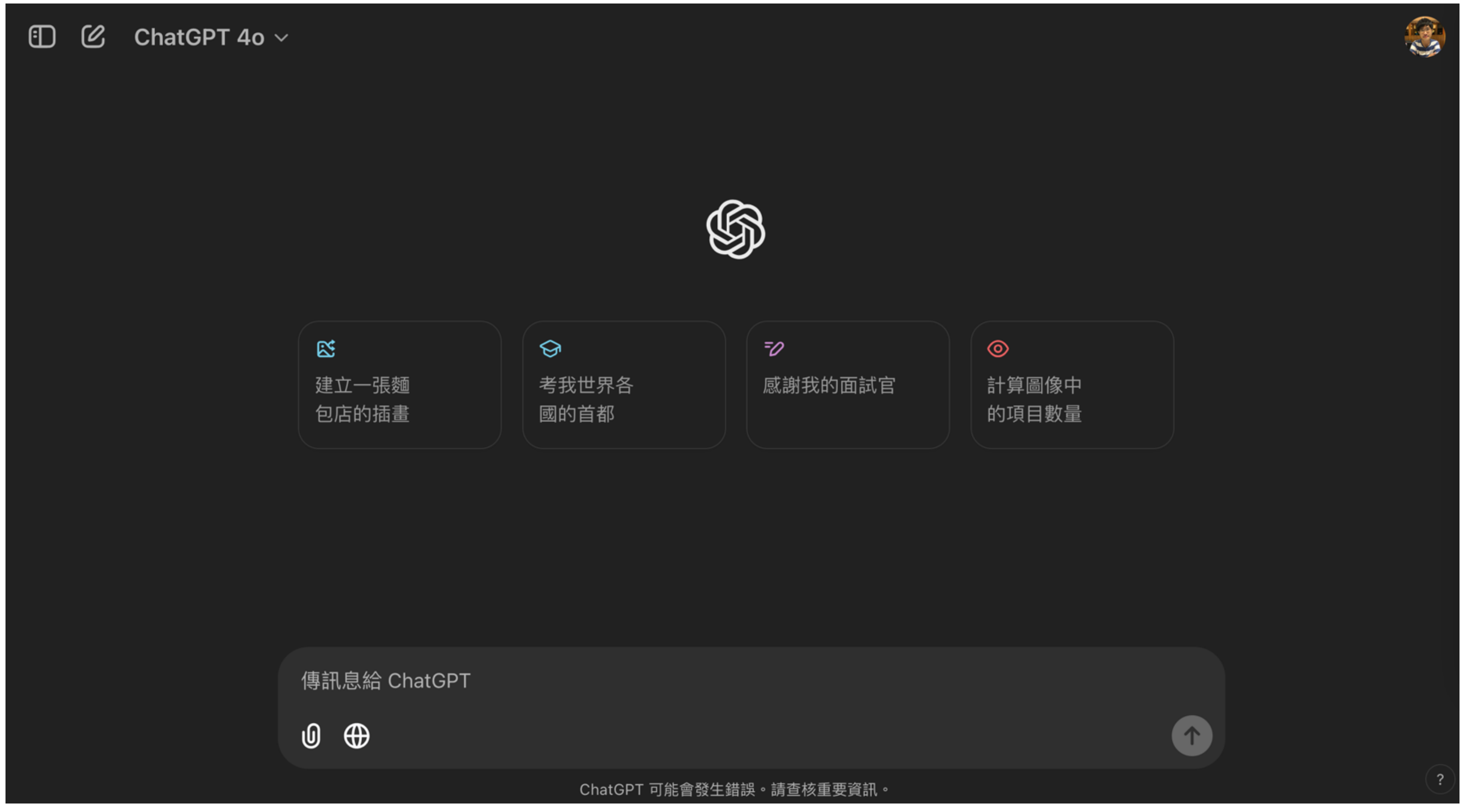
Natural Language Processing

LLM API Tutorial 2024/11/21

What you will learn in this tutorial

- Gemini API -> **Google** Gemini
- OpenAI API -> **OpenAI** ChatGPT
- Claude API - > **Anthropic** Claude
- Code (notebook) Link
  - https://github.com/IKMLab/NTHU_Natural_Language_Processing/tree/main/Reference/LLM_API_lab

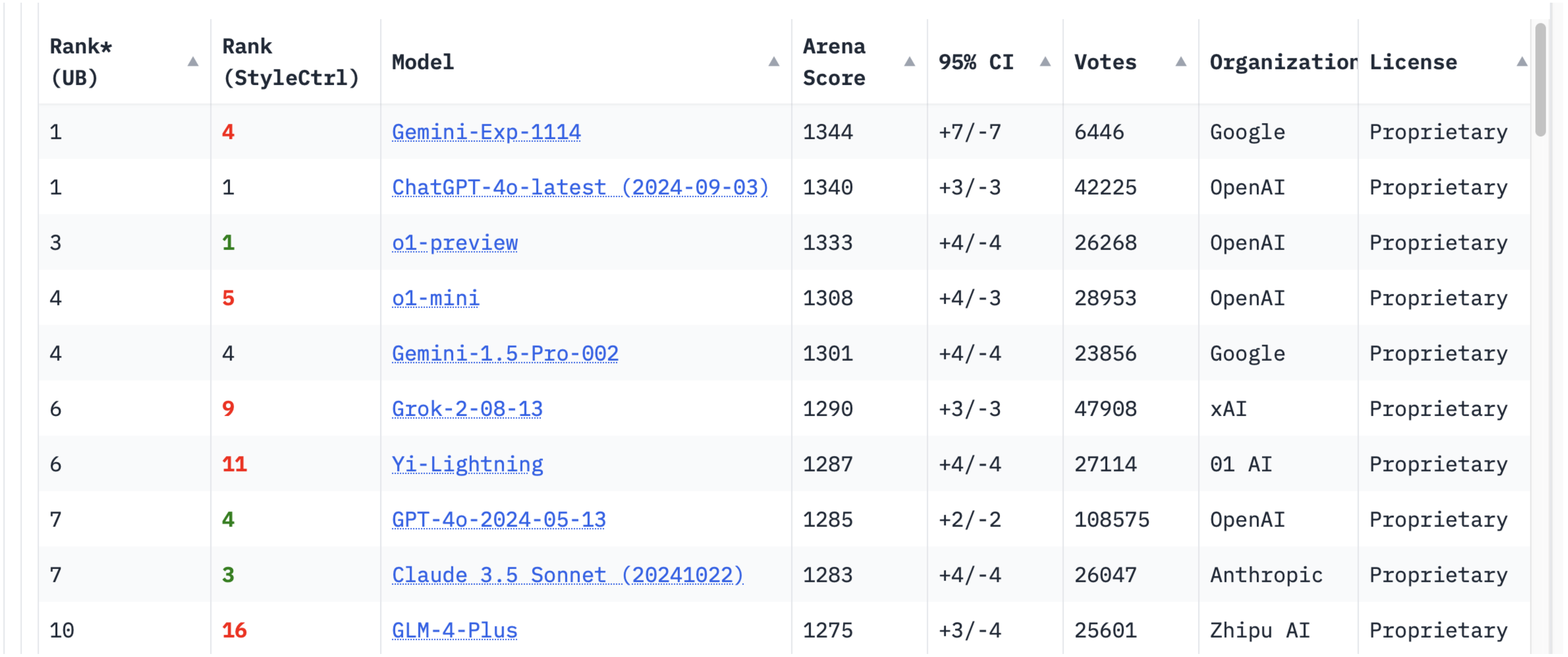
Why do we need to use API?



- It’s slow if you want to use ChatGPT for NLP tasks by manually copying and pasting from the webpage.
- If you use ChatGPT from the webpage to test data, “Too many requests in 1 hour. Try again later.” may occur.

How to choose a good LLM?

Chatbot Arena LLM Leaderboard: https://lmarena.ai/?leaderboard



Rank* (UB)	Rank (StyleCtrl)	Model	Arena Score	95% CI	Votes	Organization	License
1	4	Gemini-Exp-1114	1344	+7/-7	6446	Google	Proprietary
1	1	ChatGPT-4o-latest (2024-09-03)	1340	+3/-3	42225	OpenAI	Proprietary
3	1	o1-preview	1333	+4/-4	26268	OpenAI	Proprietary
4	5	o1-mini	1308	+4/-3	28953	OpenAI	Proprietary
4	4	Gemini-1.5-Pro-002	1301	+4/-4	23856	Google	Proprietary
6	9	Grok-2-08-13	1290	+3/-3	47908	xAI	Proprietary
6	11	Yi-Lightning	1287	+4/-4	27114	01 AI	Proprietary
7	4	GPT-4o-2024-05-13	1285	+2/-2	108575	OpenAI	Proprietary
7	3	Claude 3.5 Sonnet (20241022)	1283	+4/-4	26047	Anthropic	Proprietary
10	16	GLM-4-Plus	1275	+3/-4	25601	Zhipu AI	Proprietary

API Fee

	Gemini (Gemini-1.5-pro)	OpenAI (gpt-4o)	Claude (Claude 3.5 Sonnet)	Hugging Face
Free quota	• 2 RPM (requests per minute) • 32,000 TPM (tokens per minute) • 50 RPD (requests per day)	No	No	FREE
Price Page Link	Link	Link	Link	-
Input token Price (Every 1M tokens)	1.25 2.50 (longer than 128k tokens)	2.50	3	-
Prompt Caching	0.3125 0.625 (longer than 128k tokens) 4.5 per hour for life	1.25*	3.75 (write) 0.30 (read)*	-
Output token Price (Every 1M tokens)	5.00 10.00 (longer than 128k tokens)	10.00	15	-

