LAB:06

OBJECTIVE: SPEECH RECOGNITION IN PYTHON USING GOOGLE API

TOPIC:VISUALIZE AN AUDIO SIGNAL

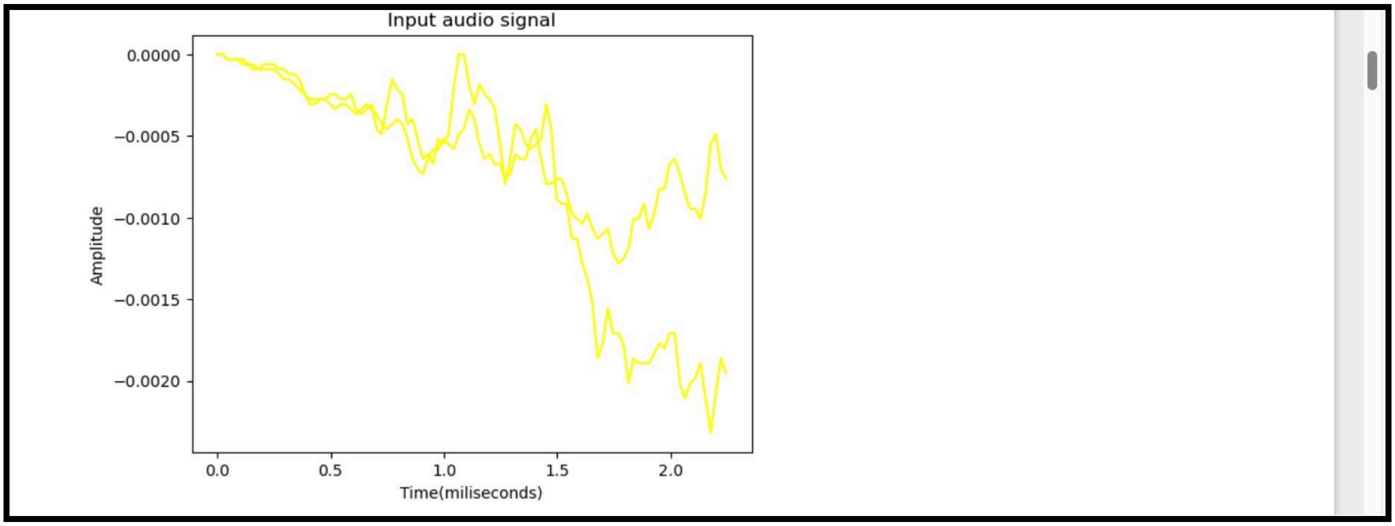
TASK1:GENERATE THE OUTPUT FOR THE ABOVE CODE(FOR WAVSOUND):

CODE:

```
!pip install scipy
import numpy as np
import matplotlib.pyplot as plt
from scipy.io import wavfile
frequency_sampling, audio_signal =
wavfile.read("C:/Users/PMLS/Downloads/Audio_waterfall.wav")
print('signal shape:', audio_signal.shape)
print('signal Datatype:', audio_signal.dtype)
print('signal Duration:', round(audio_signal.shape[0]/float (frequency_sampling),2),'seconds')
audio_signal=audio_signal /np.power(2,15)
audio_signal= audio_signal[:100]
time_axis=1000*np.arange(0,len(audio_signal),1) / float(frequency_sampling)
plt.plot(time_axis,audio_signal,color='yellow')
plt.xlabel('Time(miliseconds)')
plt.ylabel('Amplitude')
plt.title('Input audio signal')
plt.show()
```

OUTPUT:

```
signal shape: (1947349, 2)
signal Datatype: int16
signal Duration: 44.16 seconds
C:\Users\PMLS\AppData\Local\Temp\ipykernel_13776\521204116.py:4: WavFileWarning: Chunk (non-data) not understood, skipping it.
  frequency_sampling, audio_signal = wavfile.read("C:/Users/PMLS/Downloads/Audio_waterfall.wav")
```

