E:\voice.py

```python
# Imports the speech recognition library for voice commands
import speech_recognition as sr

# Imports the library for GUI automation
import pyautogui

# Imports the webbrowser library to open web pages
import webbrowser

# Imports the OpenAI library to interact with GPT-3
import openai

# Imports the os library for interacting with the operating system
import os

# Imports Google's Text-to-Speech engine
from gtts import gTTS

# Imports AudioSegment for handling audio files
from pydub import AudioSegment

# Imports function to load environment variables from a .env file
from dotenv import load_dotenv

# Loads environment variables from a .env file
load_dotenv()

# Retrieves the OpenAI API key from environment variables
OPENAI_KEY = os.getenv("OPENAI_KEY")

# Sets the OpenAI API key for use in the program
openai.api_key = OPENAI_KEY

# Gets commands from the user
def listen_for_command():
    recognizer = sr.Recognizer()

    # Opens the microphone for listening
    with sr.Microphone() as source:
        print('Listening for commands...')

        # Adjusts the recognizer sensitivity to ambient noise
        recognizer.adjust_for_ambient_noise(source)

        # Listens for the first phrase and extracts the audio
        audio = recognizer.listen(source)

    try:
        # Recognizes speech using Google's speech recognition
        command = recognizer.recognize_google(audio)
        print("Google Speech Recognition thinks you said: ", command)

        # Returns the recognized command in lowercase
        return command.lower()
    #except' catches specific exceptions that the 'try' block may encounter.
    except sr.UnknownValueError:
        print("Google Speech Recognition could not understand audio")
        return None
    #except' catches specific exceptions that the 'try' block may encounter.
    except sr.RequestError as e:
        print(f"Could not request results from Google Speech Recognition service; {e}")
        return None

# Converts text to speech
def text_to_speech(response_text):
    print(response_text)
    tts = gTTS(text=response_text, lang="en")

    # Saves the spoken text to an mp3 file
    tts.save("response.mp3")

    # Converts the mp3 file to an audio segment
    sound = AudioSegment.from_mp3("response.mp3")

    # Exports the audio segment as a wav file
    sound.export("response.wav", format="wav")

    # Plays the wav file using the system's default audio player
    os.system("afplay response.wav")

# Get Response From GPT-3
def chatGPT_response(prompt):
    # Sends the prompt to GPT-3 and returns the response
    response = openai.chat.completions.create(
        messages=[
            {
                "role": "user",
                "content": prompt,
            }
        ],
        model="gpt-3.5-turbo",
    )

    # Returns the content of the response
    return response.choices[0].message.content

# Main function that runs the program
def main():
    text_to_speech("Hello What Can I Do For You Today?")
    while True:
        # Listens for a voice command
        command = listen_for_command()
        if command:
            # Checks if the command contains certain keywords
            if any(word in command for word in ["who", "what", "when", "where", "how", "should", "why", "will", "would", "can", "could", "do", "does", "is", "are", "am", "was", "were", "have", "has", "had", "which",]):
                # Gets a response from GPT-3
                response = chatGPT_response(command)

                # Converts the response to speech
                text_to_speech(response)

            # open chrome if user says open chrome
            if "open chrome" in command:
                text_to_speech("Opening Chrome.")

                # Opens Google Chrome to the Google homepage
                webbrowser.open('http://google.com')

            # exit if user says exit
            if "exit" in command:
                text_to_speech("Goodbye.")

                # Breaks the loop, ending the program
                break
            else:
                text_to_speech("Sorry, I don't understand that command.")

# Checks if the script is the main program and runs it
if __name__ == '__main__':
    main()

```