بسته‌های کاربردی برای بازشناسی گفتار در PyPI وجود دارد. برخی از این بسته‌ها در ادامه بیان شده‌اند.

* apiai ([**+**](https://pypi.org/project/apiai/))
* assemblyai ([**+**](https://pypi.org/project/assemblyai/))
* google-cloud-speech ([**+**](https://pypi.org/project/google-cloud-speech/))
* pocketsphinx ([**+**](https://pypi.org/project/pocketsphinx/))
* SpeechRecognition ([**+**](https://pypi.org/project/SpeechRecognition/))
* watson-developer-cloud ([**+**](https://pypi.org/project/watson-developer-cloud/))
* wit ([**+**](https://pypi.org/project/wit/))

برخی از این بسته‌ها – مانند wit و apiai – ویژگی‌های توکاری مانند **«پردازش زبان طبیعی» (Natural Language Processing)** برای تعیین نیت سخنران ارائه می‌کنند که از بازشناسی گفتار پایه‌ای کاری فراتر است. دیگر بسته‌ها، مانند google-cloud-speech، صرفا روی تبدیل گفتار به متن تمرکز می‌کنند. اما «SpeechRecognition» بسته‌ای محسوب می‌شود که سهولت کار با آن مثال زدنی و فوق‌العاده به شمار می‌آید. بازشناسی گفتار نیازمند ورودی صوتی است و بسته SpeechRecognition بازیابی این ورودی را بسیار آسان می‌کند. از سوی دیگر، به جای آنکه نیاز به ساخت اسکریپتی برای دسترسی به میکروفون و پردازش فایل صوتی از پایه باشد، SpeechRecognition این کارها را تنها در چند دقیقه انجام می‌دهد.

کتابخانه SpeechRecognition مانند پوششی برای چندین «رابط برنامه‌نویسی نرم‌افزارهای کاربردی» (API) محبوب عمل کرده و بنابراین بسیار انعطاف‌پذیر است. یکی از این موارد «Google Web Speech API» است که از کلید API پیش‌فرض که در کتابخانه SpeechRecognition هاردکد (hard-coded) شده، پشتیبانی می‌کند. این بدین معنا است که کاربر بدون نیاز به ثبت‌نام برای این سرویس می‌تواند از آن استفاده کند.

انعطاف‌پذیری و سهولت کاربری بسته SpeechRecognition آن را به یک انتخاب فوق‌العاده برای هر پروژه پایتونی مبدل ساخته است. اگرچه، پشتیبانی از کلیه ویژگی‌های API‌هایی که این بسته پوششی برای آن‌ها است، تضمین نمی‌شود. کاربر باید زمانی را اختصاص داده و با جست‌و‌جو پیرامون گزینه‌های موجود اطمینان حاصل کند که بسته SpeechRecognition در مساله مورد نظر کاربر کار می‌کند یا خیر.

**نصب SpeechRecognition**

SpeechRecognition با پایتون نسخه ۲.۶، ۲.۷ و ۳.۳ به بالا سازگار است اما نیاز به یک سری مراحل نصب افزوده برای پایتون ۲ دارد. در این راهنما، فرض می‌شود که کاربر از پایتون ۳.۳ به بالا استفاده می‌کند. بسته SpeechRecognition را می‌توان از ترمینال با pip نصب کرد.

$ pip install SpeechRecognition

پس از انجام نصب، باید فرآیند نصب را با باز کردن یک «سشن» (session) مفسر و تایپ دستورات زیر تایید کرد.

>>> import speech\_recognition as sr

>>> sr.\_\_version\_\_

'3.8.1'

سشن موجود باید باز نگه داشته شود زیرا به زودی کارهای زیادی باید در آن انجام شود.  SpeechRecognition با فایل‌های صوتی موجود کار می‌کند. شایان توجه است که به بسته PyAudio برای ثبت ورودی میکروفون نیاز وجود دارد. در ادامه، لیست «وابستگی‌هایی» (dependencies) که نیاز خواهند شد بیان می‌شوند.

**کلاس Recognizer**

همه کارهای جادویی بسته SpeechRecognition با کلاس Recognizer به وقوع می‌پیوندد. البته، هدف اولیه نمونه Recognizer بازشناسی گفتار است. هر «نمونه» (instance) با تنظیمات و کارکردهای متنوعی برای بازشناسی گفتار از یک منبع صوتی همراه است. ساخت یک نمونه Recognizer آسان محسوب می‌شود. در این راستا کافی است در سشن مفسر موجود کد زیر وارد شود.

