**Lab 11: Enabling conversation using Azure Speech service**

**Introduction** The Speech service provides speech to text and text to speech capabilities with an Azure Speech resource. You can transcribe speech to text with high accuracy, produce natural-sounding text to speech voices, translate spoken audio, and use speaker recognition during conversations. Speak into the microphone to start a conversation with Azure OpenAI. The Speech service recognizes your speech and converts it into text (speech to text). Your request as text is sent to Azure OpenAI. The Speech service text to speech (TTS) feature synthesizes the response from Azure OpenAI to the default speaker.

**Prerequisites**

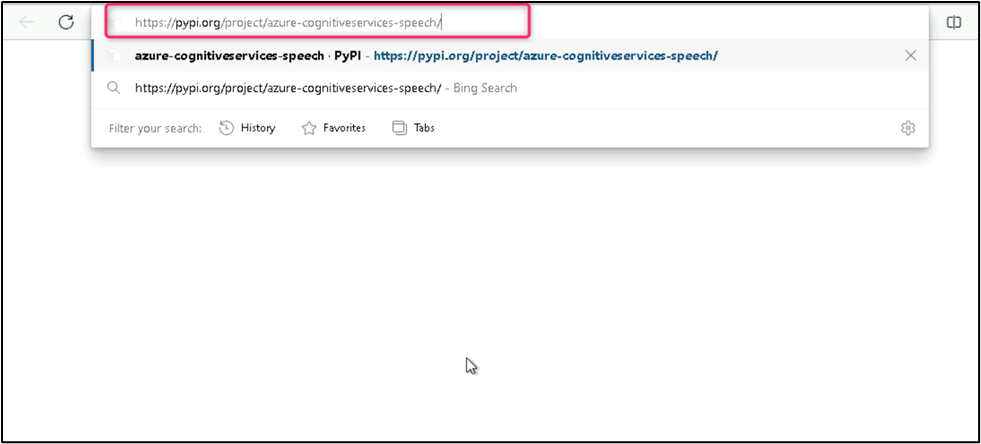
* Before starting the Lab 09 complete the Lab 01- Provisioning Azure OpenAI resource and Lab 04-Generating an E-mail content using Azure OpenAI

**Objectives**

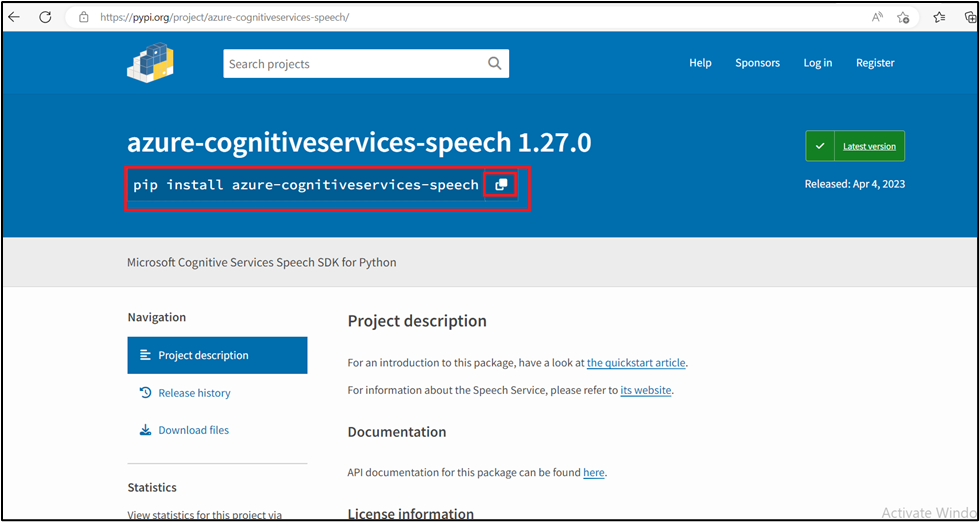
* To install the Speech SDK, python libraries, and Microsoft Visual C++ Redistributable for Visual Studio.
* To deploy the text-davinci-003 model, create a Speech resource in the Azure portal, set environment variables, and recognize speech from a microphone.

**Task 1: Use the Speech SDK and Install the Python libraries**

1. Open your browser, navigate to the address bar, and type or paste the following URL: [**https://pypi.org/project/azure-cognitiveservices-speech/**](urn:gd:lg%F0%9F%85%B0%EF%B8%8Fsend-vm-keys), then press the **Enter** button.

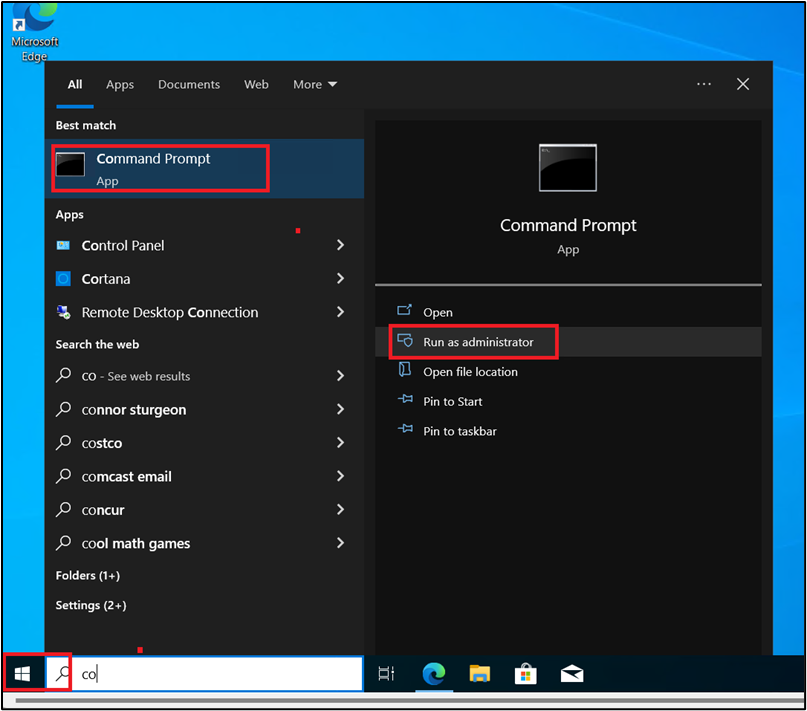


1. To install the speech SDK for Python, copy the azure-cognitiveservices-speech command – **pip install azure-cognitiveservices-speech**



1. Type **Command Prompt** in your local machine search box, and click on **Run as administrator**. On **Do you allow this app to make changes on your device** dialog box, click on the **Yes** button.