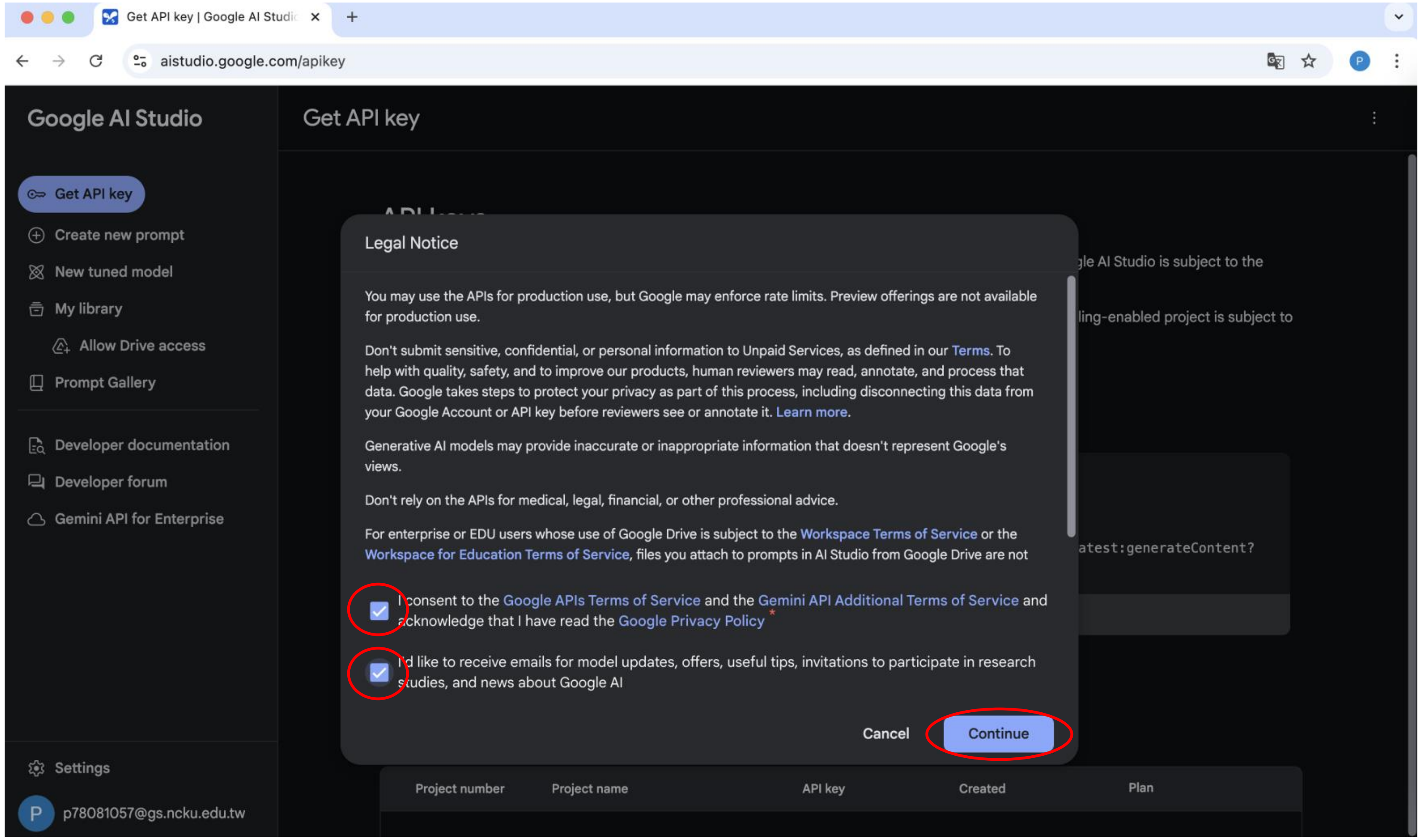
*life: 5 minutes of inactivity (automatic)

Get Gemini API Key (1)

Go to https://aistudio.google.com/apikey

And Log in your Google account

Get Gemini API Key (2)



Get API key | Google AI Studio

aistudio.google.com/apikey

Google AI Studio

Get API key

Legal Notice

You may use the APIs for production use, but Google may enforce rate limits. Preview offerings are not available for production use.

Don't submit sensitive, confidential, or personal information to Unpaid Services, as defined in our Terms. To help with quality, safety, and to improve our products, human reviewers may read, annotate, and process that data. Google takes steps to protect your privacy as part of this process, including disconnecting this data from your Google Account or API key before reviewers see or annotate it. Learn more.

Generative AI models may provide inaccurate or inappropriate information that doesn't represent Google's views.

Don't rely on the APIs for medical, legal, financial, or other professional advice.

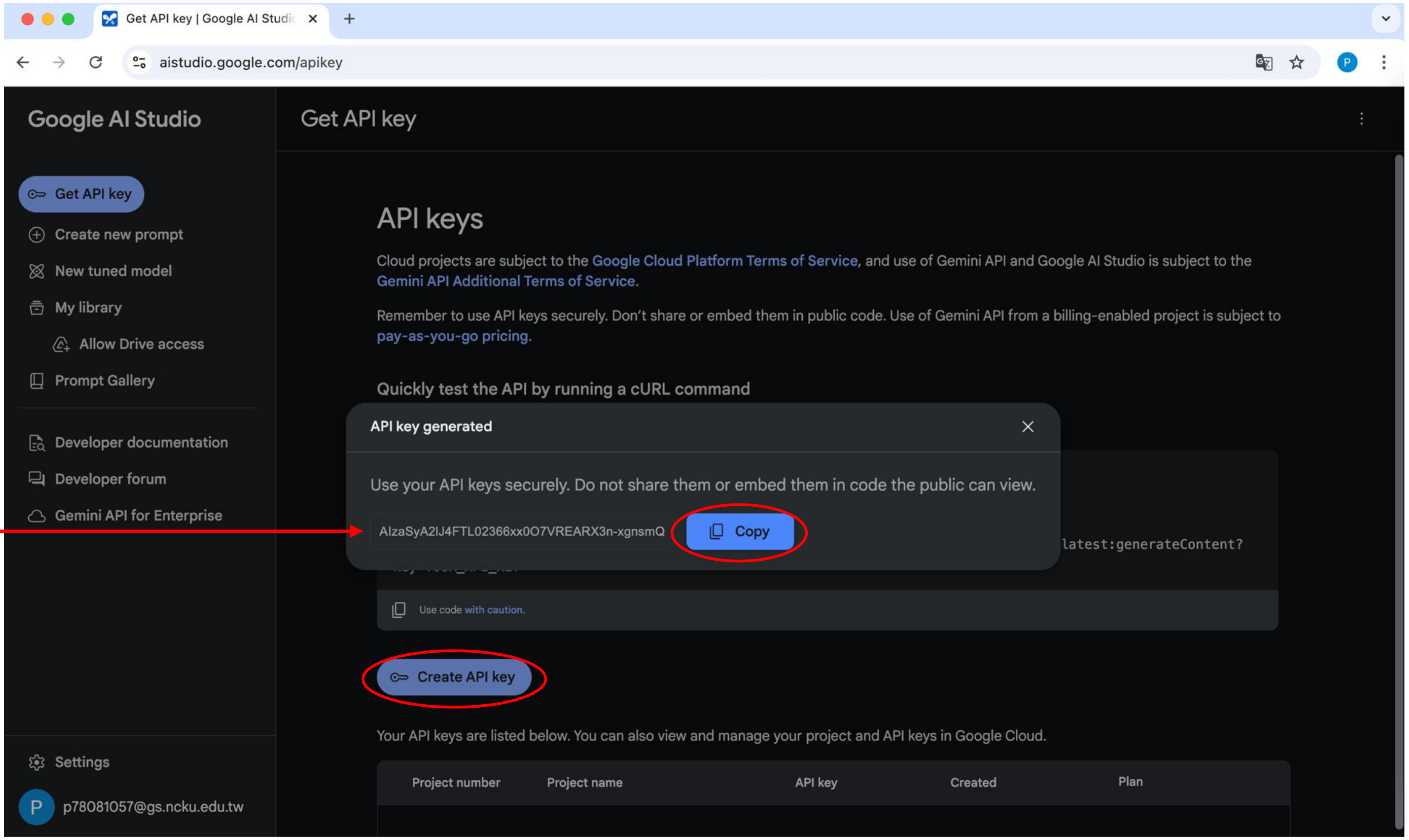
For enterprise or EDU users whose use of Google Drive is subject to the Workspace Terms of Service or the Workspace for Education Terms of Service, files you attach to prompts in AI Studio from Google Drive are not

☑ I consent to the Google APIs Terms of Service and the Gemini API Additional Terms of Service and acknowledge that I have read the Google Privacy Policy *

☑ I'd like to receive emails for model updates, offers, useful tips, invitations to participate in research studies, and news about Google AI

Project number	Project name	API key	Created	Plan

Get Gemini API Key (3)


Get API key | Google AI Studio

aistudio.google.com/apikey

Google AI Studio

Get API key

Get API key

Create new prompt

New tuned model

My library

Allow Drive access

Prompt Gallery

Developer documentation

Developer forum

Gemini API for Enterprise

Settings

p78081057@gs.ncku.edu.tw

API keys

Cloud projects are subject to the Google Cloud Platform Terms of Service, and use of Gemini API and Google AI Studio is subject to the Gemini API Additional Terms of Service.

Remember to use API keys securely. Don't share or embed them in public code. Use of Gemini API from a billing-enabled project is subject to pay-as-you-go pricing.

Quickly test the API by running a cURL command

API key generated

Use your API keys securely. Do not share them or embed them in code the public can view.

AIzaSyA2lJ4FTL02366xx0O7VREARX3n-xgnsmQ

Copy

```
latest:generateContent?
```

Use code with caution.

Create API key

Your API keys are listed below. You can also view and manage your project and API keys in Google Cloud.

