LLMs for me

Introduction to GPT & the OpenAI Ecosystem

llmsfor.me

Myles Harrison,
AI Consultant & Trainer

January 20th, 2025

NLP from scratch

Agenda

01 APIs & Making Requests

02 Working with the OpenAI API

03 Image Models

04 Next Steps

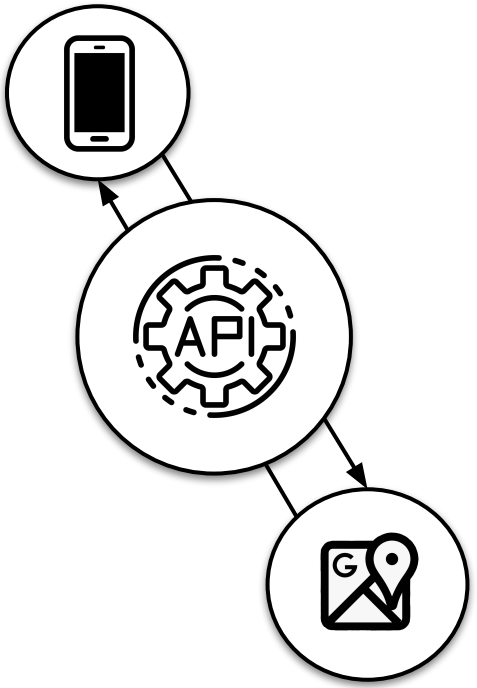
Making Requests

What is an API?

An API (**A**pplication **P**rogramming **I**nterface) is the connection point between two software applications which allows for the exchange of data as well as the ability for one software program to trigger the code of another application to run.

For example, an API might be a bridge between an application which needs to use Google Maps and the Google Maps service. The API keeps the data and logic of the two separate while still allowing them to operate with the data and functionalities they need.

In practice, the term API is use to refer both to the specification for building such a software interface, as well as infrastructure or code running to provide such a service.



Client-Server Model

Client

Uses software to make requests over the wire (web browser, other program or with code)

Server

Hosts web service via REST or other standard, performs actions and/or returns data

HTTP Requests

HTTP requests are what your web browser makes when you navigate to a given URL. Web pages are hosted by web servers and requests are made by client machines (for example, your laptop).

A server machine does not just have to serve web pages as part of a web service, it can also provide other services, such as returning data from a database or performing transactions.

There are a number of HTTP status codes that are returned as part of the response to a request. Some commonly encountered ones are detailed on the right.

200

"OK" - The request succeeded and the response was returned successfully.

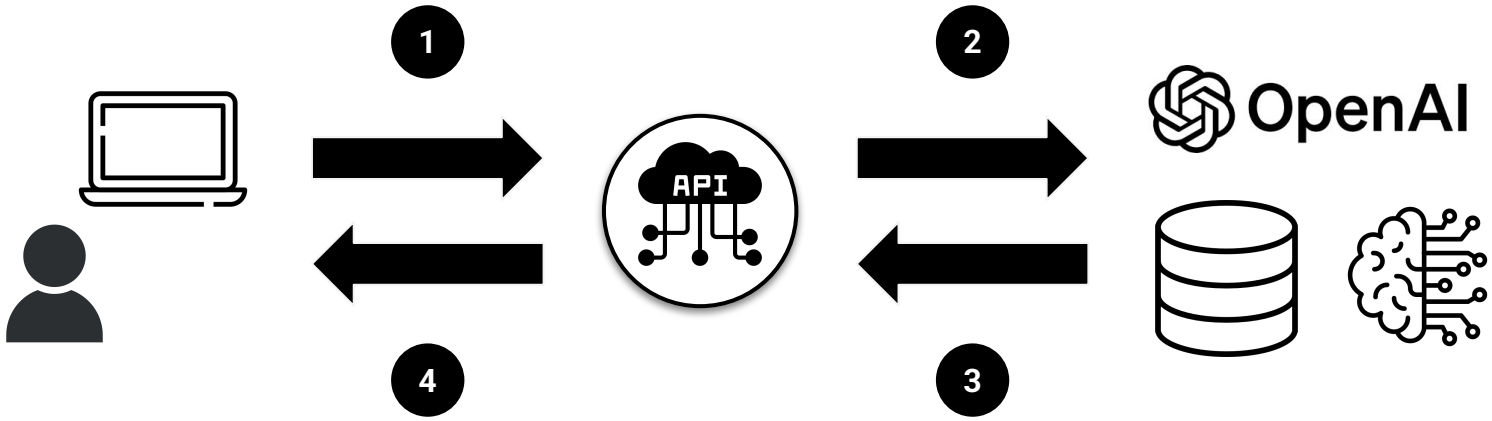
404

"Not Found" - There is nothing at the path where the request was made (resource does not exist)

503

"Unavailable" - The server is unable to return a response to the request (usually because the server is down)

Working with a Foundation Model API



1. User makes request to API via application or programmatically
2. Request is passed to provider systems and processed
3. Provider model generates response and passes back to API
4. API formats request and serves back to the user through an endpoint

Postman

Postman is a tool that streamlines API development with an easy to use interface.

It allows you to craft and test API requests and save commonly used requests, work with multiple different requests concurrently in difference tabs, and more.

