If you have a working microphone

Alternatively, use audio input from a file

Add code to process the transcribed command

# Recognize and Synthesize Speech

The Speech service is an Azure cognitive service that provides speech-related functionality, including:

- A *speech-to-text* API that enables you to implement speech recognition (converting audible spoken words into text).
- A text-to-speech API that enables you to implement speech synthesis (converting text into audible speech).

In this exercise, you'll use both of these APIs to implement a speaking clock application.

**Note**: This exercise requires that you are using a computer with speakers/headphones. For the best experience, a microphone is also required. Some hosted virtual environments may be able to capture audio from your local microphone, but if this doesn't work (or you don't have a microphone at all), you can use a provided audio file for speech input. Follow the instructions carefully, as you'll need to choose different options depending on whether you are using a microphone or the audio file.

## Clone the repository for this course

If you have not already cloned **AI-102-AIEngineer** code repository to the environment where you're working on this lab, follow these steps to do so. Otherwise, open the cloned folder in Visual Studio Code.

- 1. Start Visual Studio Code.
- Open the palette (SHIFT+CTRL+P) and run a **Git: Clone** command to clone the
   https://github.com/MicrosoftLearning/AI-102-AIEngineer
   repository to a local folder (it doesn't matter which folder).
- 3. When the repository has been cloned, open the folder in Visual Studio Code.
- 4. Wait while additional files are installed to support the C# code projects in the repo.

Note: If you are prompted to add required assets to build and debug, select Not Now.

## Provision a Cognitive Services resource

If you don't already have one in your subscription, you'll need to provision a Cognitive Services resource.

- 1. Open the Azure portal at <a href="https://portal.azure.com">https://portal.azure.com</a>, and sign in using the Microsoft account associated with your Azure subscription.
- Select the + Create a resource button, search for cognitive services, and create a Cognitive Services resource with the following settings:
  - **Subscription**: Your Azure subscription
  - **Resource group**: Choose or create a resource group (if you are using a restricted subscription, you may not have permission to create a new resource group use the one provided)
  - **Region**: Choose any available region
  - o Name: Enter a unique name
  - Pricing tier: Standard S0
- 3. Select the required checkboxes and create the resource.
- 4. Wait for deployment to complete, and then view the deployment details.
- 5. When the resource has been deployed, go to it and view its **Keys and Endpoint** page. You will need one of the keys and the location in which the service is provisioned from this page in the next procedure.

## Prepare to use the Speech service

In this exercise, you'll complete a partially implemented client application that uses the Speech SDK to recognize and synthesize speech.

**Note**: You can choose to use the SDK for either **C#** or **Python**. In the steps below, perform the actions appropriate for your preferred language.

- 1. In Visual Studio Code, in the **Explorer** pane, browse to the **07-speech** folder and expand the **C-Sharp** or **Python** folder depending on your language preference.
- 2. Right-click the **speaking-clock** folder and open an integrated terminal. Then install the Speech SDK package by running the appropriate command for your language preference:

C#



- 3. View the contents of the speaking-clock folder, and note that it contains a file for configuration settings:
  - o **C#**: appsettings.json
  - **Python**: .env

Open the configuration file and update the configuration values it contains to include an authentication **key** for your cognitive services resource, and the **location** where it is deployed. Save your changes.

- 4. Note that the **speaking-clock** folder contains a code file for the client application:
  - o C#: Program.cs
  - o Python: speaking-clock.py

Open the code file and at the top, under the existing namespace references, find the comment **Import namespaces**. Then, under this comment, add the following language-specific code to import the namespaces you will need to use the Speech SDK:

C#

```
Code

// Import namespaces
using Microsoft.CognitiveServices.Speech;
using Microsoft.CognitiveServices.Speech.Audio;
```

## **Python**



5. In the **Main** function, note that code to load the cognitive services key and region from the configuration file has already been provided. You must use these variables to create a **SpeechConfig** for your cognitive services resource. Add the following code under the comment **Configure speech service**:

C#



```
// Configure speech service
speechConfig = SpeechConfig.FromSubscription(cogSvcKey, cogSvcRegion);
Console.WriteLine("Ready to use speech service in " + speechConfig.Region);
// Configure voice
speechConfig.SpeechSynthesisVoiceName = "en-US-AriaNeural";
```

## **Python**



6. Save your changes and return to the integrated terminal for the **speaking-clock** folder, and enter the following command to run the program:

#### C#



7. If you are using C#, you can ignore any warnings about using the **await** operator in asynchronous methods - we'll fix that later. The code should display the region of the speech service resource the application will use.

