

Using Whisper API to transcribe text using your Device Microphone

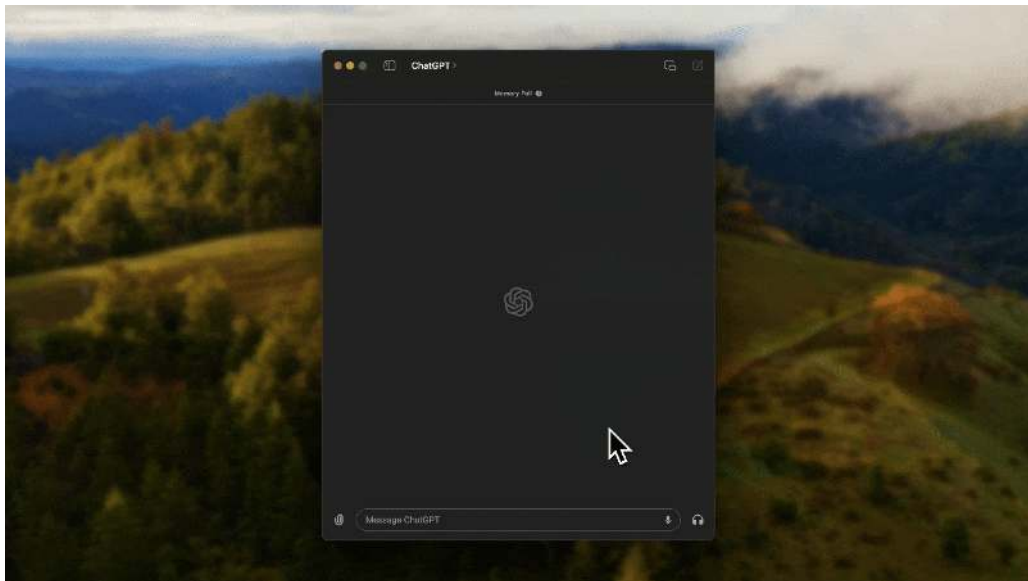
Accurate, punctuation-aware Speech-to-Text for everyone.

UPDATE: OpenAI will be including this tutorial in their [AI Cookbook website](#). I am so excited to contribute and be a part of their repository. [Pending merge](#).

Open AI's [Audio Whisper API](#) is capable of translating and transcribing —putting speech into written form. It is powered by OpenAI's [large-v2 Whisper model](#).

We will record audio from your device's microphone and transcribe the audio using OpenAI's Audio Whisper API. This functionality is similar to clicking on the microphone   icon in ChatGPT.

Note that speech-to-text is not supported in the [ChatGPT for Web](#). The code here *could* be used to enable this functionality on the web, though.



We'll be working with WAV files. Although larger than MP3, WAV files store audio in an uncompressed format, preserving audio quality. For transcription and translation purposes, high-quality audio can significantly improve accuracy.

We will go through three different code snippets making the main function:

1. **Recording** from your device microphone and storing it into a temporary file
2. **Transcribing or Translating** the audio to text using OpenAI's Whisper API
3. **Copying** the transcript to your clipboard for use (prompting, messaging, and etc.)

The entire code snippet will be available at [the end of this article](#).

Microphone Permissions

Before we start recording audio, ensure the necessary permissions to access the microphone.

For Windows

1. Open Settings.
2. Go to Privacy > Microphone.
3. Ensure that “Microphone access for this device” is turned on.
4. Ensure that the app you’re using (e.g., your Python IDE) is allowed to access the microphone.

For MacOS

1. Open System Preferences.
 2. Go to Security & Privacy > Privacy.
 3. Select Microphone from the left-hand menu.
 4. Ensure that the app you’re using (e.g., your Python IDE) is checked.
-

Setup

We need several libraries to record and process audio.

- pyaudio: To capture audio from the microphone.
- wave: To handle .wav files.
- tempfile: To create temporary files for storing recordings.
- simpleaudio: To play back audio (for debugging).

