- **메시지(message)**: 사용자가 입력할 프롬프트가 포함된 리스트입니다. 가장 핵심이 되는 부분으로 메시지를 활용하여 원하는 질문을 하고, 다양한 형태의 메시지 입력을 통해 프롬프트 엔지니어링을 할 수 있습니다. 기본적인 질문 방법은 [{role: "user", "content": "프롬프트 입력"}]과 같이 content 키 안에 질문하고자 하는 프롬프트를 입력하는 방법입니다. 역할을 부여하거나 이어서 질문하는 방법은 다음 절에서 자세히 다루겠습니다.

- **온도 조절(temperature)**: 텍스트의 랜덤성(randomness)과 관련된 파라미터입니다. 온도를 높게 설정하면(최댓값은 2.0) 모델이 생성하는 텍스트가 사람이 보기에는 예측에서 벗어나고 다양하며 창의적으로 보이는 경향을 띕니다. 반면, 온도를 낮게 설정하면 좀 더 전형적이고 보수적인 텍스트를 생성하는 경향을 보입니다. 일반적으로 온도 값은 0.5~1.5 사이의 값을 시작점으로 사용합니다. 적절한 온도 값은 사용자가 어떤 작업을 하고 싶은지, 어떤 프롬프트를 사용하는지에 따라 다르므로 실험적으로 찾는 것이 좋습니다. 설정값의 범위는 0.0~2.0이며, 값을 설정하지 않으면 기본값인 1로 설정됩니다.

- **핵 샘플링(top_p)**: 다음 단어 또는 토큰이 샘플링되는 범위를 제어합니다. 응답을 생성할 때 모델은 다음 토큰의 어휘에 대한 확률 분포를 계산합니다. 예를 들어, top_p를 0.5로 설정하면 모델이 샘플링할 때, 누적 확률이 0.5보다 큰 상위 토큰 중에서만 다음 토큰을 샘플링합니다. top_p를 1.0으로 설정하면 모든 토큰(전체 분포)에서 샘플링하고, top_p를 0.0으로 설정하면 항상 가장 확률이 높은 단일 토큰을 선택합니다. 설정값의 범위는 0.0~1.0이며, 값을 설정하지 않으면 기본값인 1로 설정됩니다.

- **존재 페널티(presence_penalty)**: 단어가 이미 생성된 텍스트에 나타난 경우 해당 단어가 등장할 가능성을 줄입니다. 빈도수 페널티와 달리 존재 페널티는 과거 예측에서 단어가 나타나는 빈도수에 따라 달라지지는 않습니다. OpenAI의 설명에 따르면 이 파라미터 값을 크게 설정할수록 모델이 새로운 주제에 대해 이야기할 가능성이 높아진다고 합니다. 설정값의 범위는 0.0~2.0이며, 값을 설정하지 않으면 기본값인 0으로 설정됩니다.

- **빈도수 페널티(frequency_penalty)**: 모델이 동일한 단어를 반복적으로 생성하지 않도록 설정하는 값입니다. 이 페널티는 어떤 단어가 어느 빈도로 등장했는지에 따라 영향을 받습니다. 빈도수 페널티의 값을 높게 설정하면 모델이 이미 생성한 단어를 다시 생성하는 것을 자제하므로 좀 더 다양하고 중복되지 않은 텍스트를 생성하게 유도할 수 있습니다. 따라서 모델이 특정 단어를 반복하는 경향을 보인다면 빈도수 페널티 값을 높게 설정합니다. 설정값의 범위는 0.0~2.0이며, 값을 설정하지 않으면 기본값인 0으로 설정됩니다.

- **응답 개수(n)**: 입력 메시지에 대해 생성할 답변의 수를 설정합니다. 값을 설정하지 않으면 기본값인 1로 설정됩니다.

- **최대 토큰(max_tokens)**: 최대 토큰 수를 제한합니다. API 사용은 토큰 수에 따라 요금이 부과되므로 최대 토큰 파라미터를 통해 답변의 길이를 조절할 수 있습니다. 값을 설정하지 않으면 모델의 최대 토큰 수에 맞춰 설정됩니다.

메시지(message): 사용자가 입력할 프롬프트가 포함된 리스트입니다. 가장 핵심이 되는 부분으로 메시 지를 활용하여 원하는 질문을 하고, 다양한 형태의 메시지 입력을 통해 프롬프트 엔지니어링을 할 수 있습 니다. 기본적인 질문 방법은 [{role: "user", "content": "프롬프트 입력"}]과 같이 content 키 안에 질문하 고자하는 프롬프트를 입력하는 방법입니다. 역할을 부여하거나 이어서 질문하는 방법은 다음 절에서 자 세히 다루겠습니다.