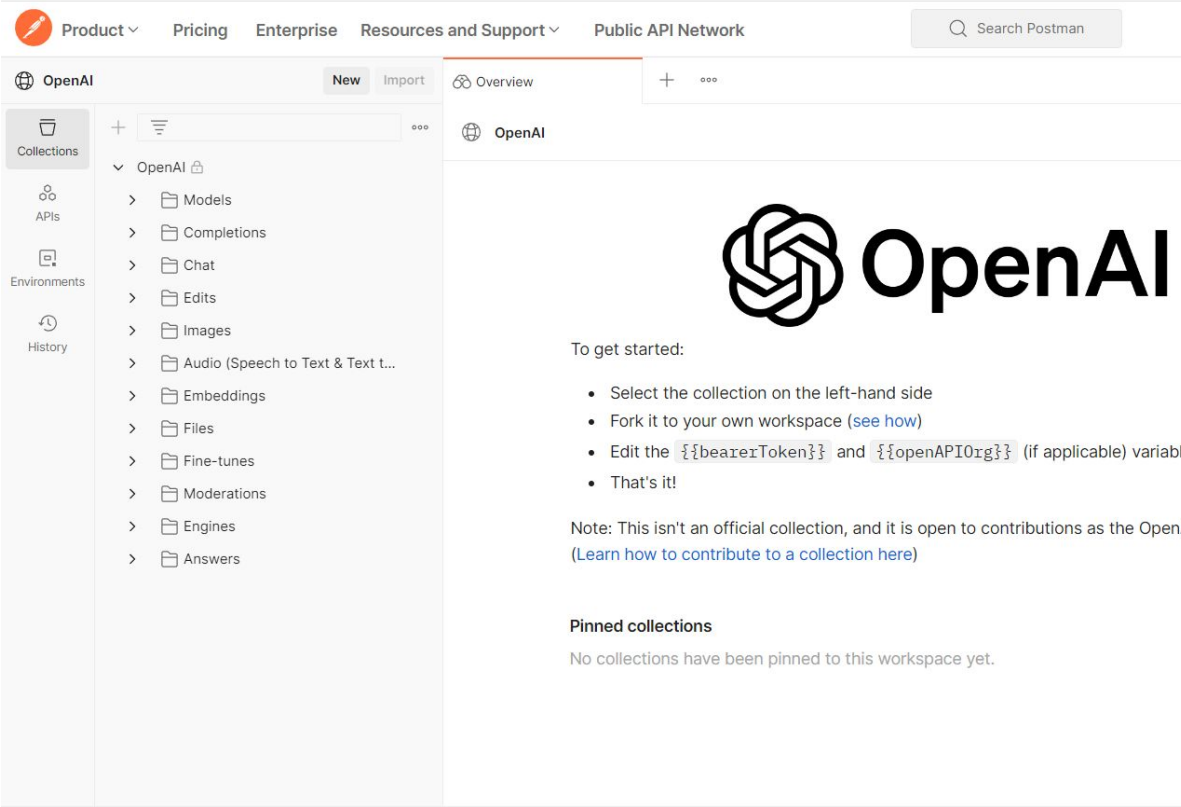
With its collaborative features, you can share workflows ensuring consistency, or access pre-existing libraries of request patterns.

For example, here is the OpenAI Postman Collection.

Postman OpenAI Documentation

There is a dedicated (unofficial) workspace of Postman requests to the OpenAI API available here:

postman.com/devrel/workspace/openai/overview



Making our first API request with Postman

Now we will make our first API request with Postman. We'll simple get a list the models available from OpenAI.
Put the following information into Postman:

- **URL:** https://api.openai.com/v1/models
- **Request Type:** GET
- **Headers:**
  - **Key:** Authorization
  - **Value:** Bearer <Your OpenAI Secret Key>

You should receive a response in JSON of the different available models.

The requests library

`requests` is a Python library that allows making HTTP requests programmatically.

One of the nice things about the requests library is that it is very simple code to write and very well documented.

