```python
import json
from dotenv import load_dotenv
import gradio as gr
import openai
import requests
```

```python
load_dotenv()
```

```
True
```

```python
client = openai.OpenAI()
```

```python
chat_history=[]   # 리스트 초기화 - 대화 내용 기록.
def respond_test(prompt,chat_history):
    print(chat_history)
    chat_history.append((prompt, "감사합니다."))    # (user 질문, assistant 답변)
    return "", chat_history
    # 튜플 리턴. chat_history 는 (user 질문, assistant 답변) 튜플을 저장하는 리스트

respond_test("안녕하세요.",chat_history)
```

```
[]
('', [('안녕하세요.', '감사합니다.')])
```

```python
respond_test("안녕히 가세요",chat_history)
```

```
[('안녕하세요.', '감사합니다.')]
('', [('안녕하세요.', '감사합니다.'), ('안녕히 가세요', '감사합니다.')])
```

```python
model = "gpt-4.1-mini"
def respond(prompt,chat_history):   # ui 와 연결할 함수
    instruction = """
    너는 피자 가격을 안내하는 챗봇이다.
    주요 임무는 다음과 같다:
    1. 현재는 가격 정보만 제공하는 임무를 맞고 있으니 그 외의 다른 정보는 제공할 수 없음.
    2. 피자의 종류에 대한 가격 안내만 할 수 있음.
    2. 불필요한 잡담은 최소화하고, 주문과 관련된 대화에 집중한다.
    3. 항상 정중하고 친근한 말투를 유지한다.
    피자 주문 이외의 다른 요청은 '챗봇 기능과 다른 질문입니다.' 라고 답변해.
    """
    # 너는 피자 주문을 돕는 챗봇이다.
    # 친절하고 간단명료하게 대화하며, 고객이 원하는 피자를 정확히 주문할 수 있도록 안내한다.
    # 1. 고객의 주문 의도를 파악한다 (피자 종류, 사이즈, 토핑, 수량, 음료, 사이드 메뉴 등).
    # 2. 필요한 경우 추가 질문을 하여 주문 정보를 완성한다.
    # 3. 주문 내용을 고객에게 다시 확인시켜준다.
    # 4. 결제나 배달 관련 정보는 기본적인 안내만 하고, 실제 결제는 외부 시스템에서 처리된다고 설명한다.
    messages = [{"role": "system", "content": instruction},{"role": "user", "content": prompt}]

    res = client.chat.completions.create(
        model=model,
        messages=messages
    )
    answer = res.choices[0].message.content
    # 여기까지가 GPT 와 통신하고 응답받는 코드

    chat_history.append((prompt, answer))
    print(chat_history)
    return "", chat_history
```

```python
chat_history=[]
result = respond('하이',chat_history)
result
```

```
[('하이', '챗봇 기능과 다른 질문입니다.')]
('', [('하이', '챗봇 기능과 다른 질문입니다.')])
```

```python
result = respond('대한민국의 수도는 어디야?',chat_history)
result
```

```
[('하이', '챗봇 기능과 다른 질문입니다.'), ('대한민국의 수도는 어디야?', '챗봇 기능과 다른 질문입니다.')]
('', [('하이', '챗봇 기능과 다른 질문입니다.'), ('대한민국의 수도는 어디야?', '챗봇 기능과 다른 질문입니다.')])
```

```python
chat_history=[]
with gr.Blocks() as app:
    gr.Markdown("#  챗봇")
```

```
    gr.Markdown(
        """
        ## Chat
        얻고 싶은 정보에 대해 질문해보세요.
        """
    )
    chatbot = gr.Chatbot(label="Chat History")  # 대화 기록을 보여주는 컴포넌트
    prompt = gr.Textbox(label="Input prompt",interactive=True)
    clear = gr.ClearButton([prompt, chatbot])
    # 텍스트박스에서 엔터-> submit. gpt api 요청 보내기
    prompt.submit(respond, [prompt, chatbot], [prompt, chatbot])
                # 함수,    함수의 입력,  함수의 리턴 ("",chat_hitory)
```

```
/tmp/ipython-input-1255310847.py:10: UserWarning: You have not specified a value for the `type` parameter. Defaulting to the 'tuples' format f
  chatbot = gr.Chatbot(label="Chat History")  # 대화 기록을 보여주는 컴포넌트
```

```
app.launch(inline=False)
```

```
It looks like you are running Gradio on a hosted Jupyter notebook, which requires `share=True`. Automatically setting `share=True` (you can tu

Colab notebook detected. To show errors in colab notebook, set debug=True in launch()
* Running on public URL: https://eec9b0e289fc5b9aad.gradio.live

This share link expires in 1 week. For free permanent hosting and GPU upgrades, run `gradio deploy` from the terminal in the working directory
```

```
app.close()
```

```
코딩을 시작하거나 AI로 코드를 생성하세요.
```