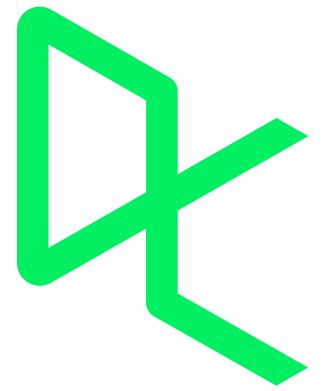


# WHAT IS THE openAI API?





# James Chapman

*Curriculum Manager,  
DataCamp*



# Course goals

- 1 Use AI models through the **OpenAI API**
- 2 Solve real-world tasks
- 3 Use **Python** code

# Expected knowledge

- Subsetting lists and dictionaries



- Control flow

if

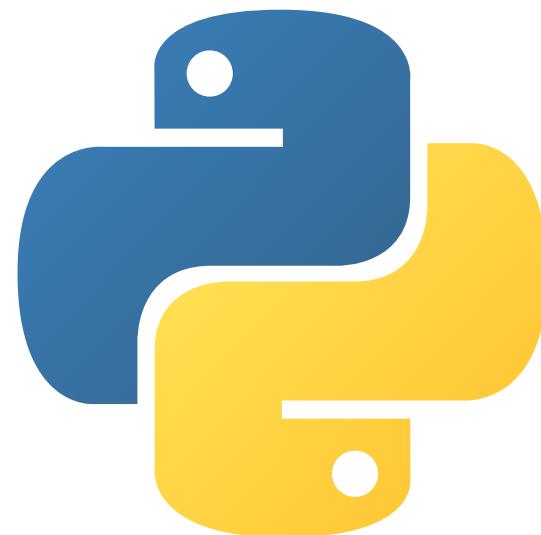
elif

else

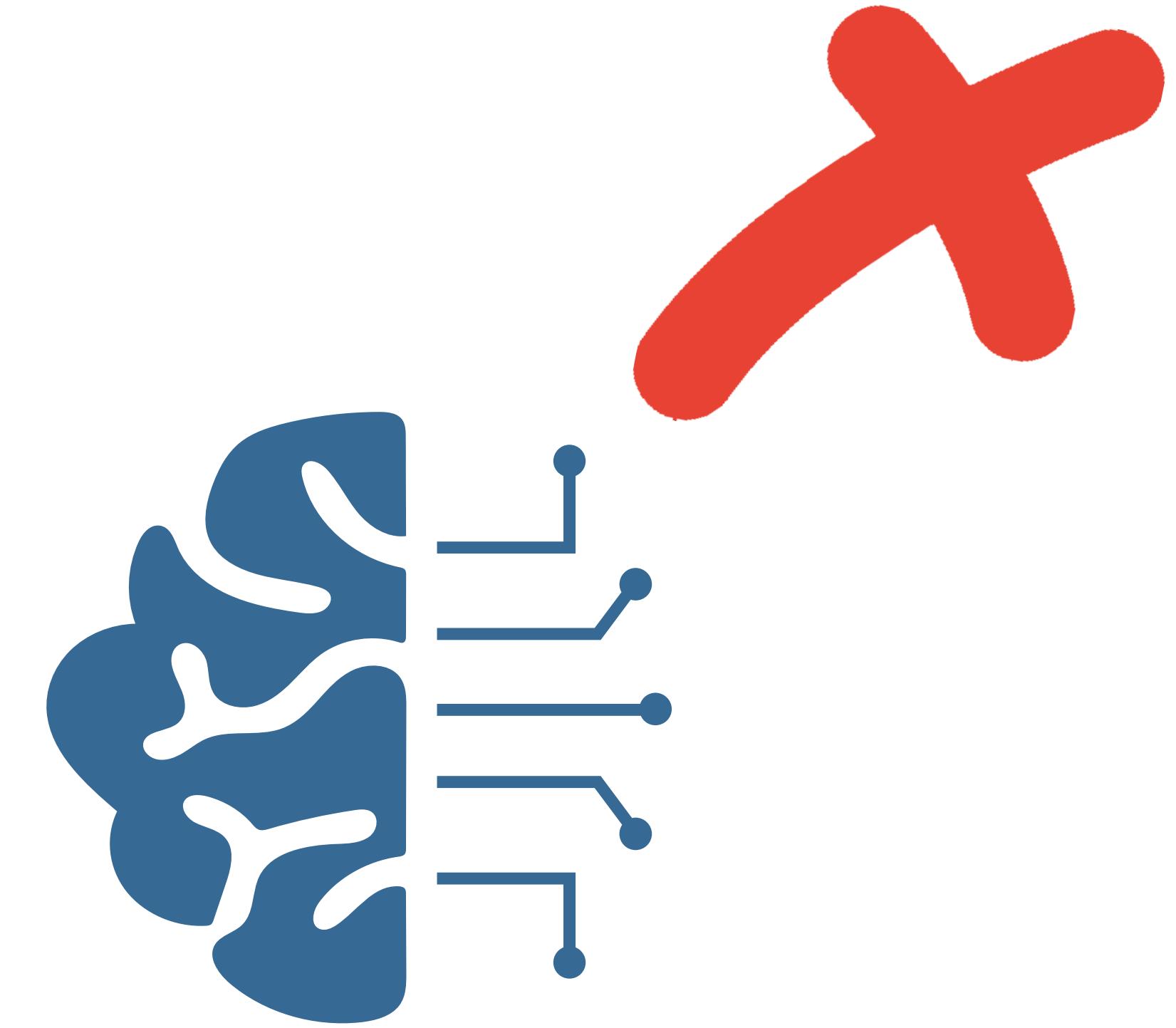
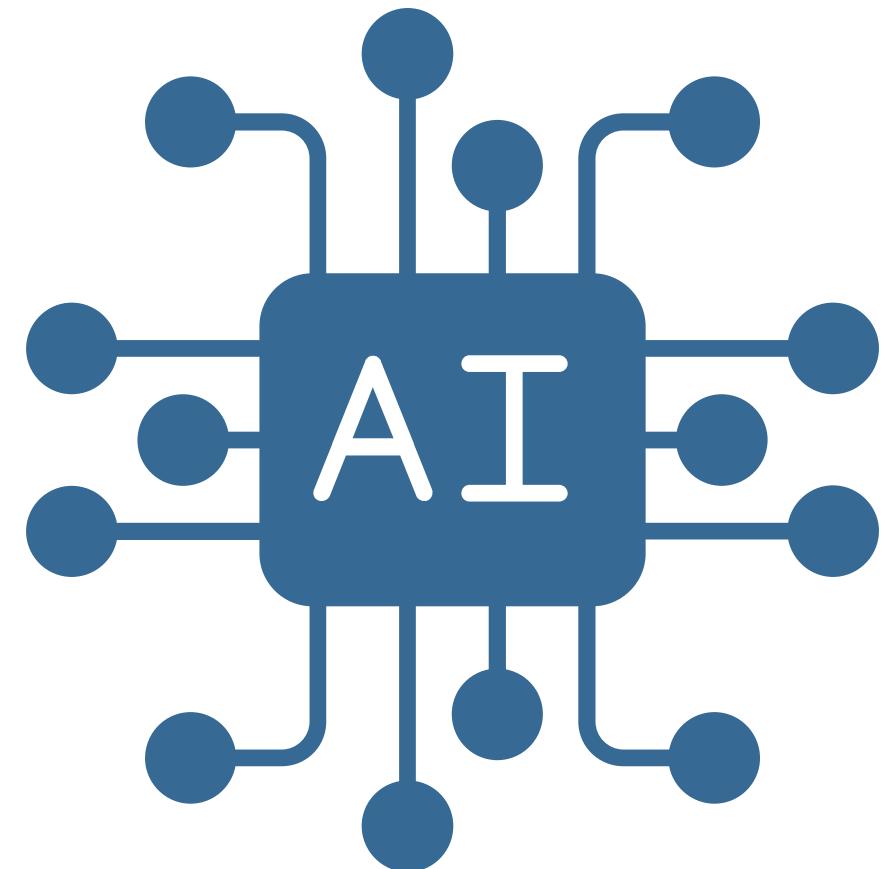
- Loops

