# 챗GPT의 API 사용 방법

# 챗GPT의 API 기본 사항

## 챗GPT의 API 기본 사항

- OpenAI의 문장 생성 API에는 'Completions API'와 'Chat Completions API' 두 가지
- 모델에 따라 'Completions API'와 'Chat Completions API' 중 어느 것을 사용할 수 있는지가 정해져
- OpenAI의 문장 생성 모델 목록 및 API 대응

| 모델 패밀리 | 모델 | |
|---|---|---|
| GPT-4 | gpt-4 | Chat Completions API /v1/chat/completions |
| | gpt-4-32k | |
| GPT-3.5 | gpt-3.5-turbo | |
| | gpt-3.5-turbo-16k | |
| | text-davinci-003 (Legacy) | Completions API (Legacy) /v1/completions |
| | text-davinci-002 (Legacy) | |
| GPT-3 | text-curie-001 (Legacy) | |
| | text-babbage-001 (Legacy) | |
| | text-ada-001 (Legacy) | |

# 챗GPT의 API 기본 사항

## Chat Completions API

- 챗GPT의 이를 사용할 때와 마찬가지로 '입력 텍스트를 주고 응답 텍스트를 받는'방식

- Chat Completions API에 대한 요청

```
from openai import OpenAI

client = OpenAI()

response = client.chat.completions.create(
  model="gpt-4",
  messages=[
    {"role": "system", "content": "You are a helpful assistant."},
    {"role": "user", "content": "Hello! I'm John."}
  ]
)

print(response)
```

# 챗GPT의 API 기본 사항

## Chat Completions API

- Chat Completions API에서는 messages라는 배열의 각 요소에 역할별 콘텐츠를 넣는 형식으로 되어 있음
- "role": "system"으로 LLM의 작동에 대한 지시를 주고, 추가로 "role": "user"로 대화를 위한 입력 메시지를 줌

# 챗GPT의 API 기본 사항

## Chat Completions API

- "role" : "assistant"를 사용하여 다음과 같이 user와 assistant(LLM)의 대화 내역을 포함한 요청을 보내기도 함

```
response = client.chat.completions.create(
  model="gpt-4",
  messages=[
        {"role": "system", "content": "You are a helpful assistant."},
        {"role": "user", "content": "Hello! I'm John."},
        {"role": "assistant", "content": "Hello John! How can I assist you today?"},
        {"role": "user", "content": "Do you know my name?"}
    ]
)

print(response.model_dump_json(indent=2))
```

# 챗GPT의 API 기본 사항

## Chat Completions API

- Chat Completions API 자체는 브라우저에서 사용할 수 있는 챗GPT와 달리 대화 기록을 기반으로 응답하는 기능을 가지고 있지 않음
- 대화 기록을 기반으로 한 응답을 원한다면 과거의 모든 대화 내용을 요청에 포함시켜야 함
- choices라는 배열 요소의 message 내용인 “Yes, you mentioned earlier that your name is John. How may I assist you today, John?”이 LLM이 생성한 문장

# 챗GPT의 API 기본 사항

## Chat Completions API

- usage 부분에는

  prompt_tokens (입력 토큰 수)

  Completion_tokens(출력 토큰 수),

  total_tokens(총 토큰 수)가

  포함


```
{
  "id": "chatcmpl-BCHmlkGBfj0qnXHuVOsemA2zduYwq",
  "choices": [
    {
      "finish_reason": "stop",
      "index": 0,
      "logprobs": null,
      "message": {
        "content": "Yes, you just told me that your name is John. How can I assist you today, John?",
        "role": "assistant",
        "function_call": null,
        "tool_calls": null,
        "refusal": null,
        "annotations": []
      }
    }
  ],
  "created": 1742268003,
  "model": "gpt-4-0613",
  "object": "chat.completion",
  "system_fingerprint": null,
  "usage": {
    "completion_tokens": 22,
    "prompt_tokens": 47,
    "total_tokens": 69,
    "prompt_tokens_details": {
      "cached_tokens": 0,
      "audio_tokens": 0
    },
    "completion_tokens_details": {
      "reasoning_tokens": 0,
      "audio_tokens": 0,
      "accepted_prediction_tokens": 0,
      "rejected_prediction_tokens": 0
    }
  },
  "service_tier": "default"
}
```