>>> r = sr.Recognizer()

هر نمونه Recognizer دارای هفت متد برای بازشناسی گفتار از منبع صوتی با استفاده از API‌های متنوع است.

* ([**+**](https://azure.microsoft.com/en-us/services/cognitive-services/speech/)) recognize\_bing(): Microsoft Bing Speech
* ([**+**](https://w3c.github.io/speech-api/speechapi.html)) recognize\_google(): Google Web Speech API
* ([**+**](https://cloud.google.com/speech/)) recognize\_google\_cloud(): Google Cloud Speech – requires installation of the google-cloud-speech package
* ([**+**](https://www.houndify.com/)) recognize\_houndify(): Houndify by SoundHound
* ([**+**](https://www.ibm.com/watson/services/speech-to-text/)) recognize\_ibm(): IBM Speech to Text
* ([**+**](https://cmusphinx.github.io/)) recognize\_sphinx(): CMU Sphinx – requires installing PocketSphinx
* ([**+**](https://wit.ai/)) recognize\_wit(): Wit.ai

در میان هفت مورد بیان شده در بالا، تنها ()recognize\_sphinx به صورت آفلاین با موتور CMU Sphinx کار می‌کند و شش گزینه دیگر نیاز به اتصال اینترنت دارند. بحث کامل پیرامون ویژگی‌ها و مزایای هر API فراتر از موضوع این راهنما است. با توجه به آنکه SpeechRecognition با کلید API پیش‌فرض Google Web Speech API همراه است، می‌توان کار با آن را در لحظه آغاز کرد. به همین دلیل، از Web Speech API در این راهنما استفاده می‌شود. شش API دیگر نیاز به احراز هویت با یک کلید API یا نام‌کاربری/گذرواژه دارند. برای کسب جزئیات بیشتر، باید به مستندات SpeechRecognition رجوع کرد ([**+**](https://github.com/Uberi/speech_recognition/blob/master/reference/library-reference.rst)).

**هشدار:** کلید پیش‌فرض فراهم شده توسط SpeechRecognition به منظور تست است و امکان دارد گوگل هر زمانی آن را باطل کند. بنابراین استفاده از Google Web Speech API در تولید ایده خوبی نیست. حتی با یک کلید API معتبر نیز کاربر به تنها ۵۰ درخواست در روز محدود می‌شود و هیچ راهکاری برای افزایش این سهمیه وجود ندارد. خوشبختانه، رابط SpeechRecognition  تقریبا در کلیه API‌ها یکسان است، بنابراین استفاده از آنچه در این راهنما ارائه شده در پروژه‌های جهان واقعی آسان خواهد بود.

هر متد ()\*ـrecognize در صورتی که API در دسترس نباشد، یک استثنای speech\_recognition.RequestError راه‌اندازی می‌کند. برای ()recognize\_sphinx این مساله در نتیجه نصب ناموجود، خراب یا ناسازگار Sphinx به وقوع می‌پیوندد. برای شش متد دیگر، در صورتی که به محدودیت API رسیده شود، سرور در دسترس نباشد و یا اتصال اینترنت وجود نداشته باشد RequestError راه‌اندازی می‌شود. اکنون در سشن مفسر ()recognize\_google فراخونی می‌شود.

r.recognize\_google()

خروجی کد بالا چیزی شبیه زیر خواهد بود.

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

TypeError: recognize\_google() missing 1 required positional argument: 'audio\_data'

هفت متد کلاس ()\*ـrecognize نیازمند یک آرگومان audio\_data هستند. در هر مورد، audio\_data باید نمونه‌ای از کلاس AudioData بسته SpeechRecognition باشد. دو راه برای ساخت یک نمونه AudioData وجود دارد. راه اول، از یک فایل صوتی  و راه دوم صوت ضبط شده توسط یک میکروفون است. کار با فایل‌های صوتی اندکی آسان‌تر است، بنابراین ابتدا این مورد بررسی می‌شود.

**کار با فایل‌های صوتی**

پیش از ادامه دادن، نیاز به دانلود یک فایل صوتی وجود دارد. SpeechRecognition با استفاده از کلاس مفید AudioFile کار با فایل‌های صوتی را آسان می‌سازد. این کلاس قابل مقداردهی اولیه با مسیر یک فایل صوتی و فراهم کردن رابط مدیریت زمینه برای خواندن و کار با محتوای این فایل‌های صوتی است.