・온도 조절(temperature): 텍스트의 랜덤성(randomness)과 관련된 파라미터입니다. 온도를 높게 설정하 면(최댓값은 2.0) 모델이 생성하는 텍스트가 사람이 보기에는 예측에서 벗어나고 다양하며 창의적으로 보 이는 경향을 띕니다. 반면, 온도를 낮게 설정하면 좀 더 전형적이고 보수적인 텍스트를 생성하는 경향을 보 입니다. 일반적으로 온도 값은 0.5~1.5 사이의 값을 시작으로 사용합니다. 적절한 온도 값은 사용자가 어떤 작업을 하고 싶은지 어떤 프롬프트를 사용하는지에 따라 다르므로 실험적으로 찾는 것이 좋습니다. 설정값 의 범위는 0.0~2.0이며, 값을 설정하지 않으면 기본값인 1로 설정됩니다.

.

핵 샘플링(top_p): 다음 단어 또는 토큰이 샘플링되는 범위를 제어합니다. 응답을 생성할 때 모델은 다음 토큰의 어휘에 대한 확률 분포를 계산합니다. 예를 들어, top_p를 0.5로 설정하면 모델이 샘플링할 때, 누 적 확률이 0.5보다 큰 상위 토큰 중에서만 다음 토큰을 샘플링합니다. top_p를 1.0으로 설정하면 모든 토 큰(전체 분포)에서 샘플링하고 top_p를 0.0으로 설정하면 항상 가장 확률이 높은 단일 토큰을 선택합니 다. 설정값의 범위는 0.0~1.0이며 값을 설정하지 않으면 기본값인 1로 설정됩니다.

・존재 페널티(presence_penalty): 단어가 이미 생성된 텍스트에 나타난 경우 해당 단어가 등장할 가능성 을 줄입니다. 빈도수 페널티와 달리 존재 페널티는 과거 예측에서 단어가 나타나는 빈도수에 따라 달라지 지는 않습니다. OpenAI의 설명에 따르면 이 파라미터 값을 크게 설정할수록 모델이 새로운 주제에 대해 이야기할 가능성이 높아진다고 합니다. 설정값의 범위는 0.0~2.0이며 값을 설정하지 않으면 기본값인 0 으로 설정됩니다.

• 빈도수 페널티(frequency_penalty): 모델이 동일한 단어를 반복적으로 생성하지 않도록 설정하는 값입니 다. 이 페널티는 어떤 단어가 어느 빈도로 등장했는지에 따라 영향을 받습니다. 빈도수 페널티의 값을 높게 설정하면 모델이 이미 생성한 단어를 다시 생성하는 것을 자제하므로 좀 더 다양하고 중복되지 않은 텍스트 를 생성하게 유도할 수 있습니다. 따라서 모델이 특정 단어를 반복하는 경향을 보인다면 빈도수 페널티 값 을 높게 설정합니다. 설정값의 범위는 0.0~2.0이며, 값을 설정하지 않으면 기본값인 0으로 설정됩니다.

응답 개수(n): 입력 메시지에 대해 생성할 답변의 수를 설정합니다. 값을 설정하지 않으면 기본값인 1로 설 정됩니다.

최대 토큰(max_tokens): 최대 토큰 수를 제한합니다. API 사용은 토큰 수에 따라 요금이 부과되므로 최 대 토큰 파라미터를 통해 답변의 길이를 조절할 수 있습니다. 값을 설정하지 않으면 모델의 최대 토큰 수 에 맞춰 설정됩니다.

2.4.2 이전 대화를 포함하여 답변하기

ChatGPT는 답변할 때 이전 질문과 답변을 모두 고려하여 답변하는 특징이 있습니다. ChatGPT API를 이용하면 ChatGPT에게 답변을 요청할 때 '앞서 네가 이런 답변을 한 상태였다'는 정보를 전달할 수 있습니다. 이것은 사용자가 가정하는 것이지만, ChatGPT는 마치 과거에 자신이 답변한 것으로 가정하고, 추가 답변을 제공합니다.

이렇게 질문을 작성하려면 `messages=[ ]` 안에 `{"role": "user", "content": ""}`를 작성한 후 `{"role": "assistant", "content": ""}`을 추가로 작성하고, 다시 `{"role": "user", "content": ""}`를 번갈아 작성하면 됩니다. 실제 코드를 통해 확인해 봅시다.