# 챗GPT의 API 기본 사항

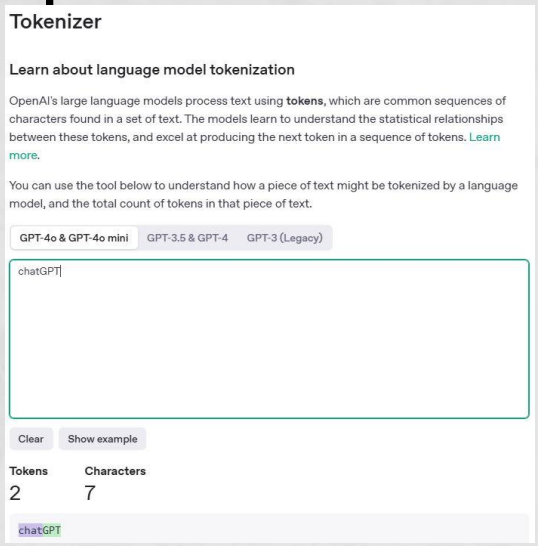
## 입출력 길이 제한과 과금에 영향을 미치는 '토큰'

- 모델은 텍스트를 '토큰'이라는 단위로 나눠 처리
- 토큰이 반드시 단어와 일치하는 것은 아니며
- 예를 들어 'ChatGPT'는 'Chat', 'GPT'라는 2개의 토큰으로 나뉘기도 함

# 챗GPT의 API 기본 사항

## Tokenizer와 tiktoken 소개

- Chat Completions API의 응답을 보면 입력과 출력의 토큰 수가 실제로 몇 개였는지 확인할 수 있음
- chat Completions API를 호출하지 않고도 토큰 수를 파악하고 싶을 때
- 사용할 수 있는 것이 OpenAI의 Tokenizer와 tiktoken






https://platform.openai.com/tokenizer

# 챗GPT의 API 기본 사항

## Tokenizer와 tiktoken 소개

- tiktoken 패키지를 설치하고 다음과 같은 코드를 작성하면 토큰 수를 확인할 수 있음

```
import tiktoken

text = "It's easy to make something cool with LLMs, but very hard to make something production-ready with them."

encoding = tiktoken.encoding_for_model("gpt-4")
tokens = encoding.encode(text)
print(len(tokens))
```

```
23
```

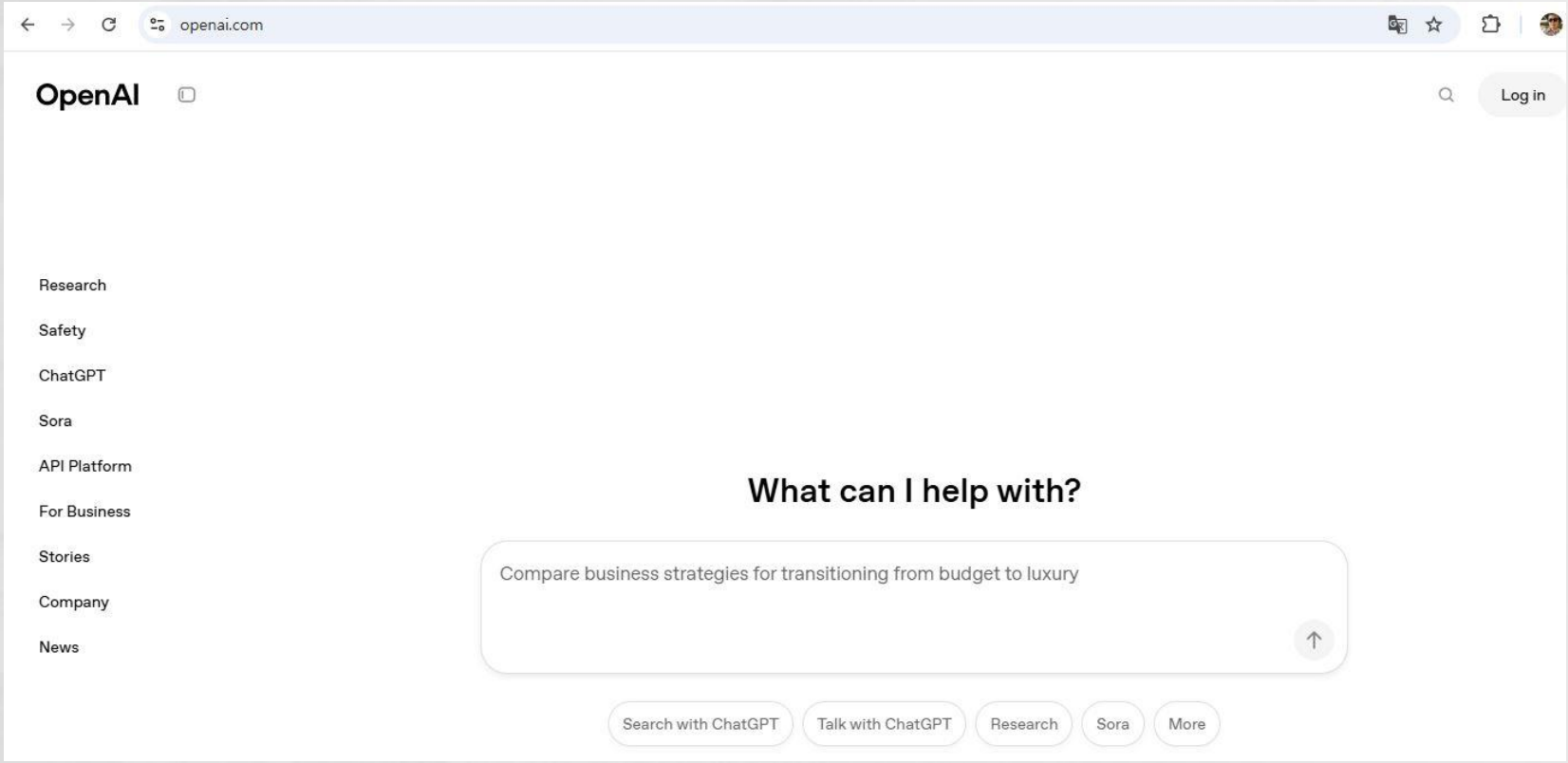
```
text = "LLM을 사용해서 멋져 보이는 것을 만들기는 쉽지만, 프로덕션 수준으로 만들어 내기는 매우 어렵다."

encoding = tiktoken.encoding_for_model("gpt-4")
tokens = encoding.encode(text)
print(len(tokens))
```

```
47
```