for

while



# Not expected



# Machine Learning

# Let's dive in!

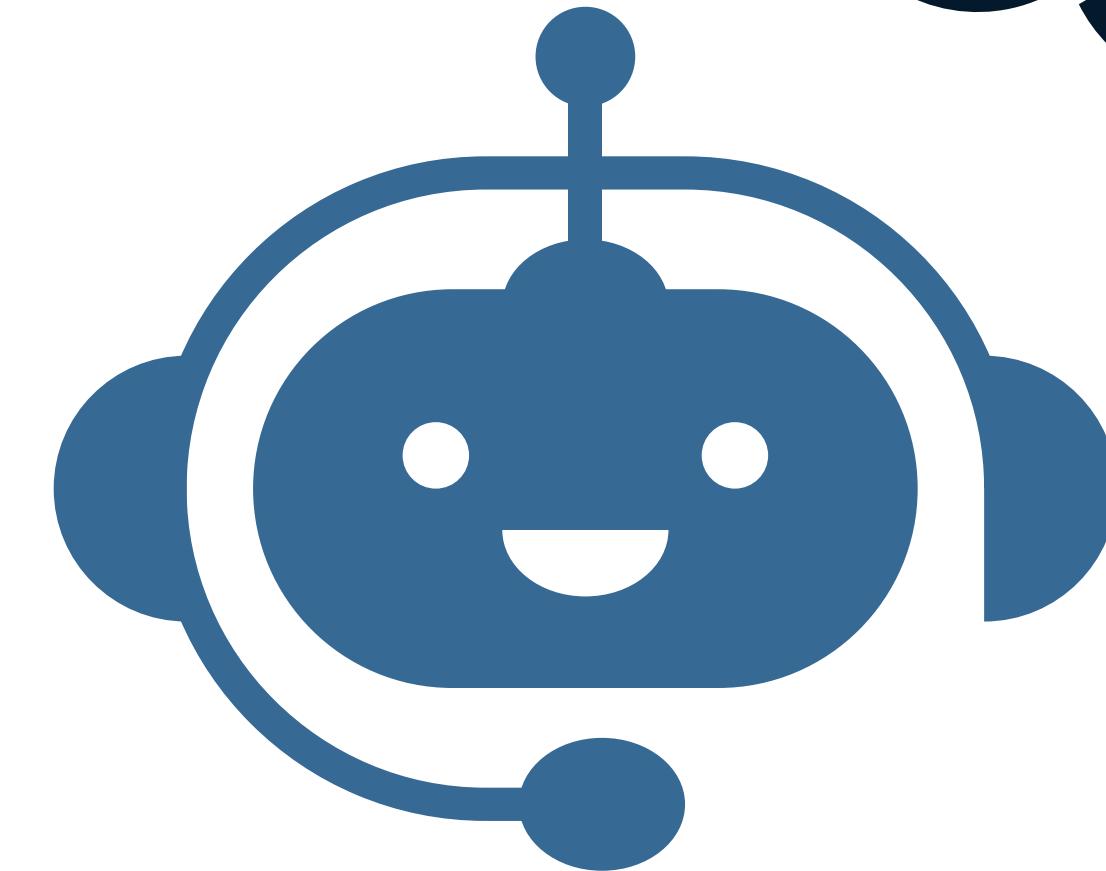
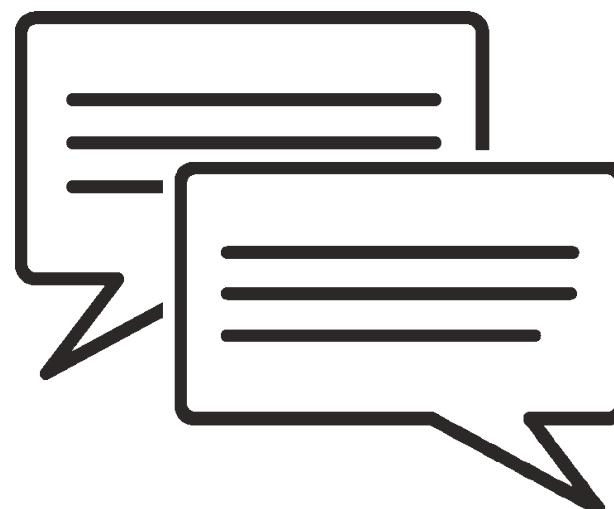
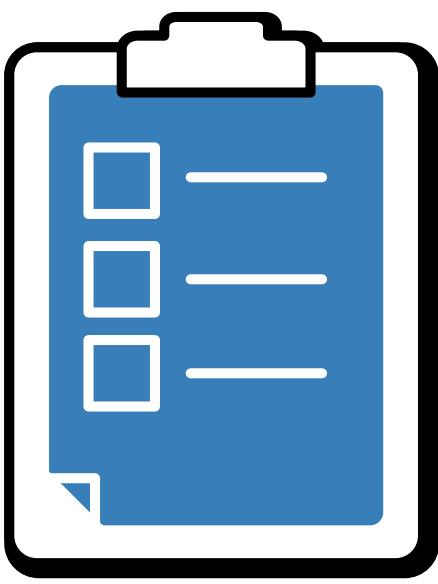
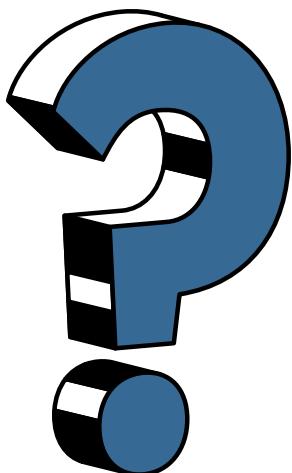
# OpenAI