다음 코드에서는 사용자가 ChatGPT에게 "2002년 월드컵에서 가장 화제가 되었던 나라는 어디야?"라고 질문한 후, ChatGPT가 "바로 예상을 뚫고 4강 진출 신화를 일으킨 한 국입니다."라고 답변한 상태라고 가정하고, 사용자가 다시 "그 나라가 화제가 되었던 이유를 자세하게 설명해 줘"라고 질문해 보겠습니다.

2.4.2 이전 대화를 포함하여 답변하기
ChatGPT는 답변할 때 이전 질문과 답변을 모두 고려하여 답변하는 특징이 있습니다. ChatGPT API를 이용하면 ChatGPT에게 답변을 요청할 때 '앞서 네가 이런 답변을 한 상 태였다'는 정보를 전달할 수 있습니다. 이것은 사용자가 가정하는 것이지만, ChatGPT는 마치 과거에 자신이 답변한 것으로 가정하고, 추가 답변을 제공합니다.

이렇게 질문을 작성하려면 messages=[] 안에 {"role":"user", "content": """} 작성한 후 {"role": "assistant", "content": ""}을 추가로 작성하고, 다시 {"role": "user", "content": """}를 번갈아 작성하면 됩니다. 실제 코드를 통 해 확인해 봅시다.

다음 코드에서는 사용자가 ChatGPT에게 "2002년 월드컵에서 가장 화제가 되었던 나 라는 어디야?"라고 질문한 후, ChatGPT가 “바로 예상을 뚫고 4강 진출 신화를 일으킨 한 국입니다."라고 답변한 상태라고 가정하고, 사용자가 다시 “그 나라가 화제가 되었던 이유 를 자세하게 설명해 줘"라고 질문해 보겠습니다.

Telegram:

텔레그램은 빠르고 강력한 보안을 제공하는 채팅 앱입니다. 특히 비밀 대화, 자동 삭제 메시지, 종단 간 암호화와 같은 고급 기능으로 전 세계에서 가장 인기가 많은 채팅 앱입니다. 무엇보다 초보자도 쉽게 챗봇을 구현할 수 있게 API와 봇파더 기능을 제공합니다.

- **텔레그램 API**: 텔레그램은 개발자들이 간단한 코드로 다양한 기능을 수행하는 챗봇을 만들 수 있는 포괄적인 API를 제공합니다. 또한, API는 웹훅 통합을 지원하여 외부 서비스와 챗봇을 쉽게 연결할 수 있습니다.
- **봇파더**: 텔레그램은 봇파더라는 전용 봇을 제공하여, 쉽게 챗봇을 생성하고 관리할 수 있습니다. 봇파더는 명령어 설정, 봇 프로필 커스터마이징, API 키 생성을 위한 간단한 인터페이스를 제공합니다. 봇파더 사용법은 뒤에서 자세히 다루겠습니다.

Telegram:
텔레그램은 빠르고 강력한 보안을 제공하는 채팅 앱입니다. 특히 비밀 대화, 자동 삭제
메시지, 종단간 암호화와 같은 고급 기능으로 전 세계에서 가장 인기가 많은 채팅 앱입니 다. 무엇보다 초보자도 쉽게 챗봇을 구현할 수 있게 API와 봇파더 기능을 제공합니다.

"텔레그램 API: 텔레그램은 개발자들이 간단한 코드로 다양한 기능을 수행하는 챗봇을 만들 수 있는 포괄적인 API를 제공합니다. 또한, API는 웹훅 통합을 지원하여 외부 서비스와 챗봇을 쉽게 연결할 수 있습니다.

봇파더: 텔레그램은 봇파더라는 전용 봇을 제공하여, 쉽게 챗봇을 생성하고 관리할 수 있습니다. 봇파더는 명령어 설정, 봇 프로필 커스터마이징, API 키 생성을 위한 간단한 인터페이스를 제공합니다. 봇파더 사용법은 뒤에서 자세히 다루겠습니다.

- HTTP 요청

HTTP 요청은 클라이언트(일반적으로 웹 브라우저)에서 특정 리소스를 요청하는 서버로 보내는 메시지입니다. 요청 메시지에는 헤더와 본문이 포함됩니다.

헤더에는 요청 유형(GET, POST 등), 요청 중인 리소스의 URL, 사용 중인 HTTP 버전 등 요청에 대한 정보가 포함됩니다. 본문에는 양식 데이터 또는 JSON 페이로드와 같은 추가 데이터가 포함됩니다.

