This Python code sets up a chat interface with the OpenAI GPT-3.5-turbo model for career counseling. The code allows a user to interact with an AI-powered career counselor by inputting questions or statements and receiving responses.

1. Import the **openai** Library:

*import* openai

This line imports the OpenAI Python library, which is used to interact with the GPT-3.5-turbo model.

1. Set the OpenAI API Key:

openai.api\_key = "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"

This line sets the API key for authentication. You should replace the API key with your own key if you want to run this code.

1. Initialize an Empty List for Messages:

messages = []

An empty list called **messages** is created to store the conversation between the user and the AI.

1. Add a System Message:

messages.append({"role": "system", "content": "You are now chatting with Career Counselor ChatGPT. Please ask any career-related questions."})

This message is a system message that serves as an introductory message when the conversation starts. It informs the user that they are chatting with a career counselor AI.

1. Print a Startup Message:

print("Your Career Counselor AI is ready!")

This line prints a message to inform the user that the AI career counselor is ready to interact.

1. Start an Infinite Loop for User Interaction:

*while* True:

This **while** loop allows the user to continuously interact with the AI until they choose to exit the program.

1. Get User Input:

user\_input = input("User: ")

The code prompts the user to input their message or question, which is stored in the **user\_input** variable.

1. Append User Message to the **messages** List:

messages.append({"role": "user", "content": user\_input})

The user's input is added to the **messages** list as a user message.

1. Generate a Response from the AI:

response = openai.ChatCompletion.create(

        model="gpt-3.5-turbo",

        messages=messages

    )

This code uses the OpenAI GPT-3.5-turbo model to generate a response based on the conversation history stored in the **messages** list.

1. Extract and Append AI's Reply:

reply = response["choices"][0]["message"]["content"]

messages.append({"role": "assistant", "content": reply})

The response generated by the AI is extracted from the API response and stored in the **reply** variable. It is then added to the **messages** list as an assistant's reply.

1. Display the AI's Reply:

print("AI: " + reply + "\n")

The AI's reply is printed to the console, and the loop continues, allowing the user to provide more input and receive further responses.

This code essentially creates a simple conversational interface with the GPT-3.5-turbo model for career counseling, where the user and the AI take turns exchanging messages. The conversation history is retained in the **messages** list to provide context for generating meaningful responses.