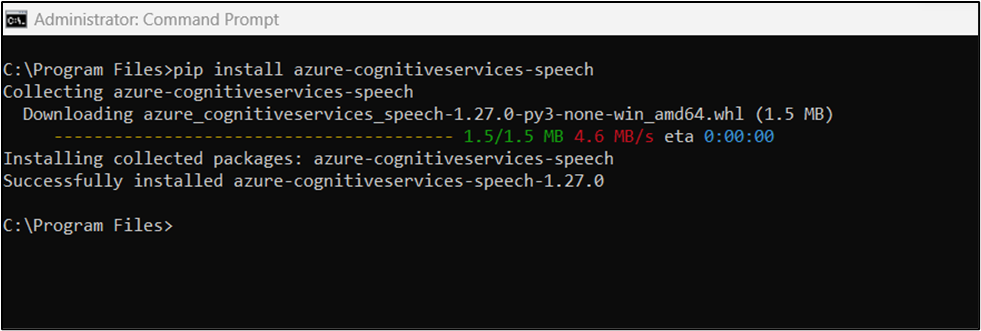
**Important Note**: You need to change the current directory to the **Program Files** directory (The command used to move back to the previous directory is **cd .. [space after cd then two dots],** the command used to move to the next directory is **cd )**



1. To install the Speech SDK for Python in Program files directory, run the following command.

**cli**

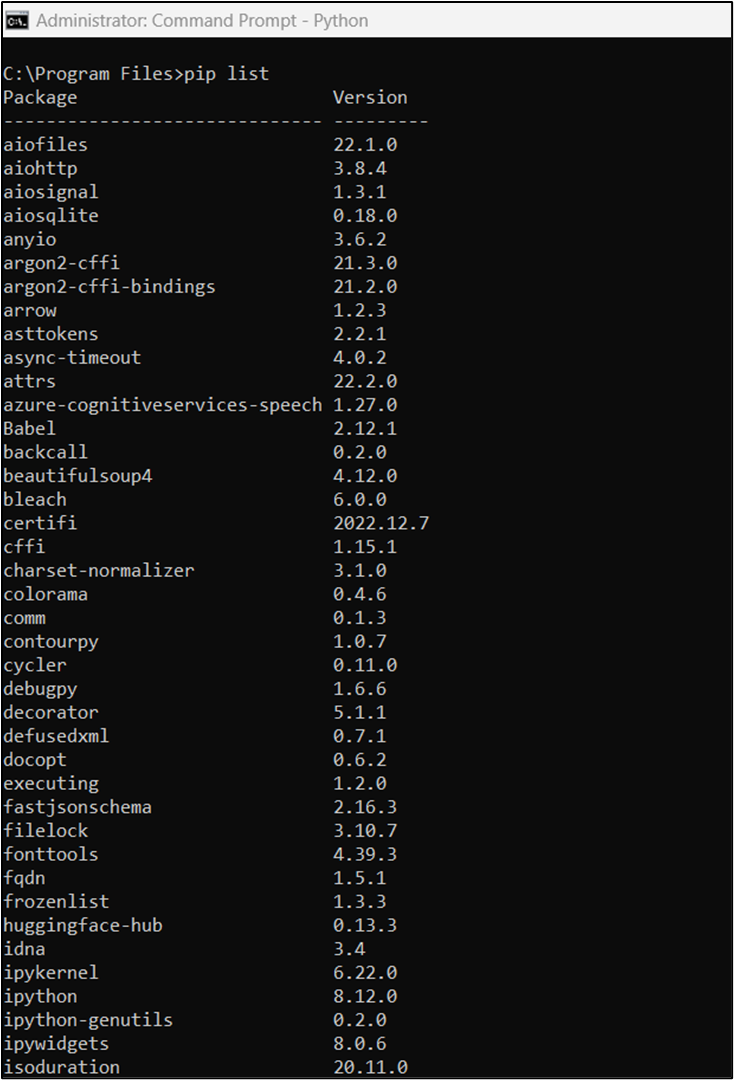
pip install azure-cognitiveservices-speech



1. You can check which Speech SDK for Python version is currently installed by inspecting the **azure.cognitiveservices.speech.version** variable. For example, run this command in a terminal:

