

# 高阶大语言模型课程

Huajun Zeng

12/13/2024 - 1/31/2025  
(12月27日和1月3日放假, 共计6次课)  
每周五 5pm-7pm PT / 8pm-10pm ET

# 课程安排

Week	Date	Content	Week	Date	Content
1	2024-12-13	Retrieval Augmented Generation (RAG) for LLM <ul style="list-style-type: none"><li>• Why augmenting LLMs?</li><li>• Methods for LLM augmentation</li><li>• Augmenting LLMs with retrieval</li><li>• Augmenting LLMs with fine tuning</li></ul>	4	2025-01-17	Pipeline for LLM Applications: From Code to Products <ul style="list-style-type: none"><li>• Full stack LLM: tools needed for an LLM application</li><li>• Case study: build an LLM app from ground</li></ul>
2	2024-12-20	Chatbot Building with LLM APIs <ul style="list-style-type: none"><li>• Environment setup</li><li>• Introduction to LLM APIs</li><li>• Using chat completion APIs</li><li>• Using fine tuning APIs</li></ul>	5	2025-01-24	More LLM Applications and Course Project <ul style="list-style-type: none"><li>• Showcase of potential LLM applications for productivity, creativity and more</li><li>• More advanced LLM applications: AI Agent, Multi-modality, etc.</li><li>• Introduction to the course project: requirement and discussion</li></ul>
3	2025-01-10	Chatbot Building with LLM Frameworks and Vector Database <ul style="list-style-type: none"><li>• Introduction to Langchain and LlamaIndex</li><li>• Case study: a chatbot from Langchain and vector database</li></ul>	6	2025-01-31	Project Presentation <ul style="list-style-type: none"><li>• Student presentation on the course project</li></ul>

# 家庭作业回顾

- Register an OpenAI account and start using the console ([platform.openai.com](https://platform.openai.com))
  - Alternatives:
    - Anthropic Account: [console.anthropic.com](https://console.anthropic.com)
    - Kimi Account: [platform.moonshot.cn](https://platform.moonshot.cn)
- Register a Pinecone account and start experiencing the embeddings and index ([pinecone.io](https://pinecone.io))
- Any questions?

## 第二课: 使用LLM APIs建立Chatbot

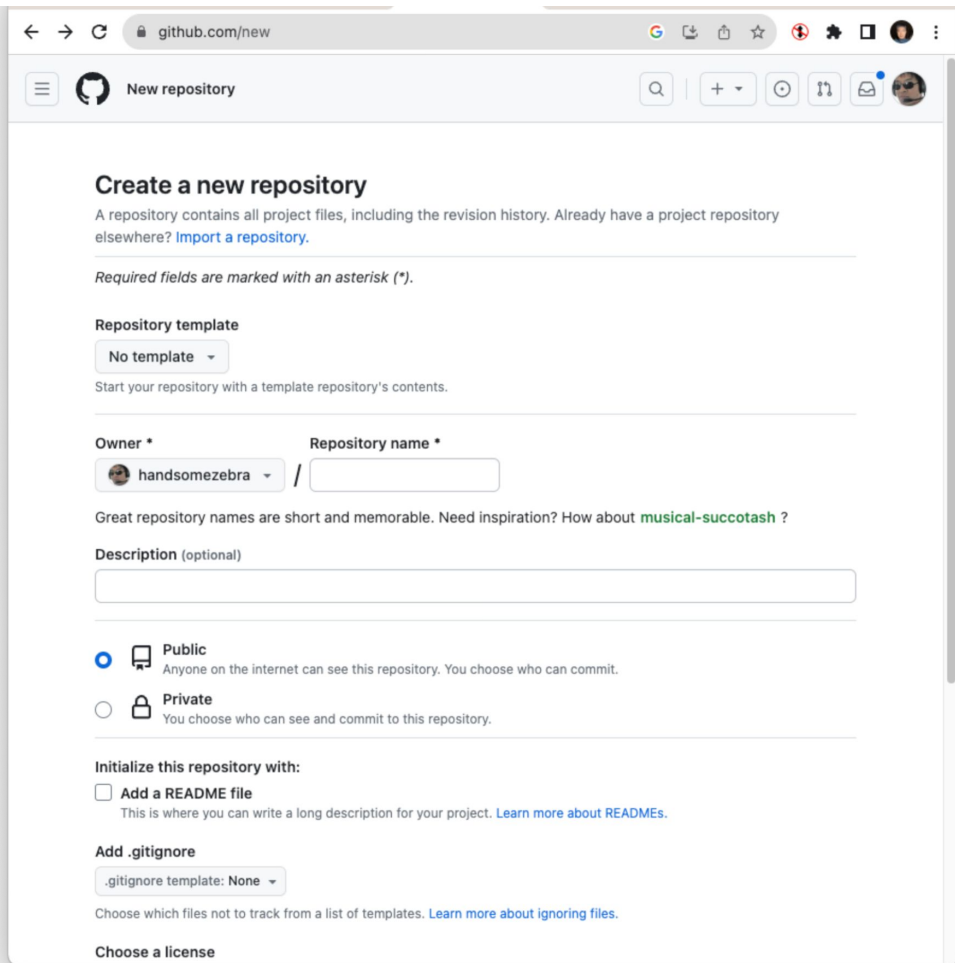
- Setup environment
- APIs Introduction
- Creating the first Chatbot