# 챗GPT의 API 기본 사항

## OpenAI의 API 키 준비

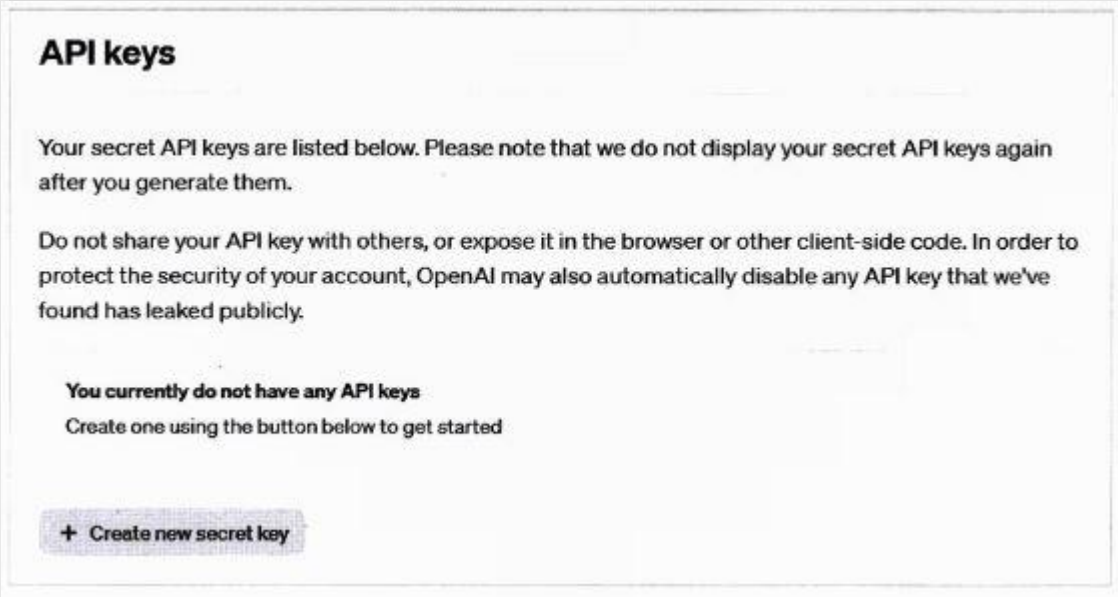
- OpenAI 웹사이트(https://openai.com/)에 접속해 화면 오른쪽 상단의 [Menu]를 클릭하고, 메뉴가 펼쳐지면 [Log in]을 클릭