**cli**

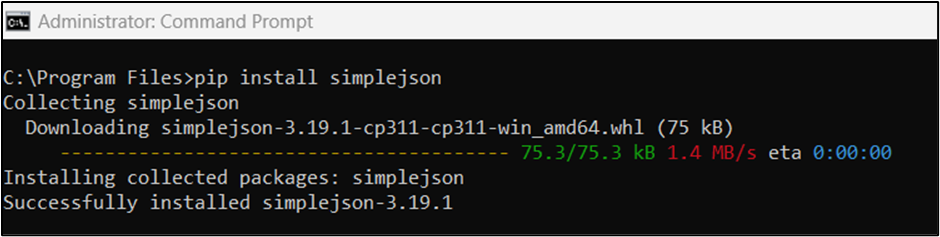
pip list



1. Run the following command to install the JSON Python libraries.

**cli**

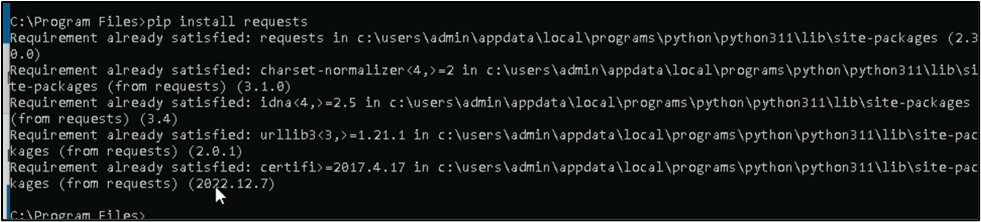
pip install simplejson



1. Run the following command to install the **requests** Python libraries.

**cli**

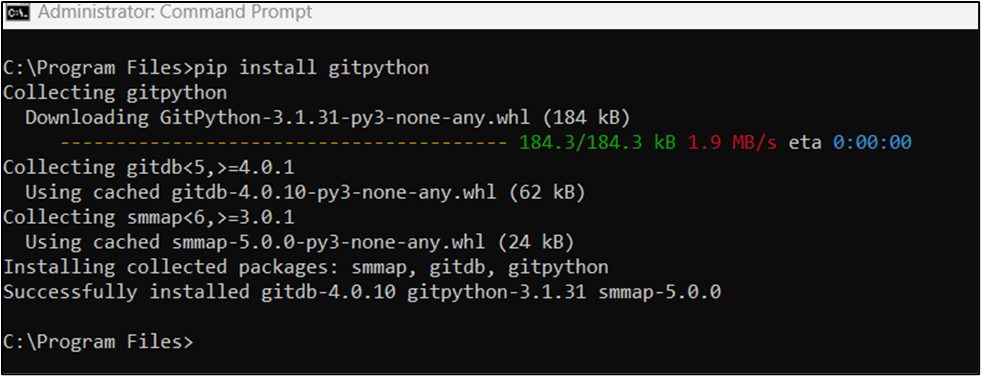
pip install requests



1. Run the following command to install the **gitpython** Python libraries.

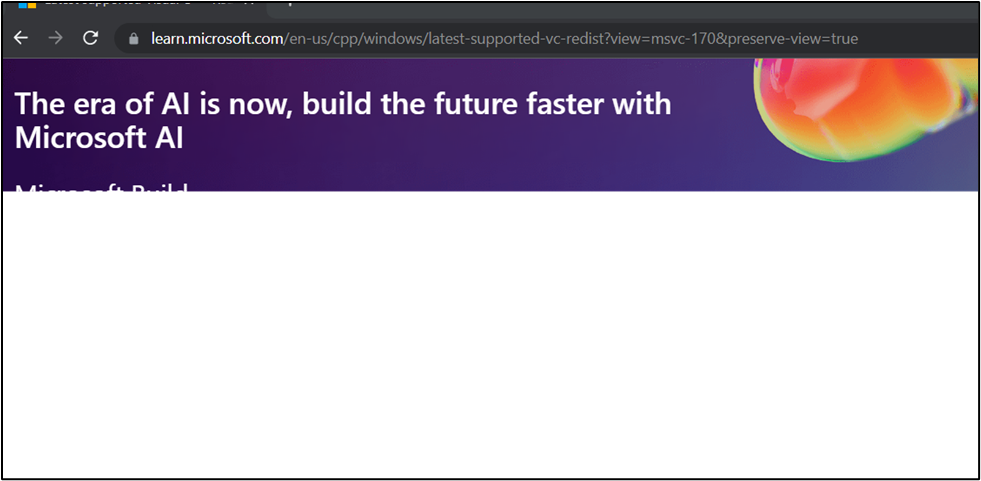
**cli**

pip install gitpython

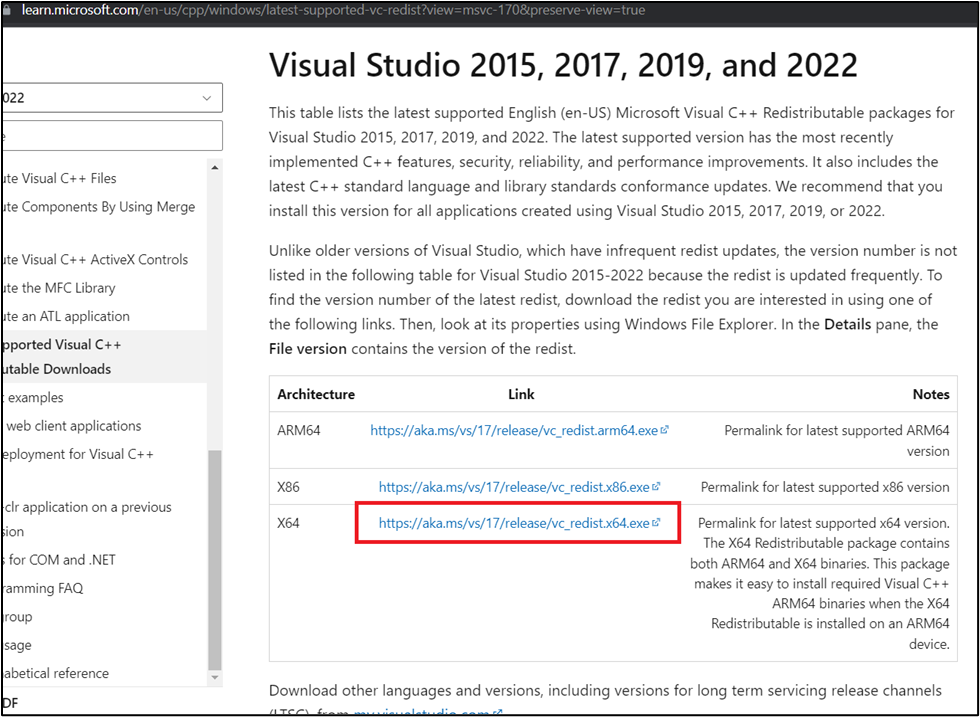


**Task 2: install the Microsoft Visual C++ Redistributable for Visual Studio**

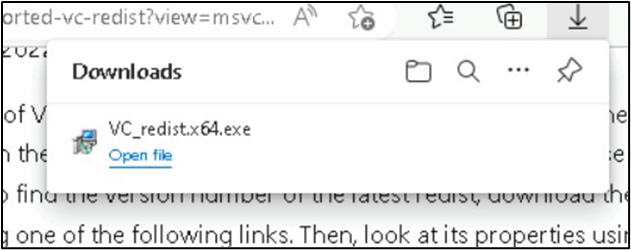
1. Open your browser, navigate to the address bar, type or paste the following URL: [**https://learn.microsoft.com/en-us/cpp/windows/latest-supported-vc-redist?view=msvc-170&preserve-view=true**](urn:gd:lg%F0%9F%85%B0%EF%B8%8Fsend-vm-keys), then press the **Enter** button.



1. Scroll down, and click on the [**https://aka.ms/vs/17/release/vc\_redist.x64.exe**](urn:gd:lg%F0%9F%85%B0%EF%B8%8Fsend-vm-keys).



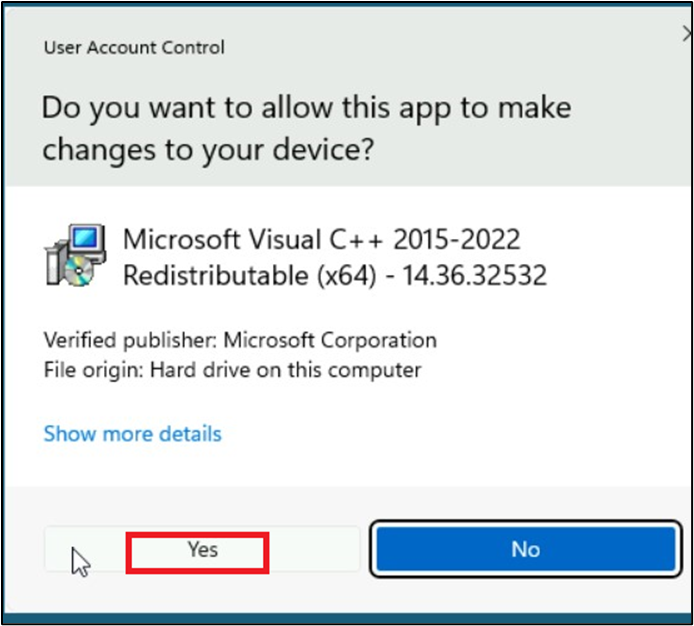
1. Microsoft Visual C++ Redistributable for Visual Studio X64 file will be downloaded. Click on the downloaded file to set up VC\_redist.x64.exe (In case, you did not find the VS Code setup file on your window, then navigate to the Downloads folder of your system and continue the installation process).



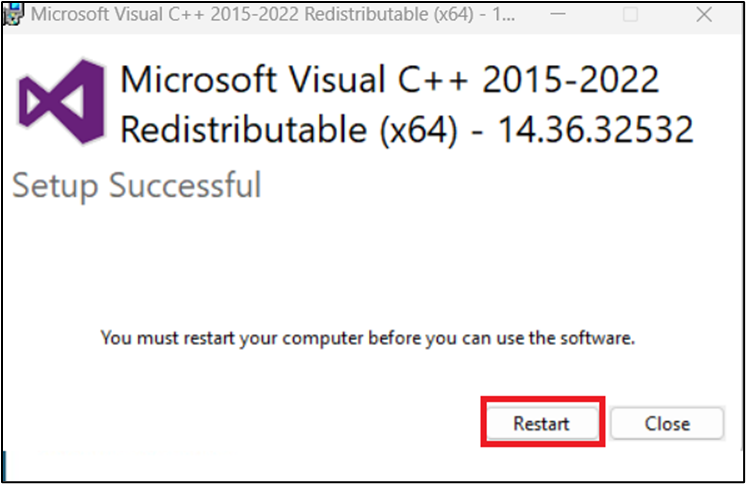
1. In the **Microsoft Visual C++ 2015-2022 Redistributable(x64)-14** window, select **I agree to the license terms and conditions** radio button and click on the **Install**



1. On User account control tab select **Yes**



1. Installing this package for the first time might require a restart. Select **Restart**



**Task 3: Deploy gpt-35-turbo model**

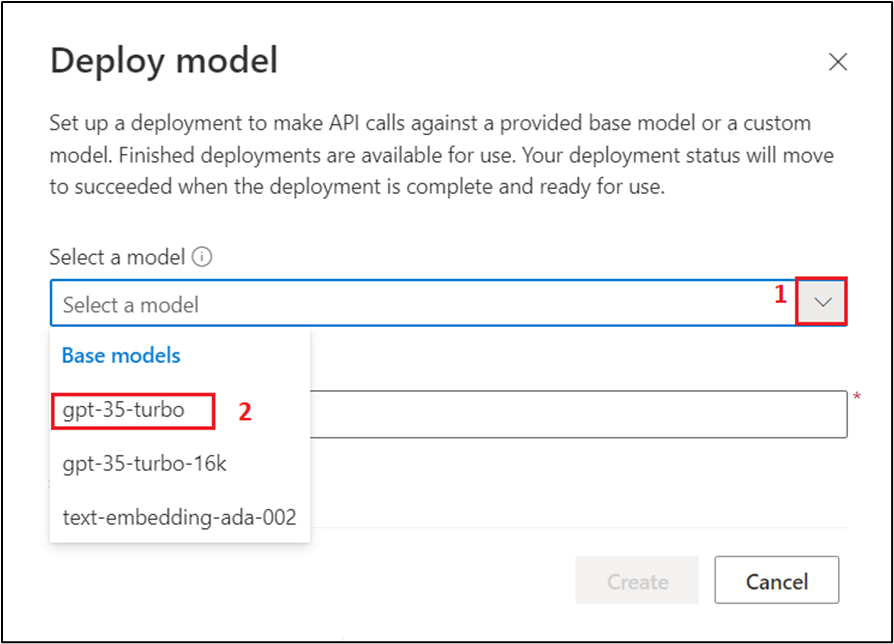
1. On the **Azure OpenAI Studio** homepage, click on **Create new deployment** button.