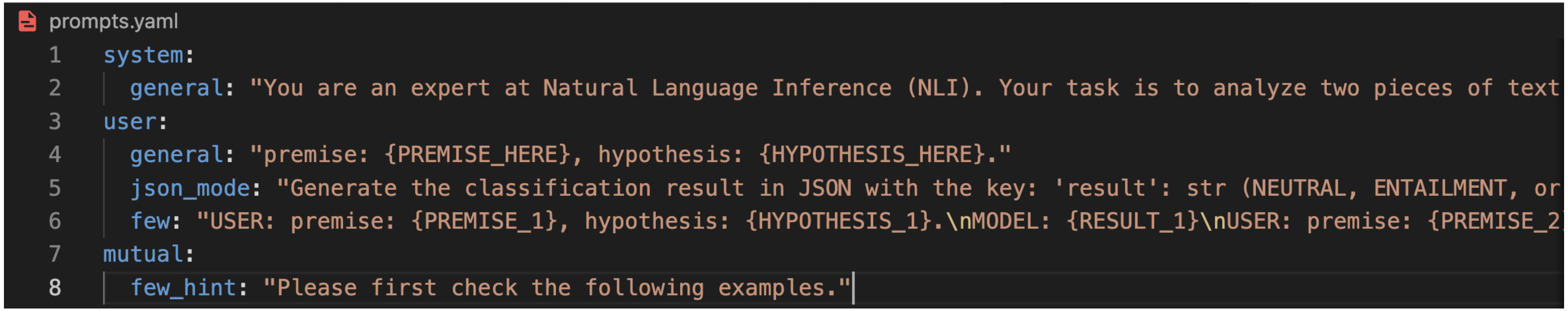
Project number	Project name	API key	Created	Plan

Your
API Key

Installation

```
!pip install google-ai-generativelanguage==0.8.3
!pip install anthropic==0.39.0
!pip install openai==1.54.5
```

The file for Prompts (prompts.yaml)



prompts.yaml

```yaml
system:
  general: "You are an expert at Natural Language Inference (NLI). Your task is to analyze two pieces of text
user:
  general: "premise: {PREMISE_HERE}, hypothesis: {HYPOTHESIS_HERE}."
  json_mode: "Generate the classification result in JSON with the key: 'result': str (NEUTRAL, ENTAILMENT, or
  few: "USER: premise: {PREMISE_1}, hypothesis: {HYPOTHESIS_1}.\nMODEL: {RESULT_1}\nUSER: premise: {PREMISE_2
mutual:
  few_hint: "Please first check the following examples."
```

System Prompt and User Prompt

System Prompt

User Prompt

Role Playing (Persona)

You are an expert at Natural Language Inference (NLI).

Task descriptions

Your task is to analyze two pieces of text - a premise and a hypothesis - and classify them into NEUTRAL, ENTAILMENT, or CONTRADICTION.

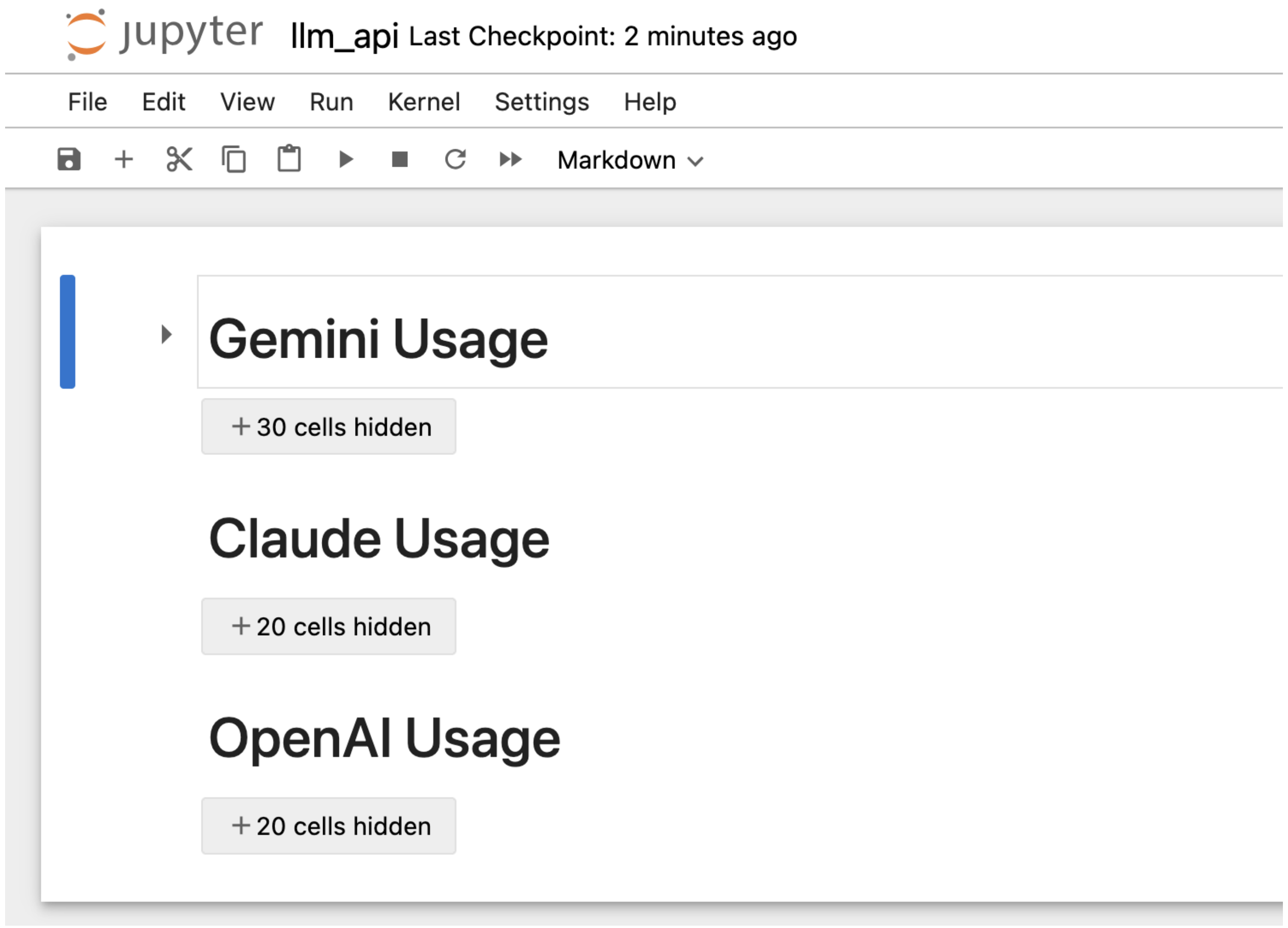
Examples

premise: {PREMISE_HERE},
hypothesis: {HYPOTHESIS_HERE}.

https://platform.openai.com/docs/guides/prompt-engineering#tactics

Notebook Executions

Each of the three parts can be executed individually.



Gemini Usage

+ 30 cells hidden

Claude Usage

+ 20 cells hidden

OpenAI Usage

+ 20 cells hidden

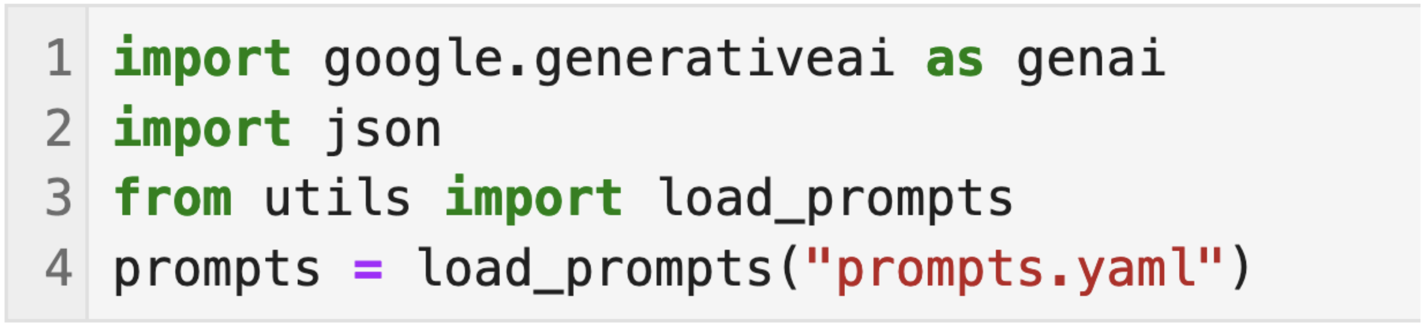
Example for Classification

- Dataset: SemEval 2014 Task1-2 (3-class classification)

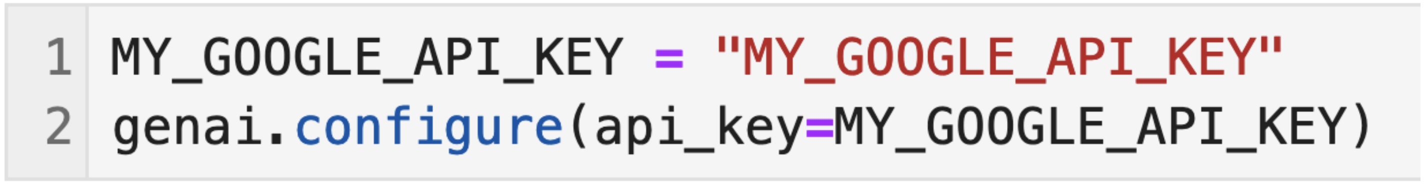
```
inputs = {
    "premise": "A group of kids is playing in a yard and an old man
is standing in the background",
    "hypothesis": "A group of boys in a yard is playing and a man is
standing in the background",
}
```

Setup (Gemini)

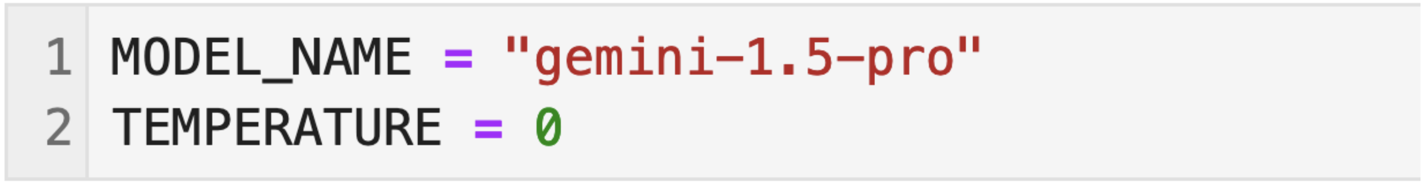
https://ai.google.dev/gemini-api/docs/models/gemini?hl=zh-tw#gemini-1.5-pro



```python
import google.generativeai as genai
import json
from utils import load_prompts
prompts = load_prompts("prompts.yaml")
```



