G. ANVITH REDDY

AIE21047

LAB-04

In [ ]:

A1. Use numpy.fft.fft() to transform the speech signal to its spectral domain. Please plot the amplitude part of the spectral components and observe it.

In [7]:

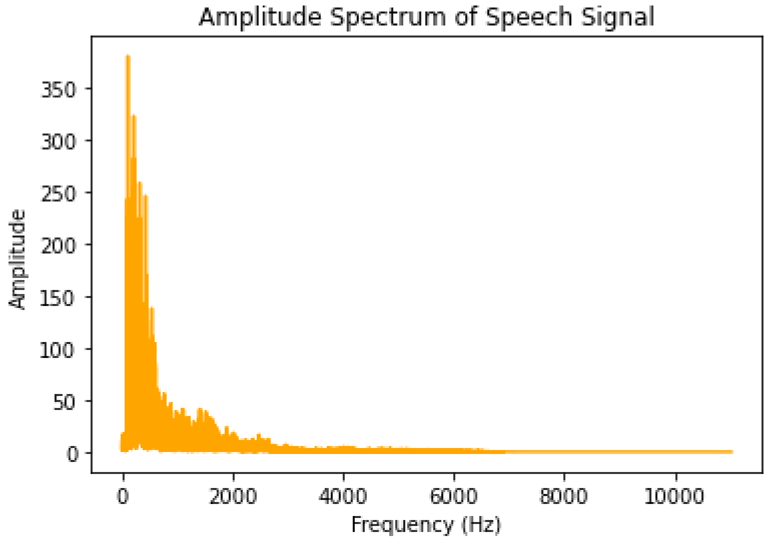
```
import numpy as np
import soundfile as sf
import matplotlib.pyplot as plt
import librosa

signal, rs = librosa.load("statement.wav")

fft_data = np.fft.fft(signal)
amplitude = np.abs(fft_data)

frequencies = np.fft.fftfreq(len(signal), d=1/rs)

print("FFT DATA:\n",fft_data)
print("Amplitude:\n",amplitude)
print("Frequencies:\n",frequencies)
```

```
FFT DATA:
 [-9.85512436+0.j          2.6204152 -4.06482669j -0.08489414+1.82845742j
 ...  2.26048267+0.93895186j -0.08489414-1.82845742j
  2.6204152 +4.06482669j]
Amplitude:
 [9.85512436 4.83625804 1.83042715 ... 2.4477362  1.83042715 4.83625804]
Frequencies:
 [ 0.          0.31459552  0.62919104 ... -0.94378656 -0.62919104
 -0.31459552]
```

In [ ]:

In [21]:

```
import numpy as np
import matplotlib.pyplot as plt
import librosa

# Load the speech signal
signal, rs = librosa.load("statement.wav")

# Perform FFT
fft_result = np.fft.fft(signal)

# Compute amplitude spectrum
amplitude_spectrum = np.abs(fft_result)

# Convert frequency axis to Hertz
freq_axis = np.fft.fftfreq(len(signal), 1 / rs)
positive_freq_axis = freq_axis[:len(freq_axis)//2]  # Keep positive frequencies

# Plot amplitude spectrum with frequency in Hz
plt.plot(positive_freq_axis, amplitude_spectrum[:len(freq_axis)//2],color = "orange")
plt.xlabel('Frequency (Hz)')
plt.ylabel('Amplitude')
plt.title('Amplitude Spectrum of Speech Signal')
plt.show()
```



In [ ]:

A2. Use numpy.fft.ifft() to inverse transform the frequency spectrum of the speech signal from frequency domain to time domain. Compare the generated time domain signal with the original signal.