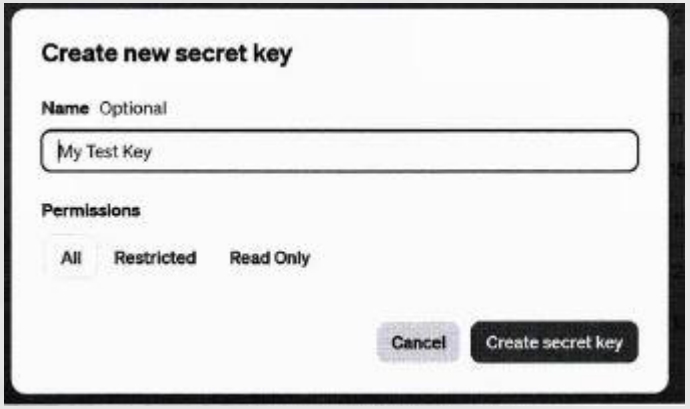


# 챗GPT의 API 기본 사항

## OpenAI의 API 키 준비

- 화면 왼쪽 메뉴에서 [API keys] 를 선택하면 API 키 목록 화면으로 이동



# 챗GPT의 API 기본 사항

## OpenAI의 API 키 준비

- OpenAI의 API 키로 0PENAI_API_KEY라는 환경 변수를 사용
- 발급받은 API 키를 복사해 OPENAI_API_KEY 환경 변수로 설정

```
import os

os.environ["OPENAI_API_KEY"] = "sk-proj-
EY0Xpe5CVqkBiNUUqkddL72W7I2P46hSG4mWTyW8tyKF_5NWN48zC1BofSKOsasvGZ
niyvapLTT3BlbkFJaP1m”
```

# 챗GPT의 API 기본 사항

## Chat Completions API

- Chat Completions API를 사용하기 위해서는 일반적으로 OpenAI의 라이브러리를 사용
- OpenAI의 라이브러리를 설치

```
!pip install openai
```

# 챗GPT의 API 기본 사항

## Chat Completions API 호출

- OpenAI 라이브러리는 환경 변수 OPENAI_API_KEY에서 가져온 API 키를 사용해 요청을 보냄
  - 요청에는 최소 model과 messages를 포함
  - "role": "system"으로 LLM의 작동에 대한 지시사항을 주고, 그 위에 "role": "user"로 대화를 위한 입력 텍스트를 주고 있음

```
from openai import OpenAI

client = OpenAI()

response = client.chat.completions.create(
  model="gpt-4",
  messages=[
    {"role": "system", "content": "You are a helpful assistant."},
    {"role": "user", "content": "Hello! I'm John."}
  ]
)

print(response)
```

# 챗GPT의 API 기본 사항

## Chat Completions API 호출

- 다음과 같은 응답을 얻을 수 있음

응답 중 choices라는 배열 요소의 message의 content를 참조하면 LLM이 생성한 'Hello John! How can I assist you today?'라는 텍스트가 포함되어 있다. 이처럼 모델을 지정하여 입력 텍스트에 대한응답 텍스트를 얻음


```
{
  "id": "chatcmpl-BCHmkSgLni4C37hU0oCYwFqdGFzqT",
  "choices": [
    {
      "finish_reason": "stop",
      "index": 0,
      "logprobs": null,
      "message": {
        "content": "Hello John! How can I assist you today?",
        "role": "assistant",
        "function_call": null,
        "tool_calls": null,
        "refusal": null,
        "annotations": []
      }
    }
  ],
  "created": 1742268002,
  "model": "gpt-4-0613",
  "object": "chat.completion",
  "system_fingerprint": null,
  "usage": {
    "completion_tokens": 11,
    "prompt_tokens": 23,
    "total_tokens": 34,
    "prompt_tokens_details": {
      "cached_tokens": 0,
      "audio_tokens": 0
    },
    "completion_tokens_details": {
      "reasoning_tokens": 0,
      "audio_tokens": 0,
      "accepted_prediction_tokens": 0,
      "rejected_prediction_tokens": 0
    }
  },
  "service_tier": "default"
}
```

# 챗GPT의 API 기본 사항

## 대화 기록에 기반한 응답 얻기

- 상태(대화 기록)를 직접 관리하지 않음(stateless)
- 대화 기록을 바탕으로 응답하게 하려면 과거 대화 내용을 요청에 포함
- 사람의 입력을 "role": "user"로, AI의 입력을 "role": "assistant"로 하여 요청을 보냄

```
response = client.chat.completions.create(
  model="gpt-4",
  messages=[
        {"role": "system", "content": "You are a helpful assistant."},
        {"role": "user", "content": "Hello! I'm John."},
        {"role": "assistant", "content": "Hello John! How can I assist you today?"},
        {"role": "user", "content": "Do you know my name?"}
    ]
)

print(response.model_dump_json(indent=2))
```