**نوع فایل‌های پشتیبانی شده**

در حال حاضر، بسته SpeechRecognition از فرمت‌های فایل زیر پشتیبانی می‌کند.

* WAV: باید در فرمت‌های PCM/LPCM باشد.
* AIFF
* AIFF-C
* FLAC: باید در فرمت FLAC محلی باشد. OGG-FLAC پشتیبانی نمی‌شود.
* .

**استفاده از ()record برای ثبت داده‌ها از یک فایل**

کد زیر به منظور پردازش محتوای فایل «harvard.wav» باید در سشن مفسر نوشته شود.

>>> harvard = sr.AudioFile('harvard.wav')

>>> with harvard as source:

... audio = r.record(source)

...

مدیر زمینه فایل را باز کرده و محتوای آن را می‌خواند، سپس داده‌ها را در یک نمونه AudioFile که source نامیده می‌شود ذخیره می‌کند. پس از آن، متد ()record داده‌ها را از کل یک فایل در یک نمونه AudioData ذخیره می‌کند. این مورد را می‌توان با بررسی نوع یک audio تایید کرد.

>>> type(audio)

<class 'speech\_recognition.AudioData'>

اکنون می‌توان ()recognize\_google را در تلاش برای شناسایی هر گفتاری در فایل صوتی فراخوانی (invoke) کرد.

>>> r.recognize\_google(audio)

'the stale smell of old beer lingers it takes heat

to bring out the odor a cold dip restores health and

zest a salt pickle taste fine with ham tacos al

Pastore are my favorite a zestful food is the hot

cross bun'

«Harvard Sentences» توسط IEEE در سال ۱۹۶۵ به منظور تست قابلیت بازشناسی گفتار در خطوط تلفن منتشر شدند. این داده‌ها همچنان در تست سلولی و VOIP مورد استفاده قرار می‌گیرند.

Harvard Sentences متشکل از ۷۲ لیست از ده عبارت است. می‌توان رکوردهای رایگان از این عبارات را در وب‌سایت «Open Speech Repository» ([**+**](http://www.voiptroubleshooter.com/open_speech/index.html)) پیدا کرد. موارد ضبط شده به زبان انگلیسی، چینی ماندارین، فرانسوی و هندی نیز موجود هستند و منبع فوق‌العاده‌ای از موارد رایگان برای ارزیابی کد را فراهم می‌کنند.

### ثبت قطعات با offset و duration

اگر نیاز به ثبت یک بخش از گفتار روی فایل باشد چه باید کرد. متد ()record آرگومان کلیدواژه duration را می‌پذیرد که ضبط را پس از تعداد ثانیه‌های مشخص شده‌ای متوقف می‌کند. برای مثال، کد زیر هر گفتاری را در چهار ثانیه اول فایل ثبت می‌کند.

>>> with harvard as source:

... audio = r.record(source, duration=4)

...

>>> r.recognize\_google(audio)

'the stale smell of old beer lingers'

متد ()record، هنگامی که درون یک بلوک with استفاده می‌شود، همیشه در جریان فایل به جلو حرکت می‌کند. این بدین معنا است که اگر یک بار برای ۴ ثانیه فایل ضبط شود و سپس دوباره برای ۴ ثانیه رکورد کردن آغاز شود، بار دوم چهار ثانیه از صوت پس از اولین چهارثانیه بازگردانده می‌شود.

>>> with harvard as source:

... audio1 = r.record(source, duration=4)

... audio2 = r.record(source, duration=4)

...

>>> r.recognize\_google(audio1)

'the stale smell of old beer lingers'

>>> r.recognize\_google(audio2)

'it takes heat to bring out the odor a cold dip'

شایان توجه است که audio2 شامل یک بخش از فاز سوم در فایل می‌شود. هنگام تعیین مدت زمان، رکورد کردن ممکن است mid-phrase یا حتی mid-word را متوقف کند که می‌تواند به صحت «رونویسی» (transcription) آسیب برساند. علاوه بر این، برای تعیین طول مدت ثبت، می‌توان به متد ()record با استفاده از آرگومان کلیدواژه offset یک نقطه شروع مشخص داد. این مقدار تعداد ثانیه‌ها از آغاز فایل که باید پیش از آغاز ضبط از آن‌ها چشم‌پوشی (ignore) شود را تعیین می‌کند. برای ثبت صرفا دومین عبارت در فایل، می‌توان با انحراف از ۴ ثانیه و ضبط برای مثلا ۳ ثانیه کار را آغاز کرد.