- Research and develop AI systems



# OpenAI

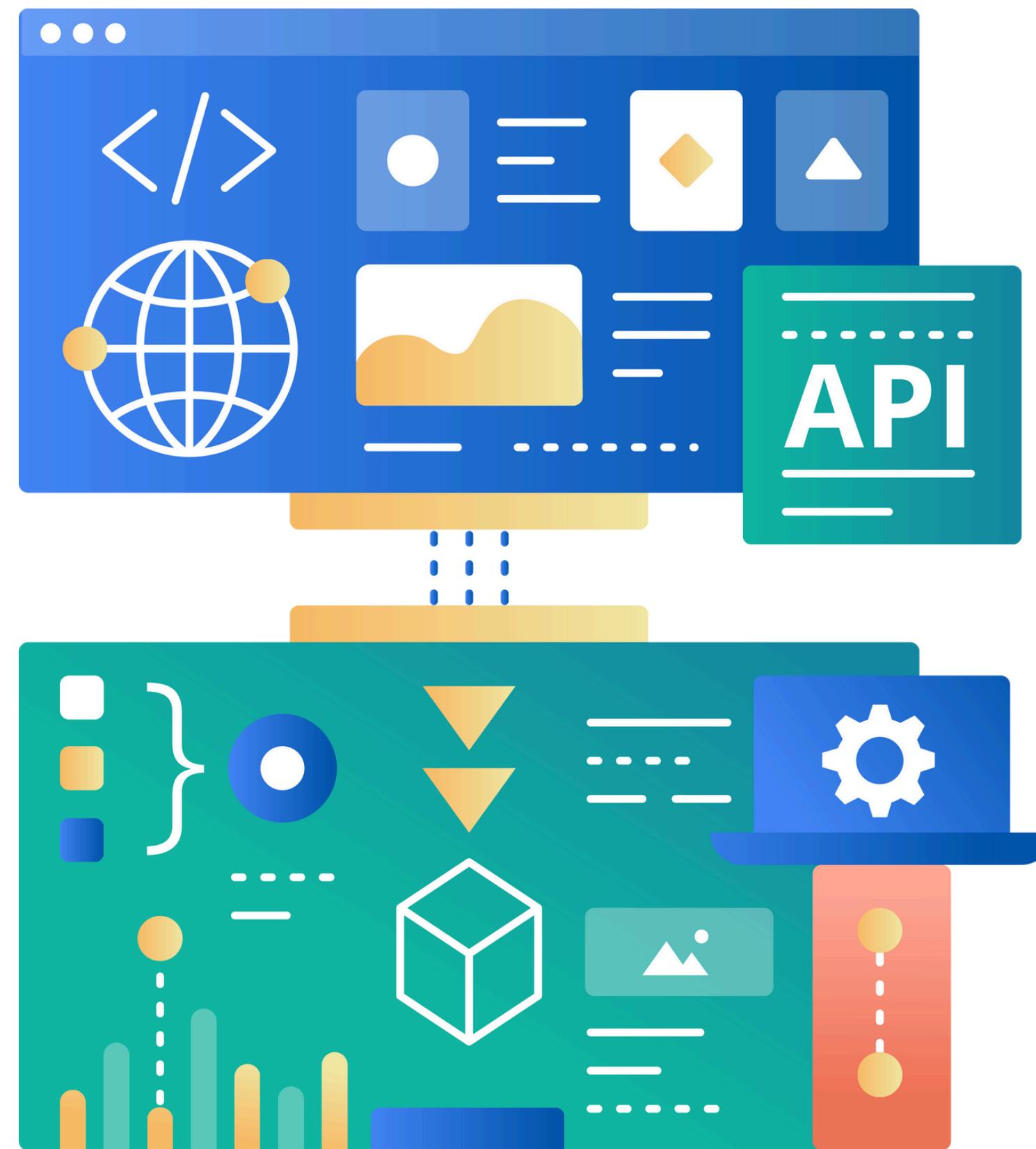
- AI-powered chatbot



ChatGPT



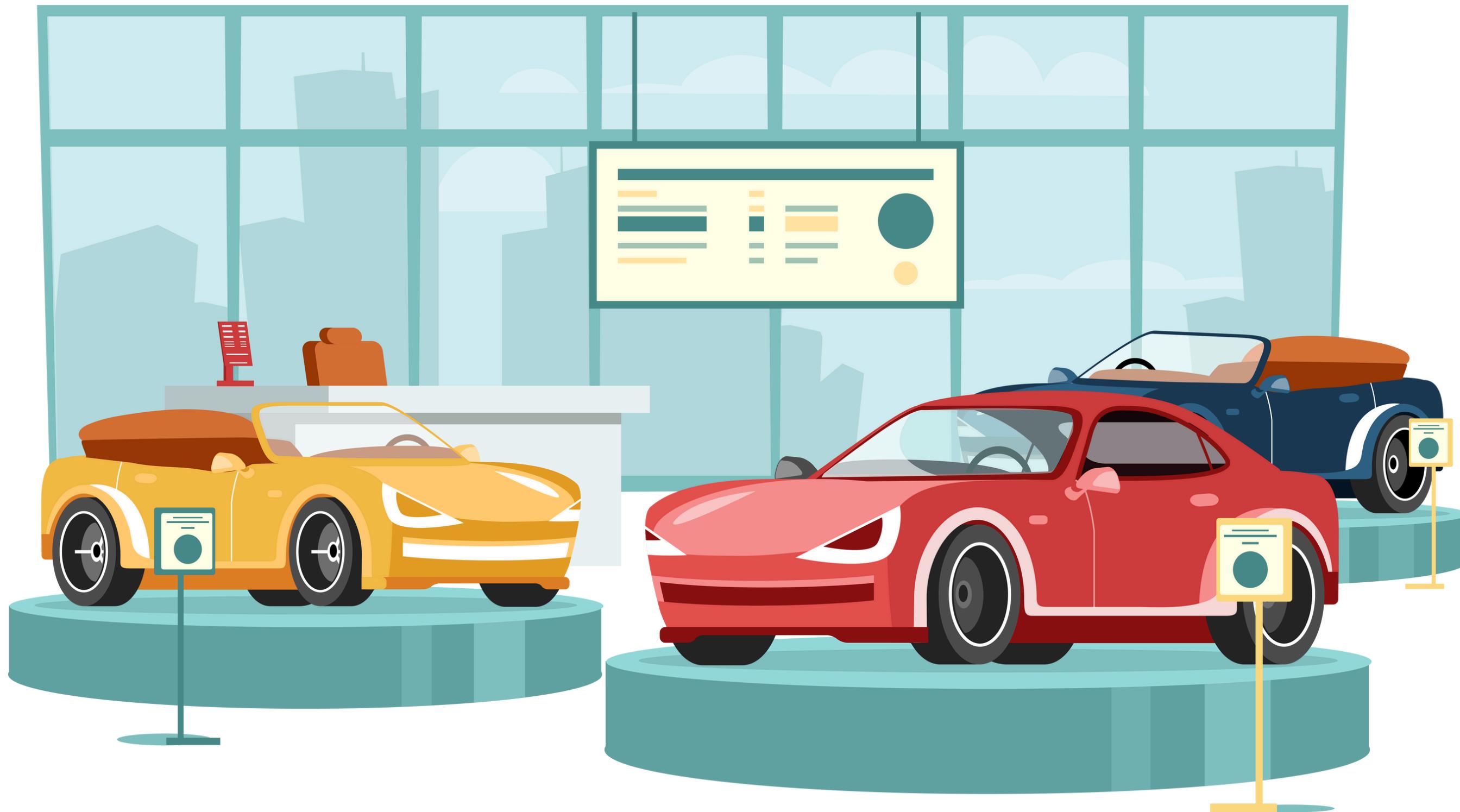
# OpenAI API



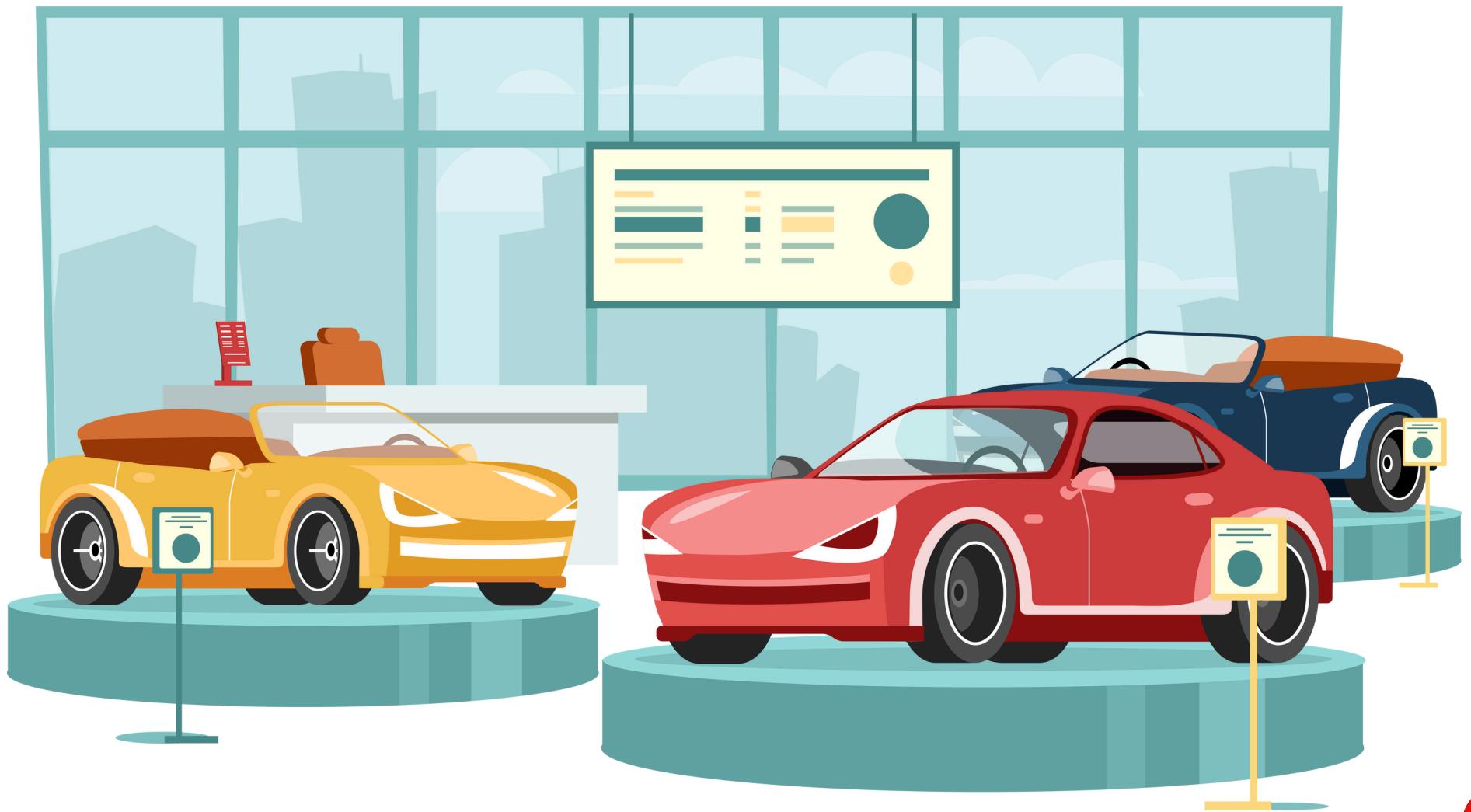
- Access and customize  
OpenAI models



# OpenAI



# OpenAI



# OpenAI



ChatGPT

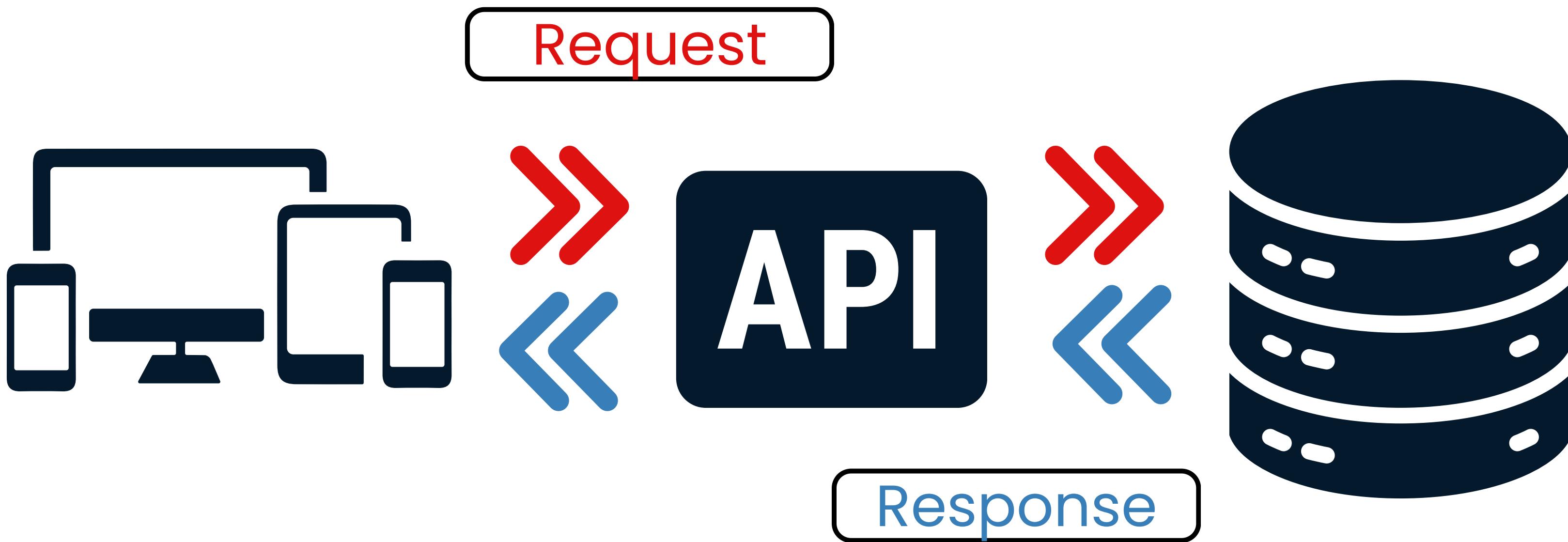


# OpenAI API



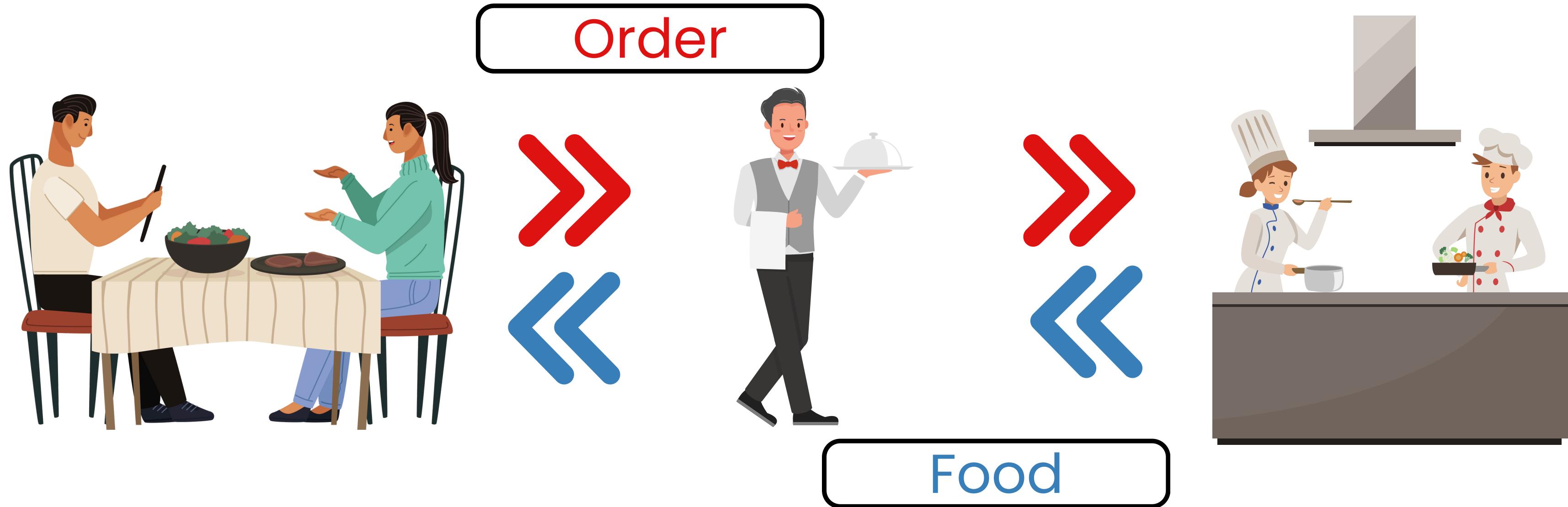
# What is an API?