1. In the **Deployments** page, click on **+Create new deployment**.



1. In the **Deploy model dialog** box, under the **Model name** field, click on the V chevron button; navigate and select carefully **gpt-35-turbo**.



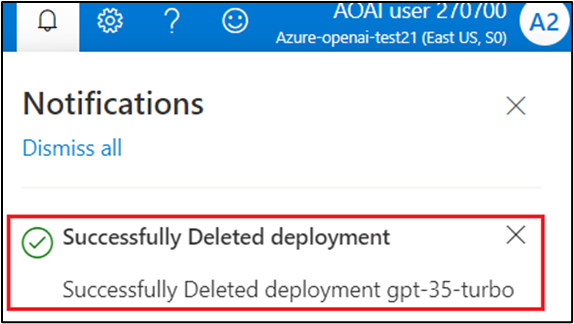
1. Select the **Model version** as **Auto-update to default**, in the **Deployment name** field, enter **gpt-35-turbo**, and click on the **Create** button.

A screenshot of a computer

Description automatically generated

**Note:** Azure OpenAI includes multiple models, each optimized for a different balance of capabilities and performance. In this exercise, you’ll use the GPT-35-Turbo model, which is a good general model for summarizing and generating natural language and code. For more information about the available models in Azure OpenAI, see Models in the Azure OpenAI documentation.

1. You will see a notification – **Successfully Created deployment** when the deployment is succeeded. (You can also view the notification by clicking on the bell icon beside **Cognitive Services | Azure OpenAI Studio**).



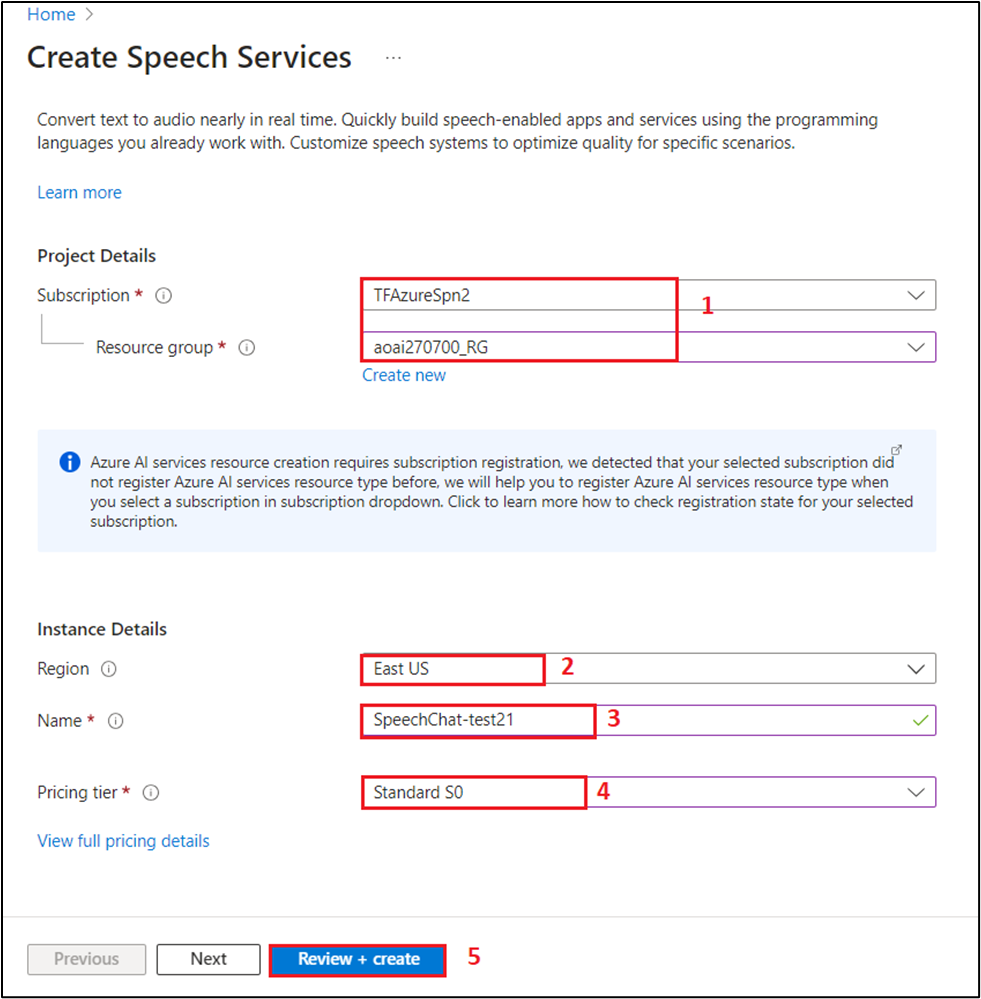
**Task 4: Create a Speech resource in the Azure portal**

1. Open your browser, navigate to the address bar, and paste the following URL: [**https://portal.azure.com/#create/Microsoft.CognitiveServicesSpeechServices**](urn:gd:lg%F0%9F%85%B0%EF%B8%8Fsend-vm-keys), then press the **Enter** button.



1. On the **Create Speech Services** page, provide the following information and then click on **Review+create** button.

| **Field** | **Description** |
| --- | --- |
| **Subscription** | Select your Azure OpenAI subscription |
| **Resource group** | Select your Resource group(that you have created in **Lab 1**) |
| **Region** | East US |
| **Name** | **SpeechChat-testXX** (XX can be unique number) |
| **Pricing Tier** | Standard S0 |

1. 
2. Once the Validation is passed, click on the **Create** button.

