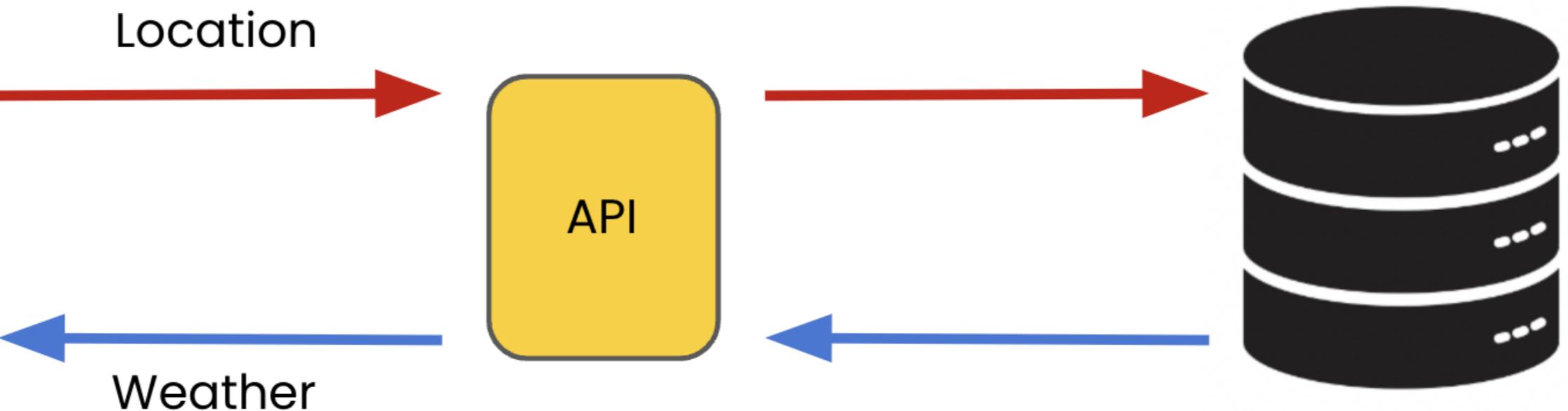


# What is an API?

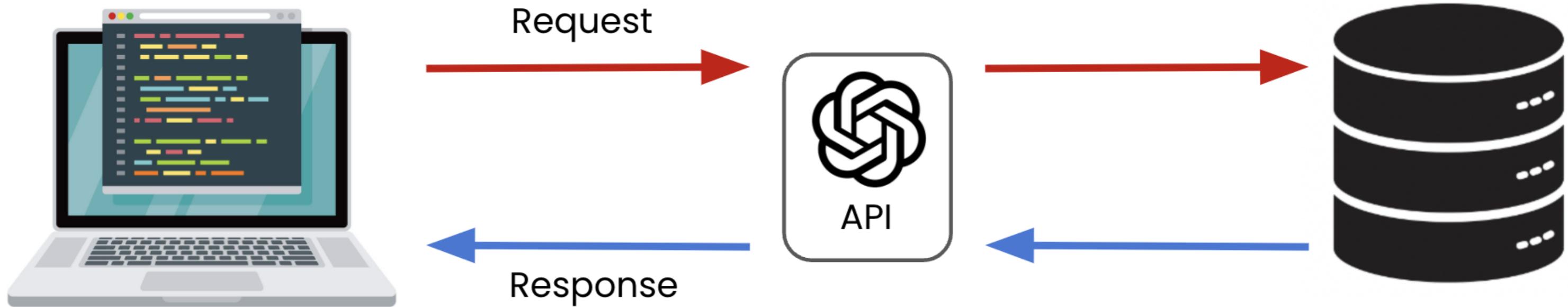
- Application Programming Interface



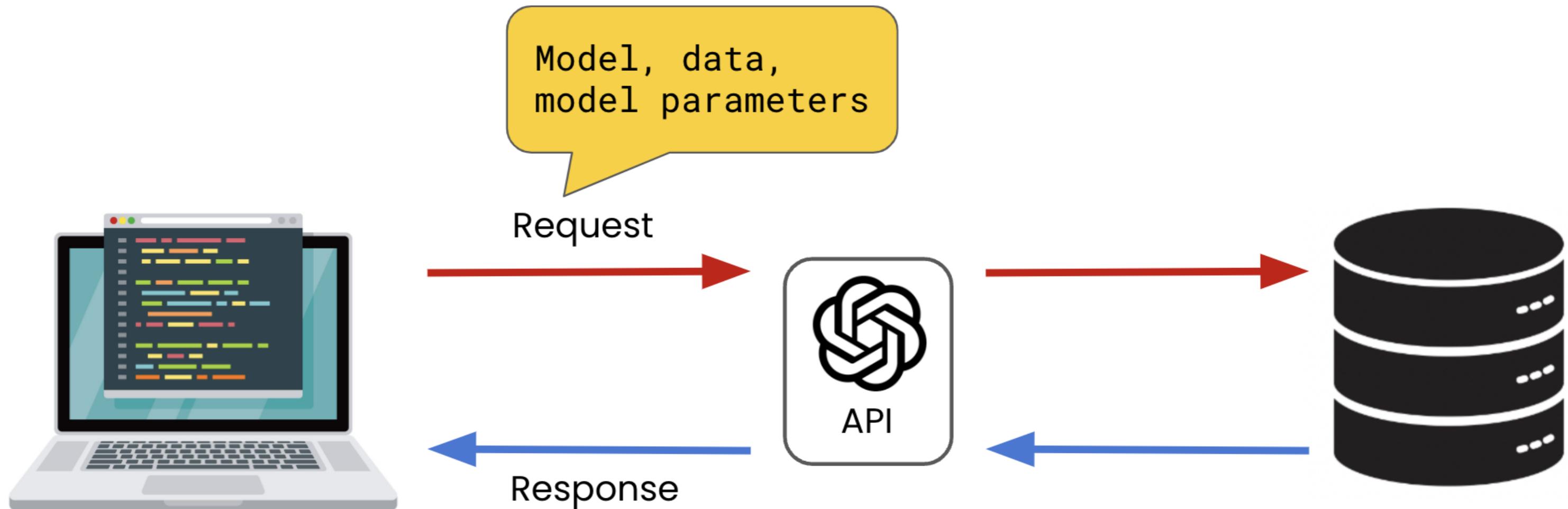
# What is an API?