# 为什么要用API

- Customization
  - APIs provide greater flexibility in how LLMs are integrated and used. Developers can create custom applications tailored to specific use cases.
- Scalability and Performance
  - LLM APIs are designed to handle large volumes of requests simultaneously, making them highly scalable for business applications.
- Integration with Existing Systems
  - APIs allow for seamless integration of LLM capabilities into existing software ecosystems and workflows.

# 创建github代码库

1. Create github account
2. Create repo on github
3. Sync code to local by using **git clone**
4. Add and run code locally
5. Sync code to remote by
  - a. **git diff**
  - b. **git add**
  - c. **git commit**
  - d. **git push**



The screenshot shows the 'New repository' page on GitHub. The browser address bar shows 'github.com/new'. The page title is 'New repository'. Below the title, there's a search bar and a '+ v' button. The main heading is 'Create a new repository'. Below it, a paragraph explains that a repository contains all project files, including the revision history, and offers a link to 'Import a repository' if the user already has one elsewhere. A note states 'Required fields are marked with an asterisk (\*)'. The 'Repository template' section has a 'No template' dropdown. Below this, the 'Owner' is set to 'handsomezebra' and the 'Repository name' is an empty field. A tip suggests 'musical-succotash' as a name. The 'Description (optional)' field is empty. The 'Public' option is selected under 'Initialize this repository with:'. Below this, the 'Add a README file' checkbox is unchecked. The 'Add .gitignore' section has a '.gitignore template: None' dropdown. At the bottom, there's a 'Choose a license' section.

github.com/new

New repository

### Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository](#).

Required fields are marked with an asterisk (\*).

**Repository template**

No template

Start your repository with a template repository's contents.

**Owner \*** handsomezebra / **Repository name \***

Great repository names are short and memorable. Need inspiration? How about [musical-succotash](#) ?

**Description (optional)**

☐ **Public**  
Anyone on the internet can see this repository. You choose who can commit.

☐ **Private**  
You choose who can see and commit to this repository.

**Initialize this repository with:**

☐ **Add a README file**  
This is where you can write a long description for your project. [Learn more about READMEs](#).

**Add .gitignore**

.gitignore template: None

Choose which files not to track from a list of templates. [Learn more about ignoring files](#).

**Choose a license**

# 创建Python环境

- Isolating Project Dependencies
  - Python environments allow you to isolate your project dependencies in a dedicated place
- Ease of Sharing and Deployment of Code
  - It will help you remember the required packages and their versions. Usually using a requirements.txt file.

```
# Download and install Anaconda...  
$ conda create -n chatbot_env python=3.11  
$ conda activate chatbot_env  
$ pip install openai  
# Create requirements.txt and add installed packages
```