- Application Programming Interface



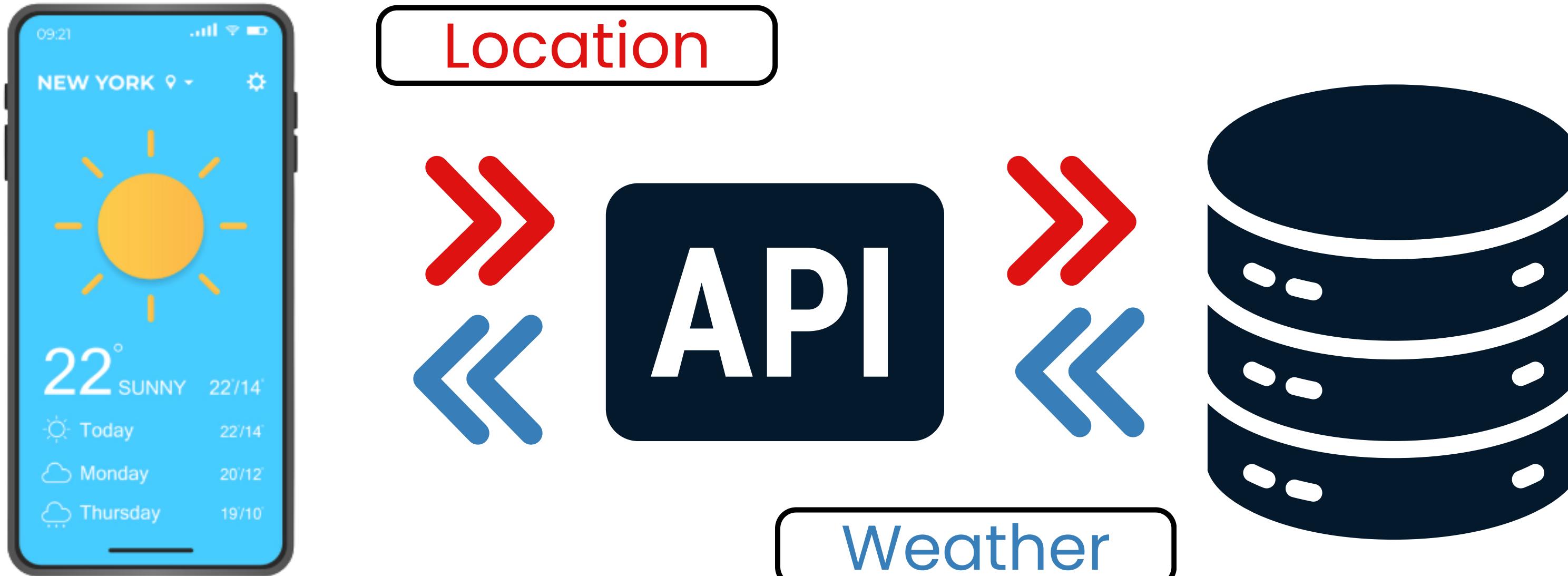
# what is an API?