A screenshot of a computer

Description automatically generated

1. Wait for few minutes, until the deployment is completed.

A screenshot of a computer

Description automatically generated

**Task 5: Get the keys for Speech service**

1. After the deployment of the speech services is successfully completed, click on **Go to resource** button.

A screenshot of a computer

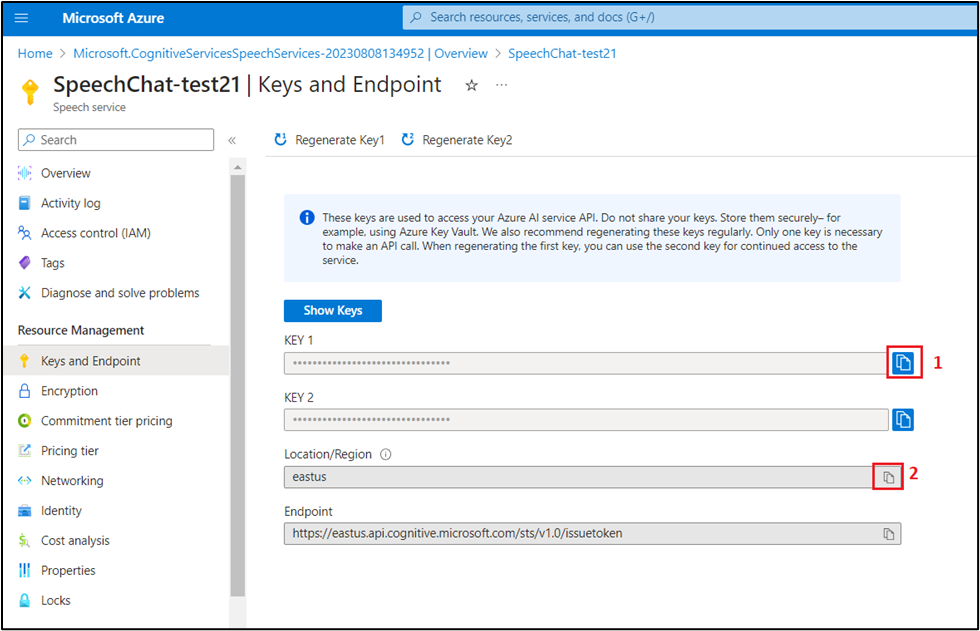
Description automatically generated

1. On **SpeechChat-testXX** window, navigate to **Resource Management** section and click on **Keys and Endpoints**.

A screenshot of a computer

Description automatically generated

1. In **Keys and Endpoints** page, copy and paste **KEY1** and **Location/Region** on a notepadand save the notepad with the name **SpeechChatKey** on your desktop to use them in upcoming labs.





**Task 6: Set environment variables**

1. To set the environment variables, go back to your window command prompt. Run the following command to set the environment variable.
2. To set the **OPEN\_AI\_KEY** environment variable, replace **your-openai-key** with one of the keys for your resource. **(Key that you have saved in Lab #4)**

setx OPEN\_AI\_KEY your-openai-key

1. To set the **OPEN\_AI\_ENDPOINT** environment variable, replace **your-openai-endpoint** with one of the regions for your resource. **(Endpoint that you have saved in Lab #4)**

setx OPEN\_AI\_ENDPOINT your-openai-endpoint

1. To set the **SPEECH\_KEY** environment variable, replace **your-speech-key** with one of the keys for your resource.

setx SPEECH\_KEY your-speech-key

1. To set the **SPEECH\_REGION** environment variable, replace **your-speech-region** with one of the regions for your resource.

setx SPEECH\_REGION your-speech-region

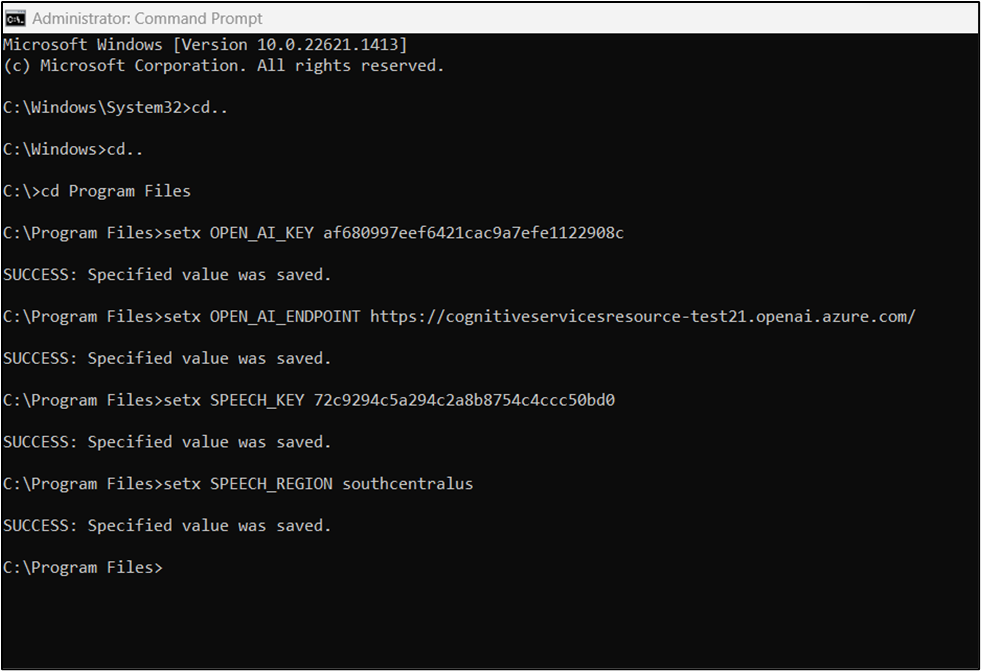
Example:

setx OPEN\_AI\_KEY af680997eef6421a7efe1122908c

setx OPEN\_AI\_ENDPOINT https://cognitiveservicesresource-test21.openai.azure.com/

setx SPEECH\_KEY 72c9294a8b8754c4ccc50bd0

setx SPEECH\_REGION eastus



**Note:** If you only need to access the environment variable in the current running console, you can set the environment variable with set instead of setx.

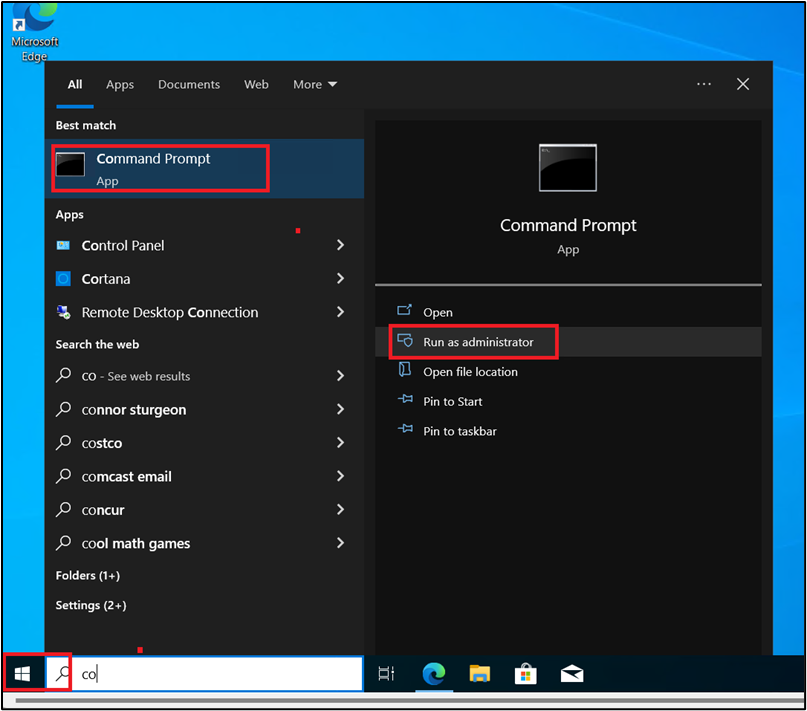
1. After you add the environment variables, you may need to restart any running programs that will need to read the environment variable, including the console window.
2. Close the Windows **command prompt**.

**Task 7: Recognize speech from a microphone**

Follow these steps to create a new console application.