# The OpenAI API

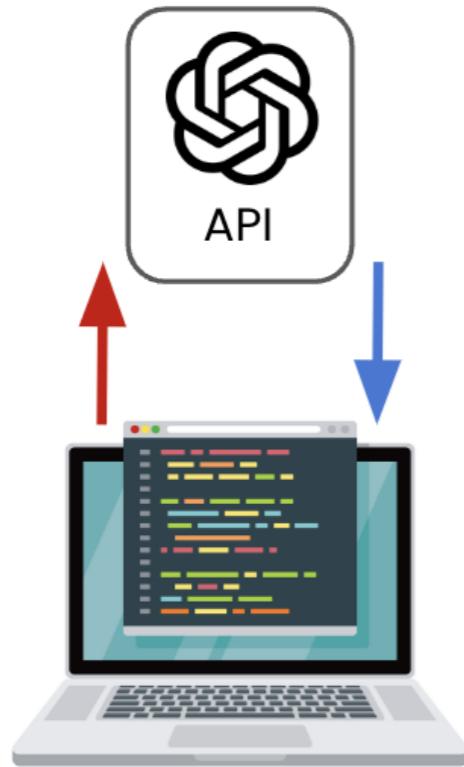


# The OpenAI API



# API vs. web interface

- Provides *flexibility* for integrating AI into products, experiences, and processes
- Interact via programming language
- Very little setup
- Sufficient for *streamlining* individual's workflows



A screenshot of a web-based AI interface. At the top, a user profile picture and the text 'Write Python code for drawing a scatter plot.' are visible. Below this, a message from the AI says: 'You can use the `matplotlib` library in Python to draw a scatter plot. Here's an example code:' followed by a code block. The code is as follows:

```
python

import matplotlib.pyplot as plt

# Example data
x = [1, 2, 3, 4, 5]
y = [2, 4, 1, 3, 5]

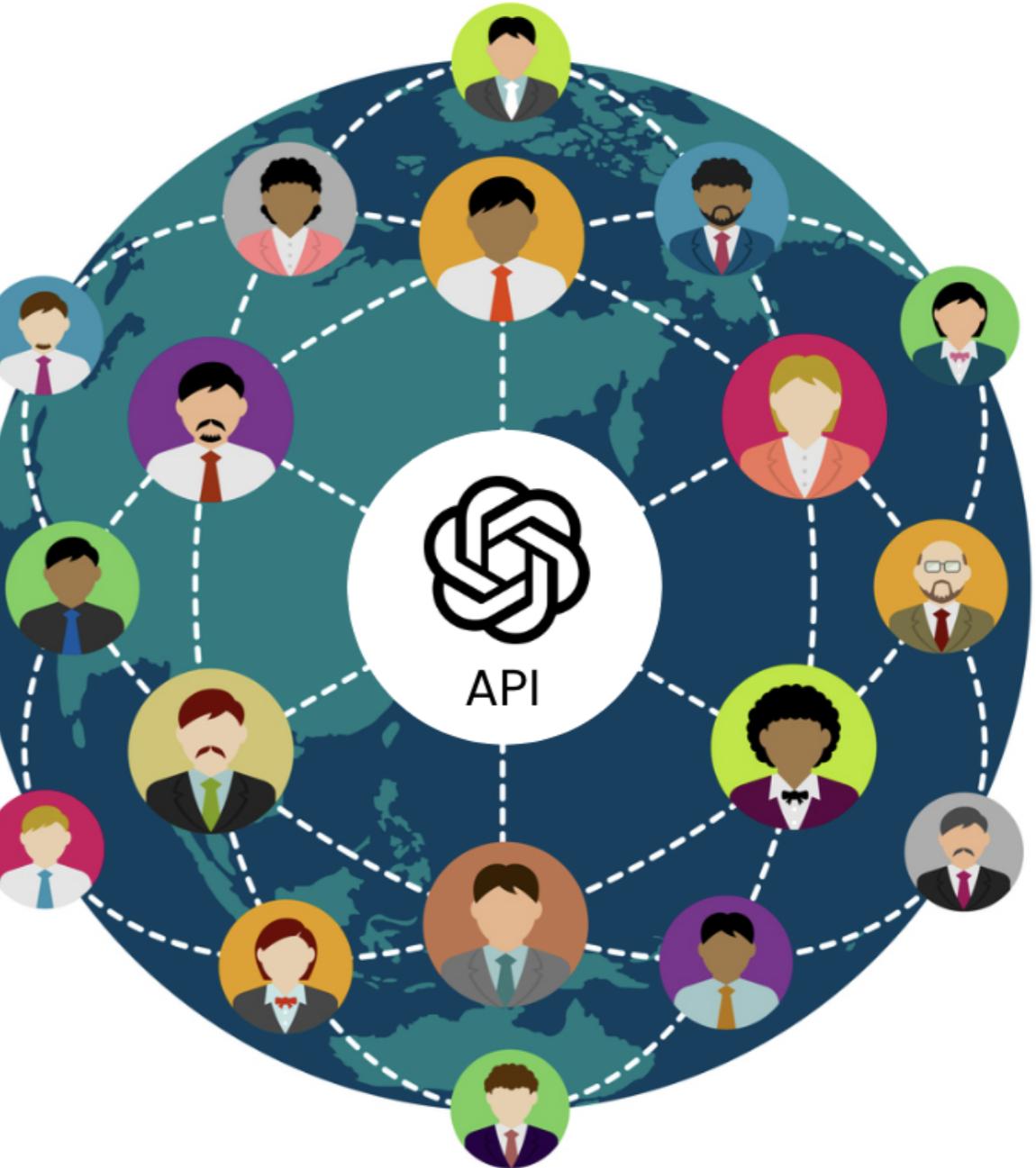
# Draw scatter plot
plt.scatter(x, y)

# Add labels and title
plt.xlabel('X-axis')
plt.ylabel('Y-axis')
plt.title('Scatter Plot Example')

# Show plot
plt.show()
```

# Why the OpenAI API?

- Accessibility to AI
- Don't require data or ML experience



# Building AI applications

DataCamp Workspace = Cloud-based IDE

The screenshot shows a DataCamp Workspace interface. On the left, a code editor window titled 'mcH' contains Python code for data analysis and visualization:

```
1 # import necessary libraries
2 import pandas as pd
3 import plotly.express as px
4
5 # read the csv file
6 df = pd.read_csv('joey_clean.csv')
7
8 # convert start_date_time column to datetime format
9 df['start_date_time'] = pd.to_datetime(df['start_date_time'])
10
11 # create a new dataframe with date and total volume columns
12 date_volume_df = df.groupby('date')[['Pump Total Volume']].sum().reset_index()
13
14 # create a bar chart of total volume over time
15 fig = px.bar(date_volume_df, x='date', y='[Pump] Total Volume', title='Total Volume Over Time')
16 fig.show()
```

To the right of the code editor is a chart viewer. The chart is titled 'Total Volume Over Time'. The y-axis is labeled '[Pump] Total Volume' and ranges from 0 to 30. The x-axis shows dates from March 5, 2023, to April 2, 2023. The chart displays daily total volume bars, showing a general upward trend with some fluctuations.