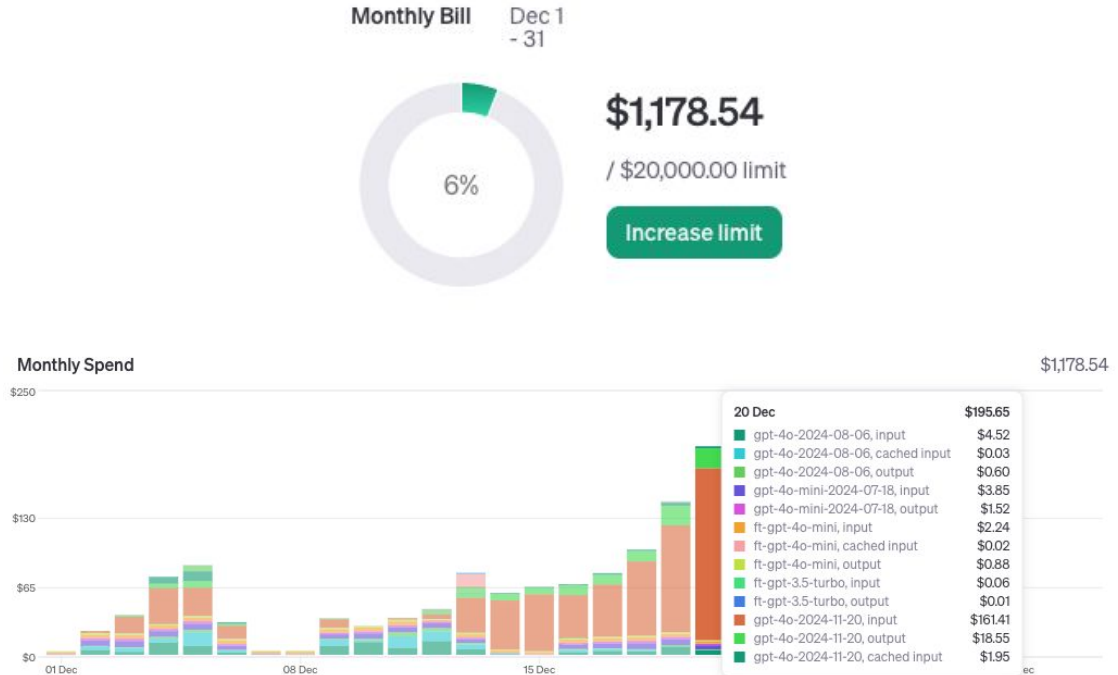
# OpenAI SDK

- <https://github.com/openai/openai-python>
- Benefit
  - Simplified code: pre-written functions and methods tailored to interact with the API, abstracting away the complexity
  - Error handling: built-in error handling and retries, which can save developers time in writing boilerplate code to manage common network issues
  - Type safety: SDKs can offer type definitions for the data sent to and received from the API



# OpenAI API Pricing and Usage

- **gpt-4o**: \$2.50 / 1M input tokens; \$10.00 / 1M output tokens
- **gpt-4o-mini**: \$0.15 / 1M input tokens; \$0.60 / 1M output tokens
- Note: monitor your usage periodically



# Token简介

One token generally corresponds to ~4 characters for English text.

This translates to roughly  $\frac{3}{4}$  of a word (so 100 tokens  $\approx$  75 words).

For Chinese, one token is about 1.5 characters

GPT-4o &amp; GPT-4o mini

GPT-3.5 &amp; GPT-4

GPT-3 (Legacy)

You are a virtual assistant create by Huajun. Today is Dec. 18, 2024. You provide responses to questions that are clear, straightforward, and factually accurate, without speculation or falsehood. Given the following context, please answer each question truthfully to the best of your abilities based on the provided information. Answer each question with a brief summary followed by several bullet points.

Clear

Show example

Tokens

81

Characters

406

You are a virtual assistant create by Huajun. Today is Dec. 18, 2024. You provide responses to questions that are clear, straightforward, and factually accurate, without speculation or falsehood. Given the following context, please answer each question truthfully to the best of your abilities based on the provided information. Answer each question with a brief summary followed by several bullet points.