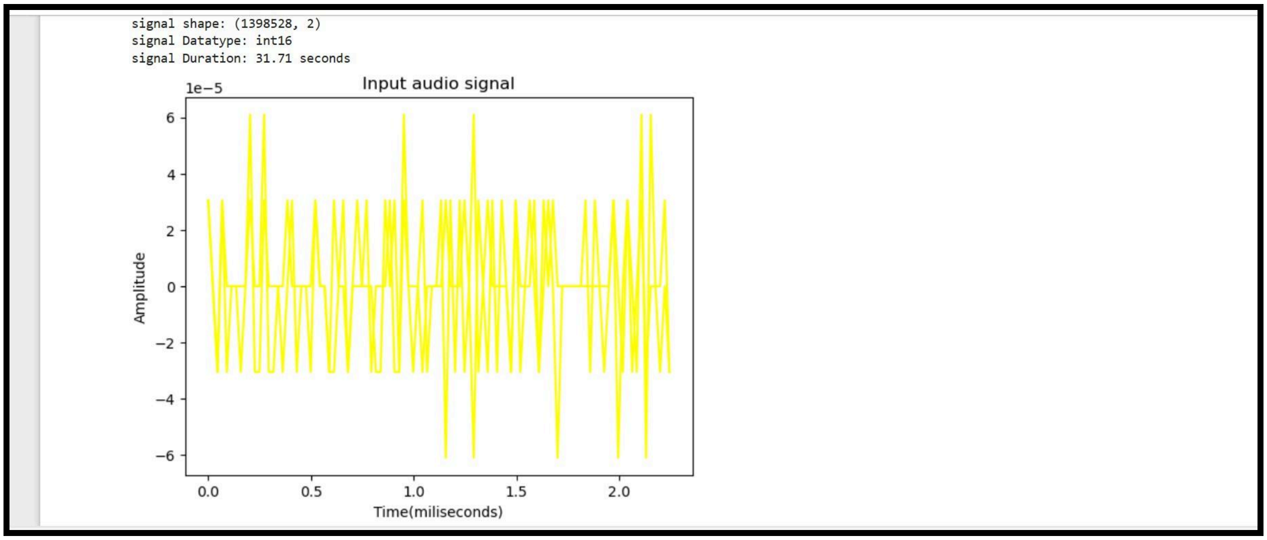


TASK2:GENERATE THE OUTPUT FOR OTHER AUDIO FILES AS WELL AND SEE THE DIFFERENCE(FOR AUDIO CAR SOUND)

CODE:

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.io import wavfile
frequency_sampling, audio_signal = wavfile.read("C:/Users/PMLS/Downloads/Audio_car.wav")
print('signal shape:', audio_signal.shape)
print('signal Datatype:', audio_signal.dtype)
print('signal Duration:', round(audio_signal.shape[0]/float (frequency_sampling),2),'seconds')
audio_signal=audio_signal /np.power(2,15)
audio_signal= audio_signal[:100]
time_axis=1000*np.arange(0,len(audio_signal),1) / float(frequency_sampling)
plt.plot(time_axis,audio_signal,color='yellow')
plt.xlabel('Time(miliseconds)')
plt.ylabel('Amplitude')
plt.title('Input audio signal')
plt.show()
```

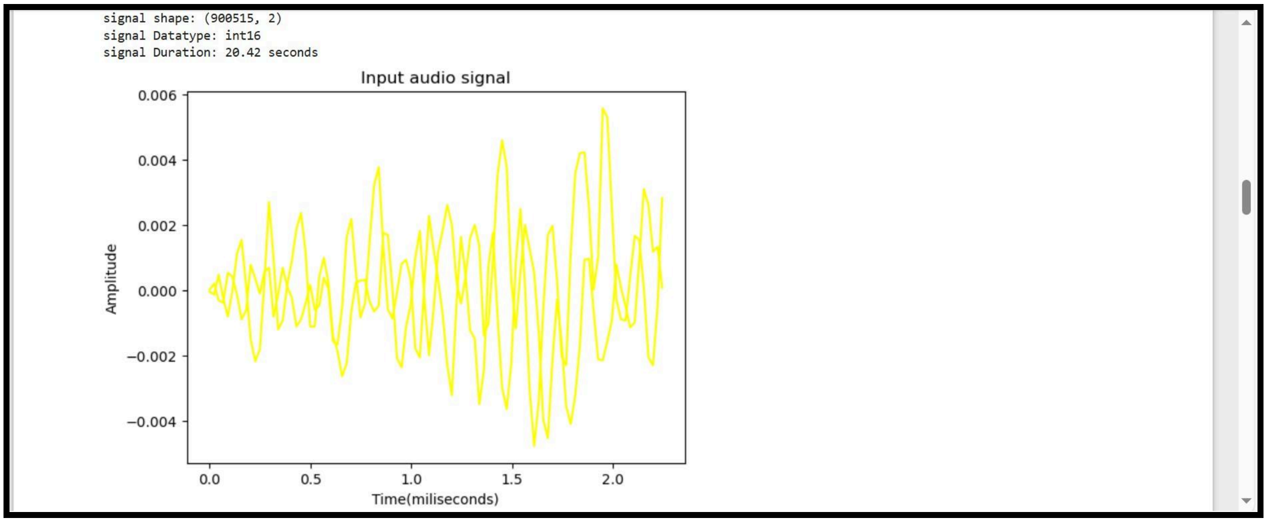
OUTPUT:



CODE:(FOR AUDIO CAR ENGINE)

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.io import wavfile
frequency_sampling, audio_signal =
wavfile.read("C:/Users/PMLS/Downloads/Audio_carengine.wav")
print('signal shape:', audio_signal.shape)
print('signal Datatype:', audio_signal.dtype)
print('signal Duration:', round(audio_signal.shape[0]/float (frequency_sampling),2),'seconds')
audio_signal=audio_signal /np.power(2,15)
audio_signal= audio_signal[:100]
time_axis=1000*np.arange(0,len(audio_signal),1) / float(frequency_sampling)
plt.plot(time_axis,audio_signal,color='yellow')
plt.xlabel('Time(miliseconds)')
plt.ylabel('Amplitude')
plt.title('Input audio signal')
plt.show()
```