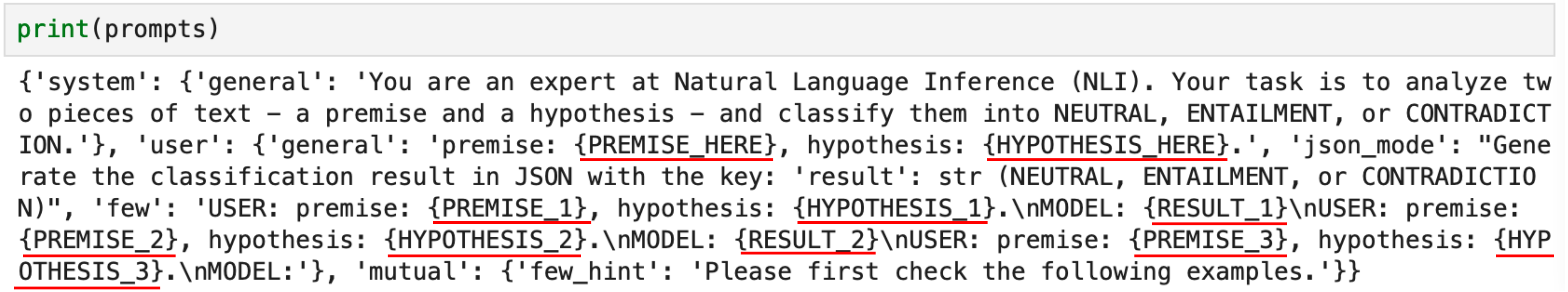
```python
MY_GOOGLE_API_KEY = "MY_GOOGLE_API_KEY"
genai.configure(api_key=MY_GOOGLE_API_KEY)
```



```python
MODEL_NAME = "gemini-1.5-pro"
TEMPERATURE = 0
```

The file for Prompts (prompts.yaml)

We use the yaml package to load the "prompts.yaml" into a Python dictionary.



```
print(prompts)
```

```
{'system': {'general': 'You are an expert at Natural Language Inference (NLI). Your task is to analyze tw
o pieces of text - a premise and a hypothesis - and classify them into NEUTRAL, ENTAILMENT, or CONTRADICT
ION.'}, 'user': {'general': 'premise: {PREMISE_HERE}, hypothesis: {HYPOTHESIS_HERE}.', 'json_mode': "Gene
rate the classification result in JSON with the key: 'result': str (NEUTRAL, ENTAILMENT, or CONTRADICTIO
N)", 'few': 'USER: premise: {PREMISE_1}, hypothesis: {HYPOTHESIS_1}.\nMODEL: {RESULT_1}\nUSER: premise:
{PREMISE_2}, hypothesis: {HYPOTHESIS_2}.\nMODEL: {RESULT_2}\nUSER: premise: {PREMISE_3}, hypothesis: {HYP
OTHESIS_3}.\nMODEL:'}, 'mutual': {'few_hint': 'Please first check the following examples.'}}
```

Basic Usage (Gemini)

```python
model = genai.GenerativeModel(
    MODEL_NAME,
    generation_config={"temperature": TEMPERATURE},
    system_instruction=system_prompt,
)
```

Set up the Gemini model

```python
cur_user_prompt = user_prompt.format(
    PREMISE_HERE=inputs["premise"],
    HYPOTHESIS_HERE=inputs["hypothesis"]
)
print(cur_user_prompt)
```

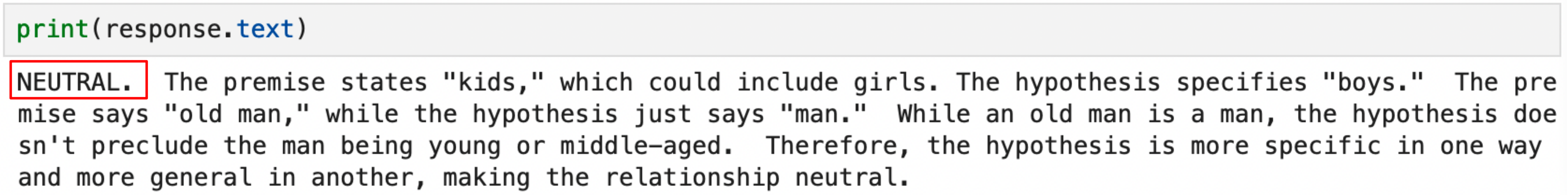
Format the inputs

```
premise: A group of kids is playing in a yard and an old man is standing in the background, hypothesis: A group of b
oys in a yard is playing and a man is standing in the background.
```

```python
response = model.generate_content(cur_user_prompt)
```

Run generation

Output of Classification (Gemini)



```python
print(response.text)
```

```
NEUTRAL.  The premise states "kids," which could include girls. The hypothesis specifies "boys."  The pre
mise says "old man," while the hypothesis just says "man."  While an old man is a man, the hypothesis doe
sn't preclude the man being young or middle-aged.  Therefore, the hypothesis is more specific in one way
and more general in another, making the relationship neutral.
```

Count tokens (Gemini)

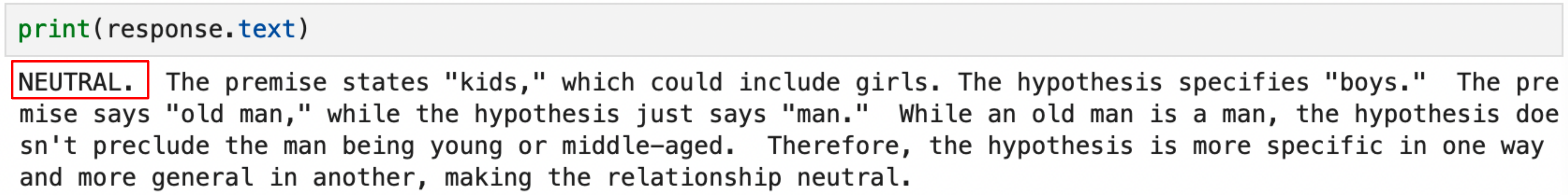
```python
num_sys_tokens = str(model.count_tokens(system_prompt)).split(": ")[1].strip()
num_usr_tokens = str(model.count_tokens(system_prompt)).split(": ")[1].strip()
num_input_tokens = int(num_sys_tokens) + int(num_usr_tokens)
print(f"Num of input tokens: {num_input_tokens}")
```

```
Num of input tokens: 176
```

```python
num_output_tokens = str(model.count_tokens(response.text)).split(": ")[1].strip()
print(f"Num of output tokens {num_output_tokens}")
```

```
Num of output tokens 128
```

Output of Classification (Gemini)

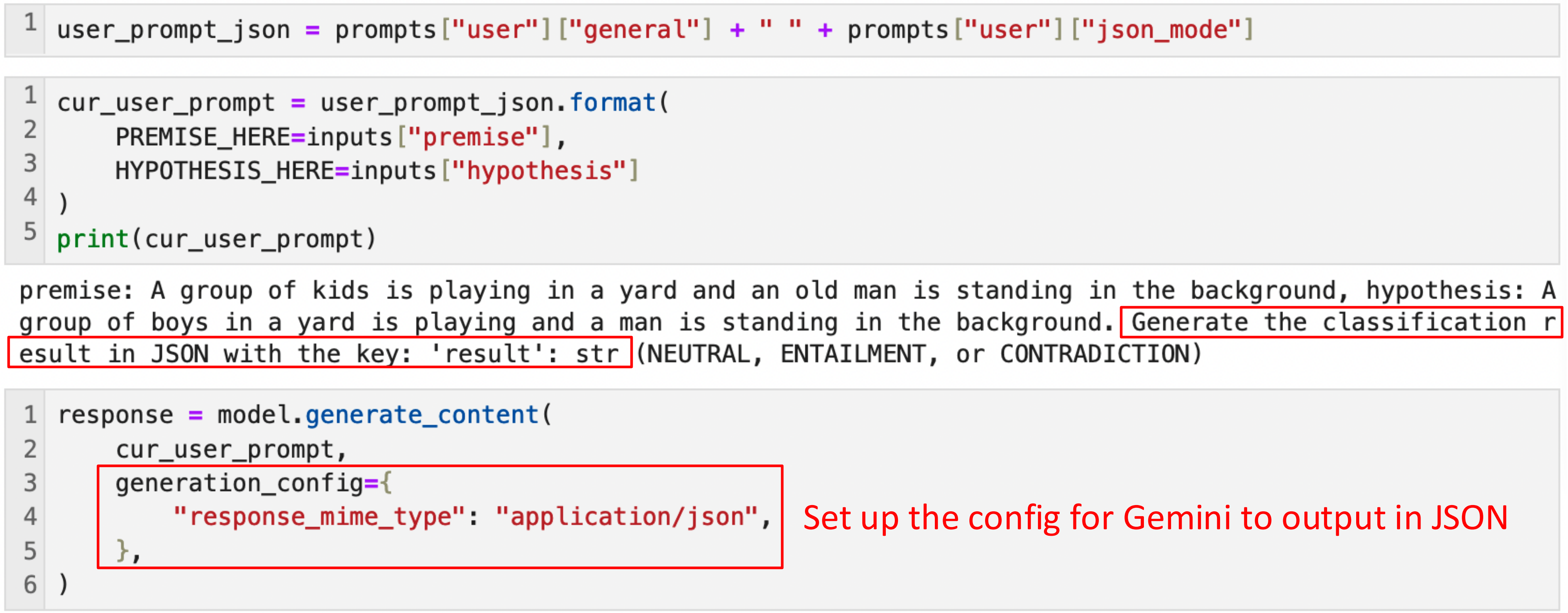


```
print(response.text)
```

```
NEUTRAL.  The premise states "kids," which could include girls. The hypothesis specifies "boys."  The pre
mise says "old man," while the hypothesis just says "man."  While an old man is a man, the hypothesis doe
sn't preclude the man being young or middle-aged.  Therefore, the hypothesis is more specific in one way
and more general in another, making the relationship neutral.
```

How to get structured output for evaluating model performance?
JSON mode -> {"result": "NEUTRAL"}

Generate structured output in JSON (Gemini)



```python
user_prompt_json = prompts["user"]["general"] + " " + prompts["user"]["json_mode"]
```

```python
cur_user_prompt = user_prompt_json.format(
    PREMISE_HERE=inputs["premise"],
    HYPOTHESIS_HERE=inputs["hypothesis"]
)
print(cur_user_prompt)
```