- Application Programming Interface

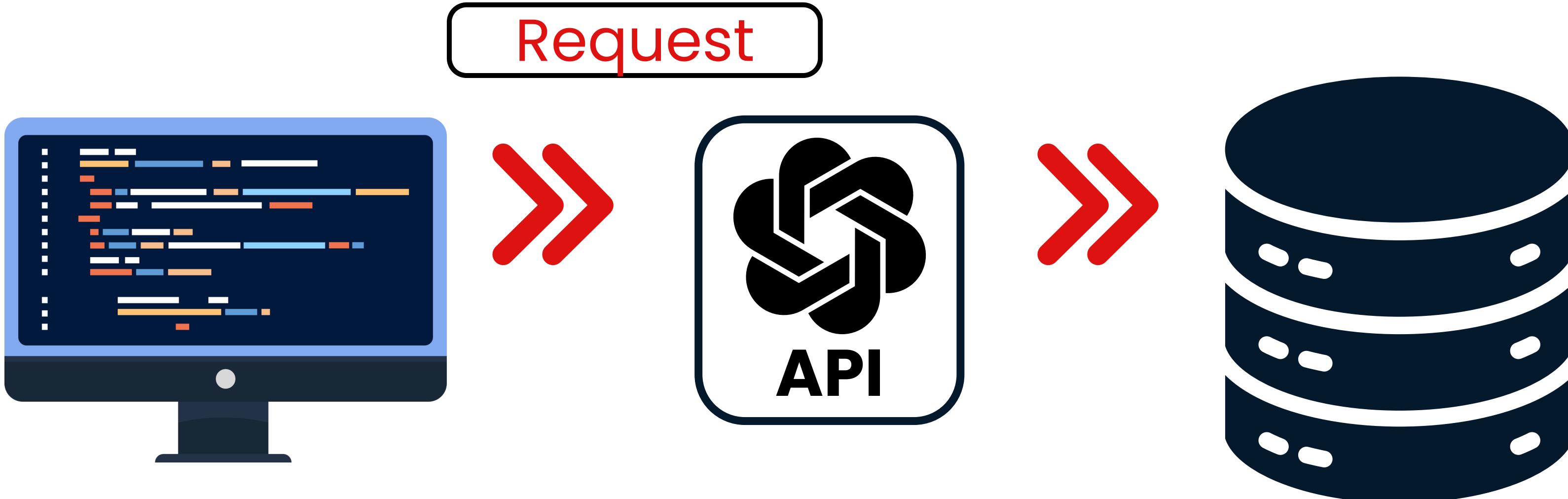


# What is an API?

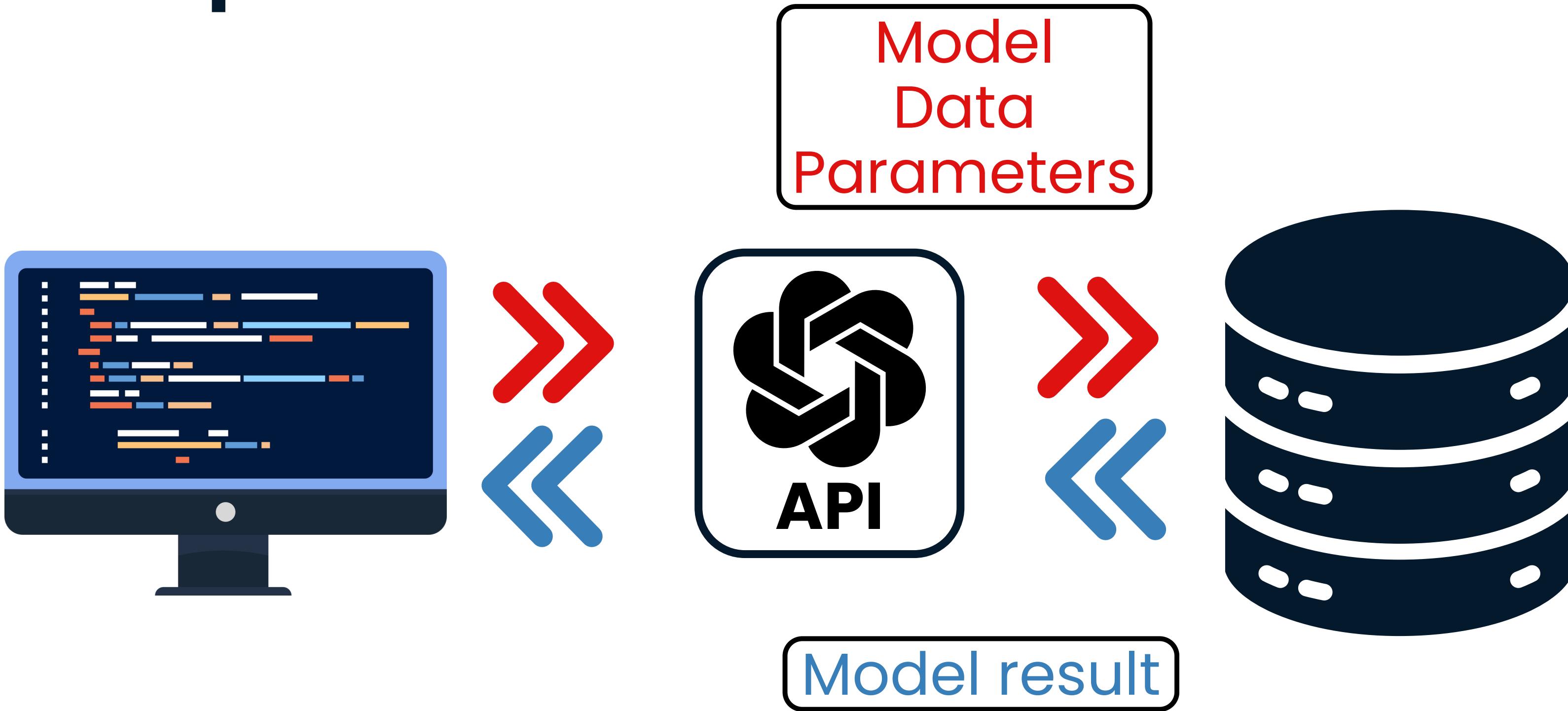
- Application Programming Interface



# The OpenAI API

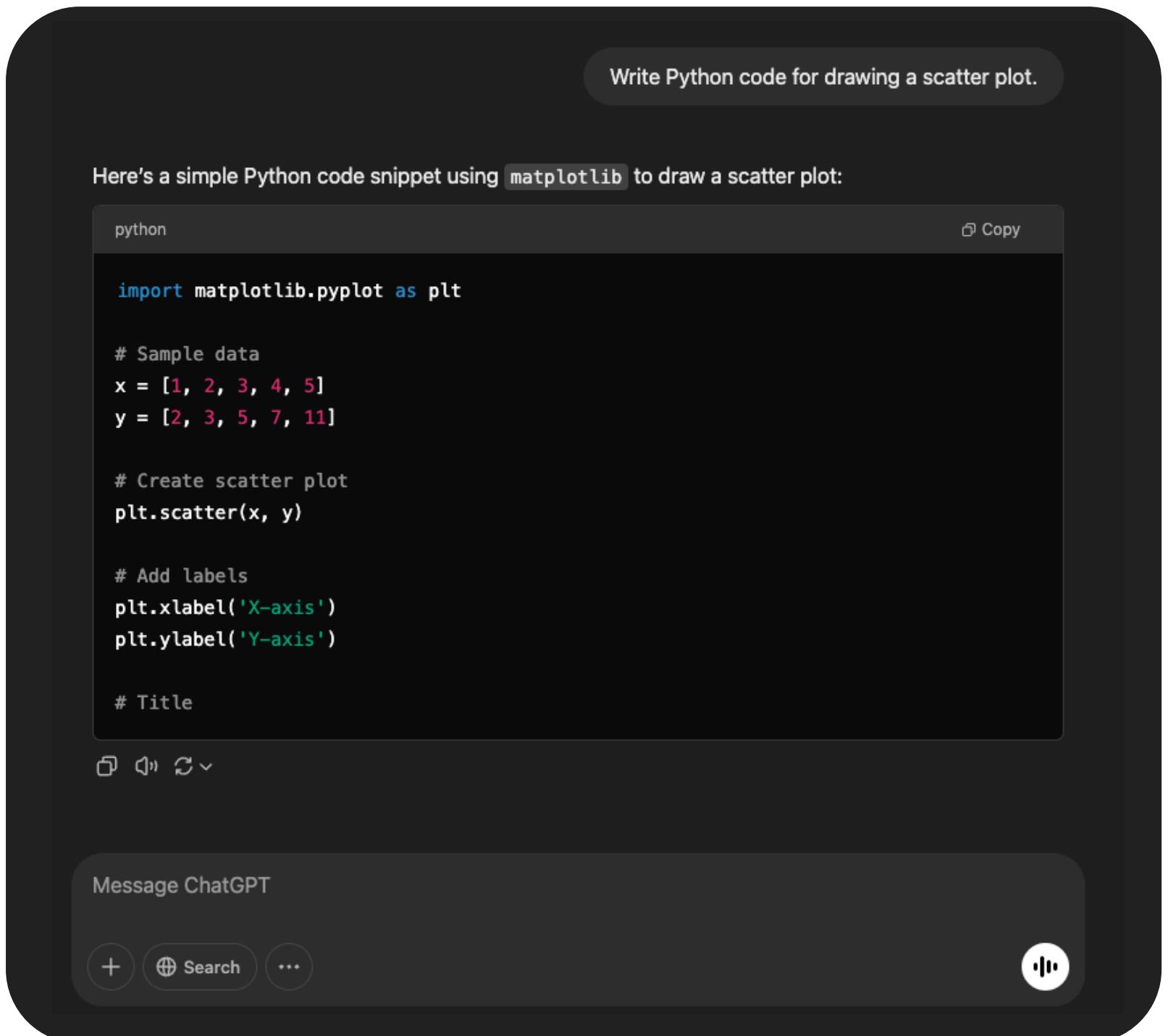


# The OpenAI API



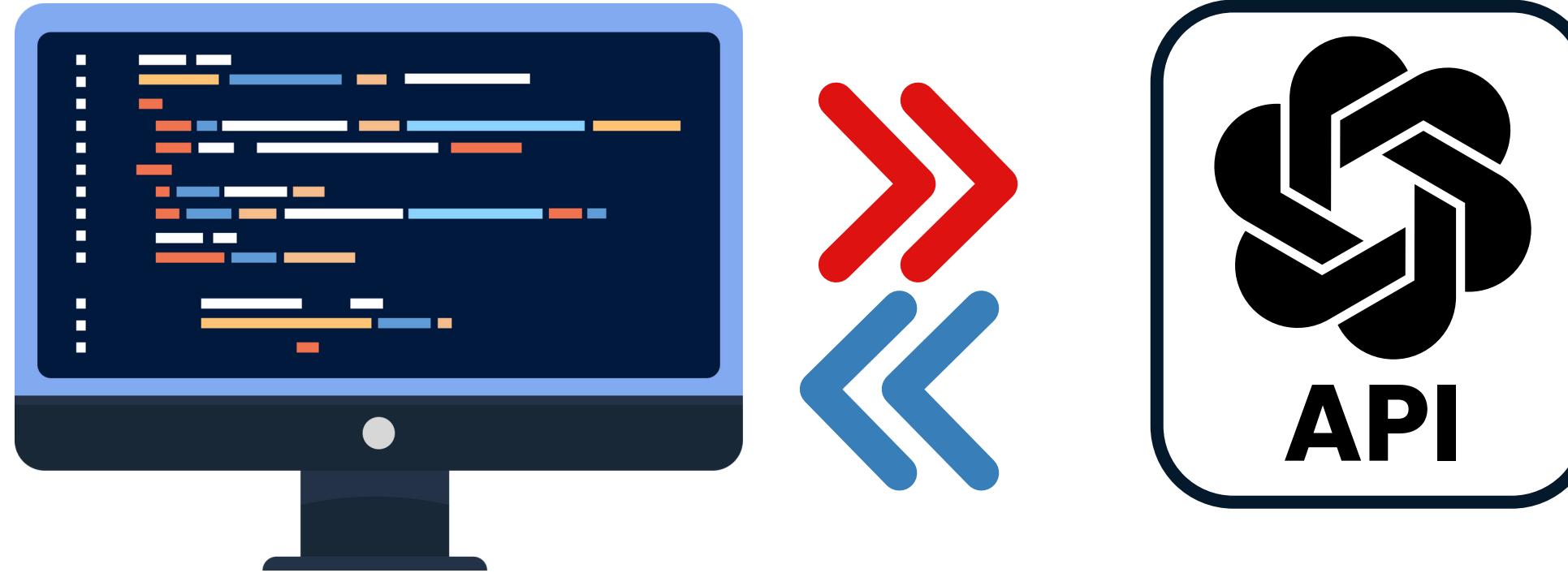
# Web interface vs. APIs

## ChatGPT



- Minimal setup
- Streamline workflows

# Web interface vs. APIs



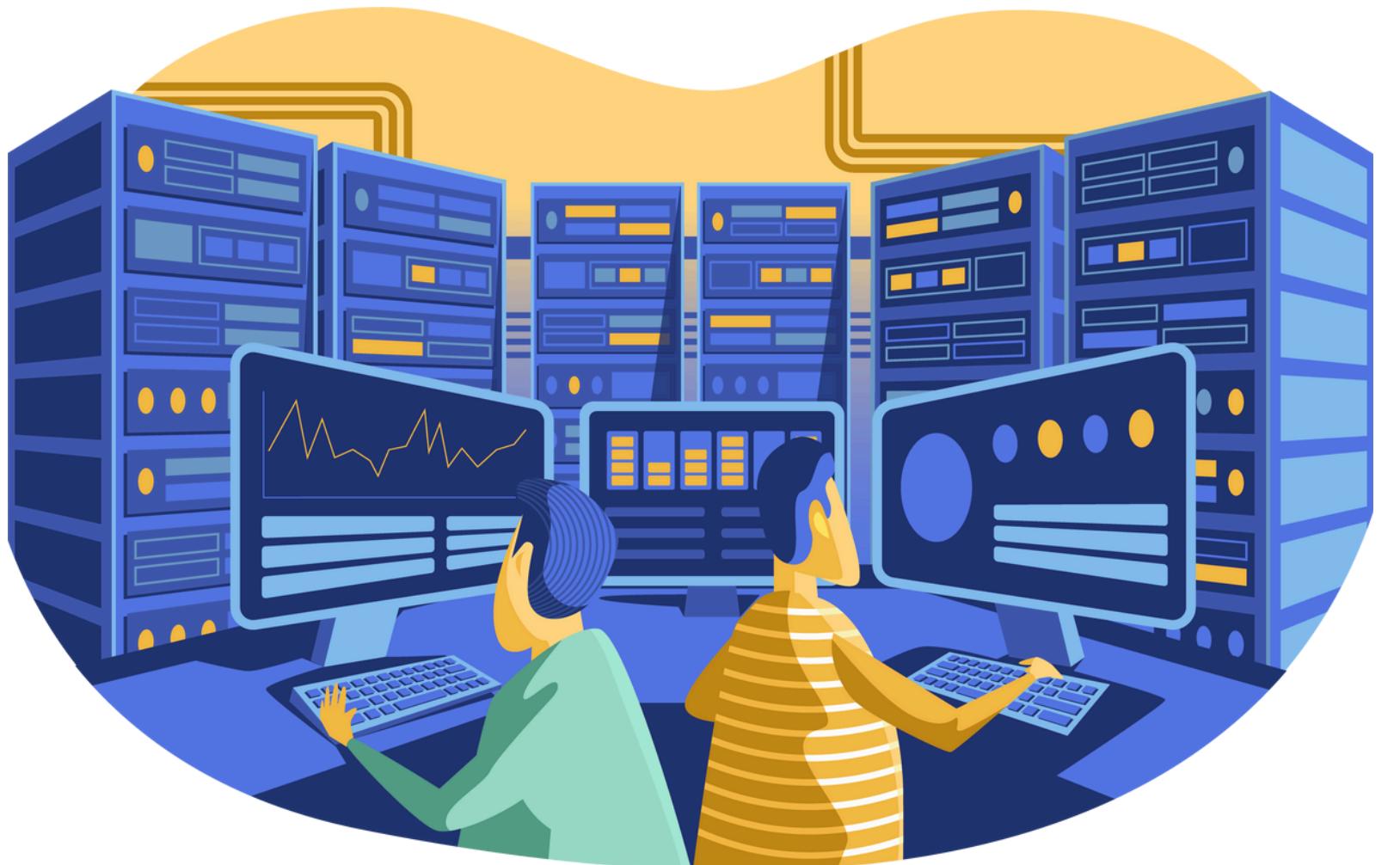
- Integrate AI into products
- Work with models programmatically