OUTPUT:

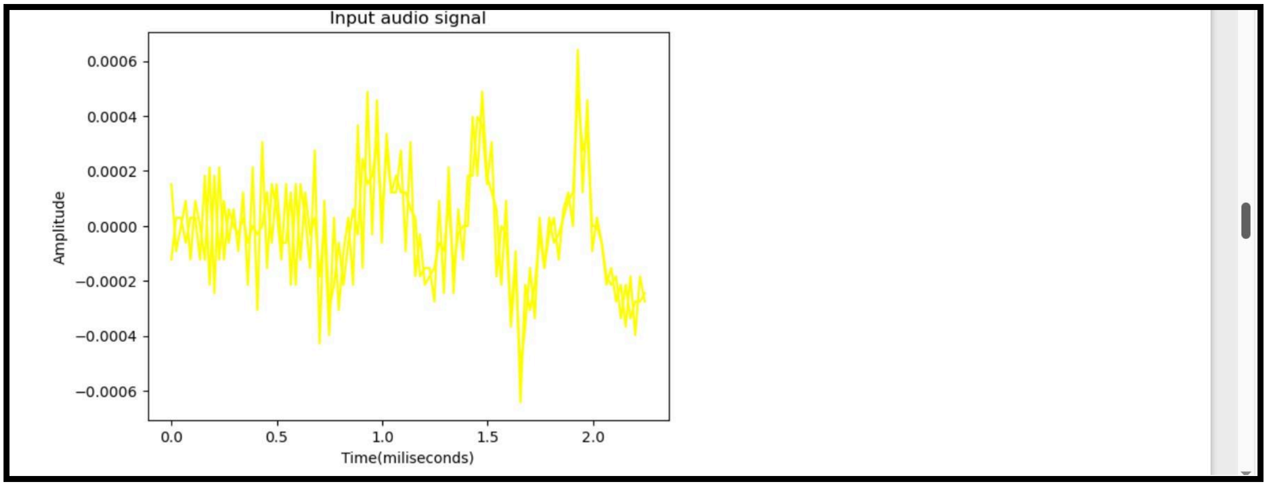


CODE:(FOR DOG AUDIO)