예를 들어, 브라우저의 주소창에 주소를 입력하고 엔터 키를 누르면 브라우저는 웹사이트를 호스팅하는 서버에 HTTP 요청을 보냅니다. 그러면 서버는 요청을 처리하고 다시 HTTP 응답을 보냅니다.

- HTTP 응답

HTTP 응답은 HTTP 요청에 대한 응답으로 서버에서 클라이언트로 보내는 메시지입니다. 응답 메시지에는 헤더와 본문도 포함됩니다.

헤더에는 사용 중인 HTTP 버전, 상태 코드(예: 200 OK, 404 찾을 수 없음), 쿠키 또는 캐싱 정보 등 응답에 대한 정보가 포함됩니다. 본문에는 HTML, CSS, 자바스크립트, 이미지 또는 JSON 데이터 등 전송되는 실제 콘텐츠가 포함됩니다.

예를 들어, 서버가 웹 페이지에 대한 HTTP 요청을 받으면 해당 웹 페이지의 HTML 코드가 포함된 HTTP 응답을 생성합니다. 그러면 브라우저는 HTTP 응답을 수신하고 HTML 코드를 읽은 후 화면에 웹 페이지를 표시합니다.

■ HTTP 요청
HTTP 요청은 클라이언트(일반적으로 웹 브라우저)에서 특정 리소스를 요청하는 서버로 보내는 메시지입니 다. 요청 메시지에는 헤더와 본문이 포함됩니다.
헤더에는 요청 유형(GET, POST 등), 요청 중인 리소스의 URL, 사용 중인 HTTP 버전 등 요청에 대한 정보 가 포함됩니다. 본문에는 양식 데이터 또는 JSON 페이로드와 같은 추가 데이터가 포함됩니다.
예를 들어, 브라우저의 주소창에 주소를 입력하고 엔터 키를 누르면 브라우저는 웹사이트를 호스팅하는 서버 에 HTTP 요청을 보냅니다. 그러면 서버는 요청을 처리하고 다시 HTTP 응답을 보냅니다.

■ HTTP 응답
HTTP 응답은 HTTP 요청에 대한 응답으로 서버에서 클라이언트로 보내는 메시지입니다. 응답 메시지에는 헤더와 본문도 포함됩니다.
헤더에는 사용 중인 HTTP 버전, 상태 코드(예: 200 OK, 404 찾을 수 없음), 쿠키 또는 캐싱 정보 등 응답에 대한 정보가 포함됩니다. 본문에는 HTML, CSS, 자바스크립트, 이미지 또는 JSON 데이터 등 전송되는 실제 콘텐츠가 포함됩니다.
예를 들어, 서버가 웹 페이지에 대한 HTTP 요청을 받으면 해당 웹 페이지의 HTML 코드가 포함된 HTTP 응 답을 생성합니다. 그러면 브라우저는 HTTP 응답을 수신하고 HTML 코드를 읽은 후 화면에 웹 페이지를 표 시합니다.

response

응답

- **status**: 서버가 반환한 HTTP 상태 코드가 포함됩니다. 상태 코드는 HTTP 요청의 결과를 나타내는 3자리 숫자입니다. 예를 들어 200은 "OK"(요청이 성공함), 404는 "Not Found"(요청된 리소스를 찾을 수 없음), 500은 "내부 서버 오류"(서버가 요청을 처리하는 동안 오류가 발생함)를 의미합니다.
- **header**: 서버가 반환한 HTTP 헤더를 포함하는 HTTPHeaderDict 클래스의 인스턴스입니다. 헤더는 콘텐츠 유형, 콘텐츠 인코딩 및 서버 소프트웨어와 같은 응답에 대한 추가 정보를 제공합니다.
- **data**: 서버가 반환한 실제 콘텐츠인 응답 본문이 바이트 객체로 포함됩니다. 응답 본문은 텍스트, JSON, 바이너리 데이터 또는 Content-Type 헤더에 지정된 기타 데이터 형식일 수 있습니다. 이 책에서는 텔레그램 채팅방에 사용자가 입력한 대화를 확인할 때 사용합니다.

```
response = http.request('POST', url, fields=data)

print(response.data)
```

response
응답
response = http.request('POST', url, fields=data)
print(response.data)

status: 서버가 반환한 HTTP 상태 코드가 포함됩니다. 상태 코드는 HTTP 요청의 결과를 나타내는 3자리 숫자입니다. 예를 들어 200은 "OK"(요청이 성공함), 404는 "Not Found"(요청된 리소스를 찾을 수 없음), 500은 "내부 서버 오류"(서버가 요청을 처리하는 동안 오류가 발생함)를 의미합니다.