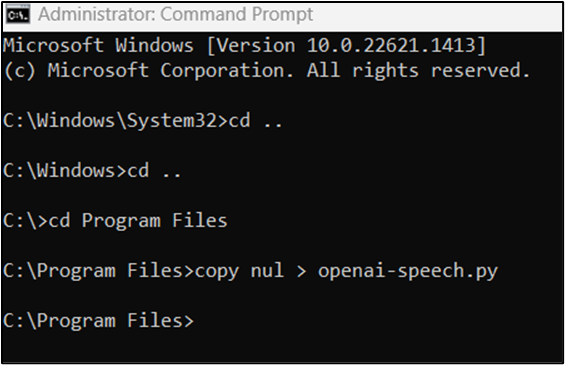
1. Type **Command Prompt** in your local machine search box, and click on **Run as administrator**. On **Do you allow this app to make changes on your device** dialog box, click on the **Yes** button.

**Important Note**: You need to change the current directory to the **Program Files** directory (The command used to move back to the previous directory is **cd .. [space after cd then two dots],** the command used to move to the next directory is **cd )**



1. Use the following command to create a new file named **openai-speech.py**.

copy nul > openai-speech.py

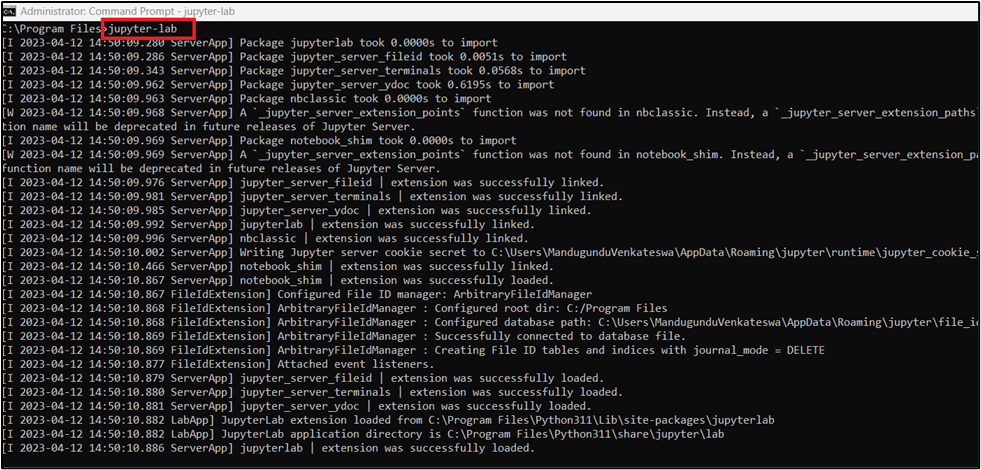


**Note:** This library is maintained by OpenAI (not Microsoft Azure). Refer to the [***release history***](https://github.com/openai/openai-python/releases)**or the**[***version.py commit history***](https://github.com/openai/openai-python/commits/main/openai/version.py)**to track the latest updates to the library.**

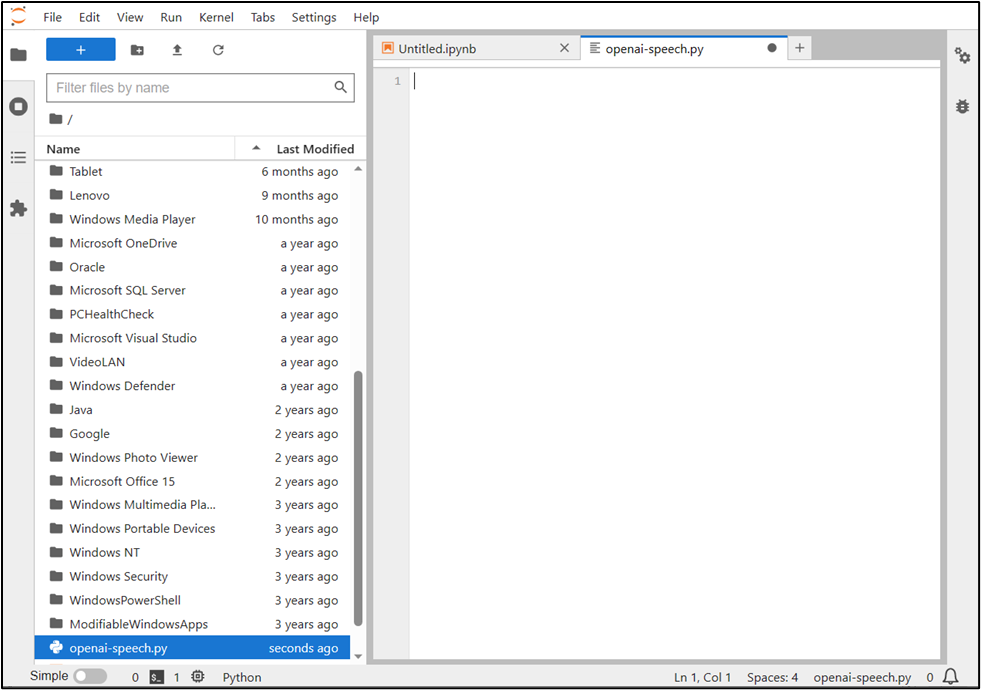
1. **Run the following command to open the Jupyter Notebook.**

**jupyter-lab**

**Note: In case, jupyter-lab command does not work, then run the following command in your command prompt: pip install jupyterlab, then run again jupyter-lab command.**

****

1. **In the Jupyter Notebook, on the left pane, navigate, select and click on openai-speech.py file. Inside openai-speech.py file, press Ctrl+A and then the Delete button.**

****

1. **Copy the following code into openai-speech.py file, then press Ctrl+S to Save the file.**

import os

import azure.cognitiveservices.speech as speechsdk

import openai

# This example requires environment variables named "OPEN\_AI\_KEY" and "OPEN\_AI\_ENDPOINT"

# Your endpoint should look like the following https://YOUR\_OPEN\_AI\_RESOURCE\_NAME.openai.azure.com/

openai.api\_key = os.environ.get('OPEN\_AI\_KEY')

openai.api\_base = os.environ.get('OPEN\_AI\_ENDPOINT')

openai.api\_type = 'azure'

openai.api\_version = ' 2022-12-01 '

# This will correspond to the custom name you chose for your deployment when you deployed a model.

deployment\_id= 'gpt-35-turbo'

# This example requires environment variables named "SPEECH\_KEY" and "SPEECH\_REGION"

speech\_config = speechsdk.SpeechConfig(subscription=os.environ.get('SPEECH\_KEY'), region=os.environ.get('SPEECH\_REGION'))

audio\_output\_config = speechsdk.audio.AudioOutputConfig(use\_default\_speaker=True)

audio\_config = speechsdk.audio.AudioConfig(use\_default\_microphone=True)

# Should be the locale for the speaker's language.

speech\_config.speech\_recognition\_language="en-US"

speech\_recognizer = speechsdk.SpeechRecognizer(speech\_config=speech\_config, audio\_config=audio\_output\_config)

# The language of the voice that responds on behalf of Azure OpenAI.

speech\_config.speech\_synthesis\_voice\_name='en-US-JennyMultilingualNeural'

speech\_synthesizer = speechsdk.SpeechSynthesizer(speech\_config=speech\_config, audio\_config=audio\_config)

# Prompts Azure OpenAI with a request and synthesizes the response.

def ask\_openai(prompt):

# Ask Azure OpenAI

response = openai.Completion.create(engine=deployment\_id, prompt=prompt, max\_tokens=100)

text = response['choices'][0]['text'].replace('\n', '').replace(' .', '.').strip()

print('Azure OpenAI response:' + text)

# Azure text-to-speech output

speech\_synthesis\_result = speech\_synthesizer.speak\_text\_async(text).get()