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.io import wavfile
frequency_sampling, audio_signal =
wavfile.read("C:/Users/PMLS/Downloads/Audio_dog.wav")
print('signal shape:', audio_signal.shape)
print('signal Datatype:', audio_signal.dtype)
print('signal Duration:', round(audio_signal.shape[0]/float (frequency_sampling),2),'seconds')
audio_signal=audio_signal /np.power(2,15)
audio_signal= audio_signal[:100]
time_axis=1000*np.arange(0,len(audio_signal),1) / float(frequency_sampling)
plt.plot(time_axis,audio_signal,color='yellow')
plt.xlabel('Time(miliseconds)')
plt.ylabel('Amplitude')
plt.title('Input audio signal')
plt.show()
```

OUTPUT:

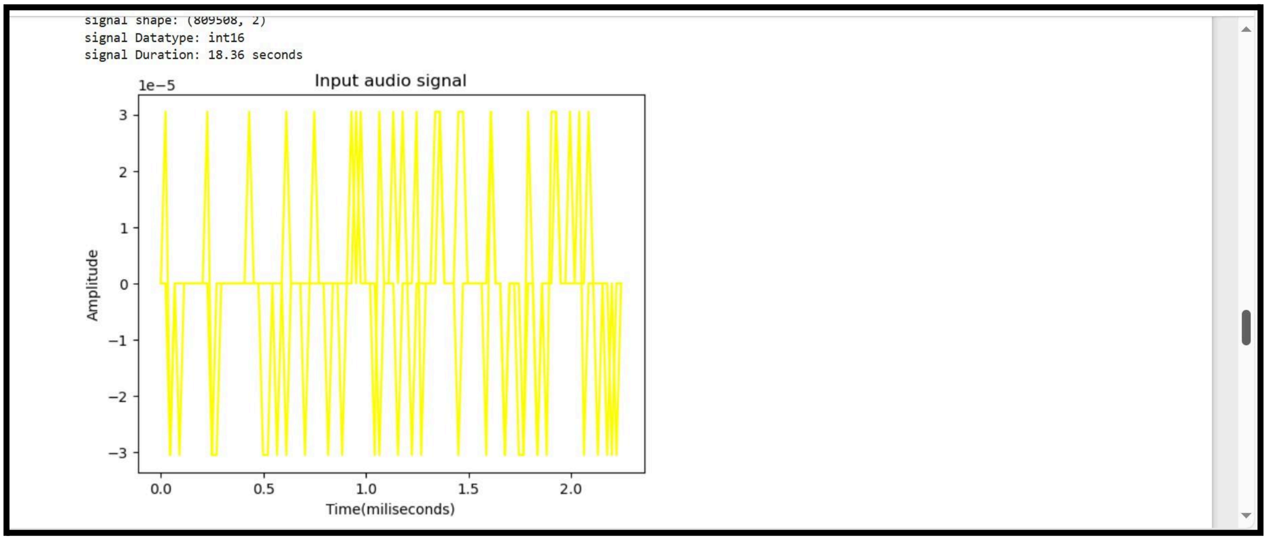
```
signal shape: (946500, 2)
signal Datatype: int16
signal Duration: 21.46 seconds
C:\Users\PMLS\AppData\Local\Temp\ipykernel_13776\4172736692.py:4: WavFileWarning: Chunk (non-data) not understood, skipping it.
  frequency_sampling, audio_signal = wavfile.read("C:/Users/PMLS/Downloads/Audio_dog.wav")
```



CODE:(FOR HARDVAR WAV):

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.io import wavfile
frequency_sampling, audio_signal = wavfile.read("C:/Users/PMLS/Downloads/harvard.wav")
print('signal shape:', audio_signal.shape)
print('signal Datatype:', audio_signal.dtype)
print('signal Duration:', round(audio_signal.shape[0]/float (frequency_sampling),2),'seconds')
audio_signal=audio_signal /np.power(2,15)
audio_signal= audio_signal[:100]
time_axis=1000*np.arange(0,len(audio_signal),1) / float(frequency_sampling)
plt.plot(time_axis,audio_signal,color='yellow')
plt.xlabel('Time(miliseconds)')
plt.ylabel('Amplitude')
plt.title('Input audio signal')
plt.show()
```

OUTPUT:



TASK3:RECOGNITION OF SPOKEN WORD

CODE:

```
!pip install SpeechRecognition
!pip install pyaudio
import speech_recognition as sr
recording=sr.Recognizer()
with sr.Microphone() as source:

    recording.adjust_for_ambient_noise(source)
    print("please say something")
    audio=recording.listen(source)

try:
    print("You said: \n"+ recording.recognize_google(audio))
except Exception as e:
    print(e)
```

OUTPUT:

```
PLEASE SAY SOMETHING
result2:
{   'alternative': [   {'transcript': 'my name is n o f i l Ahmed Khan'},
                       {'transcript': 'my name is n o f i l Ahmad Khan'}],
    'final': True}
YOU SAID:
my name is n o f i l Ahmed Khan
```

EXERCISES:

Q.1:How did different accents or languages impact the transcription process?

ANS: Different accents and languages impact transcription by causing:

1. **Misinterpretations** – Variations in pronunciation lead to incorrect transcriptions.
2. **Homophone Confusion** – Words that sound similar but have different meanings can be transcribed incorrectly.
3. **Phonetic Variability** – Accents alter vowel and consonant sounds, affecting speech recognition accuracy.
4. **Code-Switching Challenges** – Mixing languages within speech confuses transcription systems.
5. **Background Noise Sensitivity** – Accents combined with noise reduce clarity, increasing errors.
6. **Dialectal Difference**s – Regional vocabulary and slang may not be recognized by standard transcription models.

Q.2:Did background noise affect the accuracy of speech recognition? If so, how?