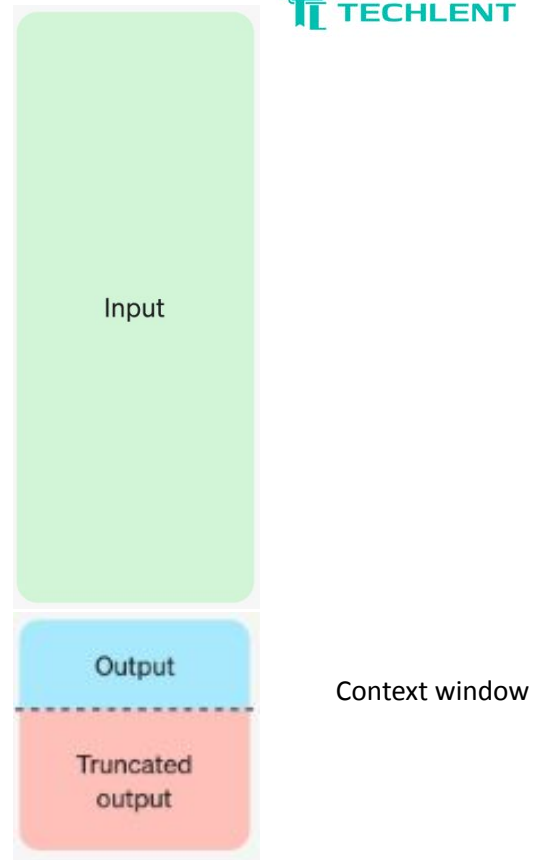
Text

Token IDs

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

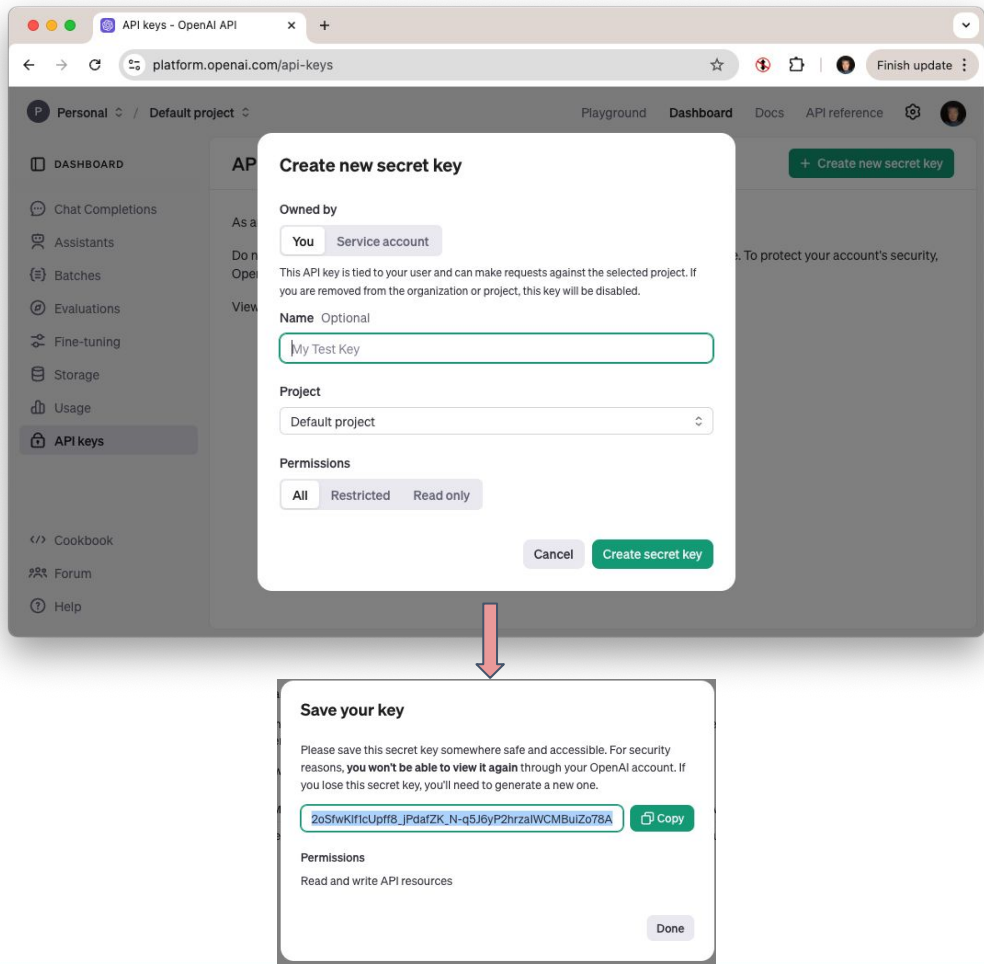
# 上下文窗口

- Context window: maximum number of tokens that can be used in a single request, inclusive of both input and output.
- Context windows of typical LLMs
  - GPT 4o: 128k tokens (roughly 380 pages)
  - Claude 3.5: 200k tokens (roughly 600 pages)



# OpenAI Key

- Make sure you have some credit in the OpenAI account
  - You will be billed by # of tokens
- Create a key, save it into a local text file
  - You won't be able to see the full key on the website any more



