Instalar dependencias

Ejecuta la siguiente celda para instalar las bibliotecas necesarias:

```
!pip install openai python-dotenv
```

Crear el archivo .env

Crea un archivo .env para almacenar las variables de configuración. Asegúrate de reemplazar los valores de los placeholders con los datos de tu recurso de **Azure OpenAI**.

```
# Crear archivo .env
with open('.env', 'w') as f:
  f.write('AZURE_OPENAI_ENDPOINT=<tu_endpoint_de_azure>\n')
  f.write('AZURE_OPENAI_KEY=<tu_clave_de_azure>\n')
  f.write('AZURE_OPENAI_DEPLOYMENT=<nombre_de_tu_modelo_deployado>\n')
```

Cargar las variables de entorno

Carga las variables de entorno desde el archivo .env que acabas de crear:

```
import os
from dotenv import load_dotenv

# Cargar las variables de entorno desde el archivo .env
load_dotenv()

# Obtener los valores desde las variables de entorno
azure_oai_endpoint = os.getenv("AZURE_OPENAI_ENDPOINT")
azure_oai_key = os.getenv("AZURE_OPENAI_KEY")
azure_oai_deployment = os.getenv("AZURE_OPENAI_DEPLOYMENT")
```

Importar la biblioteca OpenAI y configurar el cliente

Importa la biblioteca OpenAI y configura el cliente para interactuar con el servicio de Azure:

```
import openai

# Inicializar el cliente de Azure OpenAI
openai.api_key = azure_oai_key
openai.api_base = azure_oai_endpoint
openai.api_type = "azure"
openai.api_version = "2024-02-15-preview"
```

Definir el mensaje del sistema

Crea un mensaje del sistema que servirá como contexto para la interacción con el modelo:

```
# Crear un mensaje del sistema
system_message = """I am a hiking enthusiast named Forest who helps people discover hikes in their area.
    If no area is specified, I will default to near Rainier National Park.
    I will then provide three suggestions for nearby hikes that vary in length.
    I will also share an interesting fact about the local nature on the hikes when making a recommendation.
    """
```

Función para interactuar con el modelo

Crea una función que permita enviar consultas al modelo y obtener respuestas, manteniendo el historial de la conversación:

```
# Inicializar el array de mensajes
messages_array = [{"role": "system", "content": system_message}]
```

```
def get_hike_recommendations_with_history(user_input):
  # Añadir el mensaje del usuario al array de mensajes
  messages_array.append({"role": "user", "content": user_input})

  # Crear la solicitud
  response = openai.ChatCompletion.create(
    engine=azure_oai_deployment,
    messages=messages_array,
    temperature=0.7,
    max_tokens=400
  )

  # Obtener el texto generado
  generated_text = response['choices'][0]['message']['content']

  # Añadir la respuesta generada al array de mensajes
  messages_array.append({"role": "assistant", "content": generated_text})

  return generated_text
```

Conversación interactiva

Ahora, añade un bucle while para mantener una conversación continua. Esto permitirá al usuario hacer múltiples preguntas y recibir respuestas del modelo:

```
# Iniciar una conversación
print("¡Bienvenido! Puedes preguntarme sobre rutas de senderismo. Escribe 'salir' para terminar.")

while True:
  # Obtener la entrada del usuario
```

```
user_input = input("Usuario: ")


# Salir del bucle si el usuario escribe 'salir'
if user_input.lower() == 'salir':
    print("¡Hasta luego!")
    break


# Obtener la respuesta del modelo
response = get_hike_recommendations_with_history(user_input)


# Mostrar la respuesta
print("Asistente: ", response)
```

Ejecutar el cuaderno

Ejecuta cada celda de tu cuaderno de Google Colab en el orden indicado para ver cómo el modelo de Azure OpenAI genera respuestas basadas en tus preguntas.