ANS: Yes, background noise significantly affects the accuracy of speech recognition by:

1. **Reducing Signal Clarity** – Noise interferes with speech, making it harder for the ASR system to distinguish words.
2. **Misinterpretation of Words** – Background sounds may be mistaken for speech, leading to incorrect transcriptions.
3. **Lowered Feature Extraction Quality** – ASR systems rely on acoustic features; noise distorts these, decreasing recognition accuracy.
4. **Masking Phonemes** – Important speech sounds get drowned out, causing missing or altered words.
5. **Increased Word Error Rate (WER)** – The presence of noise results in more transcription mistakes.

Q.3:How did the speech recognition system perform when presented with different audio files (e.g., "Eagle" vs. "Elephant")?

ANS:The speech recognition system's performance when presented with different audio files (e.g., *"Eagle"* vs. *"Elephant"*) depends on several factors:

1. **Word Length & Phonetic Similarity** – Shorter words (*"Eagle"*) may be more prone to misrecognition than longer, distinct words (*"Elephant"*).
2. **Pronunciation Clarity** – If words have similar phonemes, ASR may struggle to differentiate them.
3. **Background Noise & Audio Quality** – Noisy or low-quality recordings increase misclassification rates.
4. **Model Training & Vocabulary Size** – Systems trained on diverse datasets perform better at distinguishing words, even those with similar phonetic structures.
5. **Speaker Accent & Enunciation** – Variations in pronunciation may cause the system to misinterpret words.

Q.4:What differences were observed in recognition accuracy when recording voices with different characteristics (e.g., "shrill" vs. "grave")?

ANS: Recognition accuracy varies based on voice characteristics like shrill (high-pitched) vs. grave (low-pitched) due to:

1. **Pitch Sensitivity** – Some ASR models struggle with extreme pitches, especially very high or very low voices.
2. **Feature Extraction Challenges** – Higher frequencies in shrill voices may be harder to detect, while low frequencies in grave voices may blend with background noise.
3. **Training Data Bias** – If an ASR system is trained mostly on mid-range voices, it may perform worse on extreme voice types.
4. **Phoneme Clarity** – Shrill voices may emphasize certain sounds more sharply, while grave voices may produce softer consonants, leading to misinterpretation.
5. **Microphone & Audio Processing Effects** – Some microphones and audio settings handle deep voices better than high-pitched ones, impacting recognition.

Q.5:Introduce yourself and recognized it in spoken word. Analyze the background noise affect.

CODE:

```
import speech_recognition as sr

def recognize_speech():

    recording = sr.Recognizer()

    with sr.Microphone() as source:

        print("Adjusting for ambient noise... Please wait.")
```

```
        # Capture noise level before recording speech
        noise_level_before = recording.energy_threshold
        recording.adjust_for_ambient_noise(source, duration=1)  # Adjust for background noise
        # Capture noise level after noise reduction
        noise_level_after = recording.energy_threshold
        print("Please introduce yourself...")
        audio = recording.listen(source)
    try:
        recognized_text = recording.recognize_google(audio)
        print("\n Recognized Speech:\n" + recognized_text)

        # Analyzing background noise effect
        print("\n Background Noise Analysis:")
        print(f"Noise Level Before: {noise_level_before}")
        print(f"Noise Level After: {noise_level_after}")
        if noise_level_after > noise_level_before:
            print(" High background noise detected. Speech accuracy might be affected.")
        else:
            print(" Noise reduced successfully.")
    except sr.UnknownValueError:
        print("\n Could not understand the audio (high noise or unclear speech).")
    except sr.RequestError as e:
```

```
print(f"\n API request failed: {e}")
    except Exception as e:
        print(f"\n Error: {e}")


# Run the function
recognize_speech()
```

OUTPUT:

```
Adjusting for ambient noise... Please wait.
Please introduce yourself...

 Recognized Speech:
hello this is a no f i l Ahmed Khan and I am 3rd year student of Computer Science and

 Background Noise Analysis:
Noise Level Before: 300
Noise Level After: 112.74289345224548
 Noise reduced successfully.
```

Theoritically:Speech Recognition Analysis:

1. **Recognition Accuracy** – If spoken clearly in a quiet environment, ASR should transcribe this accurately.
2. **Background Noise Impact:**
   - **Low Noise** – Minimal effect, recognition remains accurate.
   - **Moderate Noise (e.g., chatter, soft music)** – System might misinterpret parts of the name or add extra words.
   - **High Noise (e.g., traffic, loud music)** – Increased word error rate (WER), possible omission or incorrect words (e.g., "Hello, my aim is no Phil Ahmed gone").
3. **Voice Characteristics Effect –** A deep (grave) or high-pitched (shrill) voice may slightly impact accuracy depending on model training.
4. **Accent Influence –** Strong accents can cause phoneme shifts, affecting transcription.