# Key管理

- Keep the key safe (similar to how you protect your password)
  - Don't put the key in code
  - Don't send it online
- How to use the keys in your software
  - Use export to set the key in the environment

```
% export OPENAI_API_KEY=sk-8nAAAABBBB111122224555IIIIUUUUEEEEkkkkkDDDD1111QQQQPPPPP
```

- Add OPENAI\_API\_KEY=\*\*\* to your .env file, use python-dotenv package to load the .env file, and don't add .env file to source control
- Use Secrets Management Tools e.g. AWS Secrete Manager

# 数据隐私

- Data sent to the OpenAI API will not be used for model training unless users opt-in.
- API data might be stored for up to 30 days to detect abuse
- API data can be deleted sooner with zero data retention options available for trusted customers with sensitive applications.

# Model List API

- `test_model_list.py`
- This API shows all available models in the system
- Models
  - Chat
  - Image
  - Audio
  - Embedding
  - ...

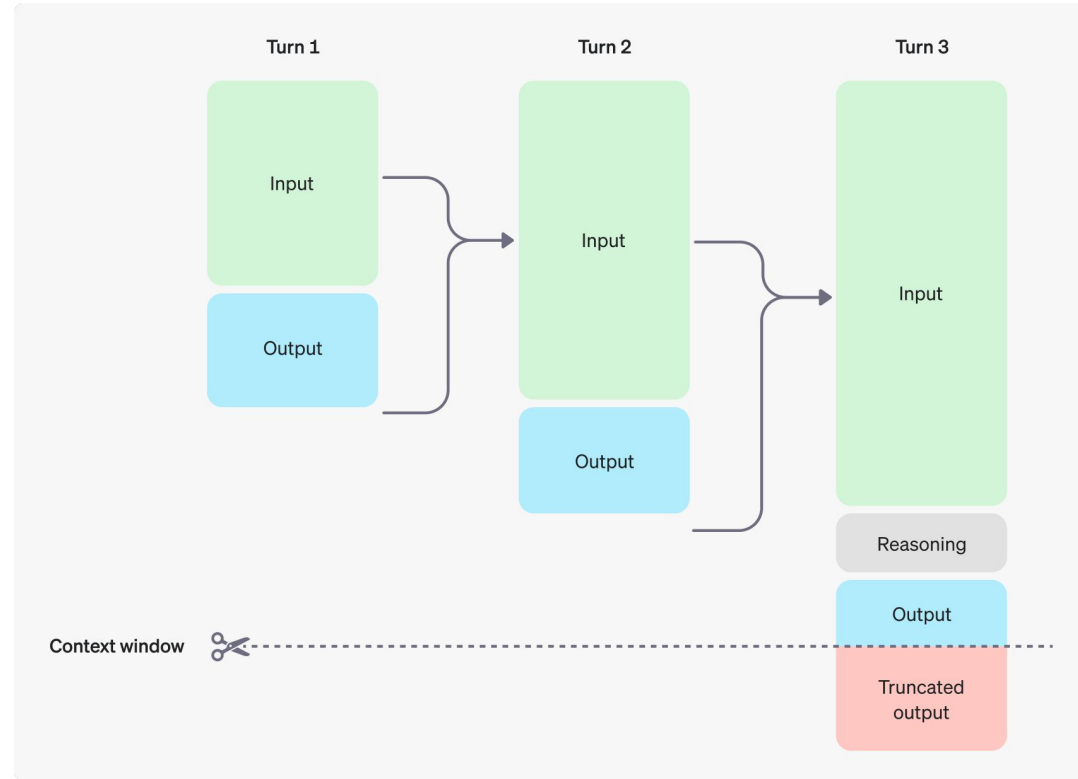
# Chat API

- Given a list of messages comprising a conversation, the model will return a response.
- System message
  - The conversation typically starts with a system message that helps set the assistant's behavior.
  - The system message is optional - the assistant will behave similarly without it.
- User messages:
  - Provide requests or comments for the assistant to respond to.
- Assistant messages:
  - Storing previous responses



# Chat API

- Models have no memory, so all relevant context must be included in each request.



# Chat API代码

test\_chat\_simple.py

test\_chat\_multi\_turn.py

# Chat API的参数

**messages** array **Required**

A list of messages comprising the conversation so far. Depending on the **model** you use, different message types (modalities) are supported, like **text**, **images**, and **audio**.

▼ Show possible types

**model** string **Required**

ID of the model to use. See the **model endpoint compatibility** table for details on which models work with the Chat API.