# 챗GPT의 API 기본 사항

## 대화 기록에 기반한 응답 얻기

- '저는 존입니다.'라고 자기소개를 한 후, 다시 '제 이름을 아시나요?'라고 물어보는 순서
- 그러자 대화 내역을 바탕으로 '예, 당신이 존이라고 말씀하셨죠. 무엇을 도와드릴까요, 존?'이라고 답


```
{
  "id": "chatcmpl-BCHmlkGBfj0qnXHuVOsemA2zduYwq",
  "choices": [
    {
      "finish_reason": "stop",
      "index": 0,
      "logprobs": null,
      "message": {
        "content": "Yes, you just told me that your name is John. How can I assist you today, John?",
        "role": "assistant",
        "function_call": null,
        "tool_calls": null,
        "refusal": null,
        "annotations": []
      }
    }
  ],
  "created": 1742268003,
  "model": "gpt-4-0613",
  "object": "chat.completion",
  "system_fingerprint": null,
  "usage": {
    "completion_tokens": 22,
    "prompt_tokens": 47,
    "total_tokens": 69,
    "prompt_tokens_details": {
      "cached_tokens": 0,
      "audio_tokens": 0
    },
    "completion_tokens_details": {
      "reasoning_tokens": 0,
      "audio_tokens": 0,
      "accepted_prediction_tokens": 0,
      "rejected_prediction_tokens": 0
    }
  },
  "service_tier": "default"
}
```

# 챗GPT의 API 기본 사항

## 응답을 스트리밍으로 받기

- Chat Completions API에서도 이와 마찬가지로 스트리밍으로 응답을 받을 수 있음

- 요청에 stream=True라는 파라미터를 추가

```
response = client.chat.completions.create(
  model="gpt-4",
  messages=[
        {"role": "system", "content": "You are a helpful assistant."},
        {"role": "user", "content": "Hello! I'm John."}
  ],
  stream=True
)

for chunk in response:
  choice = chunk.choices[0]
  if choice.finish_reason is None:
    print(choice.delta.content)
```

# 챗GPT의 API 기본 사항

## 기본 파라미터

- Chat Completions API에서 model, messages, stream 외에 지정할 수 있는 몇 가지 파라미터

| 파라미터 이름 | 개요 | 기본값 |
|---|---|---|
| temperature | 0~2 사이의 값으로 클수록 출력이 무작위적이고 작을수록 결정적임 | 1 |
| n | 생성되는 텍스트 후보의 수(응답의 choices 요소의 수) | 1 |
| stop | 등장하는 순간 생성을 멈추는 문자열(또는 그 배열) | null |
| max_tokens | 생성하는 최대 토큰 수 | 16 |
| user | OpenAI의 피드백에 도움이 되는 최종 사용자 ID를 제공 | 없음 |

# 챗GPT의 API 기본 사항

## Function calling

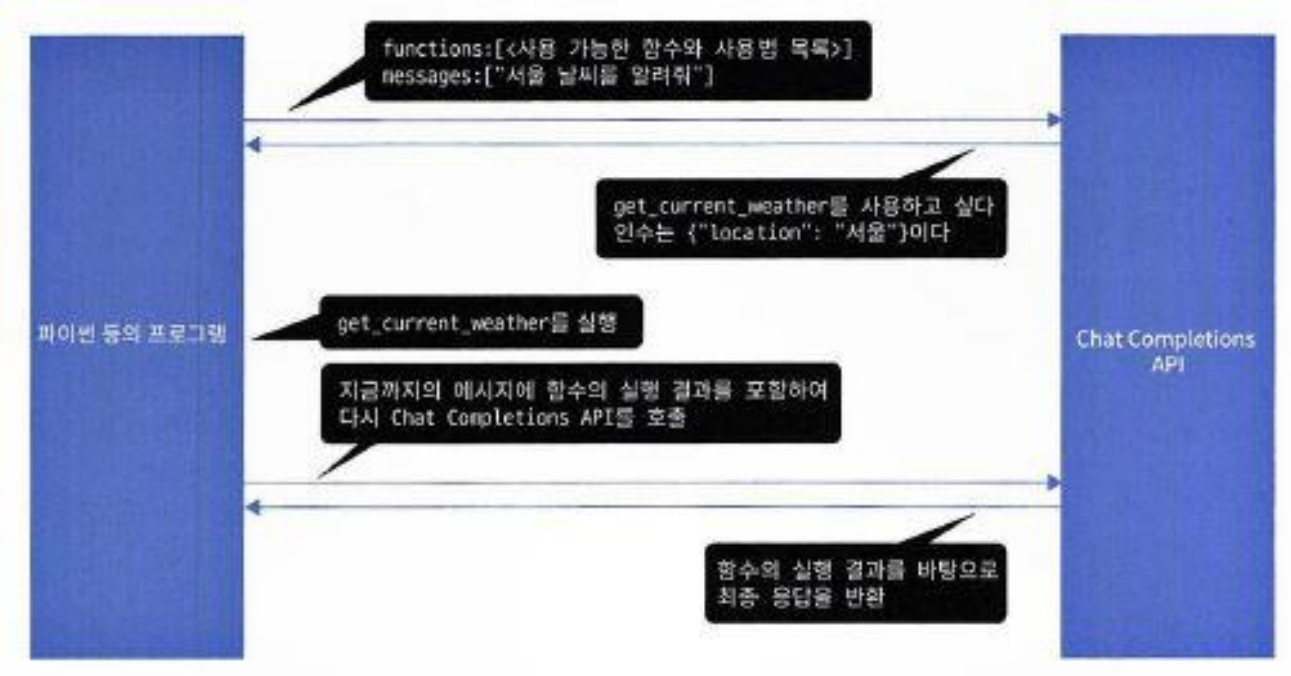
- Function calling은 2023년 6월 13일에 추가된 Chat Completions API의 새로운 기능
- 사용 가능한 함수를 LLM에 알려주고 LLM이 함수를 사용하고 싶다'는 판단을 내리게 하는 기능(LLM이 함수를 실행하는 것이 아니라, LLM이 "함수를 사용하고 싶다'는 응답을 반환할 뿐이다)
- 랭체인에는 LLM이 필요에 따라 함수를 사용할 수 있도록 하는 'Agents'라는 기능