# OpenAI API



NEW

# OpenAI API



- Computer resources
- Data

# Building AI applications



- AI-powered IDE

The screenshot shows a Jupyter-style notebook interface with an AI-powered IDE overlay. The code cell contains Python code for loading a dataset and plotting total game sales over time by platform:

```
Split by platform ←
Run cell 26+1
Run
AI
Copy link
Delete
import pandas as pd
import plotly.express as px

# Load the dataset
vgsales_df = pd.read_csv("vgsales.csv")

# Group by year and platform, then sum up global sales
sales_over_time_platform = vgsales_df.groupby(["Year", "Platform"])["Global_Sales"].sum().reset_index()

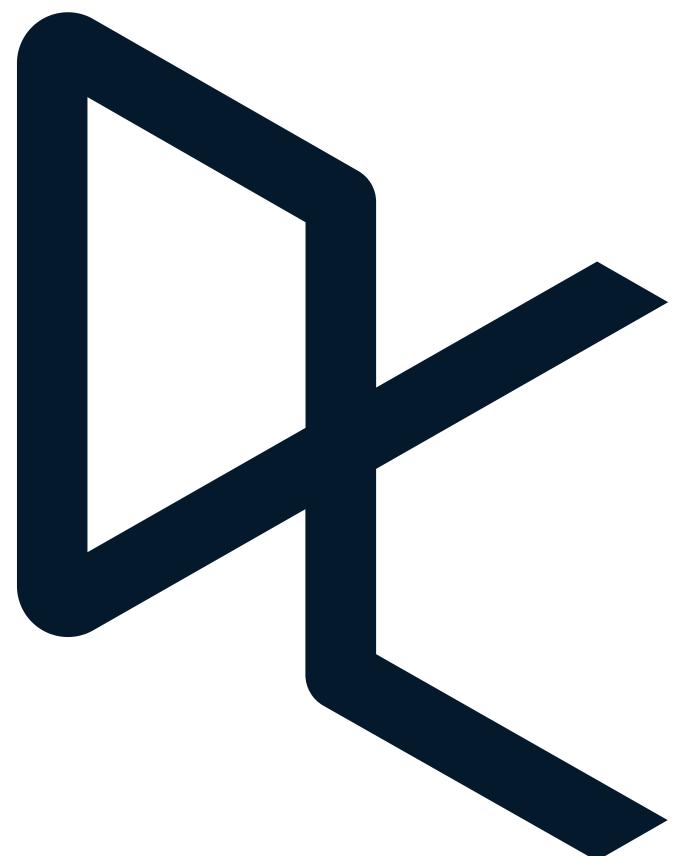
# Plot total game sales over time split by platform
fig = px.line(sales_over_time_platform, x='Year', y='Global_Sales')
```

The resulting line chart, titled "Total Video Game Sales Over Time", displays global sales in millions from 1980 to 2020. The sales show a significant increase starting around 1995, peaking around 2008, and then declining towards 2020.

Year	Global Sales (in millions)
1980	~10
1985	~50
1990	~50
1995	~100
2000	~200
2005	~450
2010	~600
2015	~250
2020	~10

At the bottom of the interface, there is a text input field for telling the AI what to do, and buttons for Done and Undo.