https://pypi.org/project/requests/

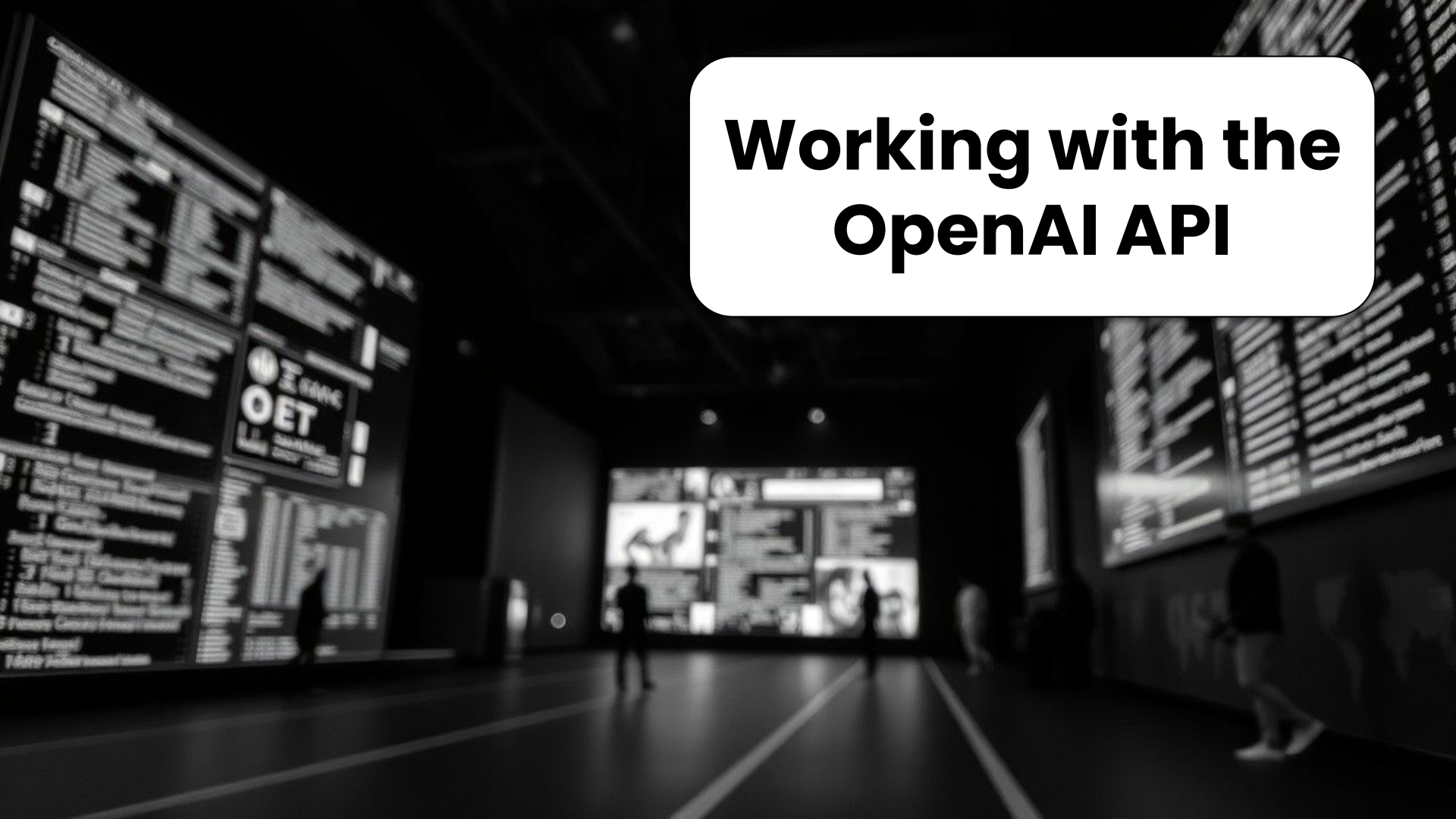
Making your first request with Python

In `requests`, getting data from a web service, whether a web server or REST API, is as simple as a few lines of code.

On the right, we make a request to get Google's homepage, and our response is the HTML code that our browser would render.

```
[1]: import requests

[10]: # Make the request
r = requests.get("https://www.google.com")

[3]: # Check out the response status code
r

[3]: <Response [200]>

[5]: # Print the results as text (first 2000 characters)
r.text[0:2000]

[5]: '<!doctype html><html itemscope="" itemtype="http://schema.org/WebPage" lang
="en-CA"><head><meta content="text/html; charset=UTF-8" http-equiv="Content-Ty
pe"><meta content="/images/branding/googleg/1x/googleg_standard_color_128dp.pn
g" itemprop="image"><title>Google</title><script nonce="x82VAMD51ra8XS_UUIuwv
g">(function(){var _g={kEI:\'DK2ZZIXcLuOv0PEPiuya8A4\',kEXPI:\'0,1359409,6059,
206,4804,2316,383,246,5,1129120,1197778,623,224,379865,16115,28684,22430,1362,
12314,17585,4998,17075,41316,2891,3926,214,4209,3405,606,50059,13245,13721,101
4,1,16916,2652,4,1528,2304,38933,3193,13659,4437,9358,13228,6651,7596,1,11943,
30211,2,16737,21269,1755,5679,1020,31122,4568,6256,23421,1252,5835,14968,4332,
7484,445,2,2,1,26632,8155,7381,2,1399,14569,873,19633,7,1922,9779,42459,20199,
927,19209,14,82,7651,12555,4070,4307,18988,2277,3098,3030,5630,480,9706,1804,7
```

NLP from scratch



Working with the OpenAI API

Costing - Language Models

Flagship models

GPT-4o New

Our fastest and most affordable flagship model

- Text and image input, text output
- 128k context length
- Input: $5 | Output: $15*

GPT-4 Turbo

Our previous high-intelligence model

- Text and image input, text output
- 128k context length
- Input: $10 | Output: $30*