# 챗GPT의 API 기본 사항

## Function calling

- 이러한 기능이 Chat Completions API 자체에 구현되어 있고, 이를 위해 모델을 미세 조정한 것이 Function calling

- Function calling을 사용하여 함수 실행 중간의 LLM과의 상호 작용



# 챗GPT의 API 기본 사항

## Function calling

- 처리 흐름은 먼저 사용 가능한 함수 목록과 함께 질문 등의 텍스트를 전송
- 이에 대해 LLM이 함수를 사용하고 싶다'는 응답을 보내면 파이썬 등의 프로그램으로 해당 함수를 실행
- 그 실행 결과를 포함한 요청을 다시 LIM에 보내면 최종 답변을 얻을 수 있는 구조
- LLM은 어떤 함수를 어떻게 사용하고 싶은지 알려줄 뿐, 함수 실행은 파이썬
- 등을 이용해 Chat Completions API의 사용자 측에서 실행

# 챗GPT의 API 기본 사항

## Function calling

- get_current_weather라는 지역을 지정하여 날씨를 구하는 파이썬 함수를 정의

```
import json

def get_current_weather(location, unit="celsius"):
    weather_info = {
        "location": location,
        "temperature": "25",
        "unit": "celsius",
        "forecast": ["sunny", "windy"],
    }
    return json.dumps(weather_info)
```

# 챗GPT의 API 기본 사항

## Function calling

- LLM이 사용할 수 있는 함수 목록을 정의
- get_current_weather라는 함수에 대한 설명과 파라미터를 정의

```
functions = [
    {
        "name": "get_current_weather",
        "description": "Get the current weather in a given location",
        "parameters": {
            "type": "object",
            "properties": {
                "location": {
                    "type": "string",
                    "description": "The city and state, e.g. Seoul",
                },
                "unit": {"type": "string", "enum": ["celsius", "fahrenheit"]},
            },
            "required": ["location"],
        },
    }
]
```

# 챗GPT의 API 기본 사항

## Function calling

- "What's the weather like in Seoul?", 즉 서울 날씨를 묻는 질문으로 Chat Completions API를 호출
- 사용할 수 있는 함수 목록을 functions라는 인수로 전달

```
messages = [{"role": "user", "content": "What's the weather like in Seoul?"}]

response = client.chat.completions.create(
    model="gpt-4",
    messages=messages,
    functions=functions
)

print(response.model_dump_json(indent=2))
```

