```
!pip install SpeechRecognition
```

```
    cting SpeechRecognition
    nloading SpeechRecognition-3.10.1-py2.py3-none-any.whl (32.8 MB)
    ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 32.8/32.8 MB 32.1 MB/s eta 0:00:00
    rement already satisfied: requests>=2.26.0 in /usr/local/lib/python3.10/dist-packages (from SpeechRecognition) (2.31.0)
    rement already satisfied: typing-extensions in /usr/local/lib/python3.10/dist-packages (from SpeechRecognition) (4.5.0)
    rement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.10/dist-packages (from requests>=2.26.0->SpeechRecognition)
    rement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-packages (from requests>=2.26.0->SpeechRecognition) (3.6)
    rement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.10/dist-packages (from requests>=2.26.0->SpeechRecognition) (2.0.
    rement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.10/dist-packages (from requests>=2.26.0->SpeechRecognition) (2023
    lling collected packages: SpeechRecognition
    ssfully installed SpeechRecognition-3.10.1
```

```
import nltk
from nltk.sentiment import SentimentIntensityAnalyzer
import speech_recognition as sr


nltk.download('vader_lexicon')
```

```
     [nltk_data] Downloading package vader_lexicon to /root/nltk_data...
     True
```

```
def speech_to_text(audio_file_path):
    recognizer = sr.Recognizer()

    with sr.AudioFile(audio_file_path) as source:
        audio_data = recognizer.record(source)  # Record the entire audio file
        text = recognizer.recognize_google(audio_data)  # Use Google Web Speech API for recognition

    return text
audio_file_path = "/content/getlost-angry.wav"
transcribed_text = speech_to_text(audio_file_path)
print("Transcribed Text:")
print(transcribed_text)
```

```
     Transcribed Text:
     just get lost from here
```

```
def map_to_emotion(compound_score):
    if compound_score >= 0.2:
        return "happy"
    elif compound_score >= 0.1:
        return "joyful"
    elif compound_score >= 0.05:
        return "content"
    elif compound_score <= -0.2:
        return "angry"
    elif compound_score <= -0.1:
        return "sad"
    elif compound_score <= -0.05:
        return "disgusted"
    elif compound_score >= -0.05 and compound_score <= 0.05:
        return "neutral"
    else:
        return "surprised"


def analyze_emotion(text):
    sia = SentimentIntensityAnalyzer()
    sentiment_scores = sia.polarity_scores(text)

    # Determine emotion category based on compound score
    compound_score = sentiment_scores['compound']
    emotion_result = map_to_emotion(compound_score)

    return emotion_result
emotion_result = analyze_emotion(transcribed_text)

print(f"Predicted Emotion: {emotion_result}")
```

```
 Predicted Emotion: angry
```

```
def speech_to_text(audio_file_path):
    recognizer = sr.Recognizer()

    with sr.AudioFile(audio_file_path) as source:
        audio_data = recognizer.record(source)  # Record the entire audio file
        text = recognizer.recognize_google(audio_data)  # Use Google Web Speech API for recognition

    return text
audio_file_path = "/content/wow-happy.wav"
transcribed_text = speech_to_text(audio_file_path)
print("Transcribed Text:")
print(transcribed_text)
def map_to_emotion(compound_score):
    if compound_score >= 0.2:
        return "happy"
    elif compound_score >= 0.1:
        return "joyful"
    elif compound_score >= 0.05:
        return "content"
    elif compound_score <= -0.2:
        return "angry"
    elif compound_score <= -0.1:
        return "sad"
    elif compound_score <= -0.05:
        return "disgusted"
    elif compound_score >= -0.05 and compound_score <= 0.05:
        return "neutral"
    else:
        return "surprised"
def analyze_emotion(text):
    sia = SentimentIntensityAnalyzer()
    sentiment_scores = sia.polarity_scores(text)

    # Determine emotion category based on compound score
    compound_score = sentiment_scores['compound']
    emotion_result = map_to_emotion(compound_score)

    return emotion_result
emotion_result = analyze_emotion(transcribed_text)

print(f"Predicted Emotion: {emotion_result}")
```

```
 Transcribed Text:
 wow that's so beautiful
 Predicted Emotion: happy
```

```
    def speech_to_text(audio_file_path):
        recognizer = sr.Recognizer()

        with sr.AudioFile(audio_file_path) as source:
            audio_data = recognizer.record(source)  # Record the entire audio file
            text = recognizer.recognize_google(audio_data)  # Use Google Web Speech API for recognition

        return text
    audio_file_path = "/content/miss-sad.wav"
    transcribed_text = speech_to_text(audio_file_path)
    print("Transcribed Text:")
    print(transcribed_text)
    def map_to_emotion(compound_score):
        if compound_score >= 0.2:
            return "happy"
        elif compound_score >= 0.1:
            return "joyful"
        elif compound_score >= 0.05:
            return "content"
        elif compound_score <= -0.2:
            return "angry"
        elif compound_score <= -0.1:
            return "sad"
        elif compound_score <= -0.05:
            return "disgusted"
        elif compound_score >= -0.05 and compound_score <= 0.05:
            return "neutral"
        else:
            return "surprised"
    def analyze_emotion(text):
        sia = SentimentIntensityAnalyzer()
        sentiment_scores = sia.polarity_scores(text)

        # Determine emotion category based on compound score
        compound_score = sentiment_scores['compound']
        emotion_result = map_to_emotion(compound_score)

        return emotion_result
```