In [9]:

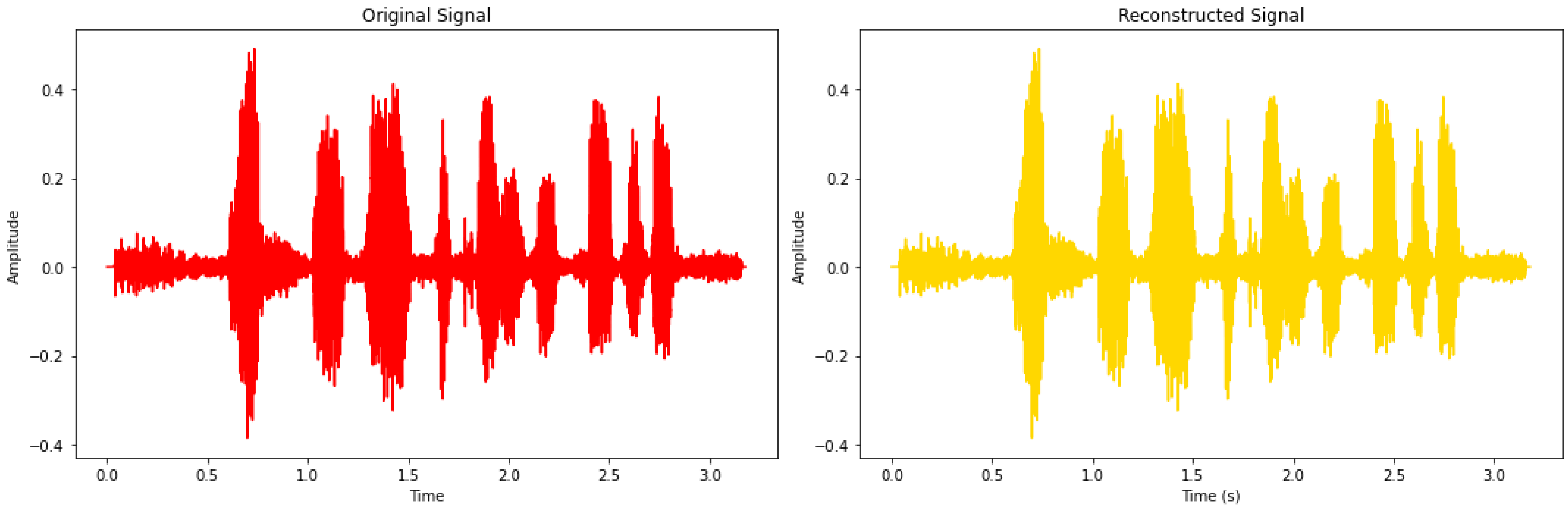
```
time_domain = np.fft.ifft(fft_data)
time_domain = time_domain[:len(signal)]
```

In [11]:

```
time = np.linspace(0, len(signal)/rs, len(signal))
plt.figure(figsize=(15, 5))

plt.subplot(1, 2, 1)
plt.plot(time,signal,color = 'red')
plt.xlabel("Time")
plt.ylabel("Amplitude")
plt.title("Original Signal")

plt.subplot(1, 2, 2)
plt.plot(time, time_domain,color = "gold")
plt.xlabel("Time (s)")
```

```
plt.ylabel("Amplitude")
plt.title("Reconstructed Signal")

plt.tight_layout()
plt.show()
```

```
C:\Users\anvit\anaconda3\lib\site-packages\matplotlib\cbook\__init__.py:1298: ComplexWarning: Casting complex values to real discards the imaginary part
  return np.asarray(x, float)
```