>>> with harvard as source:

... audio = r.record(source, offset=4, duration=3)

...

>>> recognizer.recognize\_google(audio)

'it takes heat to bring out the odor'

آرگومان‌های کلیدواژه offset و duration برای بخش‌بندی یک فایل صوتی، اگر کاربر دانش پیشینی از ساختار گفتار در فایل را داشته باشد مفید هستند. اگرچه، استفاده شتاب‌زده از آن‌ها می‌تواند منجر به رونویسی ضعیفی شود. برای مشاهده این اثر، از کد زیر در مفسر می‌توان استفاده کرد.

>>> with harvard as source:

... audio = r.record(source, offset=4.7, duration=2.8)

...

>>> recognizer.recognize\_google(audio)

'Mesquite to bring out the odor Aiko'

با آغاز ضبط از ثانیه ۴.۷، بخش «it t» از آغاز عبارت «it takes heat to bring out the odor» از دست می‌رود، بنابراین API فقط «akes heat» را دریافت می‌کند که با «Mesquite» مطابقت یافته است. به طور مشابه، در پایان ضبط «a co» ثبت می‌شود که آغاز سومین عبارت «a cold dip restores health and zest» است. این مورد با استفاده از API با «Aiko» تطبیق یافته.

دلیل دیگری نیز برای دریافت رونویسی اشتباه وجود دارد. «نویز» (Noise)! مثال بالا به این دلیل به خوبی عمل کرده که فایل صوتی به طرز منطقی تمییز است. در جهان واقعی، به جز حالتی که فرد فرصت پردازش فایل صوتی را پیش از داشته باشد، کاربر نمی‌تواند انتظار صوت بدون نویز را داشته باشد.

یکی از مواردی که می‌توان آن را مورد آزمون قرار داد استفاده از متد ()adjust\_for\_ambient\_noise برای کلاس Recognizer است.

>>> with jackhammer as source:

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

... audio = r.record(source)

...

>>> r.recognize\_google(audio)

'still smell of old beer vendors'

این کار کاربر را یک گام به عبارت واقعی نزدیک‌تر می‌کند، اما همچنان عالی نیست. همچنین، «the» از آغاز عبارت گم شده است. متد ()adjust\_for\_ambient\_noise اول دوم جریان فایل را می‌خواند و تشخیص‌دهنده را برای سطح نویز فایل صوتی کالیبره می‌کند. از این رو، بخشی از جریان پیش از آنکه ()record برای ثبت داده‌ها فراخوانی شود، صرف شده است.

می‌توان «فرم زمانی» (time-frame) را که ()adjust\_for\_ambient\_noise برای تحلیل استفاده می‌کند با آرگومان کلیدی duration تنظیم کرد. این آرگومان یک مقدار عددی را در ثانیه دریافت کرده و به طور پیش‌فرض روی ۱ قرار می‌دهد. در کد زیر تلاش شده این مقدار به ۰.۵ کاهش پیدا کند.

>>> with jackhammer as source:

... r.adjust\_for\_ambient\_noise(source, duration=0.5)

... audio = r.record(source)

...

>>> r.recognize\_google(audio)

'the snail smell like old Beer Mongers'

بسیار خب، این کار موجب می‌شود «the» در آغاز عبارت بیاید، اما اکنون باید به مشکلات جدیدی رسیدگی کرد. گاهی این امکان وجود ندارد که اثر نویز را حذف کرد، زیرا سیگنال بسیار نویزی‌تر از آن است که بتوان به شکل موفقیت‌آمیزی با آن س و کله زد. مساله این فایل نیز همین است.

به افرادی که زیاد با این مشکل مواجه می‌شوند توصیه می‌شود به انجام «پیش‌پردازش داده‌ها»  (pre-processing) اهتمام بورزند. این کار را می‌توان با نرم‌افزارهای ویرایش فایل صوتی ([**+**](https://www.audacityteam.org/)) و یا بسته‌های پایتون (مانند SciPy) که می‌توانند فیلترهایی را به فایل اضافه کنند انجام داد.

