The method **getcwd**() returns current working directory of a process.

Google has a great **Speech Recognition API**. This API converts spoken text (microphone) into written text (Python strings), briefly Speech to Text. You can simply speak in a microphone and Google API will translate this into written text. The API has excellent results for English language.

In python the **with** keyword is used when working with unmanaged resources (like file streams).

**PyAudio** provides Python bindings for PortAudio, the cross-platform audio I/O library. With PyAudio, you can easily use Python to play and record audio on a variety of platforms, such as GNU/Linux, Microsoft Windows, and Apple Mac OS X / macOS.

The primary purpose of a Recognizer instance is, of course, to recognize speech. Each instance comes with a variety of settings and functionality for recognizing speech from an audio source.

r = sr.Recognizer()

Each recognize\_\*() method will throw a speech\_recognition.RequestError exception if the API is unreachable.

All seven recognize\_\*() methods of the Recognizer class require an audio\_data argument. In each case, audio\_data must be an instance of SpeechRecognition’s AudioData class.

Now, instead of using an audio file as the source, you will use the default system microphone. You can access this by creating an instance of the Microphone class.

mic = sr.Microphone()

**listen()** to Capture Microphone Input:

Now that you’ve got a Microphone instance ready to go, it’s time to capture some input.

Just like the AudioFile class, Microphone is a context manager. You can capture input from the microphone using the listen() method of the Recognizer class inside of the with block. This method takes an audio source as its first argument and records input from the source until silence is detected.

with mic as source:

... audio = r.listen(source)

Once you execute the with block, try speaking “hello” into your microphone. Wait a moment for the interpreter prompt to display again. Once the “>>>” prompt returns, you’re ready to recognize the speech.

r.recognize\_google(audio)

If the prompt never returns, your microphone is most likely picking up too much ambient noise.

To handle ambient noise, you’ll need to use the adjust\_for\_ambient\_noise() method of the Recognizer class, just like you did when trying to make sense of the noisy audio file.

with mic as source:

... r.adjust\_for\_ambient\_noise(source)

... audio = r.listen(source)