リアルタイム音声認識

この Python コードは、Google Cloud Speech-to-Text API と PyAudio ライブラリを使用してリアルタイムの音声認識を実装します。マイクからオーディオをキャプチャし、Speech-to-Text API にストリーミングして、転写されたテキストを出力します。`MicrophoneStream` クラスはオーディオ入力処理を行い、`main` 関数は音声認識クライアントを設定し、オーディオストリームを処理します。

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

from google.cloud import speech

import pyaudio
from six.moves import queue


# 音声録音パラメータ
RATE = 16000
CHUNK = int(RATE / 10)  # 100ms


class MicrophoneStream(object):
    """ オーディオチャンクを生成するジェネレータとして録音ストリームを開きます。 """
    def __init__(self, rate, chunk):
        self._rate = rate
        self._chunk = chunk

        # PyAudio を使用してオーディオインターフェースを作成
        self._audio_interface = pyaudio.PyAudio()
        self._audio_stream = self._audio_interface.open(
            format=pyaudio.paInt16,
            # API は現在、1 チャンネル（モノラル）オーディオのみをサポートしています
            # https://goo.gl/z726ff
            channels=1, rate=self._rate,
            input=True, frames_per_buffer=self._chunk,
            # 呼び出しスレッドがネットワークリクエストなどを実行している間、入力デバイスのバッファがオー
            stream_callback=self._fill_buffer,
```

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

    def _fill_buffer(self, in_data, frame_count, time_info, status_flags):
        """ オーディオストリームからバッファに継続的にデータを取得します。 """
        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:
            # データが少なくとも 1 チャンクあることを保証するためにブロッキング get () を使用し、チャンクが
            chunk = self._buff.get()
            if chunk is None:
                return
            data = [chunk]

            # ここで、バッファリングされている他のデータも消費します。
            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
        # クライアントのストリーミング認識メソッドがプロセスの終了をブロックしないように、ジェネレータの終
        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'):
    # サポートされている言語のリストについては、http://g.co/cloud/speech/docs/languages を参照してください。
    # language_code = 'en-US'  # BCP-47 言語タグ

    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)

        # ここで、転写応答を使用します。
        transcript = ""
        for response in responses:
            print(response)
            # 転写が完了したら、結果を出力します。
            for result in response.results:
                if result.is_final:
                    alternative = result.alternatives[0]
                    transcript += alternative.transcript + " "
        print(u'Transcript: {}'.format(transcript))


if __name__ == '__main__':
```

```
parser = argparse.ArgumentParser(description=" リアルタイム音声認識（録音時間は調整可能）")
parser.add_argument('--duration', type=int, default=10, help=" 録音時間（秒）")
parser.add_argument('--language_code', type=str, default='en-US', help=" 転写用の言語コード")
args = parser.parse_args()
print(" 話してください...")
main(record_seconds=args.duration, language_code=args.language_code)
```