To install prerequisites, simply run the following code snippet.

```
# Prerequisites for the Python Modules
!brew install ffmpeg
!brew install portaudio

# Audio Processing
%pip install -q simpleaudio
%pip install -q pyaudio
%pip install -q wave

# Clipboard Management
%pip install -q pyperclip

# Speech Transcriber
%pip install -q openai
%pip install -q openai --upgrade # fix for Cannot import name 'OpenAI' from 'openai'

# Securing API keys
%pip install -q python-dotenv
```

Recording Audio from your Device Microphone

We’ll create a function that handles audio recording. This function will support both manual and timed recording:

1. **Set Up Temporary File.** We create a temporary file to store the recorded audio.

This file will be deleted after use.

```
temp_file = tempfile.NamedTemporaryFile(suffix=".wav", delete=False)
temp_file_name = temp_file.name
```

2. Callback Function. This function writes audio data to the temporary file while recording.

```
def callback(data_input, frame_count, time_info, status):
    wav_file.writeframes(data_input)
    return None, pyaudio.paContinue
```

3. Record Audio. We set up the microphone to capture audio and save it to the temporary file.

Open a .wav file for writing. Set the audio format: 1 channel, 16-bit samples, and 16000 Hz sample rate.

These values are standard for speech recognition tasks:

- **1 channel (mono):** Mono audio is sufficient for speech recognition and reduces the amount of data to process.
- **16-bit samples:** Provides a good balance between audio quality and file size.
- **16000 Hz sample rate:** Commonly used in speech recognition as it captures the necessary frequency range for human speech while keeping the file size manageable.

Initialize PyAudio and start recording.

```
import pyaudio
import wave
import tempfile
import time
```

```
def record_audio(timed_recording=False, record_seconds=5):
    temp_file = tempfile.NamedTemporaryFile(suffix=".wav", delete=False)
    temp_file_name = temp_file.name
```

```
    def callback(data_input, frame_count, time_info, status):
        wav_file.writeframes(data_input)
        return None, pyaudio.paContinue
```

```
    with wave.open(temp_file_name, "wb") as wav_file:
        wav_file.setnchannels(1) # Mono channel
        wav_file.setsampwidth(2) # 16-bit samples
        wav_file.setframerate(16000) # 16kHz sample rate
```

```
    audio = pyaudio.PyAudio()
    stream = audio.open(
        format=pyaudio.paInt16,
        channels=1,
        rate=16000,
        input=True,
        frames_per_buffer=1024,
        stream_callback=callback,
    )
```

```
    if timed_recording:
        print(f"Recording for {record_seconds} seconds...")
        time.sleep(record_seconds)
    else:
        input("Press Enter to stop recording...")
```

```
stream.stop_stream()
stream.close()
audio.terminate()

return temp_file_name
```

This function allows us to either record for a specific duration (timed_recording=True) or until the user presses Enter (timed_recording=False).

Transcribing or Translating Audio

Now, let's create a function to handle both transcription (for English) and translation (for non-English audio):

1. Import OpenAI Library

We use the OpenAI library to access the Audio Whisper API. To use the OpenAI API, you need to set up your API key. You can obtain your API key from the [OpenAI website](#).

Important

For security purposes, create a .env file in your project directory and store your OpenAI API key there. This way, you avoid hardcoding sensitive information directly in your code.

