from dotenv import load\_dotenv

from datetime import datetime

import os

# Import namespaces

# Import namespaces

import azure.cognitiveservices.speech as speech\_sdk

def main():

    try:

        global speech\_config

        # Get Configuration Settings

        load\_dotenv()

        ai\_key = os.getenv('SPEECH\_KEY')

        ai\_region = os.getenv('SPEECH\_REGION')

        # Configure speech service

                # Configure speech service

        speech\_config = speech\_sdk.SpeechConfig(ai\_key, ai\_region)

        print('Ready to use speech service in:', speech\_config.region)

        # Get spoken input

        command = TranscribeCommand()

        if command.lower() == 'what time is it?':

            TellTime()

    except Exception as ex:

        print(ex)

def TranscribeCommand():

    command = ''

    # Configure speech recognition

        # Configure speech recognition

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

    speech\_recognizer = speech\_sdk.SpeechRecognizer(speech\_config, audio\_config)

    print('Speak now...')

    # Process speech input

        # 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)

    # Return the command

    return command

def TellTime():

    now = datetime.now()

    response\_text = 'The time is {}:{:02d}'.format(now.hour,now.minute)

    # Configure speech synthesis

        # Configure speech synthesis

    speech\_config.speech\_synthesis\_voice\_name = 'en-GB-LibbyNeural' # change this

    speech\_synthesizer = speech\_sdk.SpeechSynthesizer(speech\_config)

    # Synthesize spoken output

    # Synthesize spoken output

responseSsml = " \

    <speak version='1.0' xmlns='http://www.w3.org/2001/10/synthesis' xml:lang='en-US'> \

        <voice name='en-GB-LibbyNeural'> \

            {} \

            <break strength='weak'/> \

            Time to end this lab! \

        </voice> \

    </speak>".format(response\_text)

speak = speech\_synthesizer.speak\_ssml\_async(responseSsml).get()

if speak.reason != speech\_sdk.ResultReason.SynthesizingAudioCompleted:

    print(speak.reason)

    # Print the response

    print(response\_text)

if \_\_name\_\_ == "\_\_main\_\_":

    main()