# 챗GPT의 API 기본 사항

## Function calling

- 이 요청에 대해 다음과 같은 응답을 얻을 수 있음


```
{
  "id": "chatcmpl-BCHmr8k53A87aCSzdPuBPaBM7iDhy",
  "choices": [
    {
      "finish_reason": "function_call",
      "index": 0,
      "logprobs": null,
      "message": {
        "content": null,
        "role": "assistant",
        "function_call": {
          "arguments": "{\n  \"location\": \"Seoul\"\n}",
          "name": "get_current_weather"
        },
        "tool_calls": null,
        "refusal": null,
        "annotations": []
      }
    }
  ],
  "created": 1742268009,
  "model": "gpt-4-0613",
  "object": "chat.completion",
  "system_fingerprint": null,
  "usage": {
    "completion_tokens": 18,
    "prompt_tokens": 79,
    "total_tokens": 97,
    "prompt_tokens_details": {
      "cached_tokens": 0,
      "audio_tokens": 0
    },
    "completion_tokens_details": {
      "reasoning_tokens": 0,
      "audio_tokens": 0,
      "accepted_prediction_tokens": 0,
      "rejected_prediction_tokens": 0
    }
  },
  "service_tier": "default"
}
```

# 챗GPT의 API 기본 사항

## Function calling

- LLM이 생성한 텍스트가 choices 요소의 message의 content에 포함되어 있었지만, 그 부분이 null로 되어 있음
- 대신 function_call이라는 요소가 있고, get_current_weather를 이런 인수로 실행하고 싶다'는 내용이 적혀 있음
- 주어진 함수 목록과 입력 텍스트를 통해 LLM은 이 질문에 답하기 위해서는 get_current_weather를 {"location": "Seoul"}이라는 인수로 실행해야 한다고 판단한 것.

# 챗GPT의 API 기본 사항

## Function calling

- LLM은 파이썬과 같은 함수를 실행하는 기능이 없음
- 이 함수는 LLM이 지정한 인수를 분석하여 해당 함수를 호출하는 방식으로 이 함수를 실행

```
response_message = response.choices[0].message

available_functions = {
    "get_current_weather": get_current_weather,
}
function_name = response_message.function_call.name
fuction_to_call = available_functions[function_name]
function_args = json.loads(response_message.function_call.arguments)

function_response = fuction_to_call(
    location=function_args.get("location"),
    unit=function_args.get("unit"),
)

print(function_response)
```

```
{"location": "Seoul", "temperature": "25", "unit": "celsius", "forecast": ["sunny", "windy"]}
```

# 챗GPT의 API 기본 사항

## Function calling

- 파이썬에서 함수 실행 결과가 나오면 LLM에 다시 요청을 보내기 위해 messages를 준비
- 처음 보낸 요청은 messages = [{"role": "user", "content": "What's the weather like in Seoul?"}]
- 여기에 LLM의 response_message를 추가하고, 함수 실행 결과를 "role": "function“으로 추가

# 챗GPT의 API 기본 사항

## Function calling

- messages의 내용

```
messages.append(response_message)
messages.append(
    {
        "role": "function",
        "name": function_name,
        "content": function_response,
    }
)
```

```
print(messages)
```

```
[{'role': 'user', 'content': "What's the weather like in Seoul?"}, ChatCompletionMessage(content=None, role='assistant', functi
on_call=FunctionCall(arguments='{\n  "location": "Seoul"\n}', name='get_current_weather'), tool_calls=None, refusal=None, annot
ations=[]), {'role': 'function', 'name': 'get_current_weather', 'content': '{"location": "Seoul", "temperature": "25", "unit":
"celsius", "forecast": ["sunny", "windy"]}'}]
```

# 챗GPT의 API 기본 사항

## Function calling

- 이 messages를 사용해 Chat Completions API에 다시 한번 요청

```
second_response = client.chat.completions.create(
    model="gpt-4",
    messages=messages,
)

print(second_response.model_dump_json(indent=2))
```

# 챗GPT의 API 기본 사항

## Function calling

- 최종 답변으로 방금 전의 함수 실행 결과를 바탕으로 서울의 날씨를 답변


```
{
  "id": "chatcmpl-BCHmtvJ0nxdgUjg3M3tQr2jHiqx9r",
  "choices": [
    {
      "finish_reason": "stop",
      "index": 0,
      "logprobs": null,
      "message": {
        "content": "The current weather in Seoul is 25 degrees Celsius. The forecast indicates it's sunny and windy.",
        "role": "assistant",
        "function_call": null,
        "tool_calls": null,
        "refusal": null,
        "annotations": []
      }
    }
  ],
  "created": 1742268011,
  "model": "gpt-4-0613",
  "object": "chat.completion",
  "system_fingerprint": null,
  "usage": {
    "completion_tokens": 21,
    "prompt_tokens": 72,
    "total_tokens": 93,
    "prompt_tokens_details": {
      "cached_tokens": 0,
      "audio_tokens": 0
    },
    "completion_tokens_details": {
      "reasoning_tokens": 0,
      "audio_tokens": 0,
      "accepted_prediction_tokens": 0,
      "rejected_prediction_tokens": 0
    }
  },
  "service_tier": "default"
}
```