# Check result

if speech\_synthesis\_result.reason == speechsdk.ResultReason.SynthesizingAudioCompleted:

print("Speech synthesized to speaker for text [{}]".format(text))

elif speech\_synthesis\_result.reason == speechsdk.ResultReason.Canceled:

cancellation\_details = speech\_synthesis\_result.cancellation\_details

print("Speech synthesis canceled: {}".format(cancellation\_details.reason))

if cancellation\_details.reason == speechsdk.CancellationReason.Error:

print("Error details: {}".format(cancellation\_details.error\_details))

# Continuously listens for speech input to recognize and send as text to Azure OpenAI

def chat\_with\_open\_ai():

while True:

print("Azure OpenAI is listening. Say 'Stop' or press Ctrl-Z to end the conversation.")

try:

# Get audio from the microphone and then send it to the TTS service.

speech\_recognition\_result = speech\_recognizer.recognize\_once\_async().get()

# If speech is recognized, send it to Azure OpenAI and listen for the response.

if speech\_recognition\_result.reason == speechsdk.ResultReason.RecognizedSpeech:

if speech\_recognition\_result.text == "Stop.":

print("Conversation ended.")

break

print("Recognized speech: {}".format(speech\_recognition\_result.text))

ask\_openai(speech\_recognition\_result.text)

elif speech\_recognition\_result.reason == speechsdk.ResultReason.NoMatch:

print("No speech could be recognized: {}".format(speech\_recognition\_result.no\_match\_details))

break

elif speech\_recognition\_result.reason == speechsdk.ResultReason.Canceled:

cancellation\_details = speech\_recognition\_result.cancellation\_details

print("Speech Recognition canceled: {}".format(cancellation\_details.reason))

if cancellation\_details.reason == speechsdk.CancellationReason.Error:

print("Error details: {}".format(cancellation\_details.error\_details))

print("Did you set the speech resource key and region values?")

except EOFError:

break

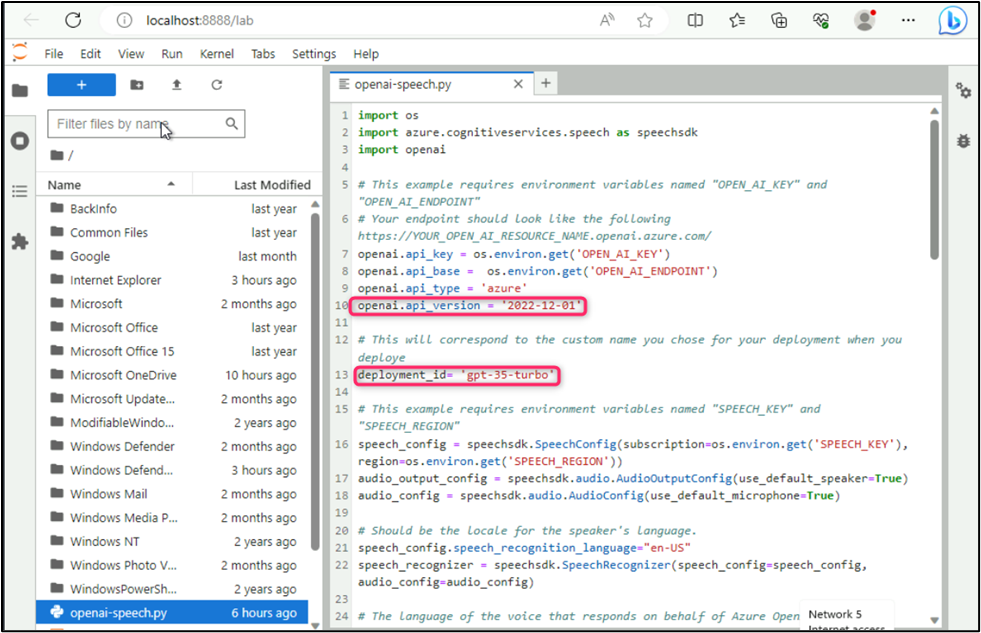
# Main

try:

chat\_with\_open\_ai()

except Exception as err:

print("Encountered exception. {}".format(err))

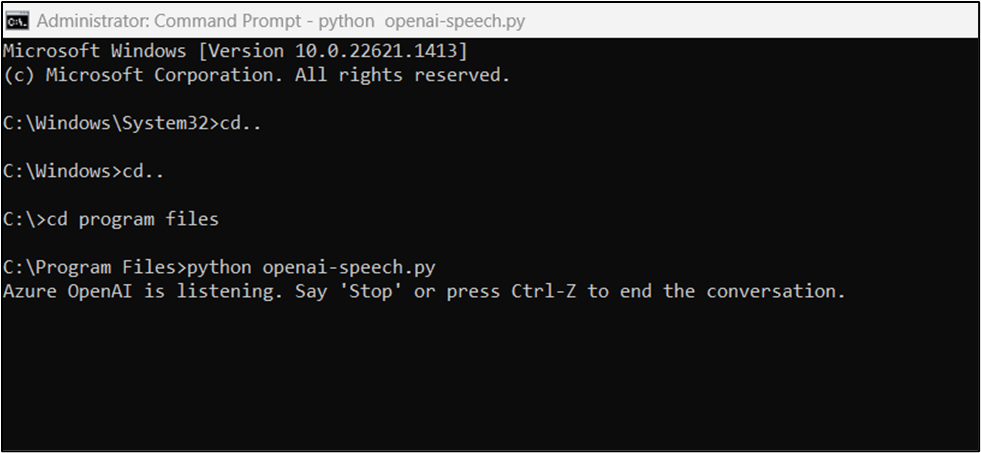


**Note**: Replace the deployment\_id with **gpt-35-turbo**

1. To increase or decrease the number of tokens returned by Azure OpenAI, change the max\_tokens parameter.
2. Run the new console application to start speech recognition from a microphone:
3. Run the following command

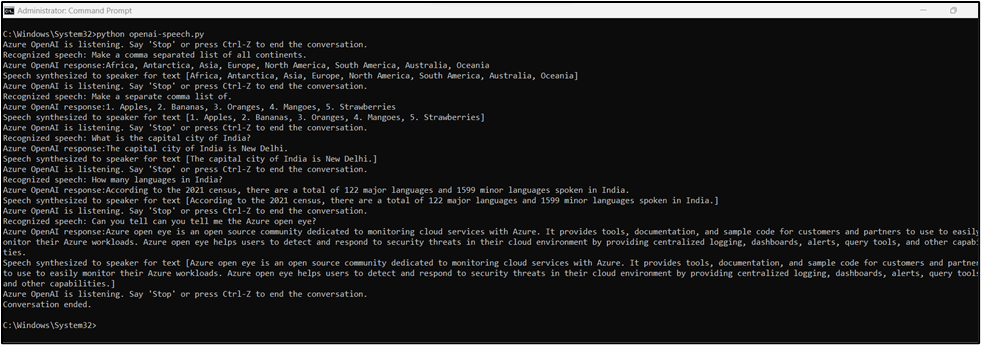
cli

python openai-speech.py



**Important:** Make sure that you set the OPEN\_AI\_KEY, OPEN\_AI\_ENDPOINT, SPEECH\_\_KEY and SPEECH\_\_REGION environment variables as described previously (i.e., **Task 7)**. If you don't set these variables, the sample will fail with an error message.

1. Speak into your microphone when prompted. The console output includes the prompt for you to begin speaking, then your request as text, and then the response from Azure OpenAI as text. The response from Azure OpenAI should be converted from text to speech and then output to the default speaker.



**Output ConsoleCopy**

PS C:\dev\openai\python> python.exe .\openai-speech.py

Azure OpenAI is listening. Say 'Stop' or press Ctrl-Z to end the conversation.

Recognized speech: Make a comma separated list of all continents.