A context menu is open on the right side of the workspace, with 'Generate' selected. Other options in the menu include Run, Comment, Move up, Move down, Hide code, Hide output, and Delete.

# **Let's practice!**

**WORKING WITH THE OPENAI API**

# Making requests to the OpenAI API

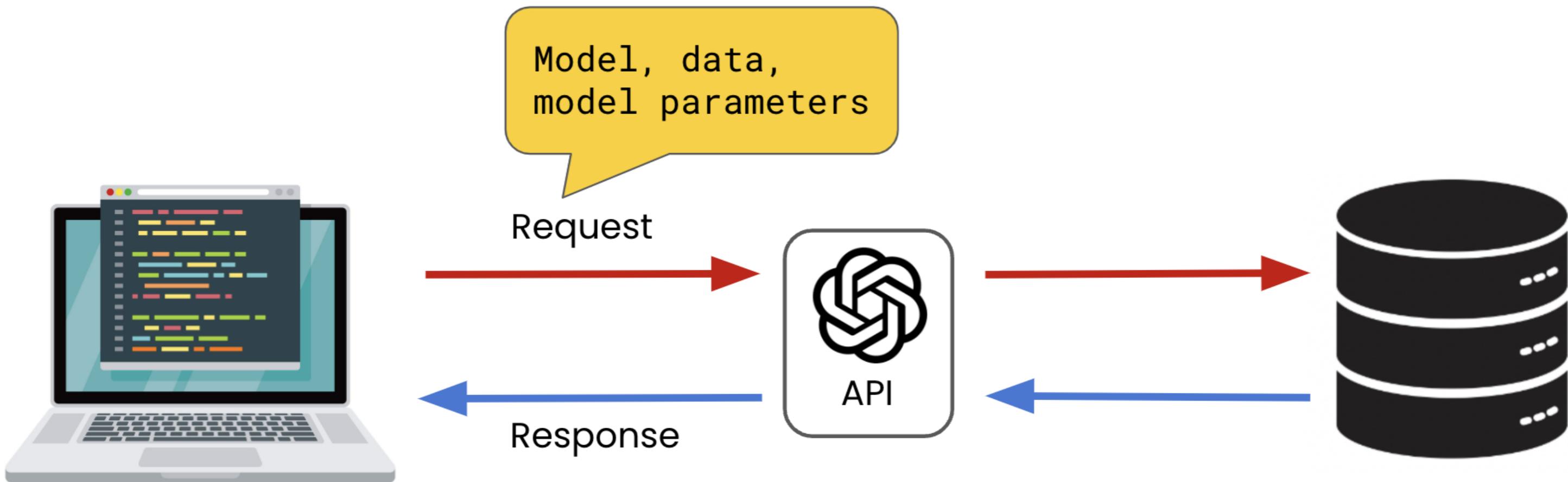
WORKING WITH THE OPENAI API



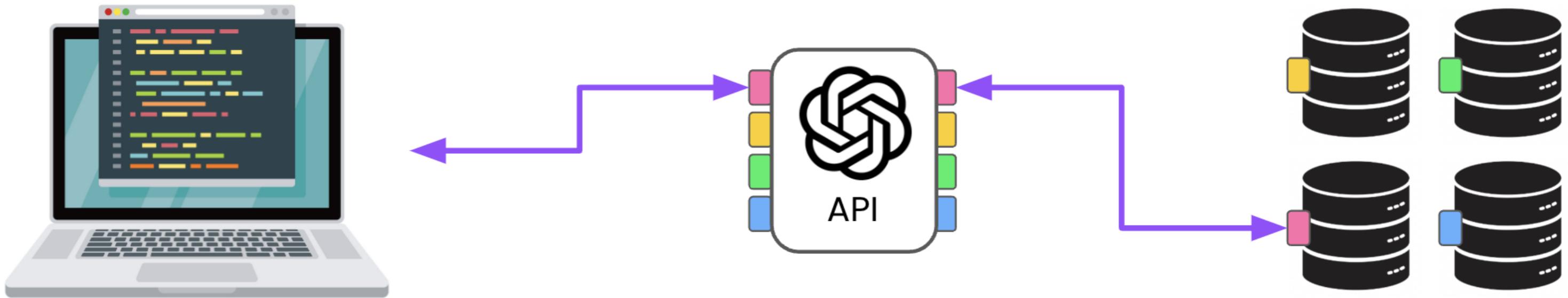
James Chapman

Curriculum Manager, DataCamp

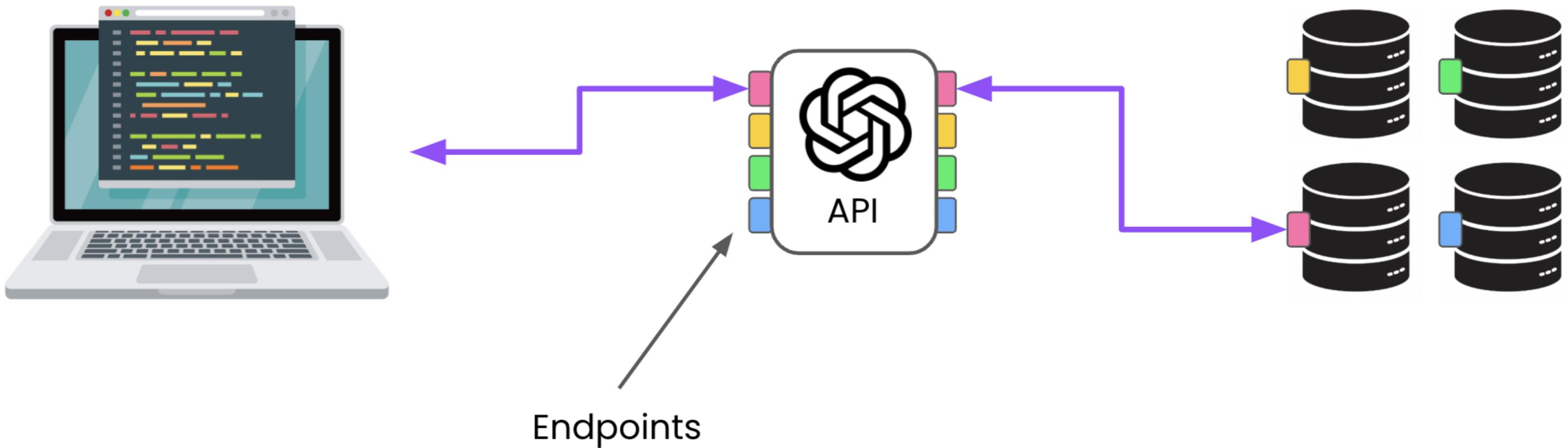
# APIs recap...