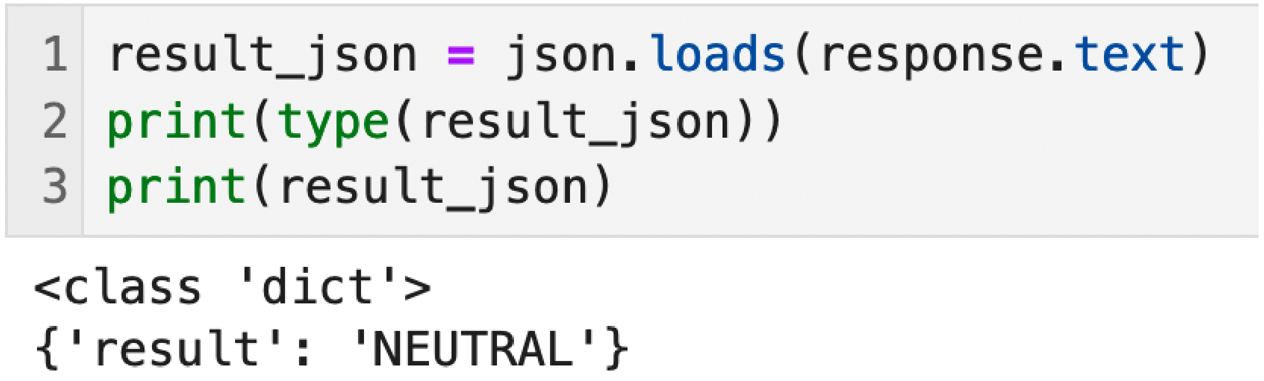
```
premise: A group of kids is playing in a yard and an old man is standing in the background, hypothesis: A
group of boys in a yard is playing and a man is standing in the background. Generate the classification r
esult in JSON with the key: 'result': str (NEUTRAL, ENTAILMENT, or CONTRADICTION)
```

```python
response = model.generate_content(
    cur_user_prompt,
    generation_config={
        "response_mime_type": "application/json",
    },
)
```

Set up the config for Gemini to output in JSON

https://ai.google.dev/gemini-api/docs/structured-output?lang=python

Output of Classification in JSON (Gemini)

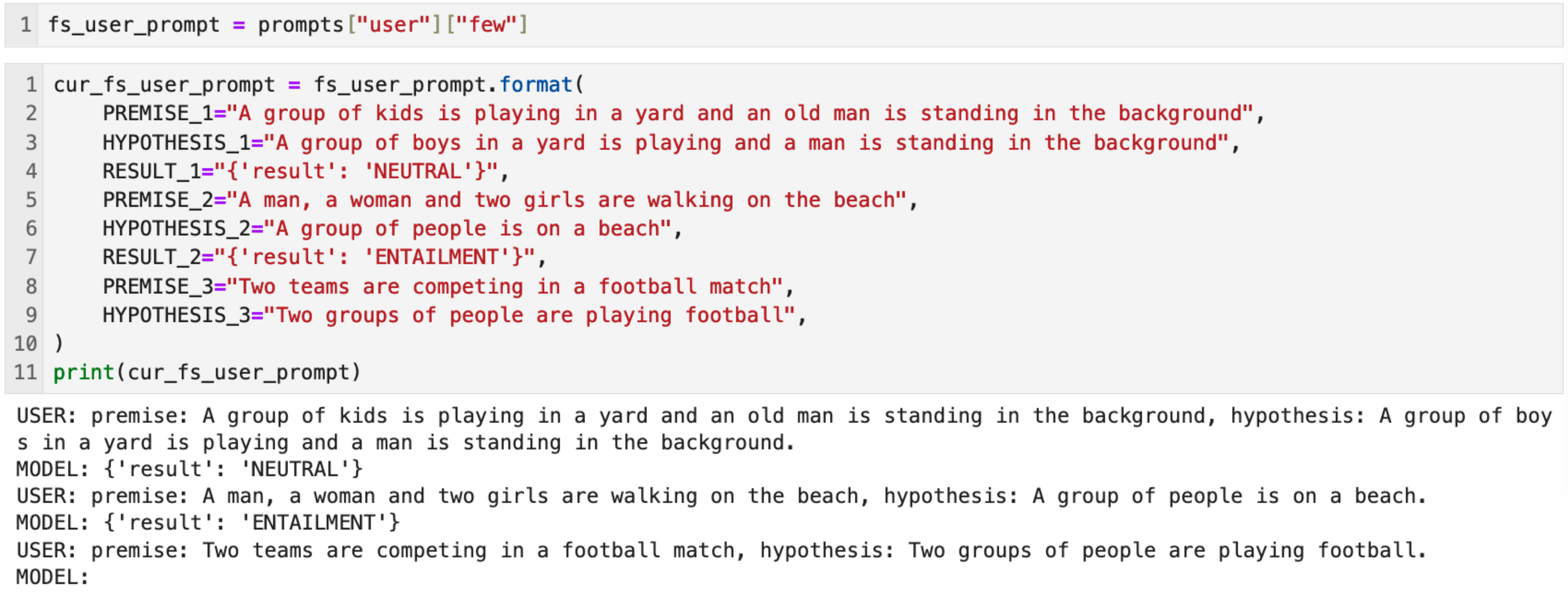


```python
result_json = json.loads(response.text)
print(type(result_json))
print(result_json)
```

```
<class 'dict'>
{'result': 'NEUTRAL'}
```

Few-shot Prompts (Gemini, ref)

- Few-shot: some examples with labels are provided to an LLM
- This approach already prompts the model to follow the output format (JSON mode may not be needed.)



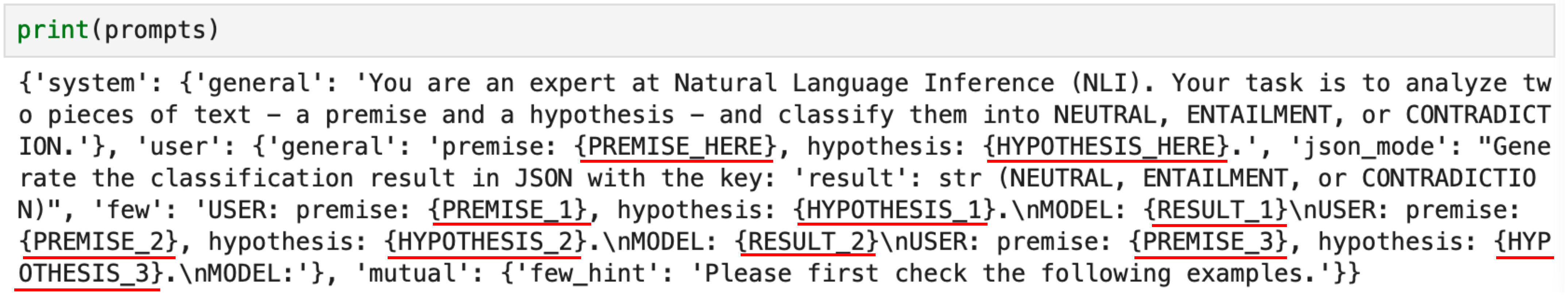
```
fs_user_prompt = prompts["user"]["few"]
```

```
cur_fs_user_prompt = fs_user_prompt.format(
    PREMISE_1="A group of kids is playing in a yard and an old man is standing in the background",
    HYPOTHESIS_1="A group of boys in a yard is playing and a man is standing in the background",
    RESULT_1="{'result': 'NEUTRAL'}",
    PREMISE_2="A man, a woman and two girls are walking on the beach",
    HYPOTHESIS_2="A group of people is on a beach",
    RESULT_2="{'result': 'ENTAILMENT'}",
    PREMISE_3="Two teams are competing in a football match",
    HYPOTHESIS_3="Two groups of people are playing football",
)
print(cur_fs_user_prompt)
```

```
USER: premise: A group of kids is playing in a yard and an old man is standing in the background, hypothesis: A group of boy
s in a yard is playing and a man is standing in the background.
MODEL: {'result': 'NEUTRAL'}
USER: premise: A man, a woman and two girls are walking on the beach, hypothesis: A group of people is on a beach.
MODEL: {'result': 'ENTAILMENT'}
USER: premise: Two teams are competing in a football match, hypothesis: Two groups of people are playing football.
MODEL:
```

The file for Prompts (prompts.yaml)