Azure OpenAI response: Africa, Antarctica, Asia, Australia, Europe, North America, South America

Speech synthesized to speaker for text [Africa, Antarctica, Asia, Australia, Europe, North America, South America]

Azure OpenAI is listening. Say 'Stop' or press Ctrl-Z to end the conversation.

Recognized speech: Make a comma separated list of 1 Astronomical observatory for each continent. A list should include each continent name in parentheses.

Azure OpenAI response: Mauna Kea Observatories (North America), La Silla Observatory (South America), Tenerife Observatory (Europe), Siding Spring Observatory (Australia), Beijing Xinglong Observatory (Asia), Naukluft Plateau Observatory (Africa), Rutherford Appleton Laboratory (Antarctica)

Speech synthesized to speaker for text [Mauna Kea Observatories (North America), La Silla Observatory (South America), Tenerife Observatory (Europe), Siding Spring Observatory (Australia), Beijing Xinglong Observatory (Asia), Naukluft Plateau Observatory (Africa), Rutherford Appleton Laboratory (Antarctica)]

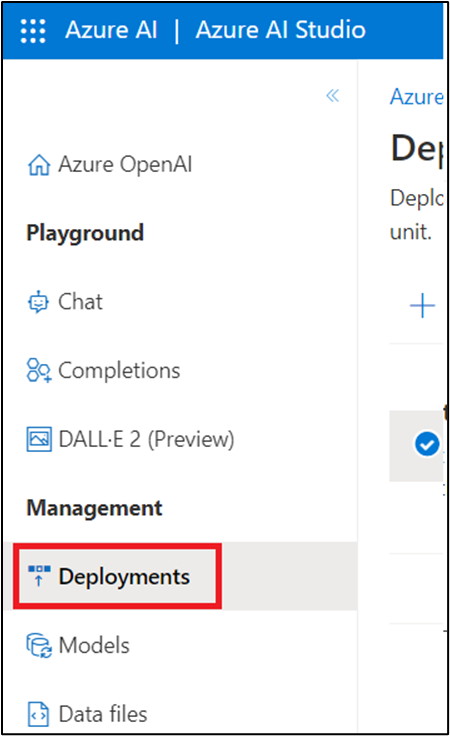
Azure OpenAI is listening. Say 'Stop' or press Ctrl-Z to end the conversation.

Conversation ended.

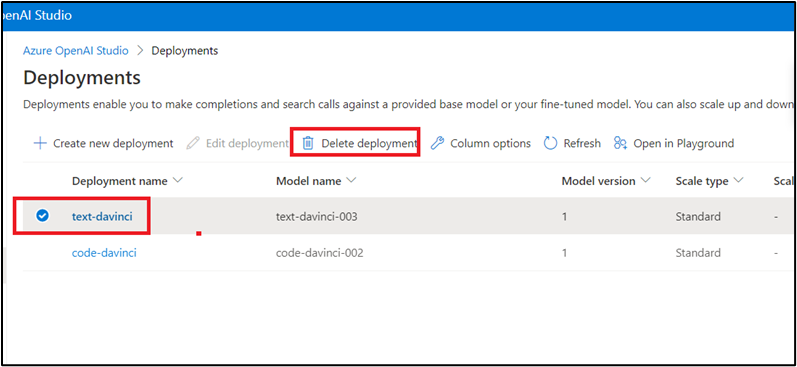
PS C:\dev\openai\python>

**Task 8: Delete the Deployed Model**

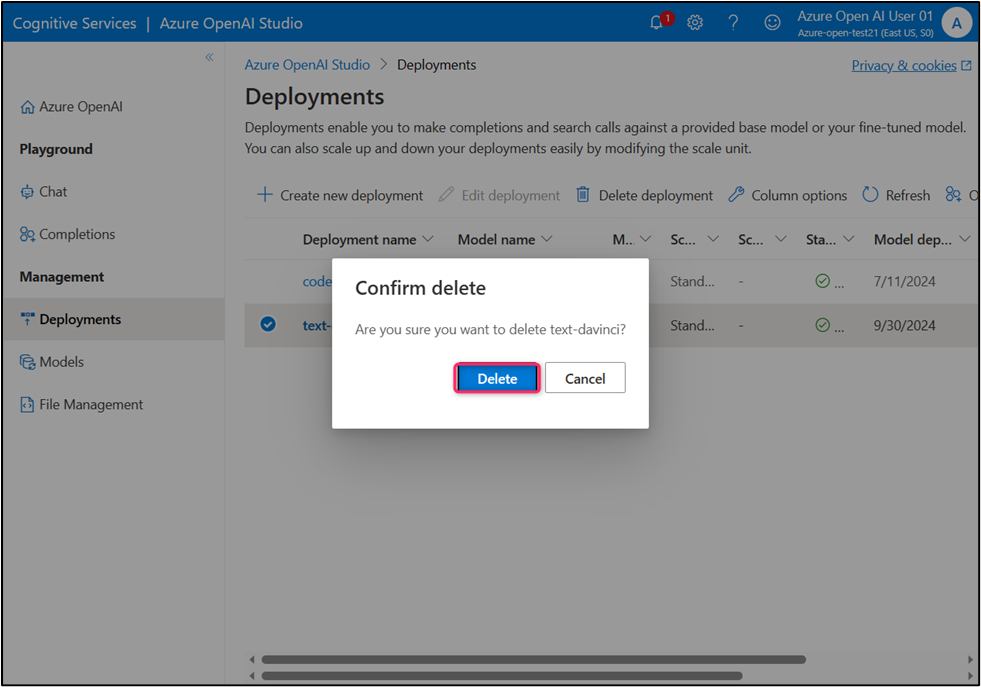
1. On the **Azure OpenAI Studio** window, under the **Management** section, click on **Deployments**.

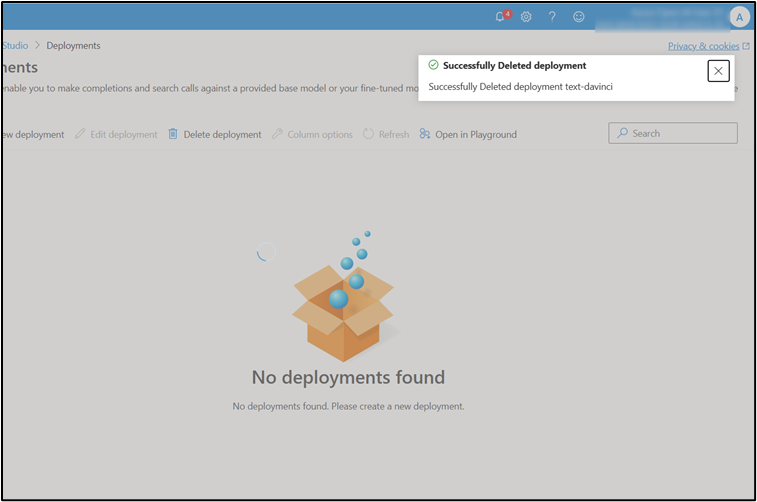


1. In the **Deployments** page, select **text-davinci** deployment name and then click on **Delete deployment**.



1. In the **Confirm delete** dialog box, click on the **Delete** button. You will see the message stating **Successfully Deleted deployment**.





**Important Note: Please do not delete the Resource group. The same Resource group will be used throughout all the labs.**

**Summary** In this lab, you’ve installed Speech SDK, python libraries, and Microsoft Visual C++ Redistributable for Visual Studio. Then, you’ve deployed text-davinci-003 model, created a Speech resource in the Azure portal, configured the environment variables, and executed the new console application to start speech recognition from a microphone. To effectively manage the OpenAI resource, you’ve deleted the deployed model.