header: 서버가 반환한 HTTP 헤더를 포함하는 HTTPHeaderDict 클래스의 인스턴스입니다. 헤더는 콘 텐츠 유형, 콘텐츠 인코딩 및 서버 소프트웨어와 같은 응답에 대한 추가 정보를 제공합니다.

data: 서버가 반환한 실제 콘텐츠인 응답 본문이 바이트 객체로 포함됩니다. 응답 본문은 텍스트, JSON, 바이너리 데이터 또는 Content-Type 헤더에 지정된 기타 데이터 형식일 수 있습니다. 이 책에서는 텔레 그램 채팅방에 사용자가 입력한 대화를 확인할 때 사용합니다.

■ 질문 입력: 사용자

사용자는 텔레그램 채팅방에서 원하는 질문을 전송합니다. 이때 ChatGPT에게 질문을 하려면 "/ask"로 메시지를 시작하고, DALL·E에게 그림을 요청하려면 "/img"로 메시지를 시작합니다. 예를 들어, "/ask 부자가 되는 법을 알려줘" 또는 "/img 차 위에 있는 고양이 그림을 그려줘"라고 작성합니다.

■ 텔레그램 서버: 텔레그램 API 활용

채팅방 내의 메시지를 로컬 PC 서버로 전송하거나, 텍스트나 사진을 채팅방에 전송하여 사용자에게 제공하는 역할을 합니다. 4.4.2절 '텔레그램 API 사용하기(120쪽)'에서 살펴본 텔레그램 API를 활용하여 로컬 PC와 통신합니다.

■ ngrok: 로컬 PC 서버를 외부로 연결해 주는 통로

로컬 PC 내부에 생성한 서버는 외부 사용자가 접근할 수 없습니다. 이때 로컬 PC 서버를 외부 사용자에게 연결해 주는 역할을 하는 것이 ngrok입니다. ngrok을 활용하면 텔레그램 서버가 실시간 채팅 정보를 로컬 서버로 전송할 수 있습니다.

■ 로컬 PC

로컬 PC는 개인이 사용하는 PC를 뜻합니다. 기본적인 파이썬 코드가 실행되고 있는 공간으로, 텔레그램 API를 활용하여 사용자와 통신합니다. 이 책에서는 FastAPI를 활용하여 로컬 PC의 서버를 열고, ngrok을 활용해 텔레그램 서버의 입력을 받습니다.

새로운 메시지가 오면 메시지를 직접 처리한 다음 OpenAI API를 통해 ChatGPT 및 DALL·E에 작업을 요청합니다. 또한, 작성이 완료된 답변 또는 그림을 텔레그램 API를 활용하여 다시 텔레그램 서버로 전송합니다.

■ 질문 입력: 사용자
사용자는 텔레그램 채팅방에서 원하는 질문을 전송합니다. 이때 ChatGPT에게 질문을 하려면 "/ask로 메시 지를 시작하고, DALL. E에게 그림을 요청하려면 "img"로 메시지를 시작합니다. 예를 들어, "/ask 부자가 되 는 법을 알려줘 또는 "Img 차 위에 있는 고양이 그림을 그려줘"라고 작성합니다.

■ 텔레그램 서버: 텔레그램 API 활용
채팅방 내의 메시지를 로컬 PC 서버로 전송하거나, 텍스트나 사진을 채팅방에 전송하여 사용자에게 제공하
는 역할을 합니다. 4.4.2절 텔레그램 API 사용하기 (120쪽)'에서 살펴본 텔레그램 API를 활용하여 로컬PC와 통신합니다.

■ngrok: 로컬 PC 서버를 외부로 연결해주는 통로
로컬PC 내부에 생성한 서버는 외부 사용자가 접근할 수 없습니다. 이때 로컬 PC 서버를 외부 사용자에게 연결해 주는 역할을 하는 것이 ngrok입니다. ngrok을 활용하면 텔레그램 서버가 실시간 채팅 정보를 로컬 서버로 전송할 수 있습니다.