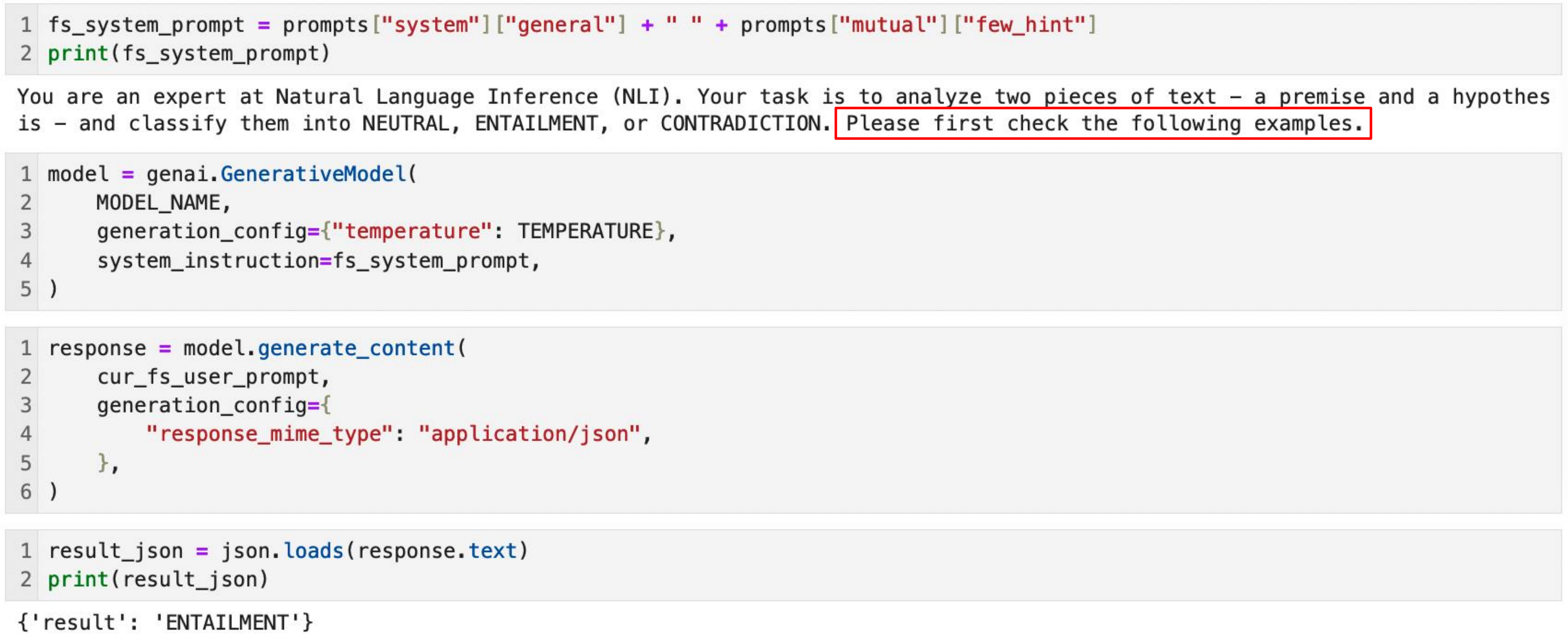
We use the yaml package to load the “prompts.yaml” into a Python dictionary.



```
print(prompts)
```

```
{'system': {'general': 'You are an expert at Natural Language Inference (NLI). Your task is to analyze tw
o pieces of text – a premise and a hypothesis – and classify them into NEUTRAL, ENTAILMENT, or CONTRADICT
ION.'}, 'user': {'general': 'premise: {PREMISE_HERE}, hypothesis: {HYPOTHESIS_HERE}.', 'json_mode': "Gene
rate the classification result in JSON with the key: 'result': str (NEUTRAL, ENTAILMENT, or CONTRADICTIO
N)", 'few': 'USER: premise: {PREMISE_1}, hypothesis: {HYPOTHESIS_1}.\nMODEL: {RESULT_1}\nUSER: premise:
{PREMISE_2}, hypothesis: {HYPOTHESIS_2}.\nMODEL: {RESULT_2}\nUSER: premise: {PREMISE_3}, hypothesis: {HYP
OTHESIS_3}.\nMODEL:'}, 'mutual': {'few_hint': 'Please first check the following examples.'}}
```

Few-shot Prompting (Gemini)



```python
fs_system_prompt = prompts["system"]["general"] + " " + prompts["mutual"]["few_hint"]
print(fs_system_prompt)
```

```
You are an expert at Natural Language Inference (NLI). Your task is to analyze two pieces of text – a premise and a hypothes
is – and classify them into NEUTRAL, ENTAILMENT, or CONTRADICTION. Please first check the following examples.
```

```python
model = genai.GenerativeModel(
    MODEL_NAME,
    generation_config={"temperature": TEMPERATURE},
    system_instruction=fs_system_prompt,
)
```

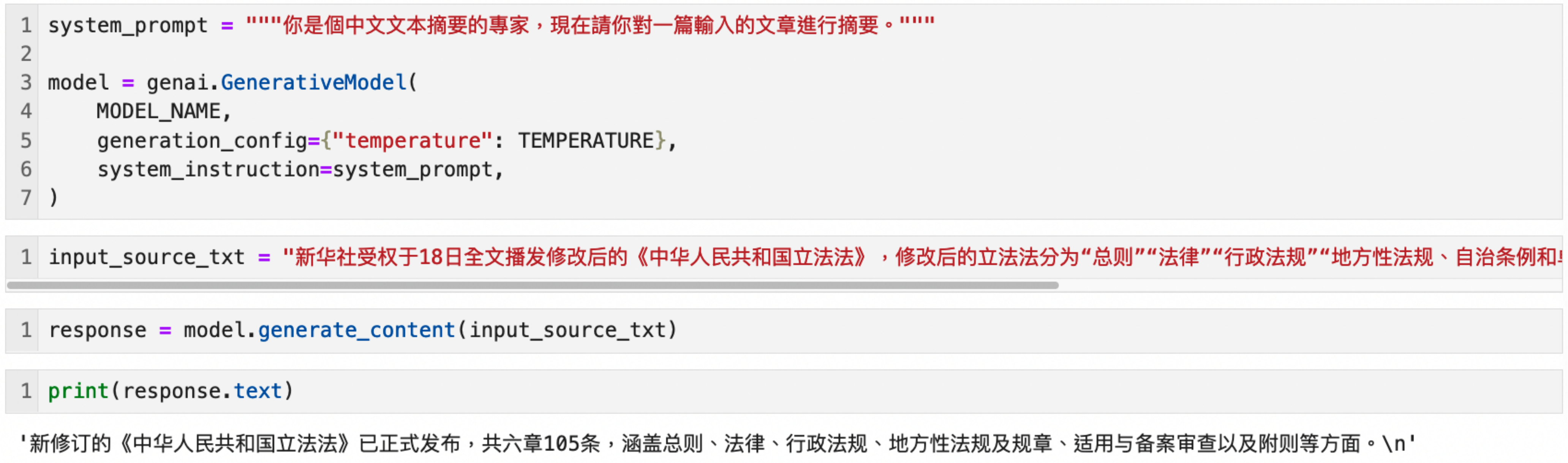
```python
response = model.generate_content(
    cur_fs_user_prompt,
    generation_config={
        "response_mime_type": "application/json",
    },
)
```

```python
result_json = json.loads(response.text)
print(result_json)
```

```
{'result': 'ENTAILMENT'}
```

Example for Summarization (Gemini)

- Dataset: LCSTS (Chinese Abstractive Summarization)



```python
system_prompt = """你是個中文文本摘要的專家，現在請你對一篇輸入的文章進行摘要。"""

model = genai.GenerativeModel(
    MODEL_NAME,
    generation_config={"temperature": TEMPERATURE},
    system_instruction=system_prompt,
)
```

```python
input_source_txt = "新华社受权于18日全文播发修改后的《中华人民共和国立法法》，修改后的立法法分为“总则”“法律”“行政法规”“地方性法规、自治条例和
```

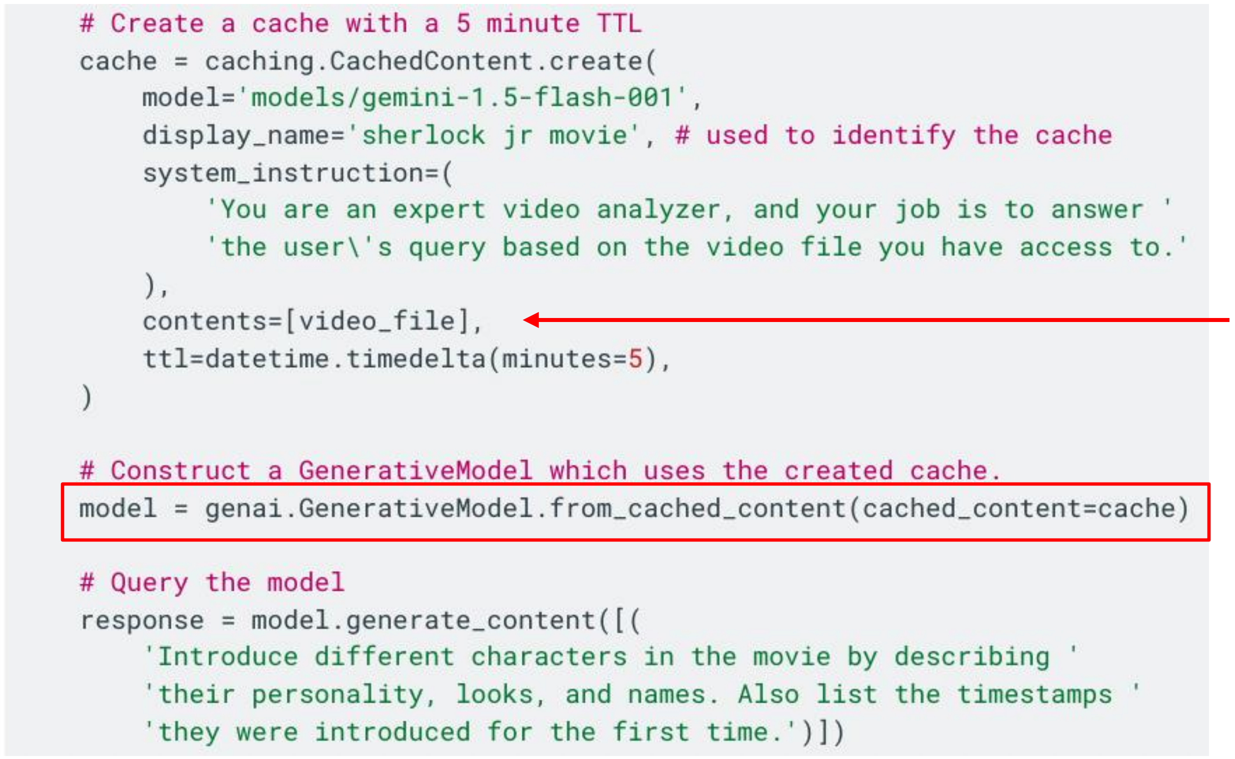
```python
response = model.generate_content(input_source_txt)
```

```python
print(response.text)
```

```
'新修订的《中华人民共和国立法法》已正式发布，共六章105条，涵盖总则、法律、行政法规、地方性法规及规章、适用与备案审查以及附则等方面。\n'
```

Prompt Caching

https://ai.google.dev/gemini-api/docs/caching?lang=python



```python
# Create a cache with a 5 minute TTL
cache = caching.CachedContent.create(
    model='models/gemini-1.5-flash-001',
    display_name='sherlock jr movie', # used to identify the cache
    system_instruction=(
        'You are an expert video analyzer, and your job is to answer '
        'the user\'s query based on the video file you have access to.'
    ),
    contents=[video_file],
    ttl=datetime.timedelta(minutes=5),
)

# Construct a GenerativeModel which uses the created cache.
model = genai.GenerativeModel.from_cached_content(cached_content=cache)

# Query the model
response = model.generate_content([(
    'Introduce different characters in the movie by describing '
    'their personality, looks, and names. Also list the timestamps '
    'they were introduced for the first time.')])
```