GPT-3.5 Turbo

Our fast, inexpensive model for simple tasks

- Text input, text output
- 16k context length
- Input: $0.50 | Output: $1.50*

** prices per 1 million tokens*

platform.openai.com/docs/models

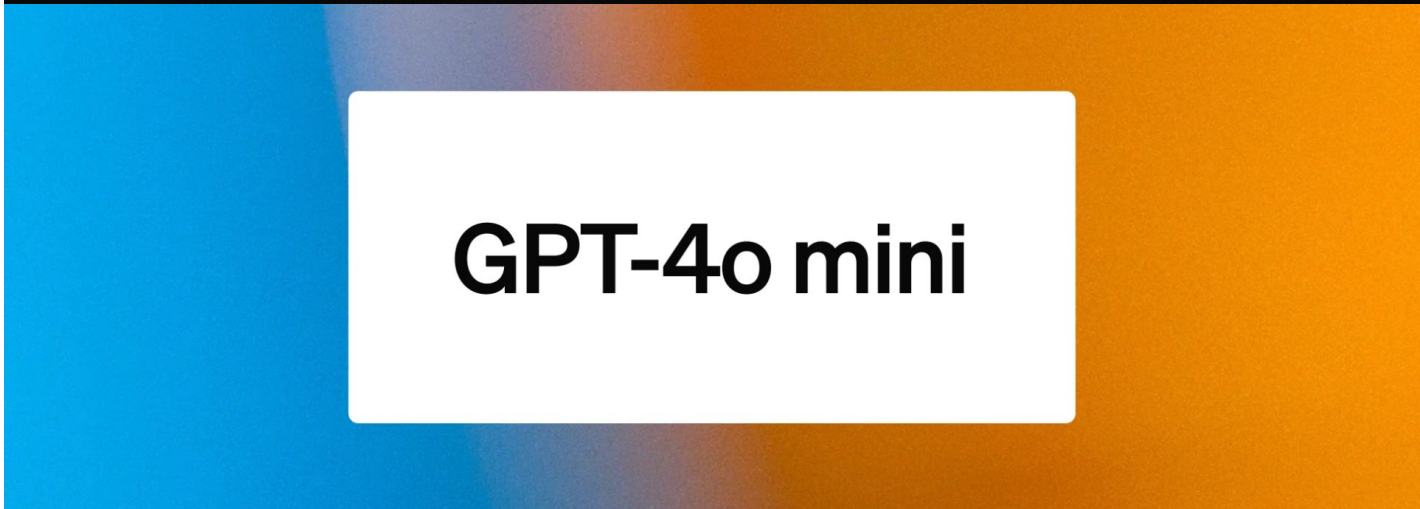
GPT-4o mini

Released 2024/07/18

July 18, 2024

GPT-4o mini: advancing cost-efficient intelligence

Introducing our most cost-efficient small model



Costing - Language Models

GPT-4o mini

New

Our affordable and intelligent small model for fast, lightweight tasks

Text and image input, text output

128k context length

Input: $0.15 | Output: $0.60*

GPT-3.5 Turbo

Our fast, inexpensive model for simple tasks

Text input, text output

16k context length

Input: $0.50 | Output: $1.50*

platform.openai.com/docs/models

And then there’s Pro… 💸

December 5, 2024

Introducing ChatGPT Pro

Broadening usage of frontier AI.

Start now ↗

As AI becomes more advanced, it will solve increasingly complex and critical problems. It also takes significantly more compute to power these capabilities.

Today, we’re adding ChatGPT Pro, a $200 monthly plan that enables scaled access to the best of OpenAI’s models and tools. This plan includes unlimited access to our smartest model, OpenAI o1, as well as to o1-mini, GPT-4o, and Advanced Voice. It also includes o1 pro mode, a version of o1 that uses more compute to think harder and provide even better answers to the hardest problems. In the future, we expect to add more powerful, compute-intensive productivity features to this plan.

ChatGPT Pro provides a way for researchers, engineers, and other individuals who use research-grade intelligence daily to accelerate their productivity and be at the cutting edge of advancements in AI.

More thinking power for more difficult problems

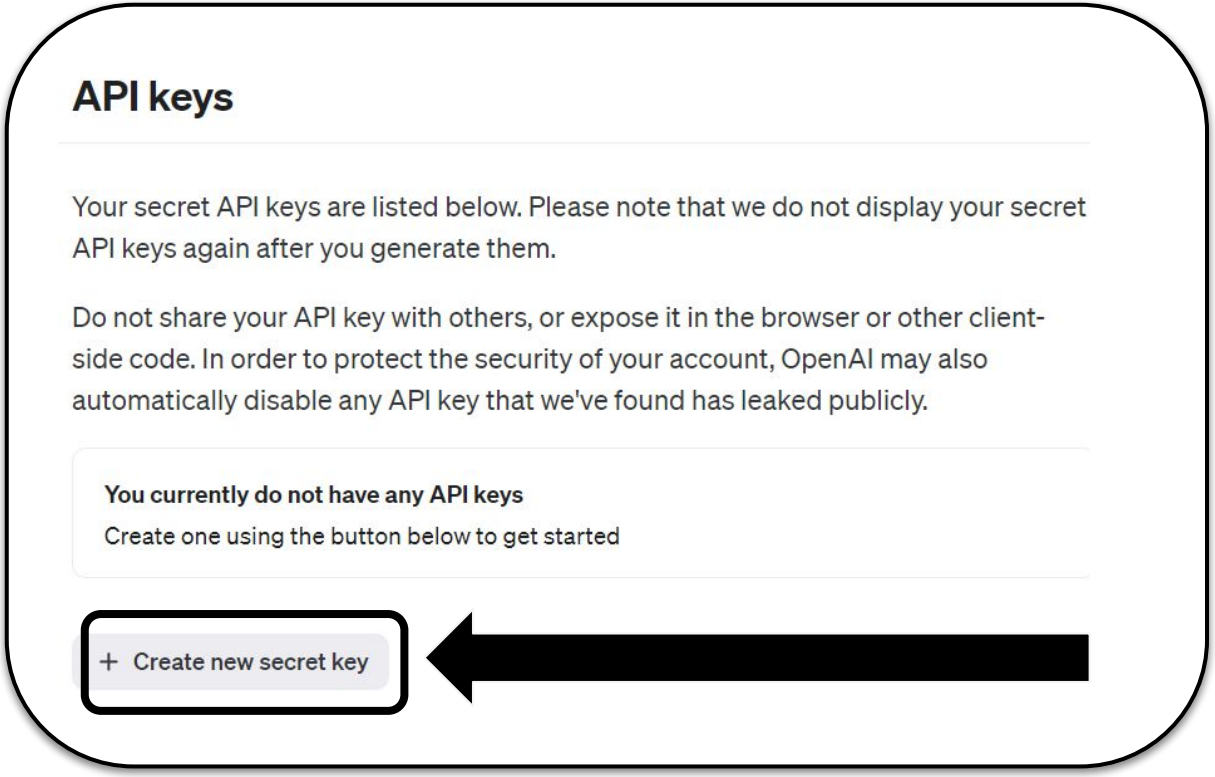
ChatGPT Pro provides access to a version of our most intelligent model that thinks longer for the most reliable responses. In evaluations from external expert testers, o1 pro mode produces more reliably accurate and comprehensive responses, especially