■ 로컬 PC
로컬 PC는 개인이 사용하는 PC를 뜻합니다. 기본적인 파이썬 코드가 실행되고 있는 공간으로, 텔레그램 API 를 활용하여 사용자와 통신합니다. 이 책에서는 FastAPI를 활용하여 로컬 PC의 서버를 열고, ngrok을 활용 해 텔레그램 서버의 입력을 받습니다.
새로운 메시지가 오면 메시지를 직접 처리한 다음 OpenAI API를 통해 ChatGPT 및 DALL.E에 작업을 요 청합니다. 또한, 작성이 완료된 답변 또는 그림을 텔레그램 API를 활용하여 다시 텔레그램 서버로 전송합 니다.

4.5.1 FastAPI를 활용한 서버 생성

1. FastAPI를 활용해 로컬 서버 생성하기
2. ngrok을 활용해 외부에서 로컬 서버에 접속할 수 있는 주소 생성하기
3. 텔레그램 API의 웹훅을 사용하여 텔레그램 서버와 로컬 서버 연결하기

챗봇 서버를 만들자, 로컬 서버로. 즉 내 컴퓨터를 서버로 만들자. 어떻게? FastAPI로 내 컴퓨터를 서버로 만들 수 있다. 진짜?

그 다음, ngrok을 이용하여, 외부(텔레그램)에서 로컬 서버로 접속하기 위한 주소를 발급 받는다. 그 다음, 이 주소를 텔레그램 API의 웹훅과 연결하면 끝! 이걸 어떻게 하는지 살펴보자.

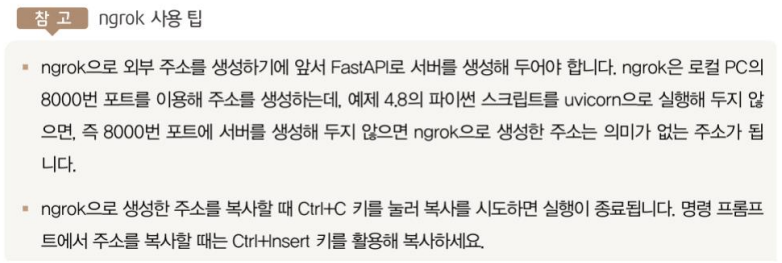
4.5.1 FastAPI를 활용한 서버 생성

1. FastAPI를 활용해 로컬 서버 생성하기
2. ngrok을 활용해 외부에서 로컬 서버에 접속할 수 있는 주소 생성하기
3. 텔레그램 API의 웹훅을 사용하여 텔레그램 서버와 로컬 서버 연결하기

챗봇 서버를 만들자, 로컬 서버로. 즉 내 컴퓨터를 서버로 만들자. 어떻게? FastAPI로 내 컴퓨터를 서버로 만들 수 있다. 진짜?

그 다음, ngrok을 이용하여, 외부(텔레그램)에서 로컬 서버로 접속하기 위한 주소를 발급 받는다. 그 다음, 이 주소를 텔레그램 API의 웹훅과 연결하면 끝! 이걸 어떻게 하는지 살펴보자.

ngrok을 FastAPI서버와 연결


참 고 ngrok 사용 팁

- ngrok으로 외부 주소를 생성하기에 앞서 FastAPI로 서버를 생성해 두어야 합니다. ngrok은 로컬 PC의 8000번 포트를 이용해 주소를 생성하는데, 예제 4.8의 파이썬 스크립트를 uvicorn으로 실행해 두지 않으면, 즉 8000번 포트에 서버를 생성해 두지 않으면 ngrok으로 생성한 주소는 의미가 없는 주소가 됩니다.
- ngrok으로 생성한 주소를 복사할 때 Ctrl+C 키를 눌러 복사를 시도하면 실행이 종료됩니다. 명령 프롬프트에서 주소를 복사할 때는 Ctrl+Insert 키를 활용해 복사하세요.

ngrok을 FastAPI서버와 연결

참고 ngrok 사용 팁

angrok으로 외부 주소를 생성하기에 앞서 FastAPI로 서버를 생성해 두어야 합니다. ngrok은 로컬 PC의 8000번 포트를 이용해 주소를 생성하는데, 예제 4.8의 파이썬 스크립트를 uvicorn으로 실행해 두지 않으면, 즉 8000번 포트에 서버를 생성해 두지 않으면 ngrok으로 생성한 주소는 의미가 없는 주소가 됩 니다.

ngrok으로 생성한 주소를 복사할 때 Ctrl+C 키를 눌러 복사를 시도하면 실행이 종료됩니다. 명령 프롬프트에서 주소를 복사할 때는 Ctrl+Insert 키를 활용해 복사하세요.