# LET'S PRACTICE!

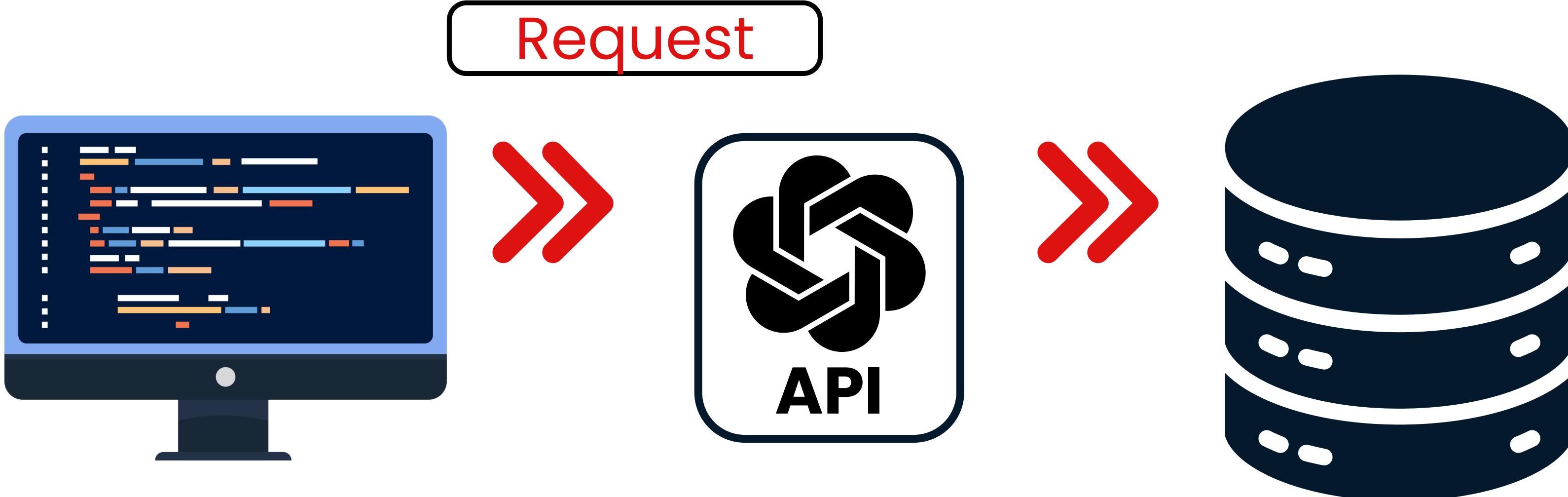


# MAKING REQUESTS TO THE

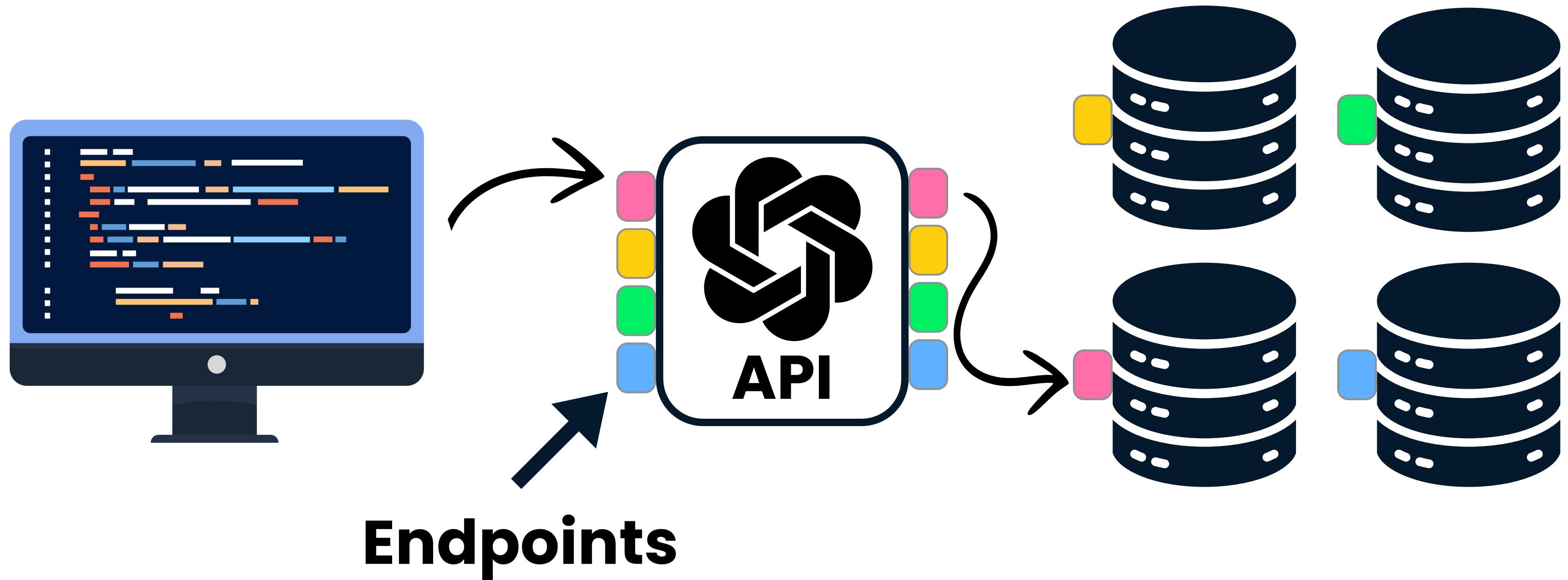
# OpenAI API



# The OpenAI API

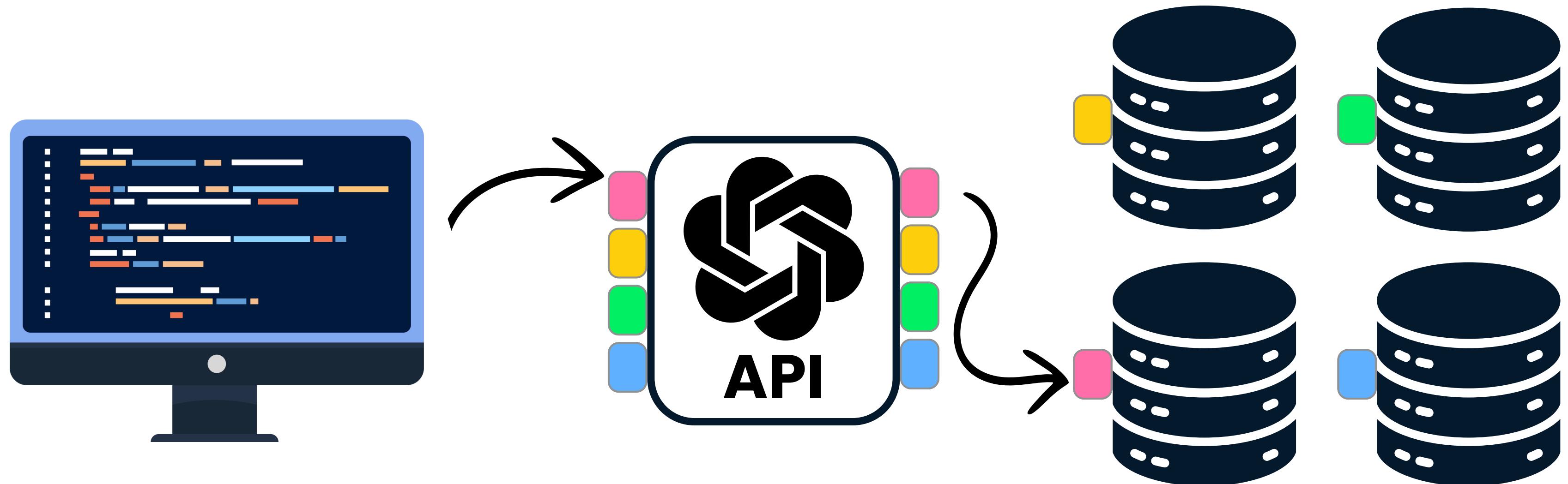


# API Endpoints

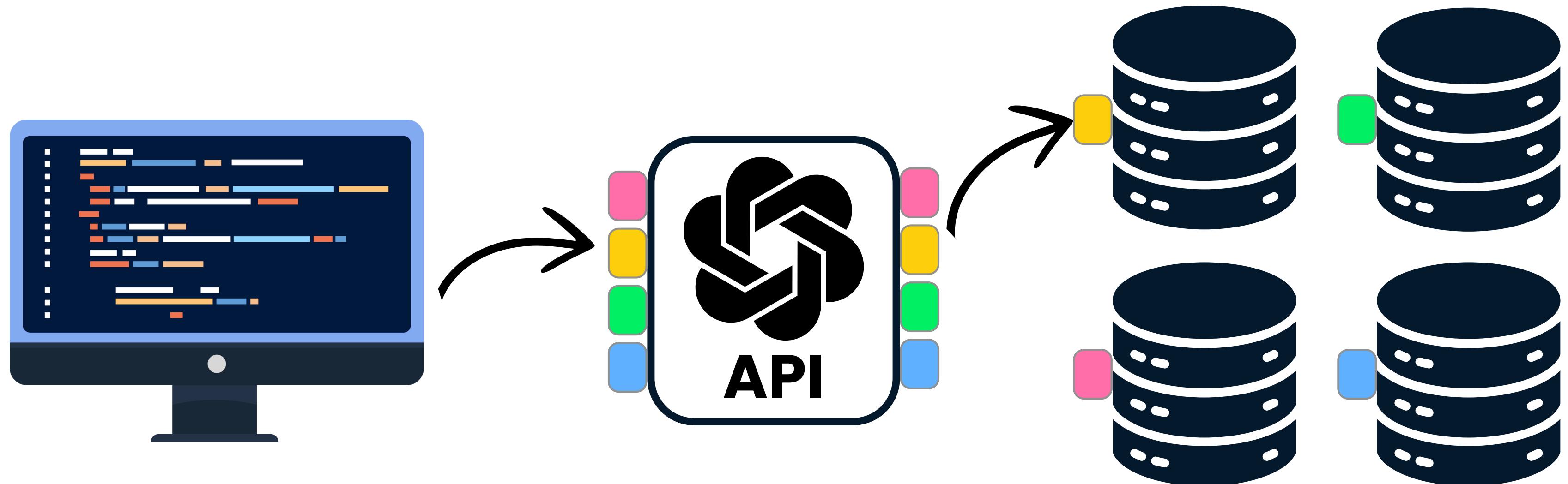




# API Endpoints



# API Endpoints



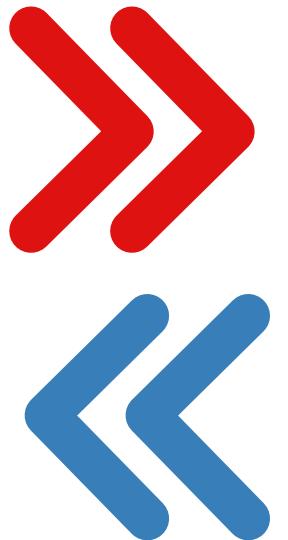
# Authentication

- Unique key

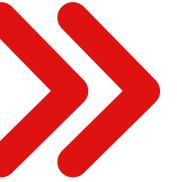
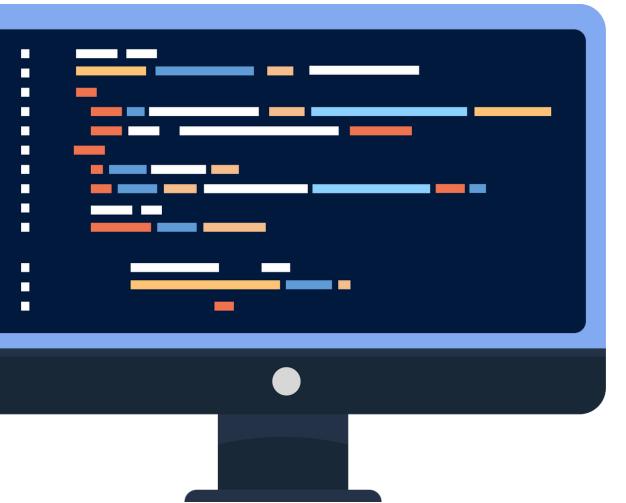


# API usage costs

- OpenAI API



# Course exercises



Exercise

## Categorizing companies

In this exercise, you'll use a Chat Completions model to categorize different companies. At first, you won't specify the categories to see how the model categorizes them. Then, you'll specify the categories in the prompt to ensure they are categorized in a desirable and predictable way.

Instructions 100 XP

- Create a request to the Chat Completions endpoint to categorize the following companies: Apple, Microsoft, Saudi Aramco, Alphabet, Amazon, Berkshire Hathaway, NVIDIA, Meta, Tesla, and LVMH; run the code to see the response.
- Alter the prompt to specify the four categories that the companies should be classified into, Tech, Energy, Luxury Goods, or Investment, and re-run the code.

Take Hint (-30 XP)

```
script.py
```

```
1 client = OpenAI(api_key="OPENAI_API_KEY")
2
3 # Define a prompt for the categorization
4 prompt = "Categorize these companies by sector into Tech, Energy, Luxury
5 Goods, or Investment: Apple, Microsoft, Saudi Aramco, Alphabet, Amazon,
6 Berkshire Hathaway, NVIDIA, Meta, Tesla, LVMH"
7
8 # Create a request to the Chat Completions endpoint
9 response = client.chat.completions.create(
10     model="gpt-4o-mini",
11     messages=[{"role": "user", "content": prompt}],
12     max_tokens=100,
13     temperature=0.5
14 )
15 print(response.choices[0].message.content)
```

IPython Shell Slides

In [1]:



# Course exercises

Exercise

## Categorizing companies

In this exercise, you'll use a Chat Completions model to categorize different companies. At first, you won't specify the categories to see how the model categorizes them. Then, you'll specify the categories in the prompt to ensure they are categorized in a desirable and predictable way.

Instructions 100 XP

- Create a request to the Chat Completions endpoint to categorize the following companies: Apple, Microsoft, Saudi Aramco, Alphabet, Amazon, Berkshire Hathaway, NVIDIA, Meta, Tesla, and LVMH; run the code to see the response.
- Alter the prompt to specify the four categories that the companies should be classified into, Tech, Energy, Luxury Goods, or Investment, and re-run the code.