Creating an API Key

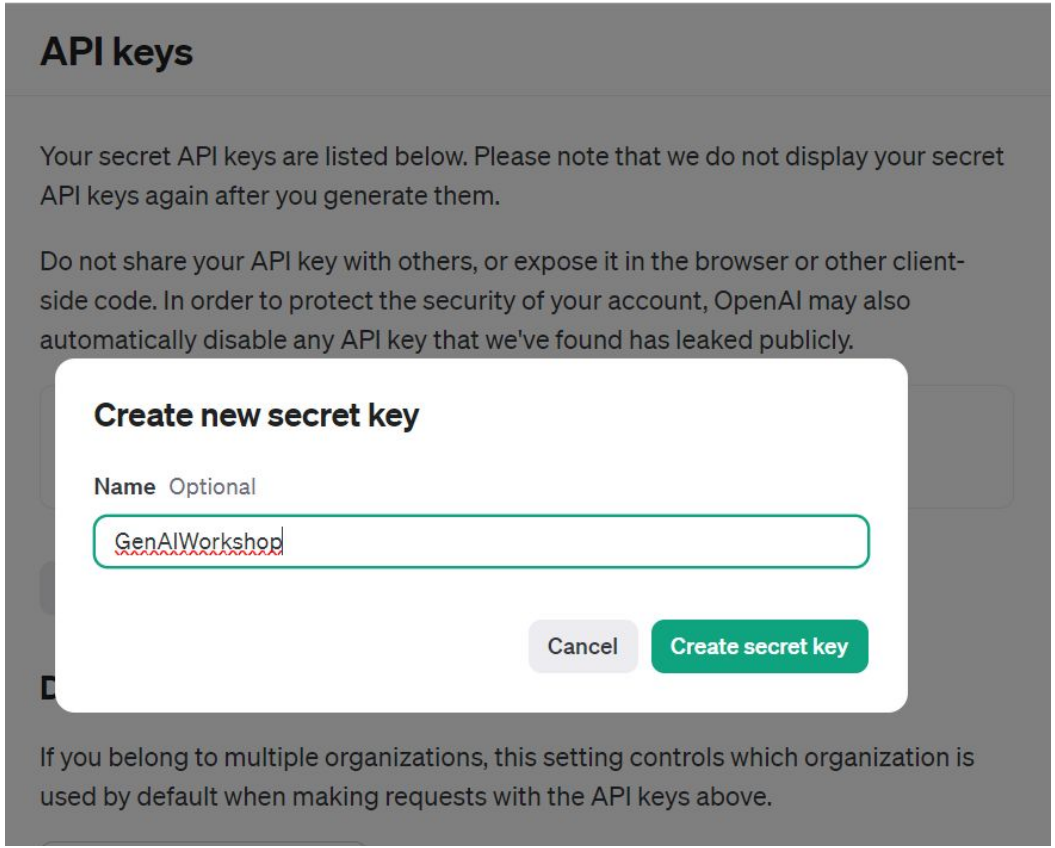
To work with the OpenAI API and get responses, you will need an OpenAI account and API key.