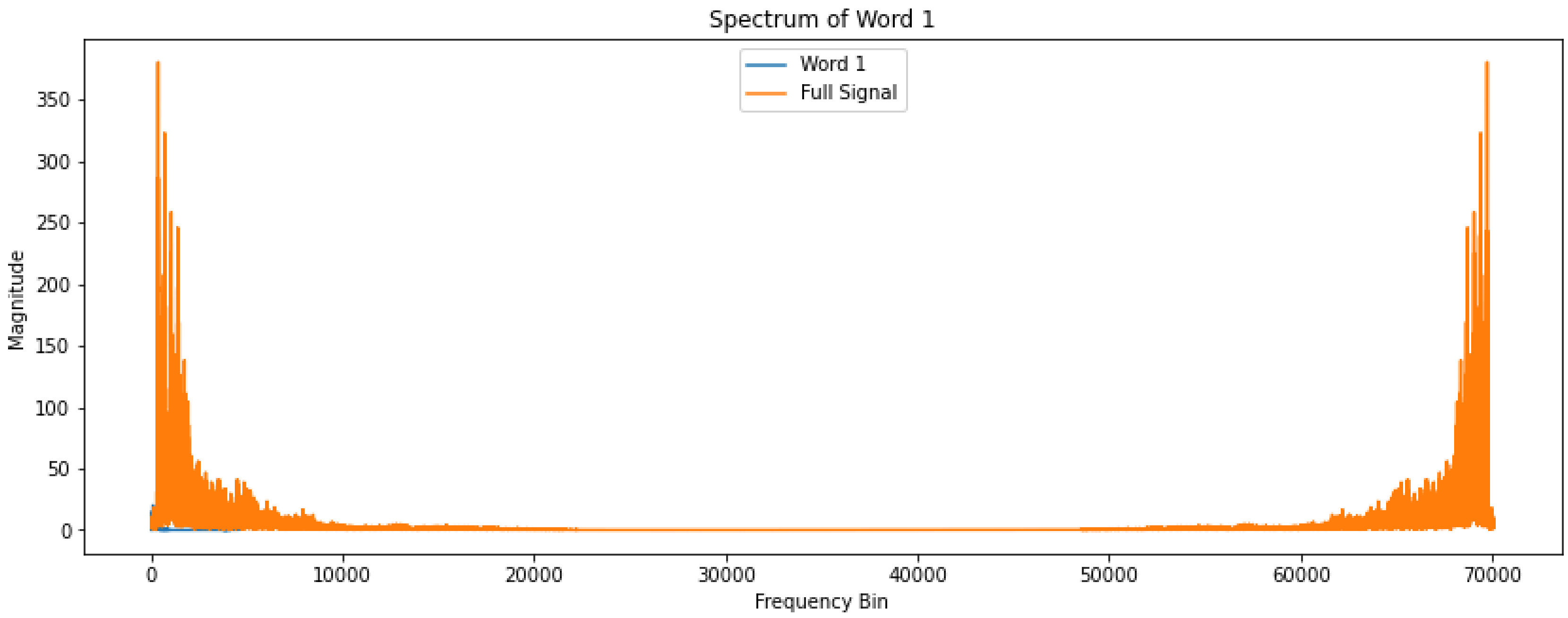
In [ ]:

