```
import base64
from io import BytesIO
from PIL import Image
```

Hàm / Lệnh	Chức năng
import base64	Nhập module để mã hóa và giải mã dữ liệu bằng Base64.
from io import BytesIO	Nhập BytesIO để xử lý dữ liệu nhị phân trong bộ nhớ thay vì lưu vào file.
from PIL import Image	Nhập thư viện Pillow để xử lý hình ảnh.

```
def artist(city):
    image_response = openai.images.generate(
            model="dall-e-3",
            prompt=f"An image representing a vacation in {city}, showing
tourist spots and everything unique about {city}, in a vibrant pop-art
style",
            size="1024x1024",
            n=1,
            response_format="b64_json",
        )
    image_base64 = image_response.data[0].b64_json
    image_data = base64.b64decode(image_base64)
    return Image.open(BytesIO(image_data))

image = artist("New York City")
display(image)
```

Hàm / Lệnh	Chức năng
def artist(city):	Hàm tạo hình ảnh về một thành phố theo phong cách pop-art.
openai.images.generate(...)	Gọi OpenAI DALL·E để tạo hình ảnh dựa trên mô tả về thành phố.
image_response.data[0].b64_json	Lấy dữ liệu hình ảnh dưới dạng Base64 từ phản hồi của API.
base64.b64decode(image_base64)	Giải mã dữ liệu Base64 thành dữ liệu nhị phân.
Image.open(BytesIO(image_data))	Chuyển đổi dữ liệu nhị phân thành ảnh PIL.
image = artist("New York City")	Gọi hàm artist để tạo hình ảnh về New York City.
display(image)	Hiển thị hình ảnh đã tạo.

```
from pydub import AudioSegment
from pydub.playback import play

def talker(message):
    response = openai.audio.speech.create(
      model="tts-1",
      voice="onyx",    # Also, try replacing onyx with alloy
      input=message
    )

    audio_stream = BytesIO(response.content)
    audio = AudioSegment.from_file(audio_stream, format="mp3")
    play(audio)

talker("Well, hi there")
```

Hàm / Lệnh	Chức năng
def talker(message):	Hàm tạo và phát giọng nói từ văn bản sử dụng OpenAI TTS.
openai.audio.speech.create(...)	Gọi API OpenAI để tạo giọng nói từ nội dung văn bản.
BytesIO(response.content)	Chuyển đổi dữ liệu âm thanh từ phản hồi API thành luồng nhị phân.
AudioSegment.from_file(audio_stream, format="mp3")	Tải dữ liệu âm thanh vào đối tượng AudioSegment.
play(audio)	Phát đoạn âm thanh đã tạo.
talker("Well, hi there")	Gọi hàm talker để tạo và phát âm thanh từ chuỗi "Well, hi there".

```
def chat(history):
    messages = [{"role": "system", "content": system_message}] + history
    response = openai.chat.completions.create(model=MODEL,
messages=messages, tools=tools)
    image = None

    if response.choices[0].finish_reason=="tool_calls":
        message = response.choices[0].message
        response, city = handle_tool_call(message)
        messages.append(message)
        messages.append(response)
        image = artist(city)
        response = openai.chat.completions.create(model=MODEL,
messages=messages)

    reply = response.choices[0].message.content
    history += [{"role":"assistant", "content":reply}]


    talker(reply)

    return history, image
```

Hàm/Lệnh	Chức năng
chat(history)	Xử lý hội thoại, gọi AI để phản hồi tin nhắn, tạo hình ảnh và phát âm thanh nếu cần.

messages = [{"role": "system", "content": system_message}] + history	Khởi tạo danh sách tin nhắn, bao gồm tin nhắn hệ thống và lịch sử hội thoại.
response = openai.chat.completions.create(model=MODEL, messages=messages, tools=tools)	Gọi OpenAI API để tạo phản hồi dựa trên hội thoại và các công cụ hỗ trợ.
if response.choices[0].finish_reason =="tool_calls":	Kiểm tra xem phản hồi có yêu cầu sử dụng công cụ không.
message = response.choices[0].message	Lấy nội dung phản hồi từ AI.
response, city = handle_tool_call(message)	Xử lý phản hồi bằng tool, trích xuất tên thành phố nếu có.
messages.append(message); messages.append(response)	Cập nhật danh sách tin nhắn với phản hồi từ tool.
image = artist(city)	Tạo hình ảnh đại diện cho thành phố nếu có.
response = openai.chat.completions.create(model=MODEL, messages=messages)	Gửi lại yêu cầu đến OpenAI sau khi xử lý tool.
reply = response.choices[0].message.content	Lấy nội dung phản hồi cuối cùng từ AI.
history += [{"role":"assistant", "content":reply}]	Cập nhật lịch sử hội thoại với phản hồi của AI.
talker(reply)	Chuyển phản hồi thành giọng nói và phát âm thanh.
return history, image	Trả về lịch sử hội thoại đã cập nhật và hình ảnh (nếu có).

```
with gr.Blocks() as ui:
    with gr.Row():
        chatbot = gr.Chatbot(height=500, type="messages")
        image_output = gr.Image(height=500)
    with gr.Row():
        entry = gr.Textbox(label="Chat with our AI Assistant:")
    with gr.Row():
        clear = gr.Button("Clear")

    def do_entry(message, history):
        history += [{"role":"user", "content":message}]
        return "", history

    entry.submit(do_entry, inputs=[entry, chatbot], outputs=[entry,
chatbot]).then(
        chat, inputs=chatbot, outputs=[chatbot, image_output]
    )
    clear.click(lambda: None, inputs=None, outputs=chatbot, queue=False)

ui.launch(inbrowser=True)
```

Hàm/Lệnh	**Chức năng**
with gr.Blocks() as ui:	Khởi tạo giao diện Gradio với bố cục linh hoạt.
with gr.Row(): chatbot = gr.Chatbot(height=500,	Tạo hộp chat có chiều cao 500px để hiển thị hội thoại.

type="messages")	
with gr.Row(): image_output = gr.Image(height=500)	Tạo vùng hiển thị hình ảnh có chiều cao 500px.
with gr.Row(): entry = gr.Textbox(label="Chat with our AI Assistant:")	Tạo ô nhập liệu để người dùng nhập tin nhắn.
with gr.Row(): clear = gr.Button("Clear")	Tạo nút "Clear" để xóa nội dung hội thoại.
def do_entry(message, history):	Định nghĩa hàm xử lý khi người dùng gửi tin nhắn.
history += [{"role":"user", "content":message}]	Cập nhật lịch sử hội thoại với tin nhắn mới từ người dùng.
return "", history	Trả về giá trị rỗng cho ô nhập liệu và cập nhật lịch sử hội thoại.
entry.submit(do_entry, inputs=[entry, chatbot], outputs=[entry, chatbot])	Gọi hàm do_entry khi người dùng nhấn Enter trong ô nhập liệu.
.then(chat, inputs=chatbot, outputs=[chatbot, image_output])	Sau khi cập nhật lịch sử hội thoại, gọi hàm chat để xử lý và hiển thị phản hồi.
clear.click(lambda: None, inputs=None, outputs=chatbot, queue=False)	Khi nhấn nút "Clear", đặt lại hội thoại về trạng thái rỗng.
ui.launch(inbrowser=True)	Khởi chạy giao diện Gradio trong trình duyệt.