After creating an OpenAI account, you will also need to create an API key. Navigate to the API Key page, and click '**Create New Secret Key**'.

platform.openai.com/api-keys



Creating an OpenAI API Key



OpenAI API Keys - Best Practices

Keep your OpenAI API key confidential, as it allows full access to your account. This means other can use it to make requests and incur costs.
Follow the best practices below to keep your account secure:

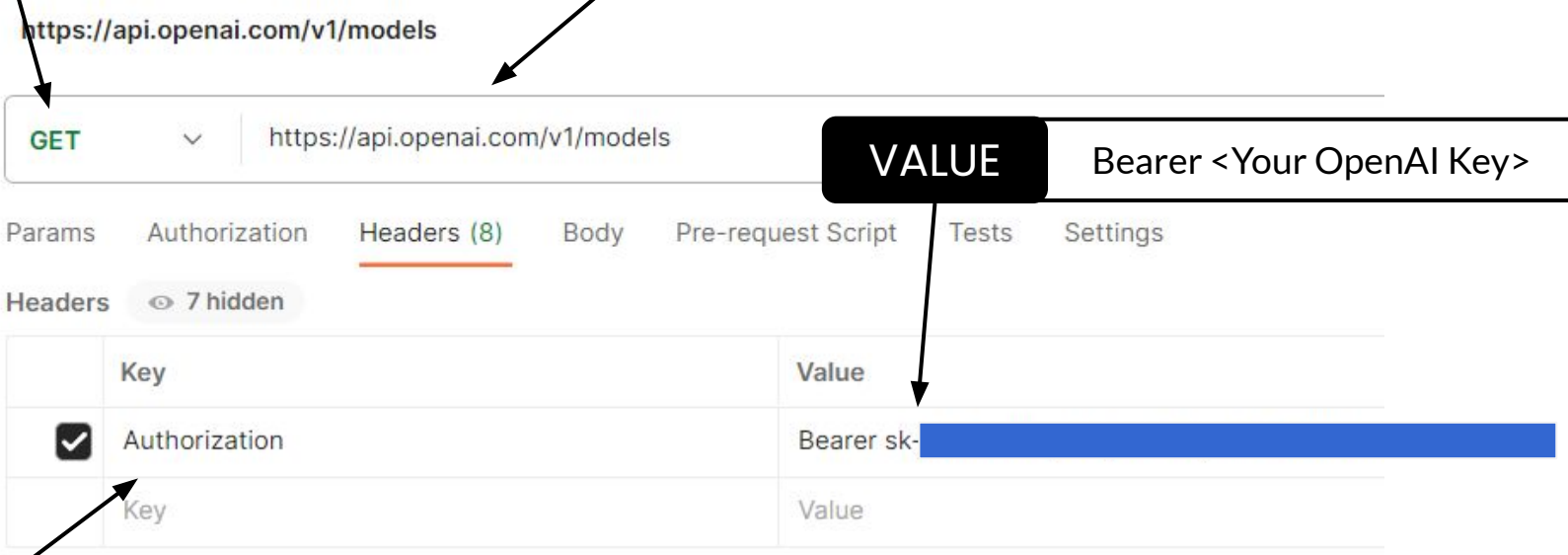
- Never share your API key publicly or store it in version control systems like Github
- Never hard-code an API key into any code (even locally)
- Use environment variables or credential managers
- Monitor usage at platform.openai.com/usage
- When in doubt, create a new API key or establish a regular cadence for key rotation / expiry

See the official documentation from OpenAI.

Making our first request to the OpenAI API

REQUEST TYPE: GET

URL: https://api.openai.com/v1/models



VALUE: Bearer <Your OpenAI Key>

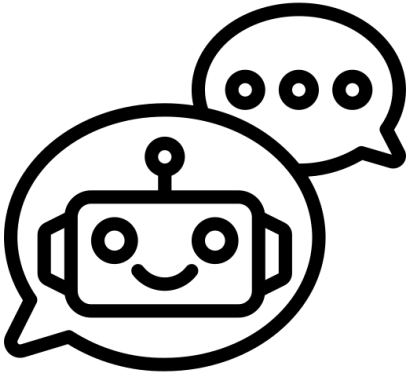
KEY: Authorization

Using GPT-3.5 to generate text

Now we will make our first API request using an OpenAI model.
We'll use the example from the documentation:

- **URL:** https://api.openai.com/v1/completions
- **Request Type:** POST
- **Headers:**
  - **Content-Type:** application/json
  - **Authorization:** Bearer <Your API key>
- **Body:**

```
{
"model": "gpt-3.5-turbo-instruct",
"prompt": "Write a limerick about APIs",
"max_tokens": 250,
"temperature": 0.7
}
```



Making the completion request

REQUEST TYPE: POST

URL: https://api.openai.com/v1/completions

https://api.openai.com/v1/completions

POST https://api.openai.com/v1/completions

Params | Authorization | Headers (10) | Body | Pre-request Scri

none | form-data | x-www-form-urlencoded | raw | bin

```
1 {
2   "model": "gpt-3.5-turbo-instruct",
3   "prompt": "Write a limmerick about APIs",
4   "max_tokens": 250,
5   "temperature": 0.7
6 }
```

BODY

```
{
"model": "gpt-3.5-turbo-instruct",
"prompt": "Write a limerick about APIs",
"max_tokens": 250,
"temperature": 0.7
}
```

OpenAI python package

- The openai python library offers simple framework to integrate OpenAI's services into applications, providing a simplified alternative to direct API use
- Offers synchronous and asynchronous responses, allowing building streaming chat applications
- Compared to using the REST API with requests, the Python library provides convenient access from any Python 3.7+ application
- Get started: `pip install openai`

OpenAI Python API library

pypi v1.25.0

The OpenAI Python library provides convenient access to the OpenAI REST API from any Python 3.7+ application. The library includes type definitions for all request params and response fields, and offers both synchronous and asynchronous clients powered by httpx.

