即時語音辨識

這段 Python 程式碼使用 Google Cloud 語音轉文字 API 和 PyAudio 函式庫實作即時語音辨識。它從麥克風擷取音訊，將其串流傳送至語音轉文字 API，並列印轉錄的文字。`MicrophoneStream` 類別處理音訊輸入，而 `main` 函式設定語音辨識用戶端並處理音訊串流。

```
import os
import argparse
import io
import sys
import time

from google.cloud import speech

import pyaudio
from six.moves import queue


# 音訊錄製參數
RATE = 16000
CHUNK = int(RATE / 10)  # 100ms


class MicrophoneStream(object):
    """ 開啟錄音串流作為產生器，產生音訊區塊。"""
    def __init__(self, rate, chunk):
        self._rate = rate
        self._chunk = chunk

        # 使用 PyAudio 建立音訊介面
        self._audio_interface = pyaudio.PyAudio()
        self._audio_stream = self._audio_interface.open(
            format=pyaudio.paInt16,
            # API 目前只支援單聲道（單聲道）音訊
            # https://goo.gl/z726ff
            channels=1, rate=self._rate,
            input=True, frames_per_buffer=self._chunk,
            # 非同步執行音訊串流以填滿緩衝區物件。
            # 這是必要的，這樣輸入裝置的緩衝區在呼叫線程發出網路請求等時不會溢位。
            stream_callback=self._fill_buffer,
```

```
        )
        self.closed = False
        self._buff = queue.Queue()

    def _fill_buffer(self, in_data, frame_count, time_info, status_flags):
        """ 持續從音訊串流收集資料到緩衝區。 """
        self._buff.put(in_data)
        return None, pyaudio.paContinue

    def generator(self, record_seconds):
        start_time = time.time()
        while not self.closed and time.time() - start_time < record_seconds:
            # 使用阻塞式 get() 以確保至少有一個資料區塊，如果區塊為 None 則停止迭代，表示音訊串流結束。
            chunk = self._buff.get()
            if chunk is None:
                return
            data = [chunk]

            # 現在消耗任何其他仍在緩衝的資料。
            while True:
                try:
                    chunk = self._buff.get(block=False)
                    if chunk is None:
                        return
                    data.append(chunk)
                except queue.Empty:
                    break

            yield b''.join(data)

    def close(self):
        self.closed = True
        # 向產生器發出信號以終止，以便用戶端的串流辨識方法不會阻塞程序終止。
        self._buff.put(None)
        self._audio_stream.close()
        self._audio_interface.terminate()

    def __enter__(self):
        return self
```

```
    def __exit__(self, type, value, traceback):
        self.close()


def main(record_seconds=10, language_code='en-US'):
    # 請參閱 http://g.co/cloud/speech/docs/languages
    # 以取得支援語言的清單。
    # language_code = 'en-US'  # BCP-47 語言標籤

    client = speech.SpeechClient()
    config = speech.RecognitionConfig(
        encoding=speech.RecognitionConfig.AudioEncoding.LINEAR16,
        sample_rate_hertz=RATE,
        language_code=language_code,
        model="latest_long",
    )

    streaming_config = speech.StreamingRecognitionConfig(
        config=config,
        interim_results=True)

    with MicrophoneStream(RATE, CHUNK) as stream:
        audio_generator = stream.generator(record_seconds)
        requests = (speech.StreamingRecognizeRequest(audio_content=content)
                    for content in audio_generator)

        responses = client.streaming_recognize(streaming_config, requests)


        # 現在，將轉錄回應應用於用途。
        transcript = ""
        for response in responses:
            print(response)
            # 轉錄完成後，列印結果。
            for result in response.results:
                if result.is_final:
                    alternative = result.alternatives[0]
                    transcript += alternative.transcript + " "
        print(u'Transcript: {}'.format(transcript))
```

```python
if __name__ == '__main__':
    parser = argparse.ArgumentParser(description=" 具備可調整持續時間的即時語音辨識。")
    parser.add_argument('--duration', type=int, default=10, help=" 錄音持續時間（秒）。")
    parser.add_argument('--language_code', type=str, default='en-US', help=" 轉錄的語言代碼。")
    args = parser.parse_args()
    print(" 請開始說話...")
    main(record_seconds=args.duration, language_code=args.language_code)
```