# API endpoints



# API endpoints



# API endpoints

- **Endpoints** → API access point designed for specific interactions



# API authentication

- Authentication → Controls on access to API endpoint services (often unique key)



# API usage costs

- For OpenAI API:
  - Larger inputs + outputs = Greater cost



<sup>1</sup> <https://openai.com/pricing>

# API documentation

The screenshot shows the OpenAI API documentation homepage. At the top, there's a navigation bar with a logo, 'Overview', 'Documentation', 'API reference' (which is highlighted in green), and 'Examples'. Below the navigation is a search bar with a magnifying glass icon, the word 'Search', and keyboard shortcut keys ('⌘ K'). On the left side, there's a sidebar with two main sections: 'GETTING STARTED' and 'ENDPOINTS'. Under 'GETTING STARTED', the sub-sections are 'Introduction', 'Authentication', 'Making requests', and 'Streaming'. Under 'ENDPOINTS', the sub-sections are 'Audio', 'Chat', 'Embeddings', 'Fine-tuning', 'Files', 'Images', 'Models', and 'Moderations'. The main content area starts with an 'Introduction' section, which says you can interact with the API through HTTP requests from any language, via official Python and Node.js bindings, or a community-maintained library. It provides a command to install the official Python bindings: `pip install openai`. Below that, it provides a command to install the official Node.js library: `npm install openai@^4.0.0`. Finally, there's an 'Authentication' section explaining that the API uses API keys for authentication and linking to the 'API Keys' page.

Overview Documentation API reference Examples

Search ⌘ K

**GETTING STARTED**

- Introduction
- Authentication
- Making requests
- Streaming

**ENDPOINTS**

- Audio
- Chat
- Embeddings
- Fine-tuning
- Files
- Images
- Models
- Moderations

## Introduction

You can interact with the API through HTTP requests from any language, via our official Python bindings, our official Node.js library, or a [community-maintained library](#).

To install the official Python bindings, run the following command:

```
pip install openai
```

To install the official Node.js library, run the following command in your Node.js project directory:

```
npm install openai@^4.0.0
```

## Authentication

The OpenAI API uses API keys for authentication. Visit your [API Keys](#) page to retrieve the API key you'll use in your requests.

<sup>1</sup> <https://platform.openai.com/docs/api-reference>

# Creating an OpenAI API key

## Create an OpenAI account

1) Go to the API signup page →

<https://platform.openai.com/signup>

2) Create your account (you'll need to provide  
your *email address* and your *phone number*)

OpenAI often provides free trial credit

## Create your account

Please note that phone verification is required for  
signup. Your number will only be used to verify  
your identity for security purposes.

Email address

Continue

Already have an account? [Log in](#)

OR



Continue with Microsoft Account



Continue with Google

# Creating an OpenAI API key

- 3) Go to the API keys page → <https://platform.openai.com/account/api-keys>
- 4) Create a new secret key and **copy** it (if you lose it, delete the key and create a new one)

NAME	KEY	CREATED	LAST USED ⓘ	
Secret key	sk-...Vpp5	5 Feb 2023	25 Apr 2023	 
<a href="#">+ Create new secret key</a>				

**Note:** DataCamp doesn't store API keys used in the exercises

# Making a request

```
from openai import OpenAI

client = OpenAI(api_key="ENTER YOUR KEY HERE")

response = client.completions.create(
    model="gpt-3.5-turbo-instruct",
    prompt="What is the OpenAI API?"
)

print(response)
```

# The response

```
Completion(id='cmpl-8SPzX0HgYV1emmoGN6MYjjaRJzkBR', choices=[CompletionChoice(finish_reason='length', index=0, logprobs=None, text='\n\nThe OpenAI API is an application programming interface provided by OpenAI, a')], created=1701783787, model='gpt-3.5-turbo-instruct', object='text_completion', system_fingerprint=None, usage=CompletionUsage(completion_tokens=16, prompt_tokens=7, total_tokens=23))
```

## Additional spacing:

```
Completion(id='cmpl-8S0D6VZBpM8Vy0fZf6atE71b1Zdvm', choices=[CompletionChoice(finish_reason='length', index=0, logprobs=None, text='\n\nThe OpenAI API is an application programming interface provided by OpenAI, a')], created=1701684684, model='gpt-3.5-turbo-instruct', object='text_completion', system_fingerprint=None, usage=CompletionUsage(completion_tokens=16, prompt_tokens=7, total_tokens=23))
```

# Interpreting the response

```
print(response.choices)
```

```
[CompletionChoice(finish_reason='length', index=0, logprobs=None,  
text='\n\nThe OpenAI API is an application programming interface provided by OpenAI, a')]
```

# Interpreting the response

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

```
CompletionChoice(finish_reason='length', index=0, logprobs=None,  
text='\n\nThe OpenAI API is an application programming interface provided by OpenAI, a')
```

# Interpreting the response

```
print(response.choices[0].text)
```

The OpenAI API is an application programming interface provided by OpenAI, a

# Converting the response into a dictionary

```
print(response.model_dump())
```

```
{'id': 'cmpl-8S0KeXQsQVl9i9CagkHZjb2Ujhy2A',
 'choices': [{'finish_reason': 'length', 'index': 0, 'logprobs': None,
              'text': '\n\nThe OpenAI API is an application programming interface provided by OpenAI, a'}],
 'created': 1701685152,
 'model': 'gpt-3.5-turbo-instruct',
 'object': 'text_completion',
 'system_fingerprint': None,
 'usage': {'completion_tokens': 16, 'prompt_tokens': 7, 'total_tokens': 23}}
}
```