Take Hint (-30 XP)

script.py

```
1 client = OpenAI(api_key="OPENAI_API_KEY")
2
3 # Define a prompt for the categorization
4 prompt = "Categorize these companies by sector into Tech, Energy, Luxury Goods, or Investment: Apple, Microsoft, Saudi Aramco, Alphabet, Amazon, Berkshire Hathaway, NVIDIA, Meta, Tesla, LVMH"
5
6 # Create a request to the Chat Completions endpoint
7 response = client.chat.completions.create(
8     model="gpt-4o-mini",
9     messages=[{"role": "user", "content": prompt}],
10    max_tokens=100,
11    temperature=0.5
12 )
13
14 print(response.choices[0].message.content)
```

IPython Shell Slides

In [1]:

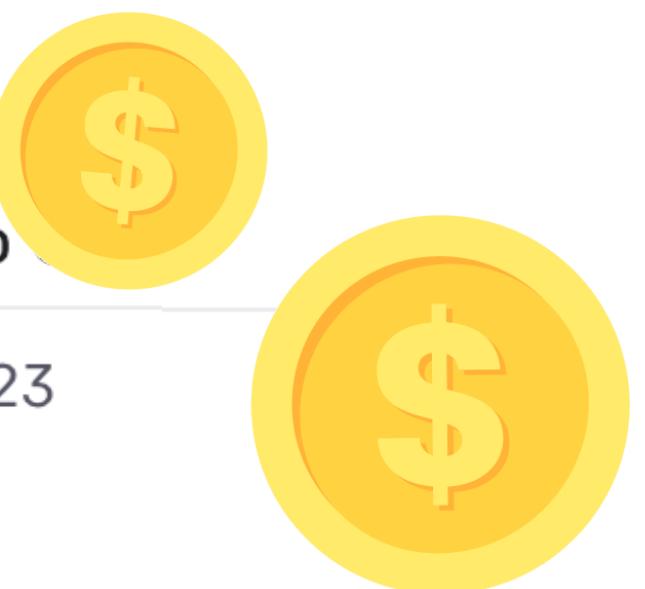


# Creating an API key

- 1 Create an OpenAI account
- 2 Go to the API keys page
- 3 Create a new secret key



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



# Making a request



# Making a request

```
from openai import OpenAI  
  
client = OpenAI(api_key="ENTER YOUR KEY HERE")
```

- **Client** - configures environment for communicating with API

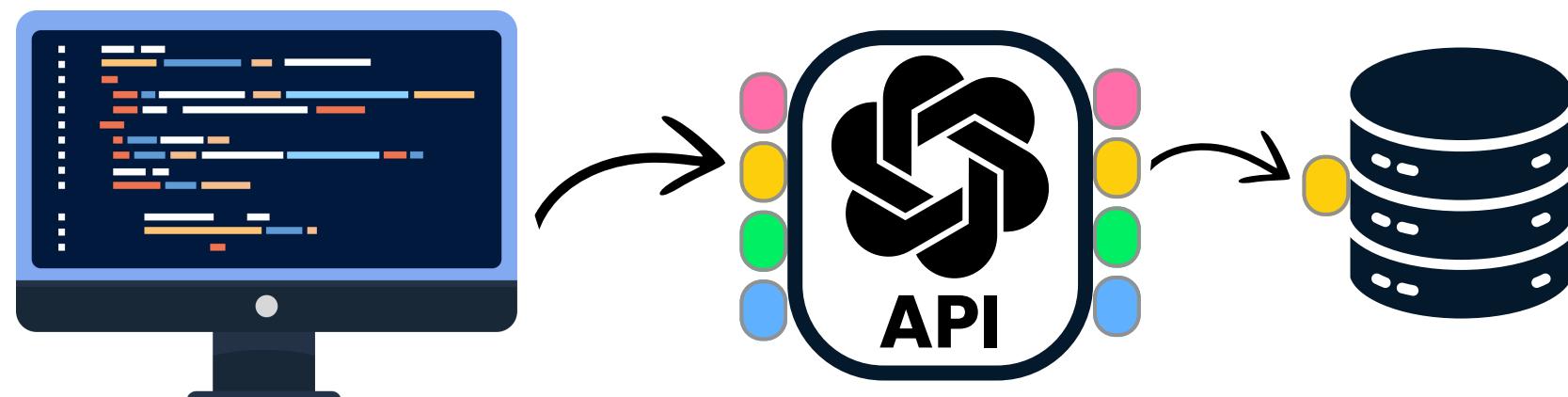
# Making a request

```
from openai import OpenAI  
  
client = OpenAI(api_key="ENTER YOUR KEY HERE")
```



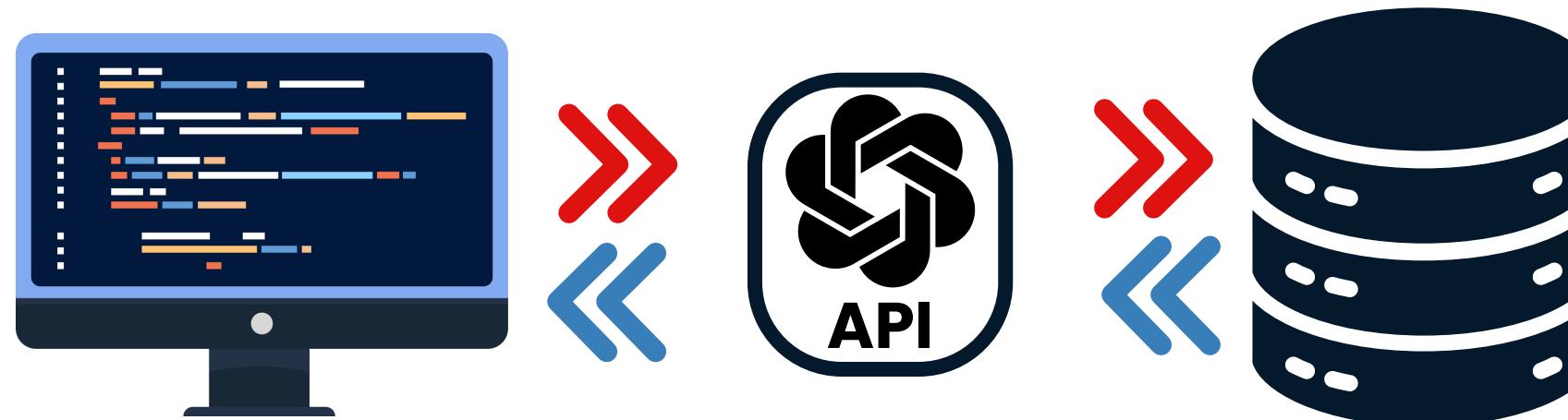
# Making a request

```
from openai import OpenAI  
  
client = OpenAI(api_key="ENTER YOUR KEY HERE")  
  
response = client.chat.completions.create()  
  
)
```



# Making a request

```
from openai import OpenAI  
  
client = OpenAI(api_key="ENTER YOUR KEY HERE")  
  
response = client.chat.completions.create(  
    model="gpt-4o-mini",  
    messages=[{"role": "user", "content": "What is the OpenAI API?"}]  
)
```



# Making a request

```
from openai import OpenAI

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

response = client.chat.completions.create(
    model="gpt-4o-mini",
    messages=[{"role": "user", "content": "What is the OpenAI API?"]
)
```

- List of dictionaries
- Content from **user** role prompts the model

# Making a request

```
from openai import OpenAI

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

response = client.chat.completions.create(
    model="gpt-4o-mini",
    messages=[{"role": "user", "content": "What is the OpenAI API?"]
)

print(response)
```

# The response

```
ChatCompletion(id='chatcmpl-AEcQbQekIzxGxVAKYVAjgUAXokgrl', choices=[Choice(finish_reason='length', index=0, logprobs=None, message=ChatCompletionMessage(content='The OpenAI API is a cloud-based service provided by OpenAI that allows developers to integrate advanced AI models into their applications.', refusal=None, role='assistant', function_call=None, tool_calls=None)), created=1728047673, model='gpt-4o-mini-2024-07-18', object='chat.completion', service_tier=None, system_fingerprint='fp_f85bea6784', usage=CompletionUsage(completion_tokens=30, prompt_tokens=14, total_tokens=44, prompt_tokens_details={'cached_tokens': 0}, completion_tokens_details={'reasoning_tokens': 0}))
```

# The response

```
ChatCompletion(id='chatcmpl-AEcQbQekIzxcxVAKYVAjgUAXokgrl',
    choices=[Choice(finish_reason='length', index=0, logprobs=None,
                    message=ChatCompletionMessage(content='The OpenAI API is a cloud-
                    based service provided by OpenAI that allows developers to
                    integrate advanced AI models into their applications.',
                    refusal=None, role='assistant', function_call=None,
                    tool_calls=None))],
    created=1728047673,
    model='gpt-4o-mini-2024-07-18',
    object='chat.completion', service_tier=None, system_fingerprint='fp_f85bea6784',
    usage=CompletionUsage(completion_tokens=25, prompt_tokens=14, total_tokens=39,
                           prompt_tokens_details={'cached_tokens': 0},
                           completion_tokens_details={'reasoning_tokens': 0}))
```

# Interpreting the response

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

```
[Choice(finish_reason='length', index=0, logprobs=None,  
message=ChatCompletionMessage(content='The OpenAI API is a cloud-  
based service provided by OpenAI that allows developers to integrate  
advanced AI models into their applications.', refusal=None,  
role='assistant', function_call=None, tool_calls=None)])]
```

- List with one element

# Interpreting the response

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

```
Choice(finish_reason='length', index=0, logprobs=None,  
message=ChatCompletionMessage(content='The OpenAI API is a cloud-  
based service provided by OpenAI that allows developers to integrate  
advanced AI models into their applications.', refusal=None,  
role='assistant', function_call=None, tool_calls=None))
```

# Interpreting the response

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

```
ChatCompletionMessage(content='The OpenAI API is a cloud-based  
service provided by OpenAI that allows developers to integrate  
advanced AI models into their applications.', refusal=None,  
role='assistant', function_call=None, tool_calls=None)
```

# Interpreting the response

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

The OpenAI API is a cloud-based service provided by OpenAI that allows developers to integrate advanced AI models into their applications.

- Response as a string

# Interpreting the response



# LET'S PRACTICE!