**max\_completion\_tokens** integer or null **Optional**

An upper bound for the number of tokens that can be generated for a completion, including visible output tokens and **reasoning tokens**.

**response\_format** object **Optional**

An object specifying the format that the model must output.

Setting to `{ "type": "json_schema", "json_schema": {...} }` enables Structured Outputs which ensures the model will match your supplied JSON schema. Learn more in the **Structured Outputs guide**.

Setting to `{ "type": "json_object" }` enables JSON mode, which ensures the message the model generates is valid JSON.

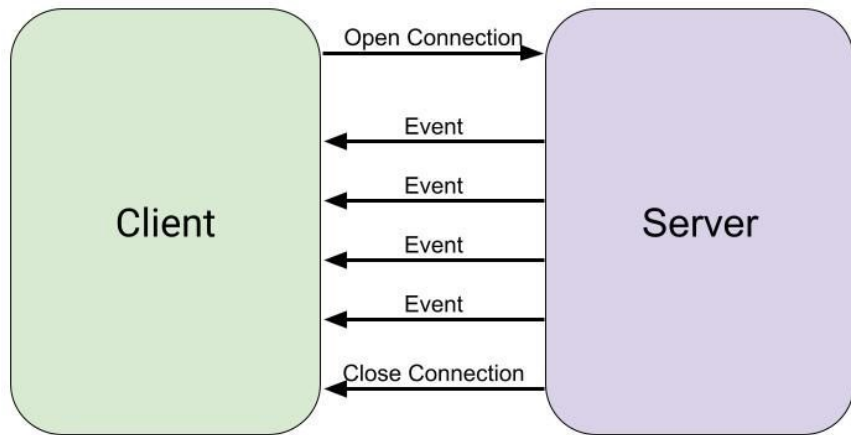
# Chat API结果

- Response is in object format.
- You can easily convert them to a dictionary by using `to_dict()`.

```
ChatCompletion(  
  id='chatcmpl-AgfYl150HwwhCgXTgaSDeGJMJIgP0',  
  choices=[  
    Choice(finish_reason='stop', index=0, logprobs=None, message=  
      ChatCompletionMessage(  
        content="In the land of code and looping rhyme...",  
        refusal=None,  
        role='assistant',  
        audio=None,  
        function_call=None,  
        tool_calls=None))  
  ],  
  created=1734732895,  
  model='gpt-4o-2024-08-06',  
  object='chat.completion',  
  service_tier=None,  
  system_fingerprint='fp_5f20662549',  
  usage=CompletionUsage(  
    completion_tokens=289,  
    prompt_tokens=25,  
    total_tokens=314,  
    completion_tokens_details=CompletionTokensDetails(  
      accepted_prediction_tokens=0,  
      audio_tokens=0,  
      reasoning_tokens=0,  
      rejected_prediction_tokens=0  
    ),  
    prompt_tokens_details=PromptTokensDetails(audio_tokens=0,  
    cached_tokens=0)  
  )  
)
```

# 流式输出

- Reason
  - Large language models are slow in generating responses
  - The predict-next-token mechanism enables the returning partial results
  - Perceived faster and nicer user experience
- Server Sent Event (SSE)
  - Single long HTTP connection
  - Fast and lightweight