It is generated from our OpenAPI specification with Stainless.

Documentation

The REST API documentation can be found on platform.openai.com. The full API of this library can be found in api.md.

Installation

> **Important**
>
> The SDK was rewritten in v1, which was released November 6th 2023. See the v1 migration guide, which includes scripts to automatically update your code.

github.com/openai/openai-python

Setting an OpenAI key environment variable

To use the openai python library, we must first set an environment variable with our API key:

WINDOWS

Open a Command prompt (not Powershell) and follow the steps below:

1. Set: **`setx OPENAI_API_KEY "<yourkey>"`**
2. Verify: **`echo %OPENAI_API_KEY%`**

This can also be done manually through "Environment Variables" in the Control Panel.

MAC

To set up your OpenAI API key on Windows, you can use the following steps:

1. Set: **`OPENAI_API_KEY=<yourkey>`**
2. Verify: **`echo $OPENAI_API_KEY`**

You can now use your API key in Python scripts or applications by accessing the environment variable OPENAI_API_KEY.

Hello GPT World

```python
import openai

import os
from openai import OpenAI

client = OpenAI(
  api_key=os.environ.get("OPENAI_API_KEY"),
)

chat_completion = client.chat.completions.create(
  messages=[
      {
        "role": "user",
        "content": "Write a poem about applesauce.",
      }
  ],
    model="gpt-3.5-turbo",
)
```

Oh luscious applesauce, so sweet and so smooth,
A comforting taste that soothes and improves.
Made from ripe apples, cooked down to a mash,
A versatile dish that creates a splash.

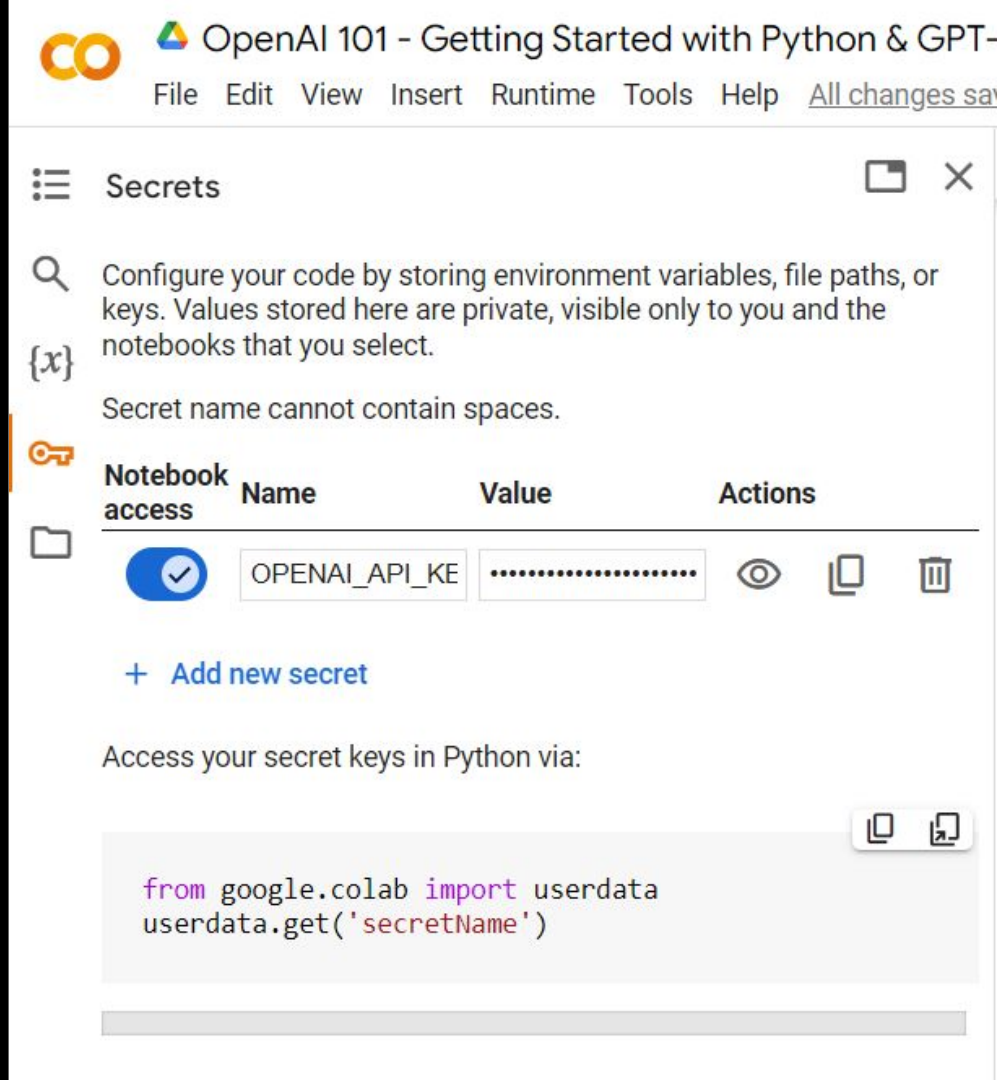
Served hot or cold, with cinnamon or plain,
Applesauce brings joy like a sweet refrain.
It pairs well with pork chops or on its own,
A delightful treat that's easily thrown.