# **Let's practice!**

**WORKING WITH THE OPENAI API**

# The OpenAI developer ecosystem

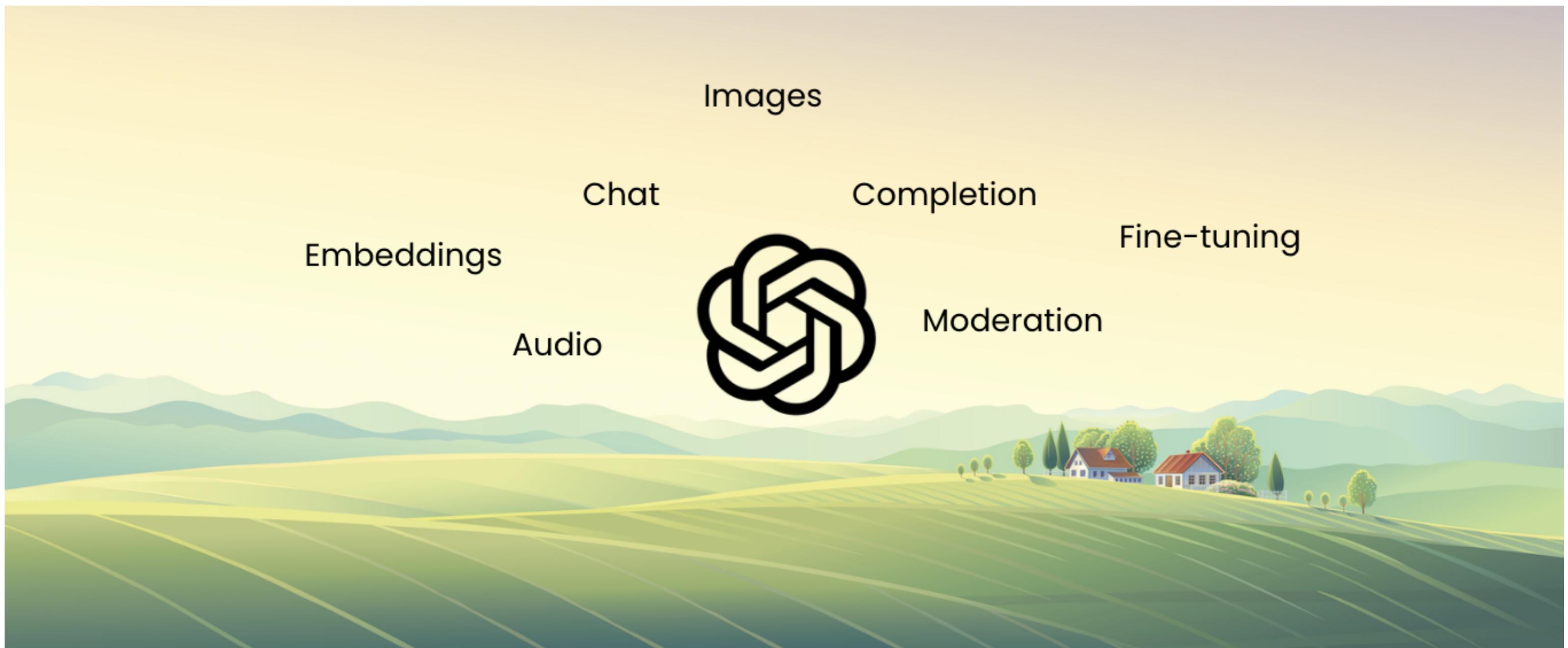
WORKING WITH THE OPENAI API

**James Chapman**

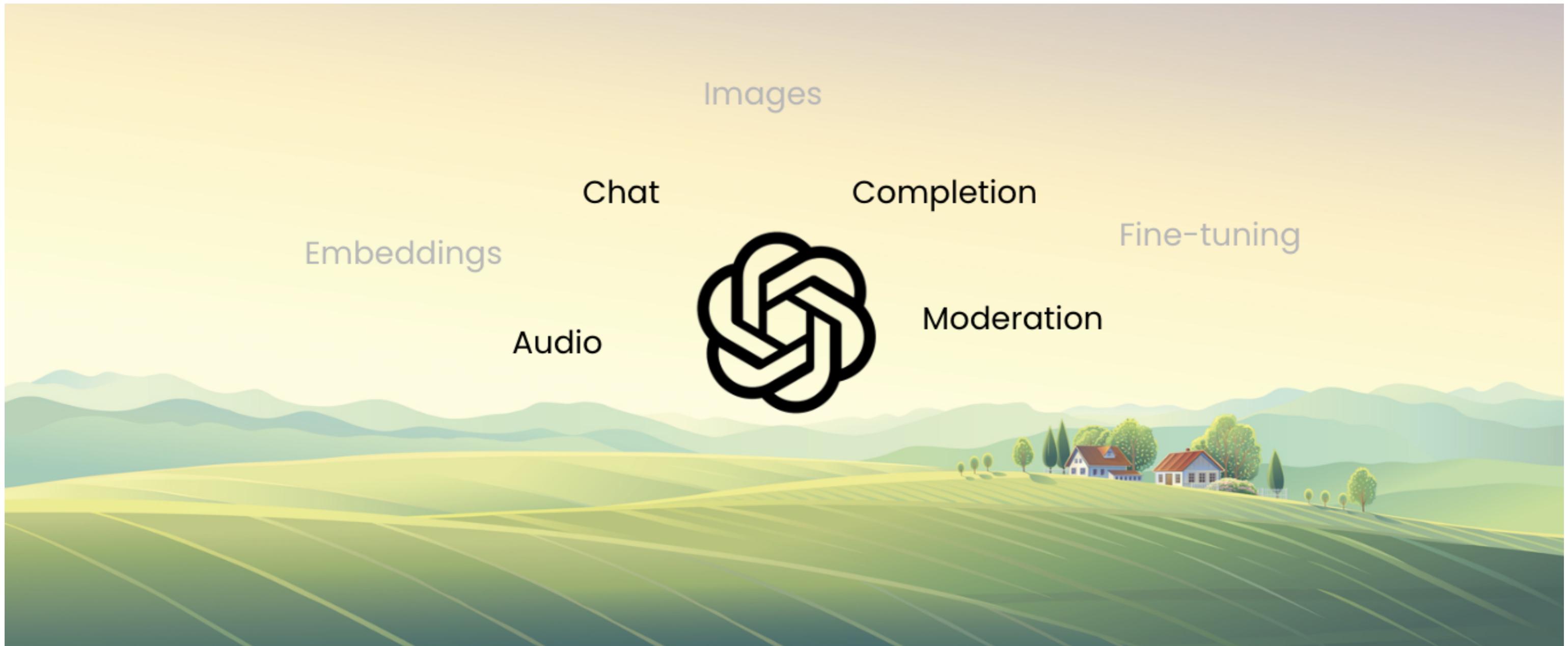
Curriculum Manager, DataCamp



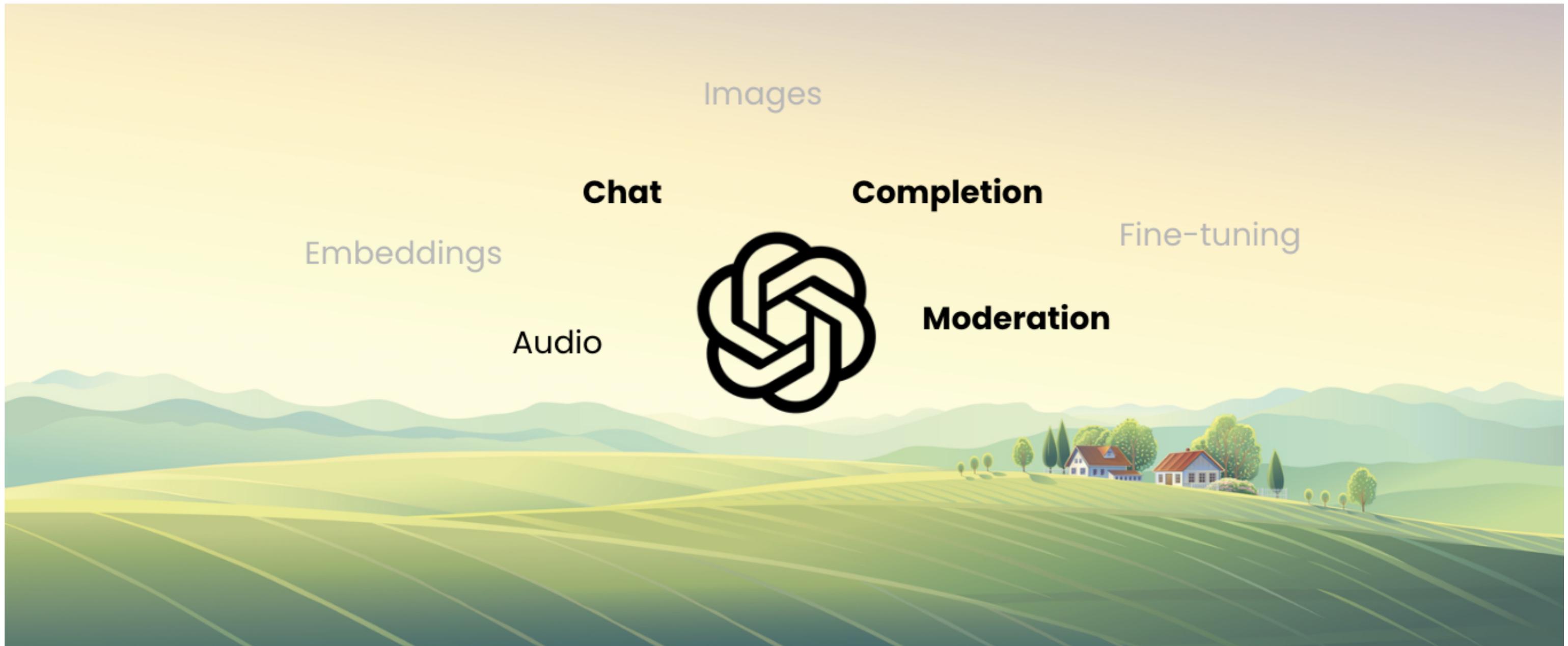
# The OpenAI landscape



# The OpenAI landscape

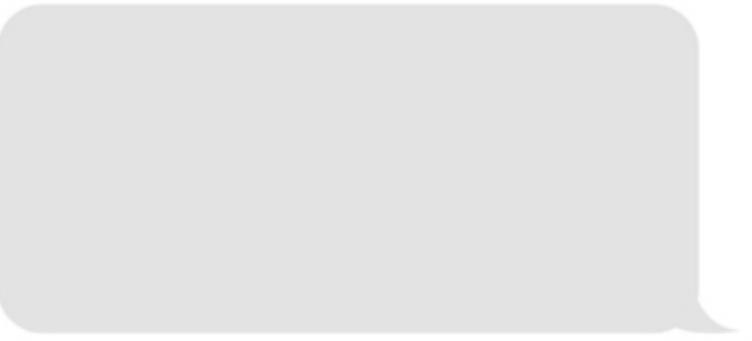


# The OpenAI landscape



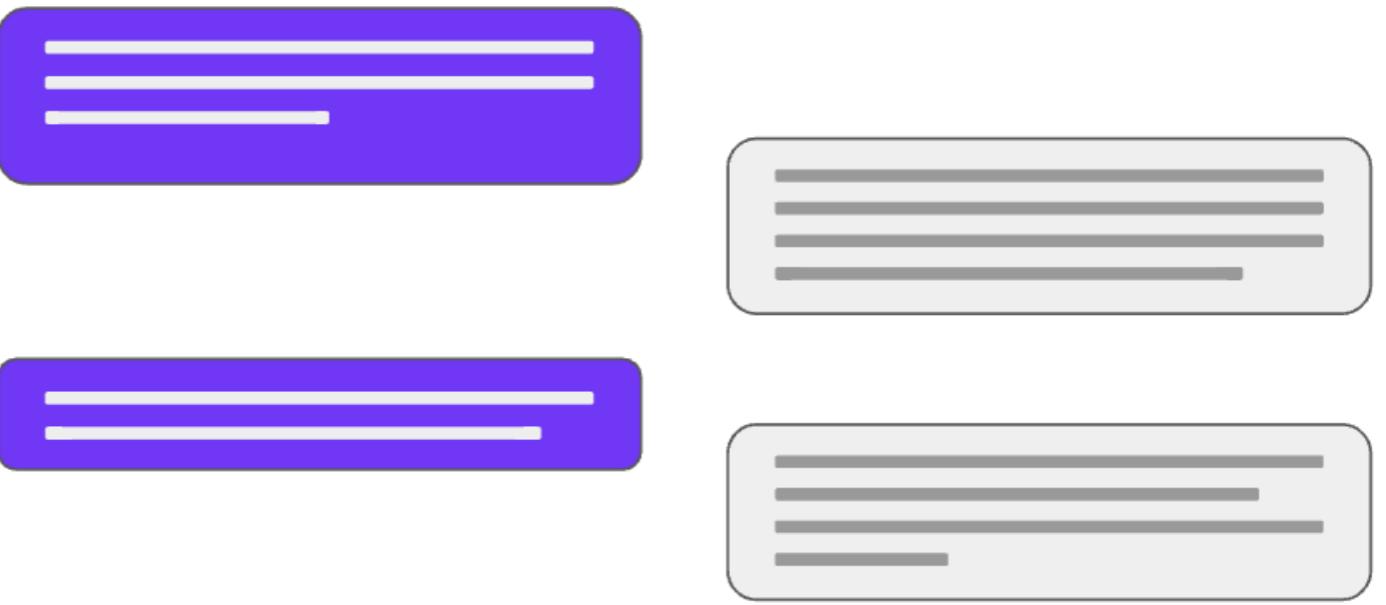
# Completions

- Receive *continuation* of a prompt
- *Single-turn tasks*
  - Answer questions
  - Classification into categories
  - Sentiment analysis
  - Explain complex topics



# Chat

- *Multi-turn conversations*
  - Ideation
  - Customer support assistant
  - Personal tutor
  - Translate languages
  - Write code
- Also performs well on *single-turn* tasks
- Chat completions → **Chapter 2**



# Moderation

- Check content for *violations* of OpenAI's *usage policies*, including:
  - Inciting violence
  - Hate speech
- Can *customize model sensitivity* to specific violations



<sup>1</sup> <https://openai.com/policies/usage-policies> <sup>2</sup> <https://platform.openai.com/docs/guides/moderation/overview>

# Organization settings

## Organization name

Human-friendly label for your organization, shown in user interfaces

DataCamp

- Better management of *access, billing, and usage limits*
- Users can be part of multiple organizations