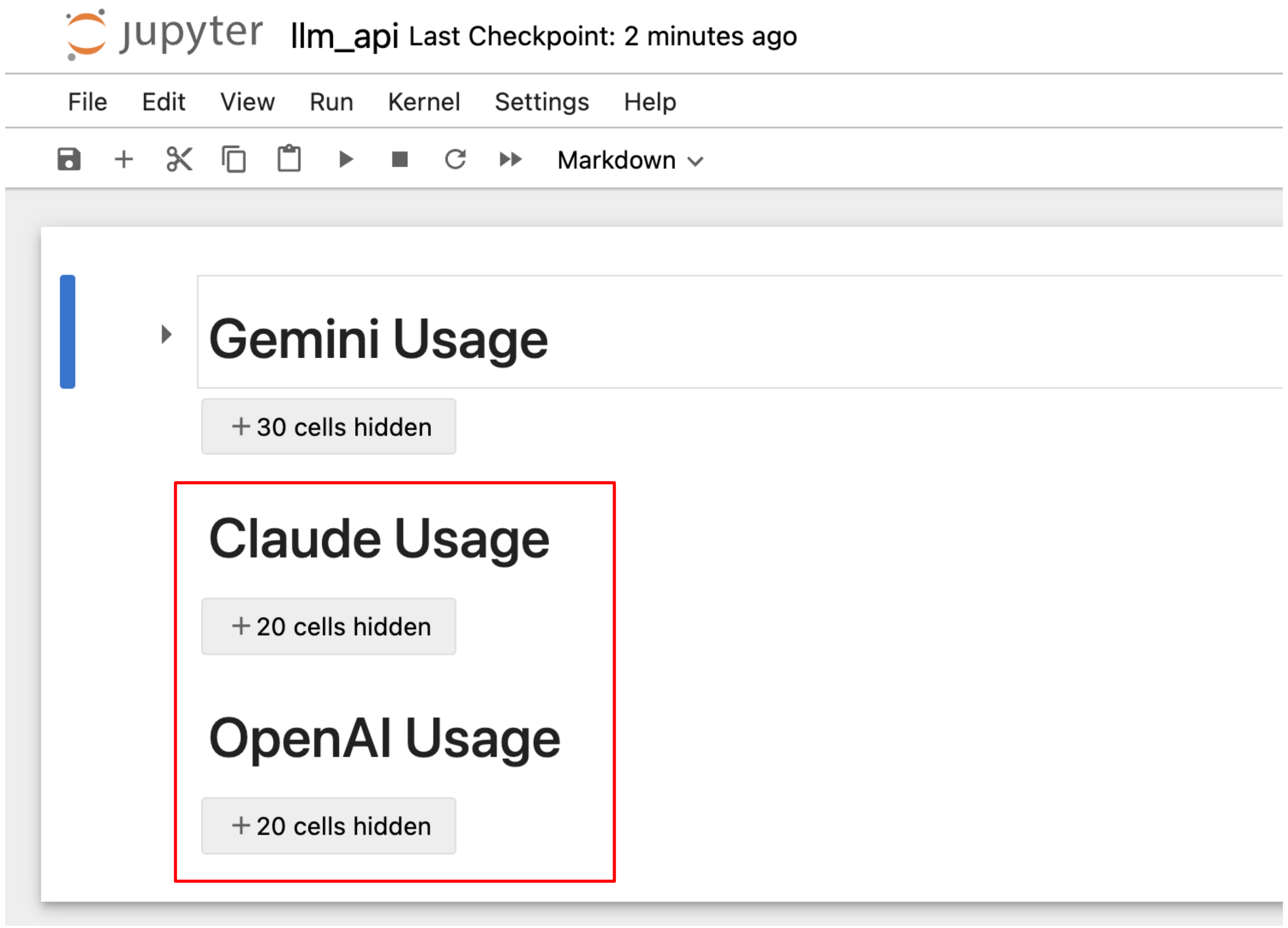
When to use?

- Chatbots with extensive system instructions
- Queries against large document sets
- Analysis of a lengthy video

A very big (long) file

System instruction and the video file is cached.
Cached tokens are in a lower price.

Notebook Executions


jupyter llm_api Last Checkpoint: 2 minutes ago

Gemini Usage

+ 30 cells hidden

Claude Usage

+ 20 cells hidden

OpenAI Usage

+ 20 cells hidden

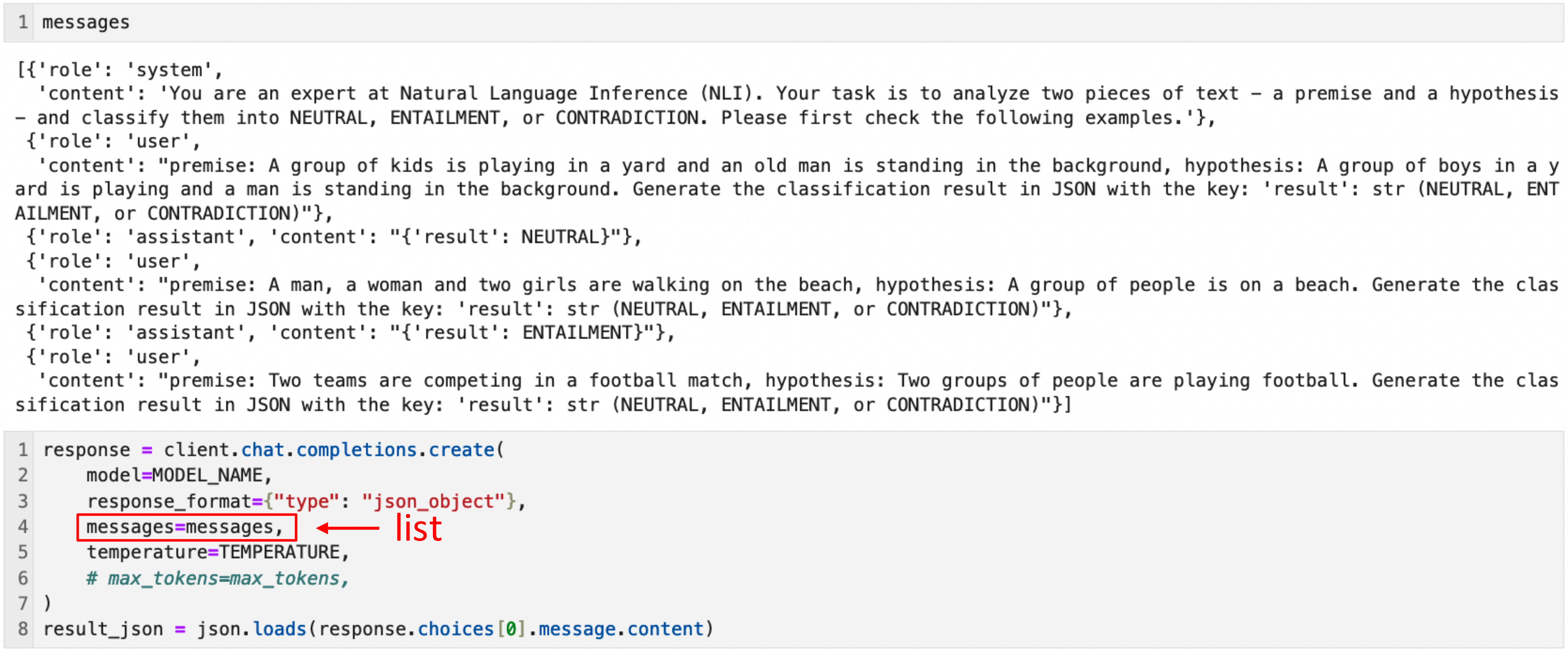
Most of them are similar.

API Keys

- OpenAI API
  - https://platform.openai.com/api-keys
- Claude API
  - https://console.anthropic.com/settings/keys

Difference in Few-shot Prompting

OpenAI API (ref)



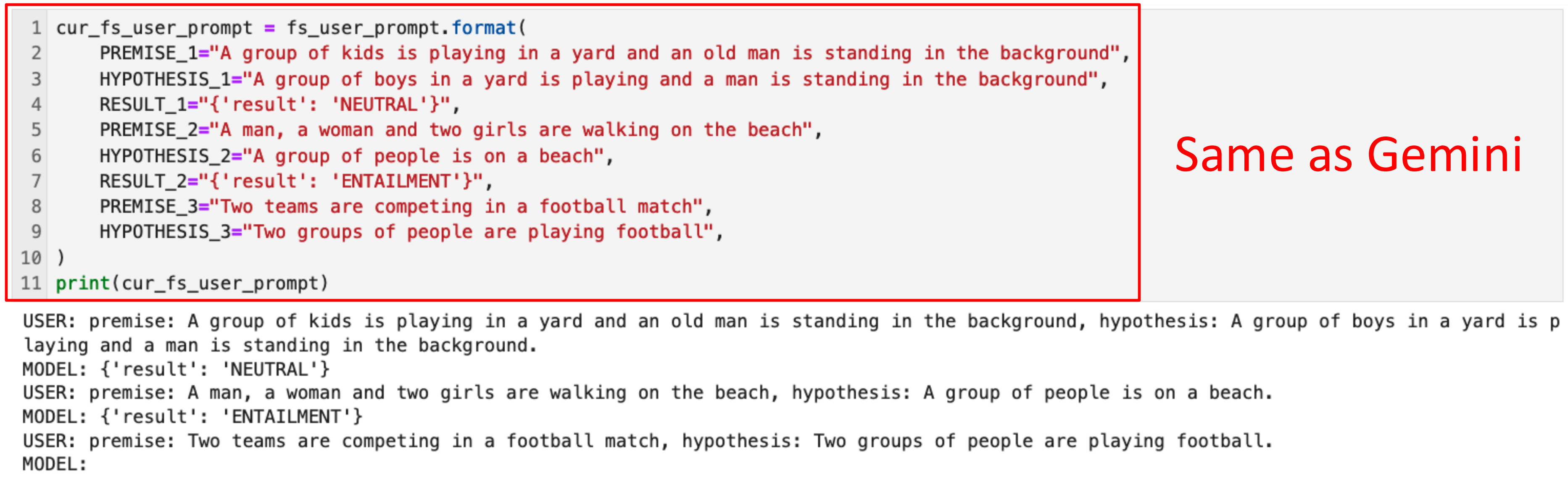
```python
messages
```

```
[{'role': 'system',
  'content': 'You are an expert at Natural Language Inference (NLI). Your task is to analyze two pieces of text - a premise and a hypothesis
- and classify them into NEUTRAL, ENTAILMENT, or CONTRADICTION. Please first check the following examples.'},
 {'role': 'user',
  'content': "premise: A group of kids is playing in a yard and an old man is standing in the background, hypothesis: A group of boys in a y
ard is playing and a man is standing in the background. Generate the classification result in JSON with the key: 'result': str (NEUTRAL, ENT
AILMENT, or CONTRADICTION)"},
 {'role': 'assistant', 'content': "{'result': NEUTRAL}"},
 {'role': 'user',
  'content': "premise: A man, a woman and two girls are walking on the beach, hypothesis: A group of people is on a beach. Generate the clas
sification result in JSON with the key: 'result': str (NEUTRAL, ENTAILMENT, or CONTRADICTION)"},
 {'role': 'assistant', 'content': "{'result': ENTAILMENT}"},
 {'role': 'user',
  'content': "premise: Two teams are competing in a football match, hypothesis: Two groups of people are playing football. Generate the clas
sification result in JSON with the key: 'result': str (NEUTRAL, ENTAILMENT, or CONTRADICTION)"}]
```

```python
response = client.chat.completions.create(
    model=MODEL_NAME,
    response_format={"type": "json_object"},
    messages=messages,    ← list
    temperature=TEMPERATURE,
    # max_tokens=max_tokens,
)
result_json = json.loads(response.choices[0].message.content)
```