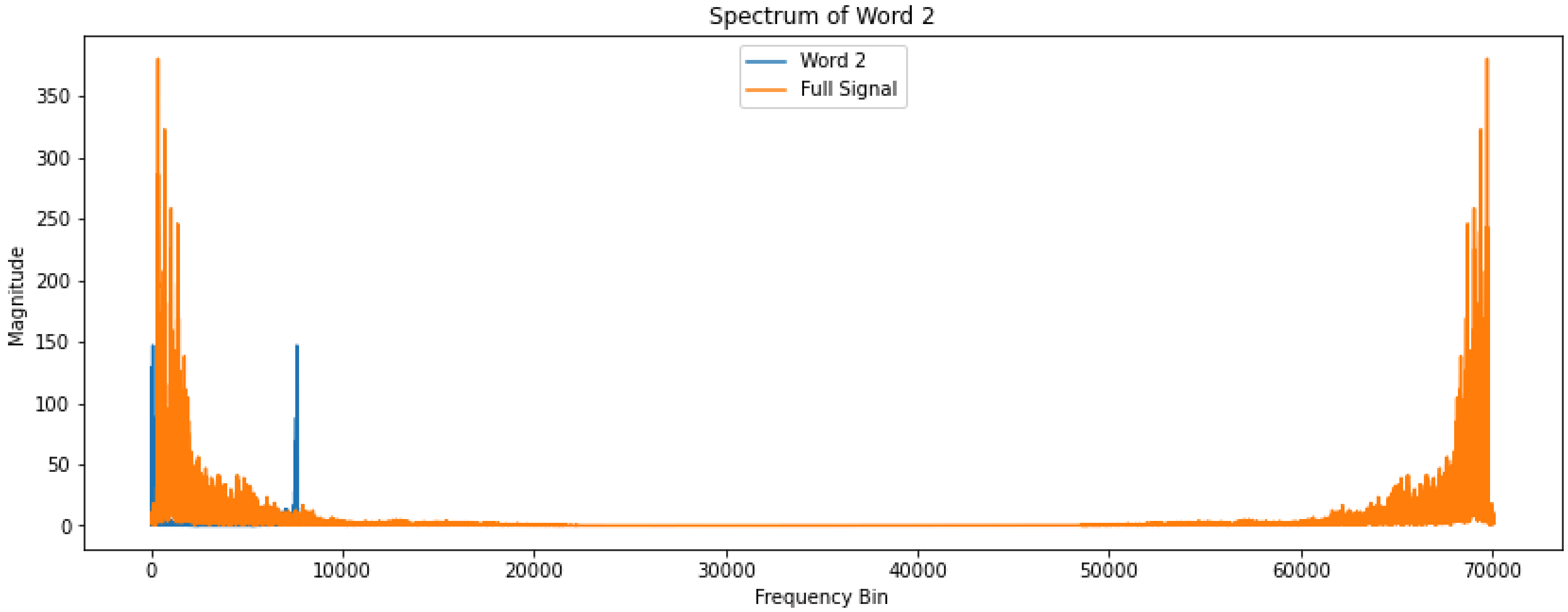
A3. Perform the spectral analysis of a word present in the recorded speech. Compare the spectrum with the spectrum of the full signal.

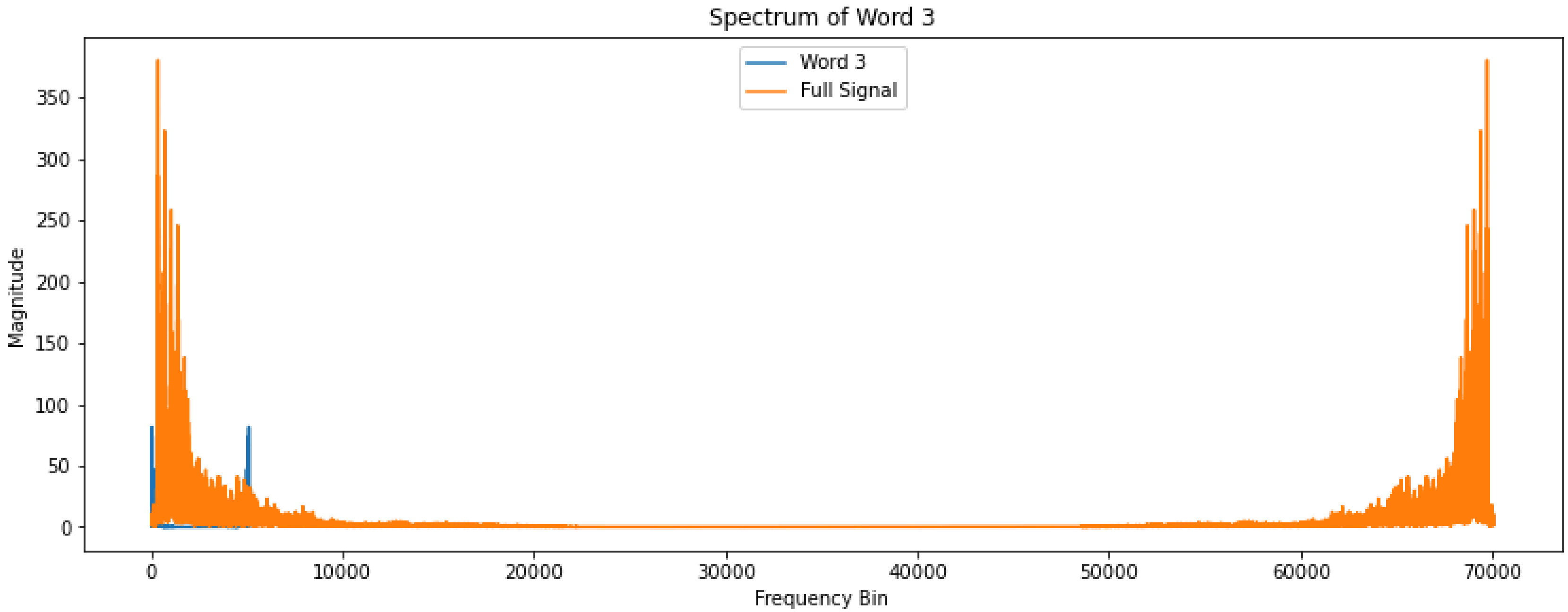
In [ ]:

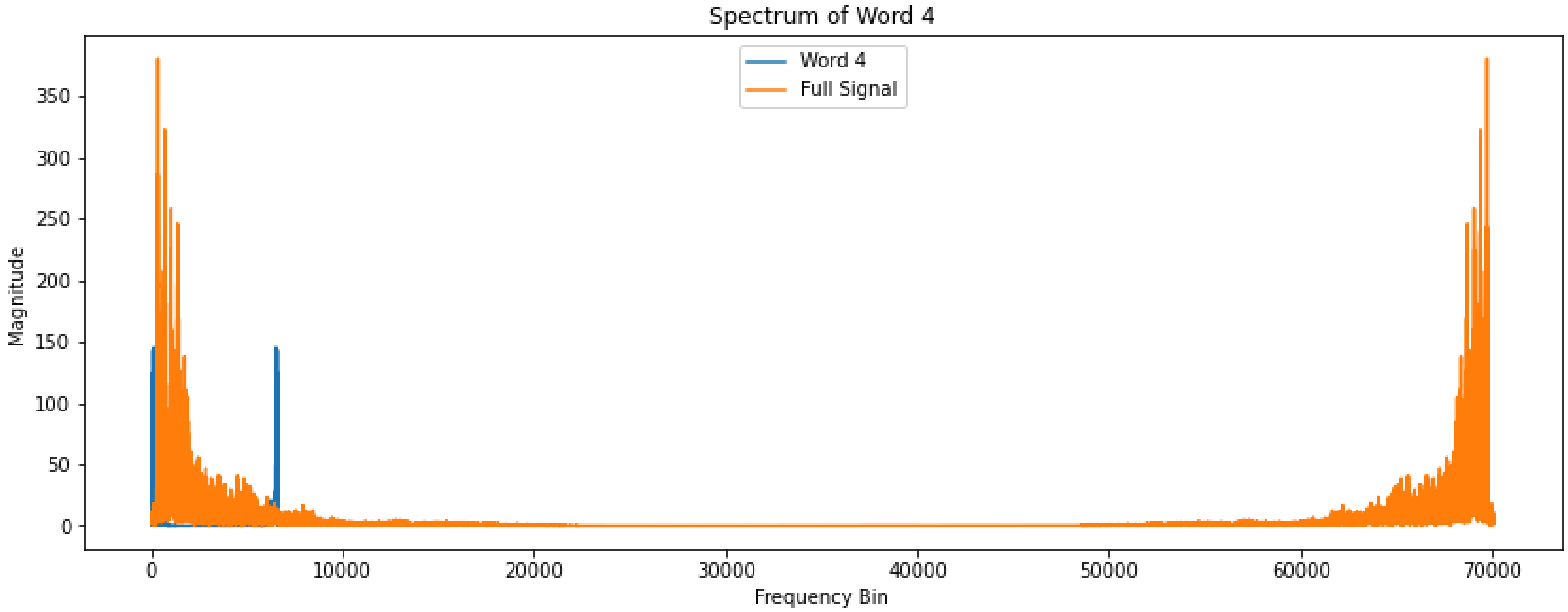
In [24]:
```
import numpy as np
import matplotlib.pyplot as plt
import librosa
```