هنگام کار با فایل‌های دارای نویز، مشاهده پاسخ API واقعی مفید خواهد بود. اغلب API‌ها یک رشته JSON باز می‌گردانند که دربرگیرنده رونویس‌های بسیاری است. متد ()recognize\_google همیشه محتمل‌ترین رونویس را باز می‌گرداند مگر اینکه کاربر آن را مجبور به ارائه پاسخ کامل کند. می‌توان این کار را با تنظیم آرگومان کلیدواژه show\_all برابر با مقدار True در متد ()recognize\_google انجام داد.

>>> r.recognize\_google(audio, show\_all=True)

{'alternative': [

{'transcript': 'the snail smell like old Beer Mongers'},

{'transcript': 'the still smell of old beer vendors'},

{'transcript': 'the snail smell like old beer vendors'},

{'transcript': 'the stale smell of old beer vendors'},

{'transcript': 'the snail smell like old beermongers'},

{'transcript': 'destihl smell of old beer vendors'},

{'transcript': 'the still smell like old beer vendors'},

{'transcript': 'bastille smell of old beer vendors'},

{'transcript': 'the still smell like old beermongers'},

{'transcript': 'the still smell of old beer venders'},

{'transcript': 'the still smelling old beer vendors'},

{'transcript': 'musty smell of old beer vendors'},

{'transcript': 'the still smell of old beer vendor'}

], 'final': True}

همانطور که مشهود است، ()recognize\_google یک دیکشنری با کلید «alternative» باز می‌گرداند که به لیست رونویس‌های ممکن اشاره می‌کند. ساختار این پاسخ ممکن است از یک API به API دیگر متفاوت باشد و این موضوع برای «اشکال‌زدایی» (debugging) مفید است.

**کار با میکروفون**

برای دسترسی به میکروفون با SpeechRecognizer، باید بسته PyAudio ([**+**](https://people.csail.mit.edu/hubert/pyaudio/)) را نصب کرد.

**نصب PyAudio**

فرآیند نصب PyAudio بسته به سیستم‌عامل متفاوت است.

در ویندوز می‌توان PyAudio را با pip نصب کرد.

$ pip install pyaudio

**تست نصب**

پس از نصب PyAudio، می‌توان نصب را از کنسول تست کرد.

$ python -m speechcognition

اکنون کاربران باید اطمینان حاصل کنند که میکروفون روشن است و صدای آن بسته نیست. اگر نصب به طور صحیح انجام شده بود، کاربر باید چیزی شبیه آنچه در زیر آمده را مشاهده کند.

A moment of silence, please...

Set minimum energy threshold to 600.4452854381937

Say something!

اکنون می‌توان با حرف زدن در میکروفون مشاهده کرد که SpeechRecognition چقدر خوب گفتار را رونویس می‌کند.

**کلاس Microphone**

باید یک سشن مفسر دیگر را باز کرد و یک نمونه از کلاس recognizer ساخت.

>>> import speech\_recognition as sr

>>> r = sr.Recognizer()

اکنون، به جای استفاده از فایل صوتی به عنوان منبع، از میکروفون پیش‌فرض سیستم استفاده می‌شود. می‌توان به این میکروفون از طریق کلاس Microphone دسترسی داشت.

>>> mic = sr.Microphone()

اگر سیستم هیچ میکروفون پیش‌فرضی ندارد (مثلا در زربری‌پای)، یا کاربر قصد دارد که از میکروفونی غیر از پیش‌فرض سیستم استفاده کند، نیاز به تغیین میکروفون مورد استفاده با دادن اندیس دستگاه است. می‌توان لیست نام میکروفون‌ها را با فراخوانی متد استاتیک ()list\_microphone\_names از کلاس Microphone دریافت کرد.

>>> sr.Microphone.list\_microphone\_names()

['HDA Intel PCH: ALC272 Analog (hw:0,0)',

'HDA Intel PCH: HDMI 0 (hw:0,3)',

'sysdefault',

'front',

'surround40',

'surround51',

'surround71',

'hdmi',

'pulse',

'dmix',

'default']

توجه به این نکته لازم است که خروجی در سیستم‌های گوناگون امکان دارد با مثال بالا متفاوت باشد. اندیس دستگاه میکروفون، در واقع اندیس نام آن در لیست بازگردانده شده توسط ()list\_microphone\_names است. برای مثال، در خروجی بالا، برای استفاده از میکروفونی که «front» نامیده می‌شود و اندیس آن «3» است، باید یک نمونه از میکروفون به صورت زیر ساخته شود.

>>> # This is just an example; do not run

>>> mic = sr.Microphone(device\_index=3)

در اغلب پروژه‌ها، کاربر احتمالا از میکروفون پیش‌فرض سیستم استفاده می‌کند.