## Organization ID

Identifier for this organization sometimes used in API requests

Save

<sup>1</sup> <https://platform.openai.com/account/org-settings>

# Organizations

```
from openai import OpenAI

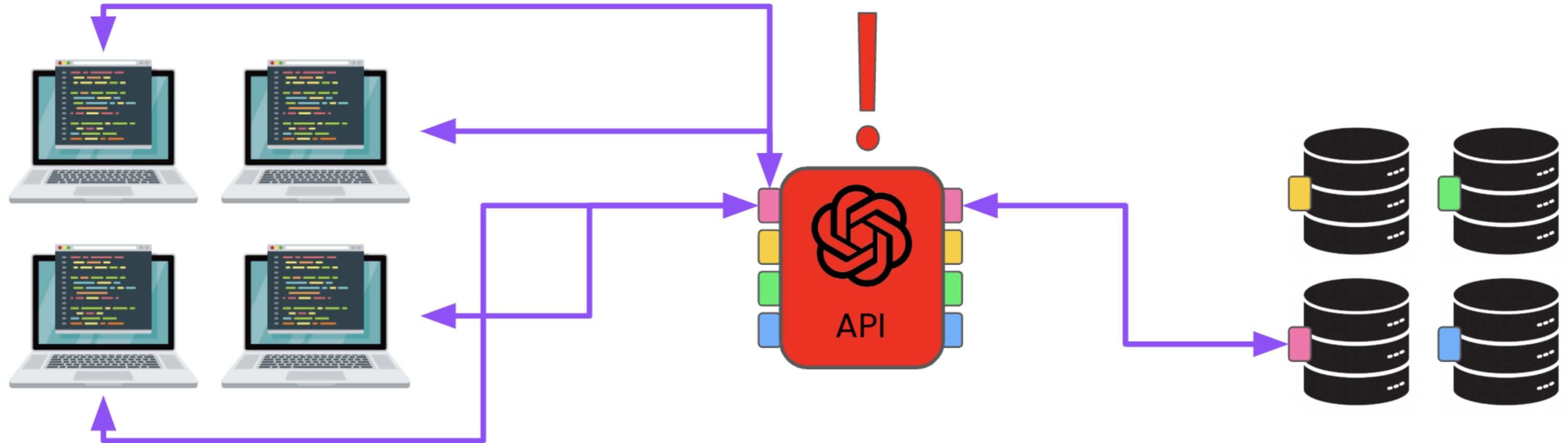
client = OpenAI(api_key="ENTER YOUR KEY HERE",
                organization = "ENTER ORG ID")

response = client.completions.create(
    model="gpt-3.5-turbo-instruct",
    prompt="What is the OpenAI API?"
)

print(response)
```

