

# تحويل الصوت إلى نص

دالة يلي في ما `transcribe_audio` من نص إلى الكلام بتحويل الخاصة `transcribe_audio` التطبيقات برمجة واجهة بتجربة مؤخرًا قدمت ل قد `transcribe_audio` النسخ. عملية لإجراء استخدمتها

```
import os
import json
import time
import argparse
from google.cloud import speech
from pydub import AudioSegment
import tempfile

#
OUTPUT_DIRECTORY = "assets/transcriptions"

def speech_to_text(audio_file, output_filename):
    print(f"          : {output_filename}")
    try:
        client = speech.SpeechClient()

        #
        # pydub
        audio_segment = AudioSegment.from_file(audio_file)
        sample_rate = audio_segment.frame_rate
        channels = audio_segment.channels

        #
        file_extension = os.path.splitext(audio_file)[1].lower()
        if file_extension == '.mp3':
            encoding = speech.RecognitionConfig.AudioEncoding.MP3
        elif file_extension in ['.wav', '.wave']:
            encoding = speech.RecognitionConfig.AudioEncoding.LINEAR16
        elif file_extension == '.flac':
            encoding = speech.RecognitionConfig.AudioEncoding.FLAC
        else:
            print(f"          : {file_extension}")
            return

        #
```

```

config = speech.RecognitionConfig(
    encoding=encoding,
    sample_rate_hertz=sample_rate,
    audio_channel_count=channels,
    language_code="en-US", #
)

with open(audio_file, "rb") as f:
    audio_content = f.read()

audio = speech.RecognitionAudio(content=audio_content)

#
try:
    operation = client.long_running_recognize(config=config, audio=audio)
    response = operation.result(timeout=300) #
except Exception as e:
    print(f"          :   {e}")
    return

print(response.results)

transcription = ""
for result in response.results:
    transcription += result.alternatives[0].transcript + "\n"

with open(output_filename, "w", encoding="utf-8") as f:
    f.write(transcription)
print(f"          }output_filename}")

except Exception as e:
    print(f"          }output_filename}: {e}")

def process_audio_files(input_dir, output_dir):
    os.makedirs(output_dir, exist_ok=True)

    all_audio_files = [f for f in os.listdir(input_dir) if f.endswith(('mp3', '.wav', '.m4a'))]
    total_files = len(all_audio_files)
    print(f"          :   }total_files}")

```

```

if total_files == 0:
    print(f"                                '{input_dir}' ".)
    return

files_processed = 0

for filename in all_audio_files:
    audio_file_path = os.path.join(input_dir, filename)
    output_filename = os.path.join(output_dir, f"{os.path.splitext(filename)[0]}.txt")
    if os.path.exists(output_filename):
        print(f"    {filename}: {output_filename} ". (
            continue
    print(f"\n    {files_processed + 1}/{total_files}: {filename}")
    try:
        #
        if filename.endswith('-zh.mp3') or filename.endswith('-zh.wav') or filename.endswith('-zh.m4a'):
            language_code = "cmn-CN"
        else:
            language_code = "en-US"

        #          speech_to_text
        #          language_code          speech_to_text

        speech_to_text(
            audio_file=audio_file_path,
            output_filename=output_filename,
        )
        files_processed += 1
        print(f"    {files_processed}/{total_files}.\n")
    except Exception as e:
        print(f"    {filename}: {e}")
        continue

    print(f"    !    {files_processed}/{total_files} ". (

if __name__ == "__main__":
    parser = argparse.ArgumentParser(description= " ". (
    parser.add_argument('--input_dir', type=str, default="assets/audios", help= " ". (

```

```
args = parser.parse_args()
```

```
process_audio_files(  
    input_dir=args.input_dir,  
    output_dir=OUTPUT_DIRECTORY,  
)
```