# 챗GPT의 API 기본 사항

## Function calling

- Function calling을 사용하면 LLM이 필요에 따라 "함수를 사용하고 싶다"고 판단
- 그 인수까지 고려해 중
- 내용을 바탕으로 여기서 함수를 실행
- 실행 결과를 포함하여 다시 LLM을 호출하면 LLM이 최종적인 답을 돌려주는 것

# 챗GPT의 API 기본 사항

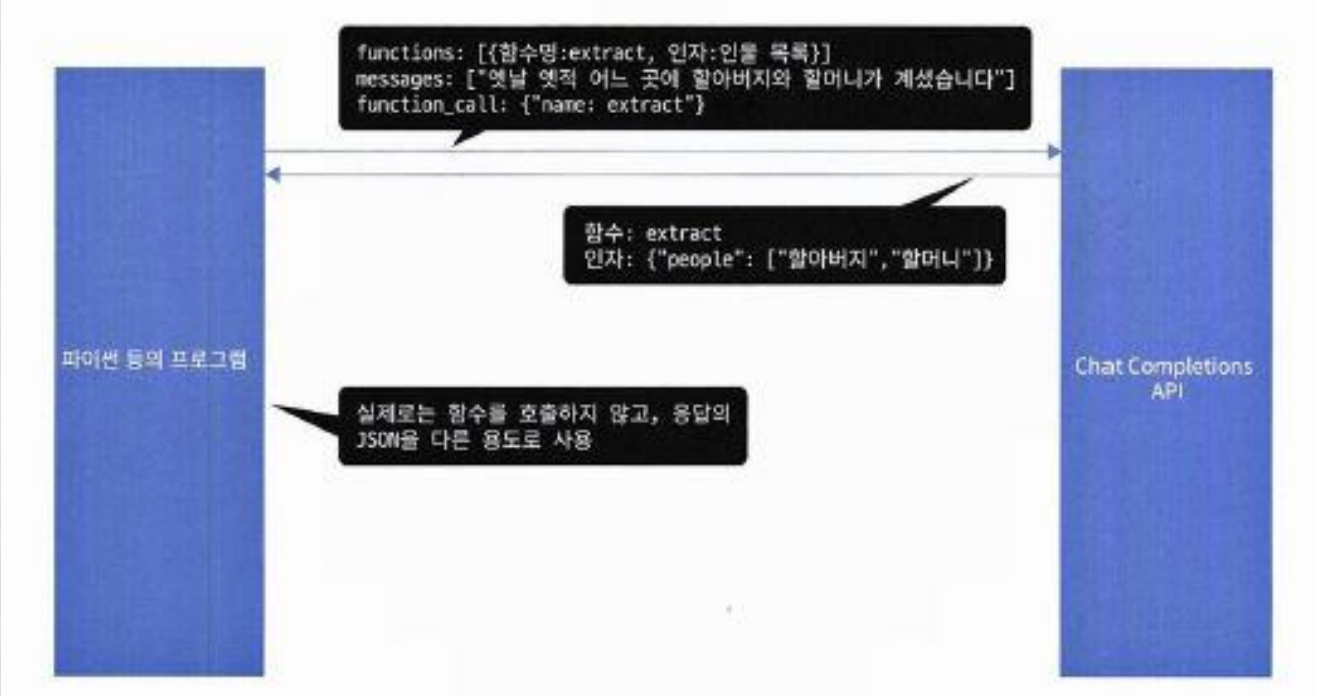
## Function calling

- 파라미터 ‘function_call’
  - Function calling의 등장'과 함께 Chat Completions API 요청에 ‘function_calf 이라는 파라미터가 추가
  - function_call에 "none"을 지정하면 LLM은 함수 호출과 같은 응답을 하지 않고 일반 텍스트를 반환
  - function_call을 "auto"로 설정하면 LLM은 입력에 따라 지정된 함수를 사용해야 한다고 판단되면 함수 이름과 인수를 응답
  - Function_call 파라미터의 기본 작동은 functions를 제공하지 않으면 "none", functions를 제공하면 "auto” 로 설정

# 챗GPT의 API 기본 사항

## Function calling을 응용한 JSON 생성

- LLM이 함수를 호출할 생각으로 JSON 형식의 데이터를 생성하게 하고 실제로는 함수를 호출하지 않고 인수의 값을 다른 용도로 사용



# 정리

## 정리

- 챗GPT의 API 기본 사항