**استفاده از ()listen برای ثبت ورودی‌های میکروفون**

اکنون که نمونه Microphone آماده است، زمان ثبت ورودی‌ها فرارسیده. درست مانند کلاس AudioFile ، نمونه Microphone نیز مدیر زمینه محسوب می‌شود. می‌توان ورودی‌ها را با میکروفون با استفاده از متد ()listen از کلاس Recognizer درون بلوک with ثبت کرد. این روش یک منبع صوتی را به عنوان اولین آرگومان خود دریافت کرده و ورودی را از منبع تا زمانی که سکوت تشخیص داده شود دریافت می‌کند.

>>> with mic as source:

... audio = r.listen(source)

...

پس از آنکه بلوک with اجرا شد، کاربر می‌تواند با گفتن «hello» امتحان و یک دقیقه برای اینکه خط مفسر (interpreter prompt) نمایش داده شود صبر کند. هنگامی که «>>>» باز می‌گردد، کاربر آماده بازشناسی گفتار است.

>>> r.recognize\_google(audio)

'hello'

اگر خط مفسر باز نگردد، به احتمال زیاد میکروفون نویز زیادی از محیط دریافت کرده است. می‌توان با استفاده از کلیدهای ترکیبی «Ctrl + C» در فرآیند وقفه ایجاد کرد تا prompt بازگردد. برای مدیریت نویز محیط، نیاز به استفاده از متد ()adjust\_for\_ambient\_noise از کلاس Recognizer است، از آنجا که ورودی از میکروفون کمتر از ورودی از فایل صوتی قابل پیش‌بینی محسوب می‌شود، بهتر است در هر بار استفاده از ورودی میکروفون از روش گفته شده استفاده شود.

>>> with mic as source:

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

... audio = r.listen(source)

...

پس از اجرای کد بالا، باید یک ثانیه برای ()adjust\_for\_ambient\_noise صبر کرد تا کارش را انجام دهد، سپس مجددا می‌توان در میکروفون «hello» گفت. پس از آن باید صبر کرد تا خط مفسر پیش از آنکه سعی کند گفتار را تشخیص دهد بازگردد. با فراخوانی ()adjust\_for\_ambient\_noise منابع صوتی برای یک ثانیه آنالیز می‌شوند. اگر این زمان به نظر کاربر زیاد است، می‌تواند آن را با آرگومان کلیدواژه duration تنظیم کند.

در مستندات SpeechRecognition پیشنهاد شده که duration کمتر از ۰/۵ ثانیه نباشد. در برخی موارد، ممکن است کاربر تشخیص دهد که طول مدت بیشتر از یک ثانیه نتایج بهتری را تولید می‌کند. حداقل مقداری که کاربر نیاز دارد، بسته به محیط میکروفون است. متاسفانه، این اطلاعات اغلب در طول توسعه ناشناخته هستند. براساس تجربه نویسنده این مطلب، به نظر می‌رسد طول مدت پیش‌فرض برای اغلب کاربردها مناسب باشد.

**مدیریت گفتار غیرقابل تشخیص**

اکنون باید کد مثال پیشین را در مفسر  نوشته و نویزهای نامفهوم در میکروفون ایجاد شود. در پاسخ باید پیامی شبیه آنچه در زیر آمده داده شود.

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

File "/home/david/real\_python/speech\_recognition\_primer/venv/lib/python3.5/site-packages/speech\_recognition/\_\_init\_\_.py", line 858, in recognize\_google

if not isinstance(actual\_result, dict) or len(actual\_result.get("alternative", [])) == 0: raise UnknownValueError()

speech\_recognition.UnknownValueError

صوتی که نمی‌تواند توسط API تشخیص داده شود، منجر به نمایش پیام استثنای UnknownValueError می‌شود. کاربر باید همیشه فراخوانی‌های خود به API را با بلوک try and except پوشش دهد تا این استثناها را مدیریت کند.

**تذکر:**برای راه‌اندازی بلوک‌های «try and except» جهت مدیریت «استثناها» امکان دارد به تلاشی بیش از آنچه به نظر می‌رسد نیاز باشد. API برای رونویس هرگونه صوتی به شدت تلاش می‌کند. حتی خرخرهای اندک نیز به کلماتی مانند «how» رونویس می‌شوند. سرفه کردن، دست زدن و تپق‌های زبانی همواره استثنا را افزایش می‌دهند.