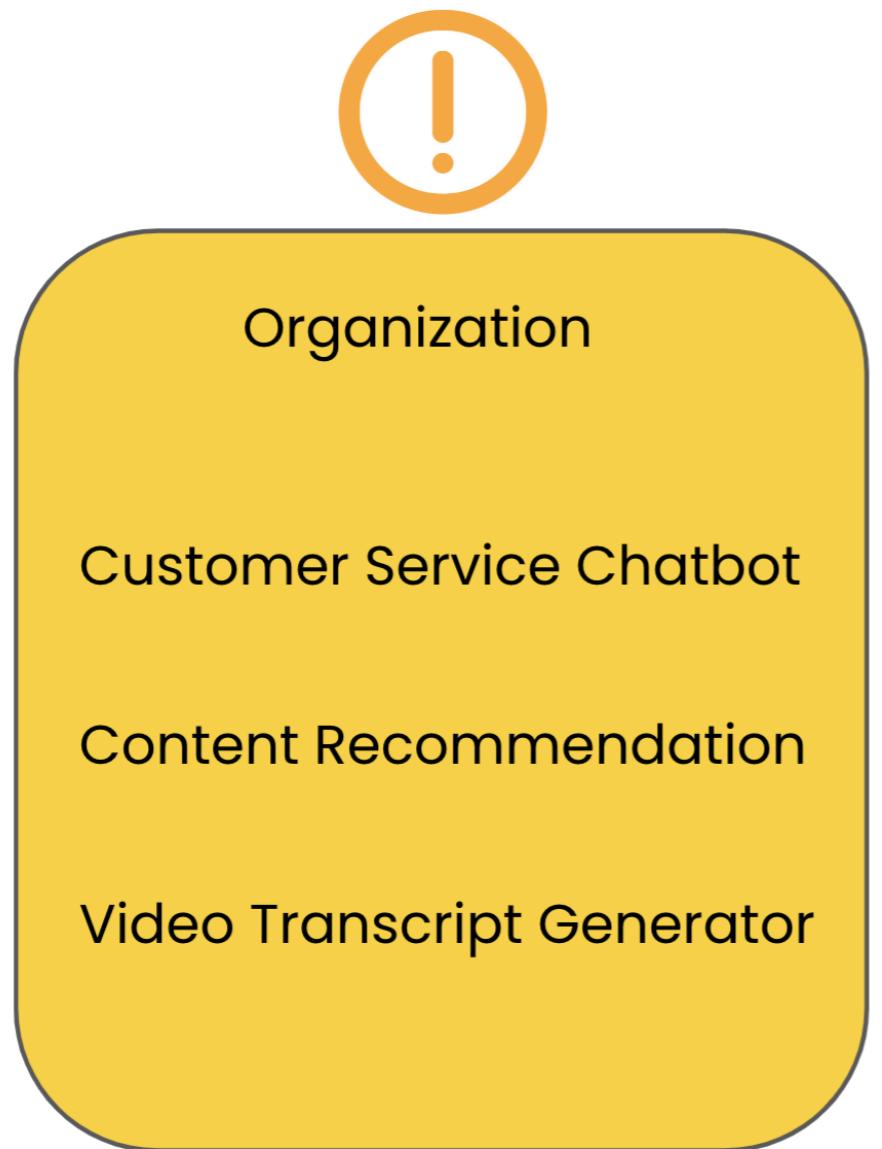
# Rate limits

- Cap on *frequency* and *size* of API requests

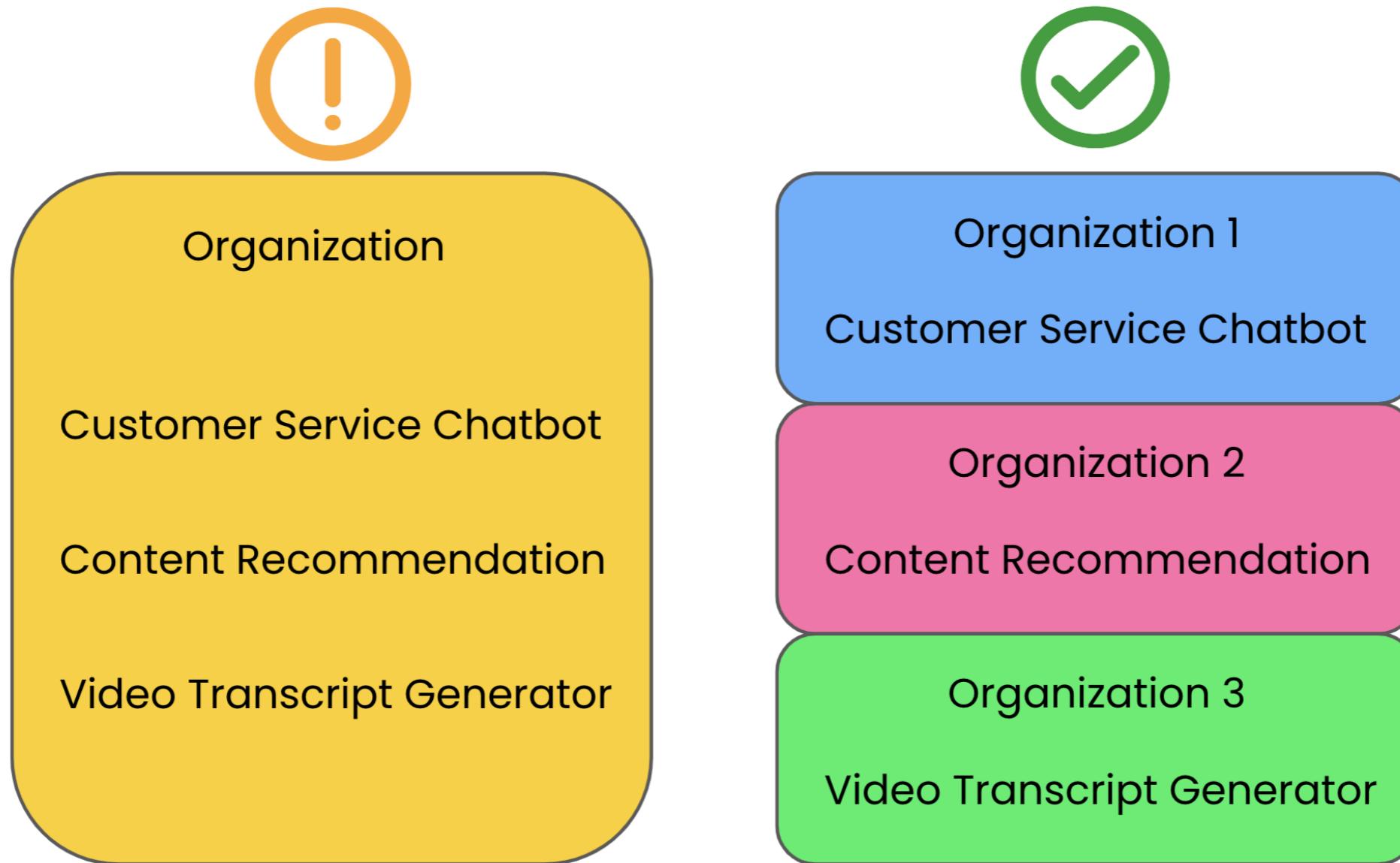


<sup>1</sup> <https://platform.openai.com/docs/guides/rate-limits>

# Organization structure

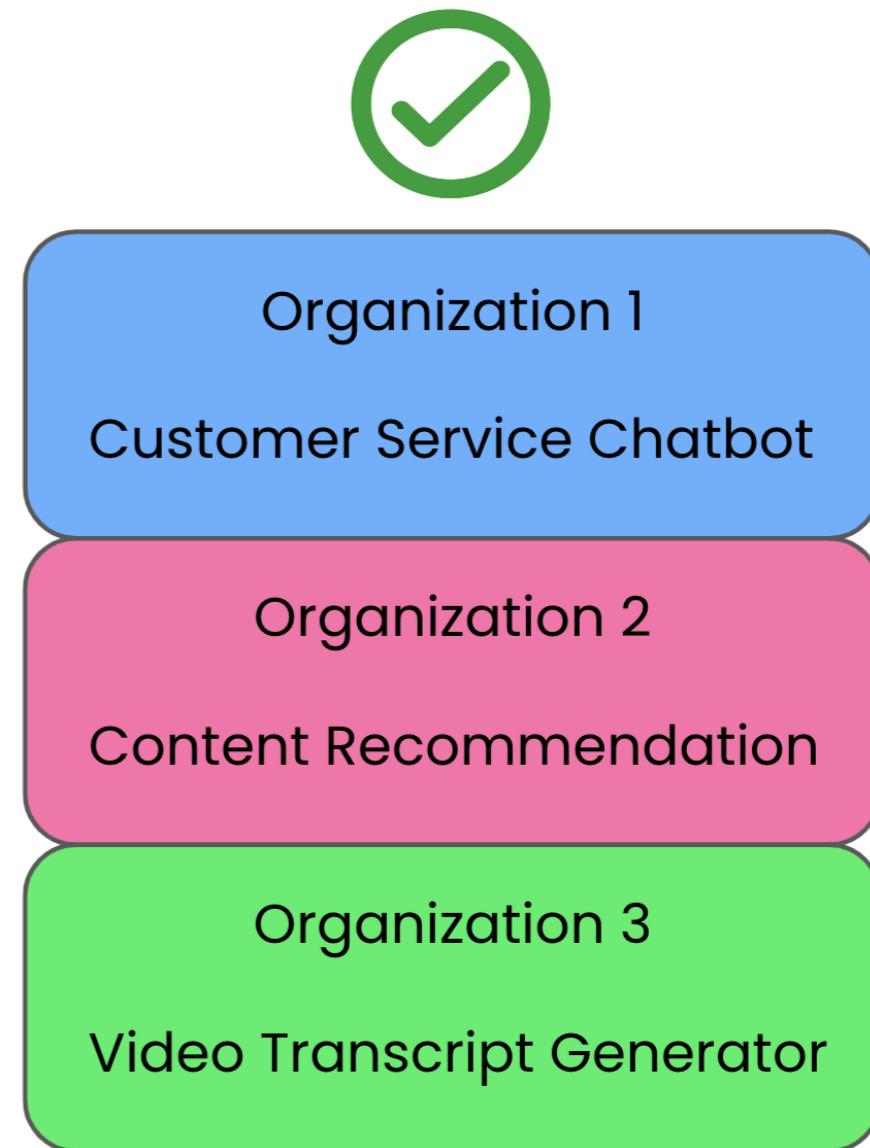


# Organization structure



# Organization structure

- Distributes requests across organizations
  - Reduced likelihood of hitting rate limits
- Removes *single failure point*
  - Issue with one organization doesn't break all features
- Better usage and billing management



# **Let's practice!**

**WORKING WITH THE OPENAI API**