# Importing Libraries

In [ ]:
```
import numpy as np
from scipy.io import wavfile
from scipy.signal import butter, lfilter
import matplotlib.pyplot as plt
import librosa
from scipy.stats import pearsonr
```

# Filter Audio Function

- It takes the path of input audio file and returns the filtered audio file
- it first extracts the audio features and then applies the filter on the audio file
- It uses butterworth filter to filter the audio file

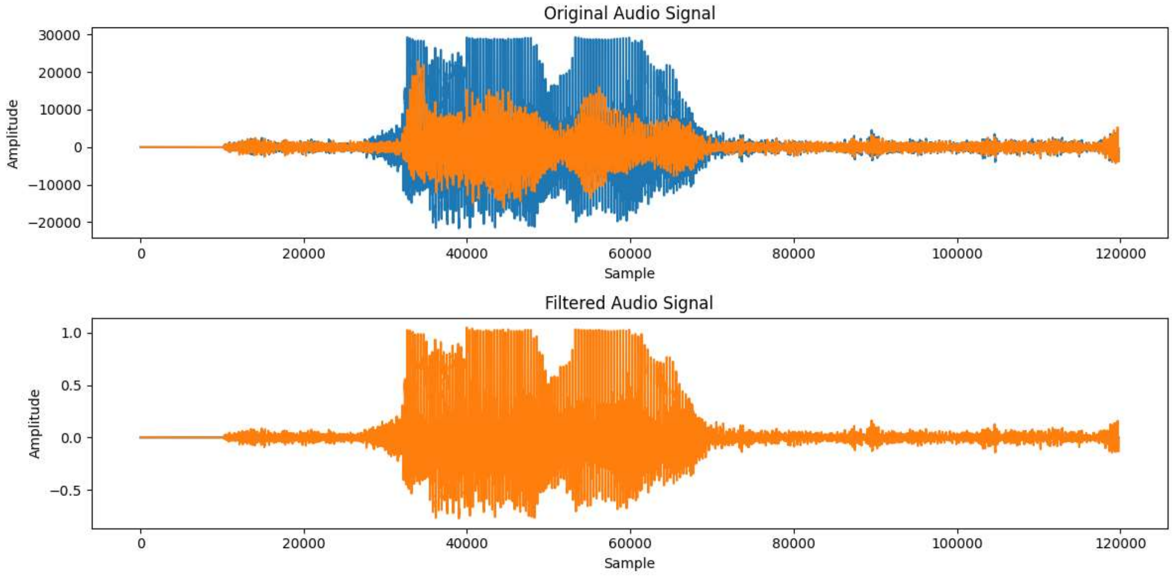
# What is Butterworth Filter?

- Butterworth filter is a type of signal processing filter designed to have as flat a frequency response as possible in the passband. It is also referred to as a maximally flat magnitude filter.
- Butterworth filters are one of the most commonly used digital filters in motion analysis and in audio circuitry. They are called maximally flat magnitude filters because they maximize the flatness of the frequency response in the passband. This means that Butterworth filters have a magnitude response that is as flat as mathematically possible in the passband.

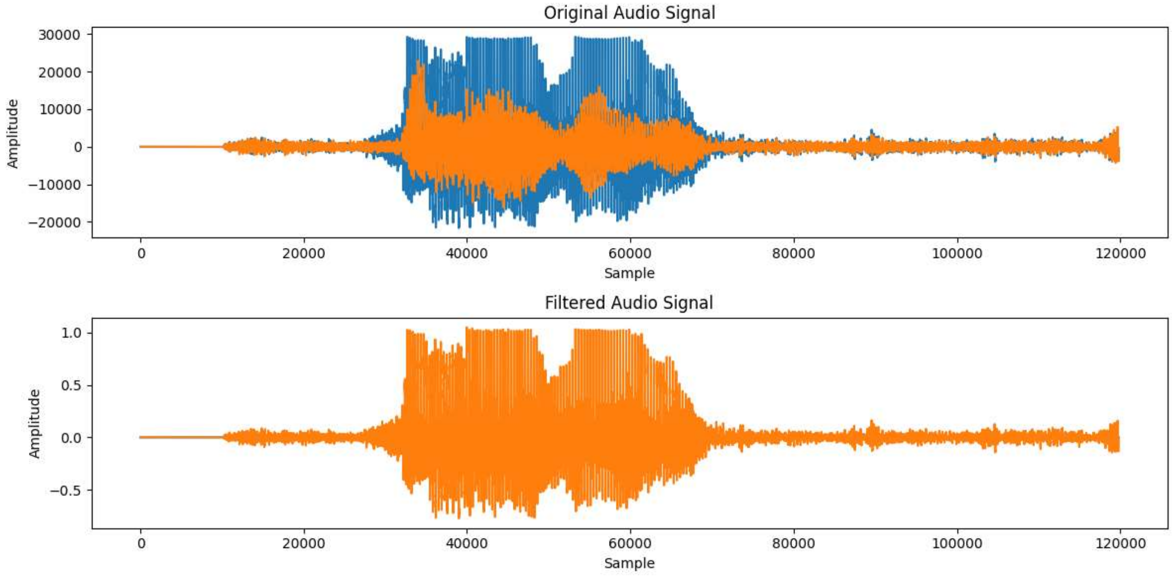
In [ ]:
```
def filter_audio(ipath,opath):
    sample_rate, audio_data = wavfile.read(ipath)
    cutoff_frequency = 2000
    filter_order = 6
    filter_type = 'low'
    nyquist = 0.5 * sample_rate
    normal_cutoff = cutoff_frequency / nyquist
    b, a = butter(filter_order, normal_cutoff, btype=filter_type, analog=False)
    filtered_audio_data = lfilter(b, a, audio_data)
    plt.figure(figsize=(12, 6))
    plt.subplot(2, 1, 1)
    plt.title('Original Audio Signal')
    plt.plot(audio_data)
    plt.xlabel('Sample')
    plt.ylabel('Amplitude')
    plt.subplot(2, 1, 2)
    plt.title('Filtered Audio Signal')
    plt.plot(filtered_audio_data)
    plt.xlabel('Sample')
    plt.ylabel('Amplitude')
    plt.tight_layout()
    plt.show()
    wavfile.write(opath, sample_rate, np.int16(filtered_audio_data))
```

# Filtering the Audio Files

```
In [ ]: filter_audio('./Audio/Original_Password.wav','./Audio/Filtered_Password.wav')
```




```
In [ ]: filter_audio('./Audio/Matching_Password.wav','./Audio/Filtered_Matching_Password
```




# Loading the Audio Files

```
In [ ]: audio_data_original,sample_rate_original = librosa.load('./Audio/Original_Passwo
```

```
In [ ]: audio_data_matching,sample_rate_matching = librosa.load('./Audio/Matching_Passwo
```

# Calculating FFT of the Audio Files

```
In [ ]: X = np.fft.fft(audio_data_original)
        Y = np.fft.fft(audio_data_matching)
```

# Finding Pearson Correlation Coefficient

In [ ]:
```
min_len = min(len(X),len(Y))
X = X[0:min_len]
Y = Y[0:min_len]
```

In [ ]:
```
X_mod = np.abs(X)**2
Y_mod = np.abs(Y)**2
```

In [ ]:
```
r = pearsonr(X_mod,Y_mod)
```

In [ ]:
```
print("Correlation Coefficient: ",r)
```

```
Correlation Coefficient:  PearsonRResult(statistic=1.0, pvalue=0.0)
```