Difference in Few-shot Prompting

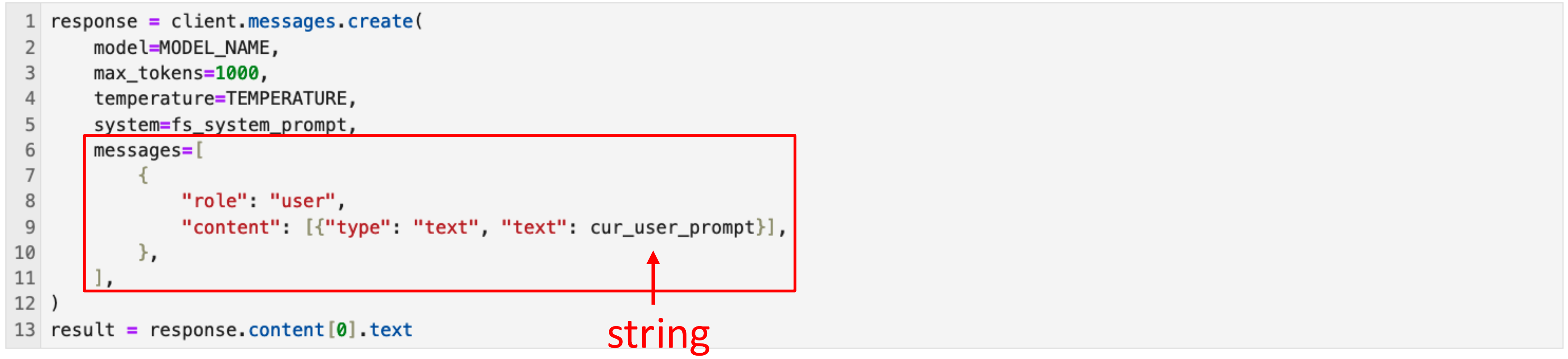
Claude API (ref)



```python
cur_fs_user_prompt = fs_user_prompt.format(
    PREMISE_1="A group of kids is playing in a yard and an old man is standing in the background",
    HYPOTHESIS_1="A group of boys in a yard is playing and a man is standing in the background",
    RESULT_1="{'result': 'NEUTRAL'}",
    PREMISE_2="A man, a woman and two girls are walking on the beach",
    HYPOTHESIS_2="A group of people is on a beach",
    RESULT_2="{'result': 'ENTAILMENT'}",
    PREMISE_3="Two teams are competing in a football match",
    HYPOTHESIS_3="Two groups of people are playing football",
)
print(cur_fs_user_prompt)
```

```
USER: premise: A group of kids is playing in a yard and an old man is standing in the background, hypothesis: A group of boys in a yard is p
laying and a man is standing in the background.
MODEL: {'result': 'NEUTRAL'}
USER: premise: A man, a woman and two girls are walking on the beach, hypothesis: A group of people is on a beach.
MODEL: {'result': 'ENTAILMENT'}
USER: premise: Two teams are competing in a football match, hypothesis: Two groups of people are playing football.
MODEL:
```

Same as Gemini



```python
response = client.messages.create(
    model=MODEL_NAME,
    max_tokens=1000,
    temperature=TEMPERATURE,
    system=fs_system_prompt,
    messages=[
        {
            "role": "user",
            "content": [{"type": "text", "text": cur_user_prompt}],
        },
    ],
)
result = response.content[0].text
```

string

Count the number of tokens

OpenAI API

```
print(f"Num of input tokens {response.usage.prompt_tokens}")
print(f"Num of output tokens {response.usage.completion_tokens}")
```

Claude API

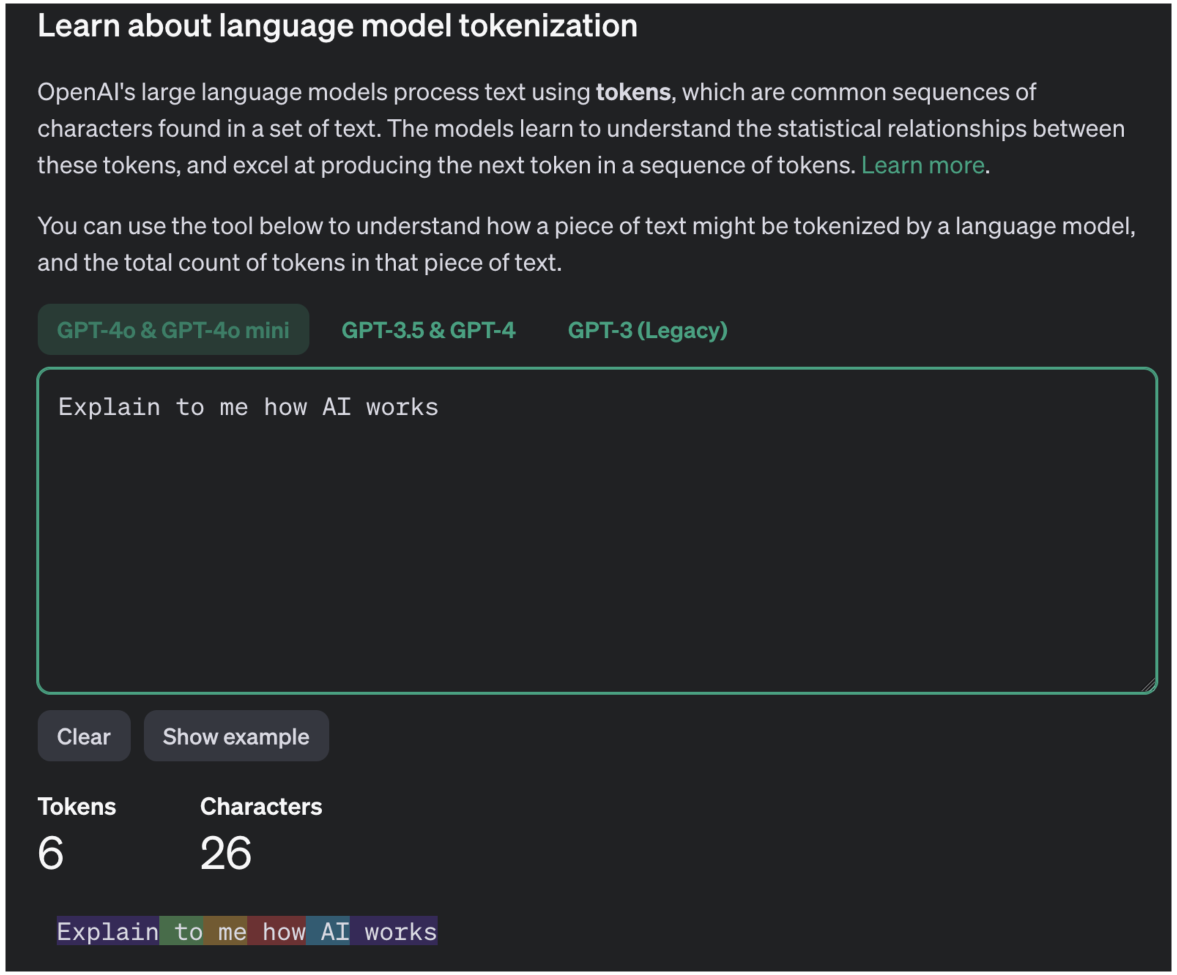
```
print(f"Num of input tokens {response.usage.input_tokens}")
print(f"Num of output tokens {response.usage.output_tokens}")
```

Count the number of tokens (OpenAI)

OpenAI Tokenizer:
https://platform.openai.com/tokenizer

Question: How to calculate the number of tokens for Claude and Gemini?

- All the three APIs (OpenAI, Anthropic, and Gemini) can calculate number of tokens after processing a request.
- Only **OpenAI** offers **pre**-calculator to count tokens.



Learn about language model tokenization

OpenAI's large language models process text using **tokens**, which are common sequences of characters found in a set of text. The models learn to understand the statistical relationships between these tokens, and excel at producing the next token in a sequence of tokens. Learn more.

You can use the tool below to understand how a piece of text might be tokenized by a language model, and the total count of tokens in that piece of text.

```
Explain to me how AI works
```

Tokens	Characters
6	26

```
Explain to me how AI works
```

Further Learning

- Prompt caching:
  - Practical Implementations using Gemini API
  - OpenAI API
  - Claude API (beta)
- Batch API (Batch predictions):
  - OpenAI API
  - Claude API (beta)

Thank you!