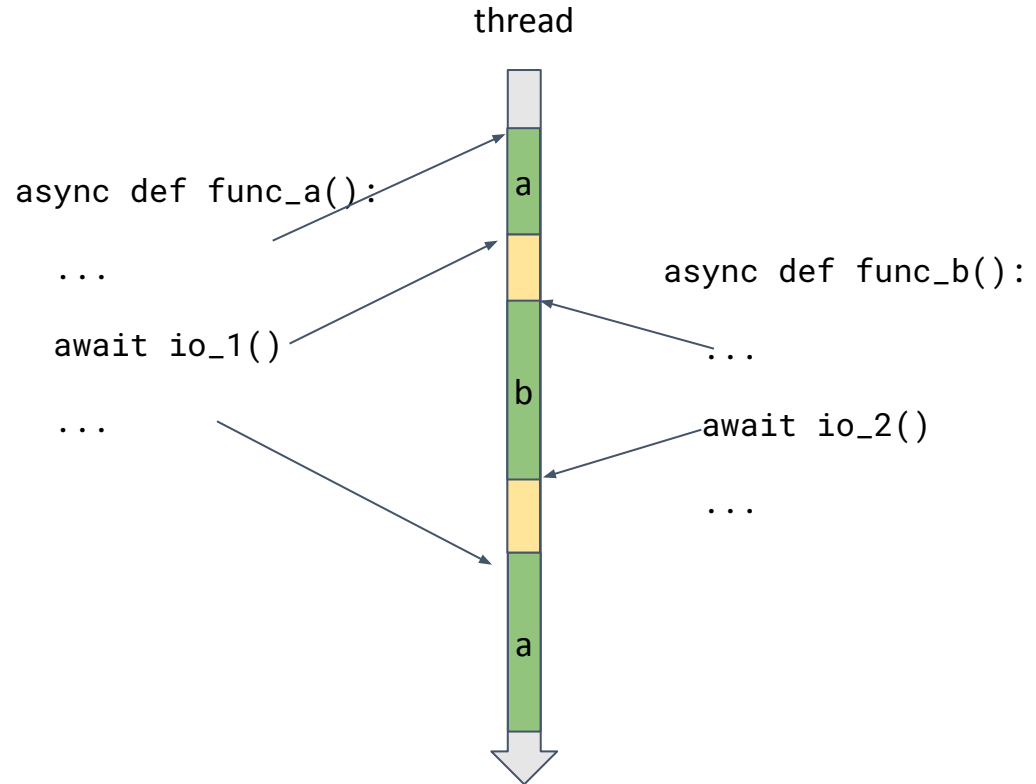


# 流式输出代码

test\_chat\_streaming.py

# Python异步编程

- **coroutine**: A function that can suspend its execution and yield control back to its caller without blocking.
- **async/await**: Python keywords used to define coroutines and call them for suspension and resumption.
- Good for slow network I/O operations.
- May not be good for computation intensive tasks



# 异步编程代码

test\_chat\_async\_streaming.py



# 创建提示词

- Raw string
  - Problem: Not modularized
- String concatenation
  - `prompt = instruction1 + role + instruction2 + "Context:" + context + "Examples" + examples`
    - Problem: Not easy to see the overall structure
- String template

# 使用字符串模版来创建提示词

- String with variables
- Using string.format function to replace variables with actual values

```
You are {name}, a virtual  
assistant create by  
{owner}. Today is {date}.  
Please answer each  
question truthfully.
```

# 基于模版创建提示词

test\_chat \_with\_context.py

# Tokenizer

- Needed if you want to calculate the number of tokens before sending the prompt to LLMs

```
import tiktoken

enc = tiktoken.encoding_for_model("gpt-4o")

def num_tokens(string):
    num_tokens = len(enc.encode(string))
    return num_tokens
```

# Embedding API

test\_embedding.py

# 持续运行的Chatbot

test\_chat\_continuous.py

# 其他OpenAI APIs

- Function call API
- Moderation API
- Fine-tuning API
- Audio API
- Images API
- ...

# Anthropic API

```
import anthropic

client = anthropic.Anthropic()

message = client.messages.create(
    model="claude-3-5-sonnet-20241022",
    max_tokens=1000,
    temperature=0,
    system="You are a helpful assistant.",
    messages=[
        {
            "role": "user",
            "content": [{"type": "text", "text": "Write a poem about recursion in
programming."}]
        }
    ]
)

print(message.content)
```



# 月之暗面API

```
from openai import OpenAI

client = OpenAI(
    api_key = "$MOONSHOT_API_KEY",
    base_url = "https://api.moonshot.cn/v1",
)

completion = client.chat.completions.create(
    model = "moonshot-v1-8k",
    messages = [
        {"role": "system", "content": "你是 Kimi, 由 Moonshot AI 提供的人工智能助手, 你更擅长中文和英文的对话。你会为用户提供安全, 有帮助, 准确的回答。"},
        {"role": "user", "content": "你好, 我叫李雷, 1+1等于多少?"}
    ],
    temperature = 0.3,
)

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

# 家庭作业

- Using your private domain data (for example, a set of research papers, a set of financial report documents, ...)
- Write a python program to answer questions about your data
  - Receive question from console and generate responses in streaming way
  - Using OpenAI SDK (or Anthropic)
- Refer to the examples in [https://github.com/hzeng-otterai/chatbot-example/tree/main/backend\\_api](https://github.com/hzeng-otterai/chatbot-example/tree/main/backend_api)

# Questions?