1. Create a .env file in the same directory as your notebook.
2. Add your OpenAI API key to the .env file in the following format:

```
OPEN_AI_API_KEY=your_actual_api_key_here
```

3. Use the dotenv library to load the environment variables in your notebook.

Once you have your key, you can set it up in your code as follows:

```
from openai import OpenAI
from dotenv import load_dotenv
import os

# Load the OpenAI API key from the .env file
load_dotenv()
openai_api_key = os.getenv("OPEN_AI_API_KEY")

# Set up your OpenAI API client
client = OpenAI(api_key=openai_api_key)
```

2. Transcribe Audio

Open the recorded audio file and send it to the OpenAI Audio Whisper API for transcription. The API returns the text.

```
def process_audio(file_name, is_english=True, prompt=""):
    with open(file_name, "rb") as audio_file:
        if is_english:
            response = client.audio.transcriptions.create(
                model="whisper-1", file=audio_file, prompt=prompt
            )
        else:
            response = client.audio.translations.create(
                model="whisper-1", file=audio_file
            )

    return response.text.strip()
```

Note: You can use prompt to *guide* the transcription as you record. This is useful for various reasons, such as spelling correction, language specification, acronym recognition, filler word removal or inclusion, punctuation, and more.

View [Audio Whisper API's reference](#) for more information. Alternatively, you can also look at prestontuggle's [AI Cookbook Recipe](#).

Copying to Clipboard

1. Import pyperclip

This library helps copy text to the clipboard.

```
import pyperclip
```

2. Copy Transcription

Copy the transcribed text to the clipboard and print a confirmation message.

```
def copy_to_clipboard(text):
    pyperclip.copy(text)
    print("Result copied to clipboard!")
```

Main Code Snippet

Here's the complete function that records audio, transcribes it, and copies the resulting text to the clipboard.

```
import simpleaudio as sa
import os

def transcribe_audio(
    debug: bool = False,
    prompt: str = "",
    timed_recording: bool = False,
```

```

record_seconds: int = 5,
is_english: bool = True,
) -> str:
    """
    Records audio from the microphone and transcribes or translates it using OpenAI's API.

    Args:
        debug (bool): If True, plays back the recorded audio for verification.
        prompt (str): A prompt to guide the transcription (only used for English).
        timed_recording (bool): If True, records for a set duration. If False, records until user input.
        record_seconds (int): The number of seconds to record if timed_recording is True.
        is_english (bool): If True, uses transcription. If False, uses translation to English.

    Returns:
        str: The transcription or translation of the recorded audio.
    """

    # Record audio
    temp_file_name = record_audio(timed_recording, record_seconds)

    # Debug playback
    if debug:
        print("Playing back recorded audio...")
        playback = sa.WaveObject.from_wave_file(temp_file_name)
        play_obj = playback.play()
        play_obj.wait_done()

    # Process audio (transcribe or translate)
    result = process_audio(temp_file_name, is_english, prompt)

    # Clean up temporary file
    os.remove(temp_file_name)

    # Copy result to clipboard
    copy_to_clipboard(result)

    return result

```

Demo

```

# Demo: Transcribe 5 seconds of spoken English with proper grammar and punctuation
result = transcribe_audio(
    debug=True,
    prompt="English spoken. Proper grammar and punctuation. Skip fillers.",
    timed_recording=True,
    record_seconds=5,
    is_english=True,
)
print("\nTranscription/Translation:", result)

```

Troubleshooting

If you encounter issues, ensure that:

- Your microphone is correctly set up and permissions are granted.
- The OpenAI API key is valid and correctly set in your .env file.
- You are using the correct versions of the libraries.

FAQ

1. Can I record for longer than 5 minutes?

Yes, adjust the `record_seconds` parameter to a longer duration.

2. What audio formats does Whisper support?

Whisper supports various audio formats; however, using `.wav` is recommended for optimal performance.

3. Is there a way to transcribe languages other than English?

Yes, simply set the `is_english` parameter to `False` to use Whisper's translation features.

Conclusion

With these code snippets, you can easily record audio from your device microphone, transcribe or translate it using OpenAI's Whisper API, and copy the result to your clipboard. You can expand or modify these functions for additional features, such as user interfaces or integration with other applications.

Feel free to use this code as a starting point for your own projects!
Happy coding!

By [Carl Kho](#) on [September 21, 2024](#).

[Canonical link](#)

Exported from [Medium](#) on October 31, 2025.