## Recognize speech

Now that you have a **SpeechConfig** for the speech service in your cognitive services resource, you can use the **Speech-to-text** API to recognize speech and transcribe it to text.

### If you have a working microphone

python speaking-clock.py

- 1. In the **Main** function for your program, note that the code uses the **TranscribeCommand** function to accept spoken input.
- 2. In the TranscribeCommand function, under the comment Configure speech recognition, add the appropriate code below to create a SpeechRecognizer client that can be used to recognize and transcribe speech using the default system microphone:

## C#

```
// Configure speech recognition

using AudioConfig audioConfig = AudioConfig.FromDefaultMicrophoneInput();

using SpeechRecognizer speechRecognizer = new SpeechRecognizer(speechConfig, audioConfig);

Console.WriteLine("Speak now...");
```

## **Python**

```
# Configure speech recognition
audio_config = speech_sdk.AudioConfig(use_default_microphone=True)
speech_recognizer = speech_sdk.SpeechRecognizer(speech_config, audio_config)
print('Speak now...')
```

3. Now skip ahead to the Add code to process the transcribed command section below.

## Alternatively, use audio input from a file

pip install playsound==1.2.2

1. In the terminal window, enter the following command to install a library that you can use to play the audio file:

#### C#



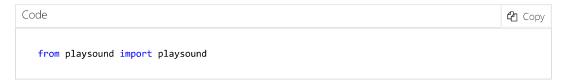
2. In the code file for your program, under the existing namespace imports, add the following code to import the library you just installed:

#### C#

```
Code

using System.Media;
```

## **Python**



3. In the Main function, note that the code uses the TranscribeCommand function to accept spoken input. Then in the TranscribeCommand function, under the comment Configure speech recognition, add the appropriate code below to create a SpeechRecognizer client that can be used to recognize and transcribe speech from an audio file:

### C#

```
// Configure speech recognition
string audioFile = "time.wav";
SoundPlayer wavPlayer = new SoundPlayer(audioFile);
wavPlayer.Play();
using AudioConfig audioConfig = AudioConfig.FromWavFileInput(audioFile);
using SpeechRecognizer speechRecognizer = new SpeechRecognizer(speechConfig, audioConfig);
```

```
# Configure speech recognition
audioFile = 'time.wav'
playsound(audioFile)
audio_config = speech_sdk.AudioConfig(filename=audioFile)
speech_recognizer = speech_sdk.SpeechRecognizer(speech_config, audio_config)
```

## Add code to process the transcribed command

1. In the TranscribeCommand function, under the comment Process speech input, add the following code to listen for spoken input, being careful not to replace the code at the end of the function that returns the command:

C#

```
Code
                                                                                          ℃ Copy
   // Process speech input
  SpeechRecognitionResult speech = await speechRecognizer.RecognizeOnceAsync();
   if (speech.Reason == ResultReason.RecognizedSpeech)
   {
       command = speech.Text;
       Console.WriteLine(command);
  }
  else
   {
       Console.WriteLine(speech.Reason);
       if (speech.Reason == ResultReason.Canceled)
           var cancellation = CancellationDetails.FromResult(speech);
           Console.WriteLine(cancellation.Reason);
           Console.WriteLine(cancellation.ErrorDetails);
   }
```

## Python

```
# Process speech input

speech = speech_recognizer.recognize_once_async().get()

if speech.reason == speech_sdk.ResultReason.RecognizedSpeech:
    command = speech.text
    print(command)

else:
    print(speech.reason)
    if speech.reason == speech_sdk.ResultReason.Canceled:
        cancellation = speech.cancellation_details
        print(cancellation.reason)
        print(cancellation.error_details)
```

2. Save your changes and return to the integrated terminal for the **speaking-clock** folder, and enter the following command to run the program:



3. If using a microphone, speak clearly and say "what time is it?". The program should transcribe your spoken input and display the time (based on the local time of the computer where the code is running, which may not be the correct time where you are).

The SpeechRecognizer gives you around 5 seconds to speak. If it detects no spoken input, it produces a "No match" result.

If the SpeechRecognizer encounters an error, it produces a result of "Cancelled". The code in the application will then display the error message. The most likely cause is an incorrect key or region in the configuration file.

## Synthesize speech

Your speaking clock application accepts spoken input, but it doesn't actually speak! Let's fix that by adding code to synthesize speech.