Its smooth texture and tartness combine,
To make a dish that's simply divine.
A dollop on oatmeal, a swirl in yogurt,
Applesauce is truly a culinary effort.

So here's to applesauce, so humble yet grand,
A versatile treat made by hand.
Whether chunky or smooth, hot or cold,
Applesauce is a treasure to behold.

OpenAI API on Google Colab

Adding the API Key to Google Colab



Hello GPT World

```python
from openai import OpenAI
client = OpenAI()

completion = client.chat.completions.create(
  model="gpt-4o",
  messages=[
    {"role": "system", "content": "You are a poetic assistant, skilled in explaining complex programming concepts with creative flair."},
    {"role": "user", "content": "Compose a poem that explains the concept of recursion in programming."}
  ]
)

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

Step 3: Sending your first API request

Making an API request

After you have Python configured and set up an

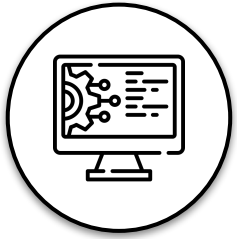
platform.openai.com/docs/quickstart/step-3-sending-your-first-api-request

Inside the file, copy and paste one of the examples below:

ChatCompletions

```python
from openai import OpenAI
client = OpenAI()

completion = client.chat.completions.cre
  model="gpt-3.5-turbo",
  messages=[
    {"role": "system", "content": "You a
```

Message Roles

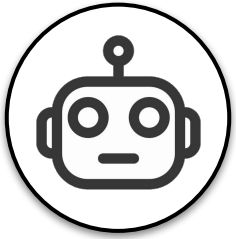


SYSTEM

Sets the behavior of the assistant - how it should behave at the conversation level (optional)

USER

Provide requests or input to which the assistant will respond (*i.e.* the prompts)



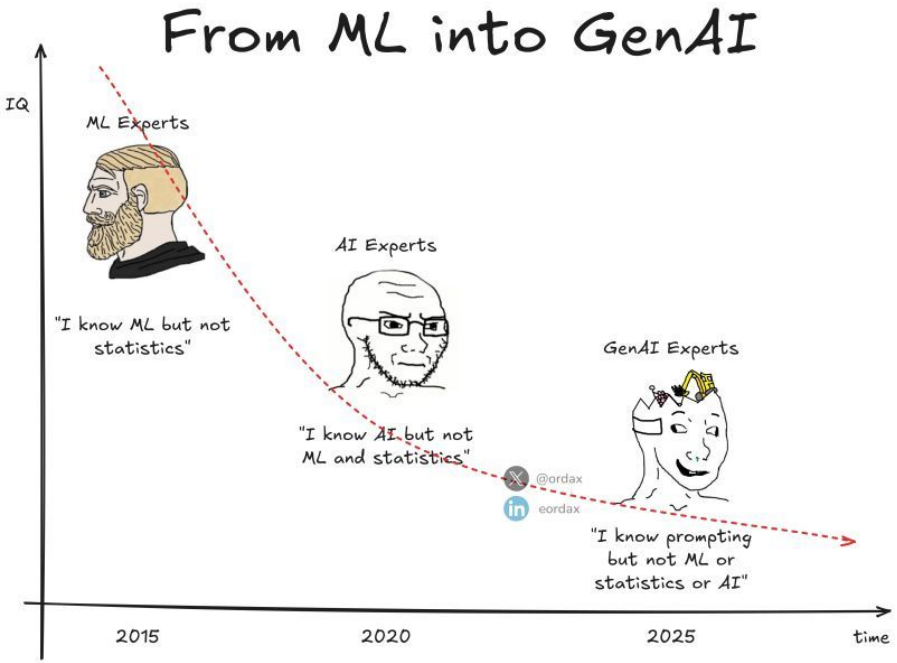
ASSISTANT

Responses from the model. Can be used to include conversation history when it is important (optional)

“Prompt Engineering” a.k.a writing

For generative AI language models, the model is given a **prompt** - a piece of input text, usually in the form of a question or instruction - to produce a desired output: a response of text in return, or something like an image or video in the case of “multimodal” modals.

Prompt engineering is the process of crafting and optimizing input prompts to effectively guide the behavior and outputs of AI models, particularly large language models (LLMs) like GPT. This practice involves designing clear, specific, and contextually relevant prompts to ensure the AI produces useful, accurate, and relevant responses - *i.e.* being articulate.



System prompts - Xena, Warrior Princess

```
completion = client.chat.completions.create(
  model="gpt-4o",
  messages=[
    {"role": "system", "content": "You are Xena, Warrior Princess. You will only answer in ALL CAPS, incorporate characters from the TV show in your replies, and end each reponse with 'I AM XENA, HEAR ME ROAR!'"},
    {"role": "user", "content": "Compose a poem that explains the concept of recursion in programming."}
  ]
)
```

platform.openai.com/docs/guides/text-generation/

End of Part 3

LLMsfor.me
PWYC Microcourse in LLMs and Generative AI
January 2025

Part 3 - GPT & the OpenAI Ecosystem

Monday, January 20th, 2025

llmsfor.me

NLP from scratch