- 1. In the **Main** function for your program, note that the code uses the **TellTime** function to tell the user the current time.
- 2. In the **TellTime** function, under the comment **Configure speech synthesis**, add the following code to create a **SpeechSynthesizer** client that can be used to generate spoken output:

C#

```
Code

// Configure speech synthesis

speechConfig.SpeechSynthesisVoiceName = "en-GB-RyanNeural";

using SpeechSynthesizer speechSynthesizer = new SpeechSynthesizer(speechConfig);
```

#### **Python**

```
# Configure speech synthesis

speech_config.speech_synthesis_voice_name = "en-GB-RyanNeural"

speech_synthesizer = speech_sdk.SpeechSynthesizer(speech_config)
```

Note: The default audio configuration uses the default system audio device for output, so you don't need to explicitly provide an AudioConfig. If you need to redirect audio output to a file, you can use an AudioConfig with a filepath to do so.

3. In the **TellTime** function, under the comment **Synthesize spoken output**, add the following code to generate spoken output, being careful not to replace the code at the end of the function that prints the response:

```
Code

// Synthesize spoken output

SpeechSynthesisResult speak = await speechSynthesizer.SpeakTextAsync(responseText);

if (speak.Reason != ResultReason.SynthesizingAudioCompleted)

{
    Console.WriteLine(speak.Reason);
}
```

#### **Python**

```
# Synthesize spoken output

speak = speech_synthesizer.speak_text_async(response_text).get()

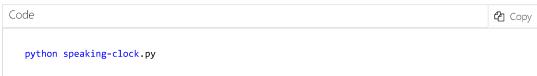
if speak.reason != speech_sdk.ResultReason.SynthesizingAudioCompleted:

print(speak.reason)
```

4. Save your changes and return to the integrated terminal for the **speaking-clock** folder, and enter the following command to run the program:

### C#





5. When prompted, speak clearly into the microphone and say "what time is it?". The program should speak, telling you the time.

## Use a different voice

Your speaking clock application uses a default voice, which you can change. The Speech service supports a range of *standard* voices as well as more human-like *neural* voices. You can also create *custom* voices.

**Note**: For a list of neural and standard voices, see <u>Language and voice support</u> in the Speech service documentation.

1. In the **TellTime** function, under the comment **Configure speech synthesis**, modify the code as follows to specify an alternative voice before creating the **SpeechSynthesizer** client:

## C#

```
// Configure speech synthesis

speechConfig.SpeechSynthesisVoiceName = "en-GB-LibbyNeural"; // change this

using SpeechSynthesizer speechSynthesizer = new SpeechSynthesizer(speechConfig);
```

## **Python**

Code Copy

```
# Configure speech synthesis
speech_config.speech_synthesis_voice_name = 'en-GB-LibbyNeural' # change this
speech_synthesizer = speech_sdk.SpeechSynthesizer(speech_config)
```

2. Save your changes and return to the integrated terminal for the **speaking-clock** folder, and enter the following command to run the program:

### C#



3. When prompted, speak clearly into the microphone and say "what time is it?". The program should speak in the specified voice, telling you the time.

## Use Speech Synthesis Markup Language

Speech Synthesis Markup Language (SSML) enables you to customize the way your speech is synthesized using an XML-based format.

1. In the **TellTime** function, replace all of the current code under the comment **Synthesize spoken output** with the following code (leave the code under the comment **Print the response**):

### C#

```
Code
                                                                                          ℃ Copy
  // Synthesize spoken output
   string responseSsml = $@"
       <speak version='1.0' xmlns='http://www.w3.org/2001/10/synthesis' xml:lang='en-US'>
           <voice name='en-GB-LibbyNeural'>
               {responseText}
               <break strength='weak'/>
               Time to end this lab!
           </voice>
       </speak>";
  SpeechSynthesisResult speak = await speechSynthesizer.SpeakSsmlAsync(responseSsml);
   if (speak.Reason != ResultReason.SynthesizingAudioCompleted)
   {
       Console.WriteLine(speak.Reason);
   }
```

### **Python**

Code Copy

2. Save your changes and return to the integrated terminal for the **speaking-clock** folder, and enter the following command to run the program:

### C#



3. When prompted, speak clearly into the microphone and say "what time is it?". The program should speak in the voice that is specified in the SSML (overriding the voice specified in the SpeechConfig), telling you the time, and then after a pause telling you it's time to end this lab - which it is!

## More information

For more information about using the **Speech-to-text** and **Text-to-speech** APIs, see the <u>Speech-to-text</u> <u>documentation</u> and <u>Text-to-speech documentation</u>.