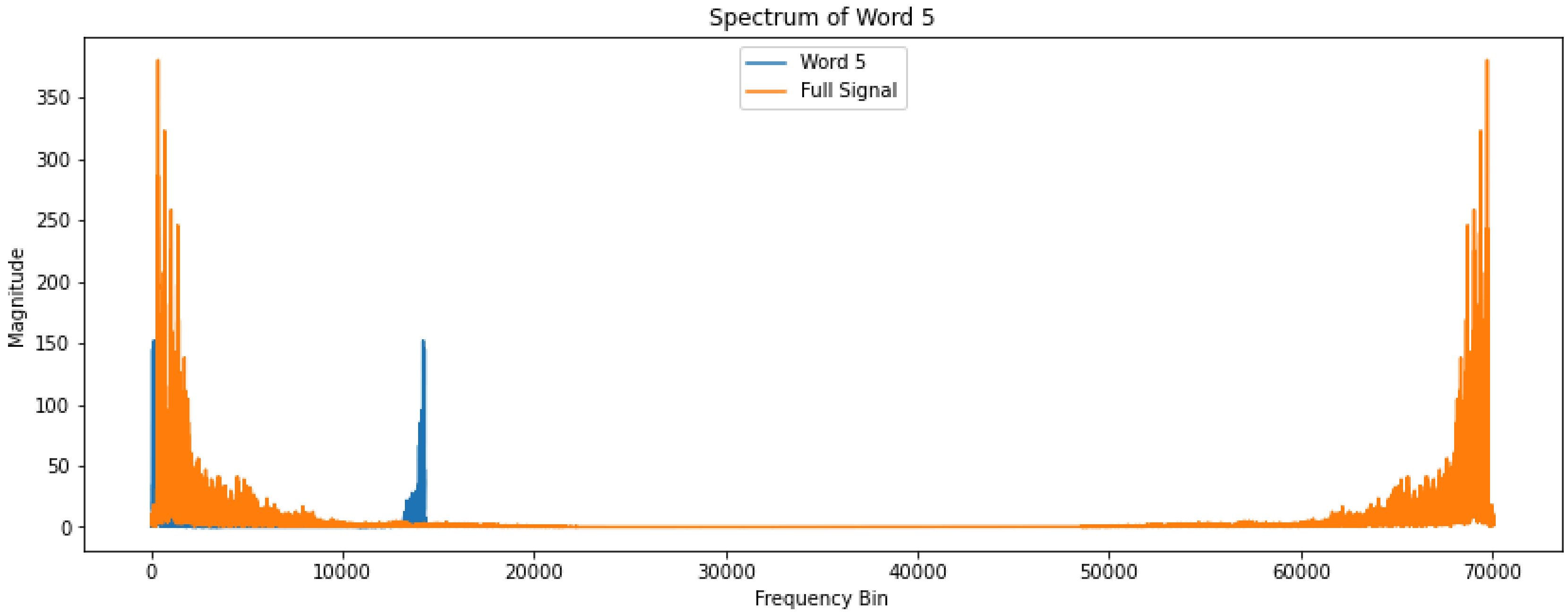
```
speech_signal, rs = librosa.load("statement.wav")

threshold = np.percentile(np.abs(speech_signal), 92)
segments = librosa.effects.split(speech_signal, top_db=-20 * np.log10(threshold))
for i, (start, end) in enumerate(segments):
    word = speech_signal[start:end]
    D_full = np.fft.fft(speech_signal)
    D_word = np.fft.fft(word)
    plt.figure(figsize=(14, 5))
    plt.plot(np.abs(D_word), label=f'Word {i+1}')
    plt.plot(np.abs(D_full), label='Full Signal')

    plt.title(f'Spectrum of Word {i+1}')
    plt.xlabel('Frequency Bin')
    plt.ylabel('Magnitude')
    plt.legend()
    plt.show()
```

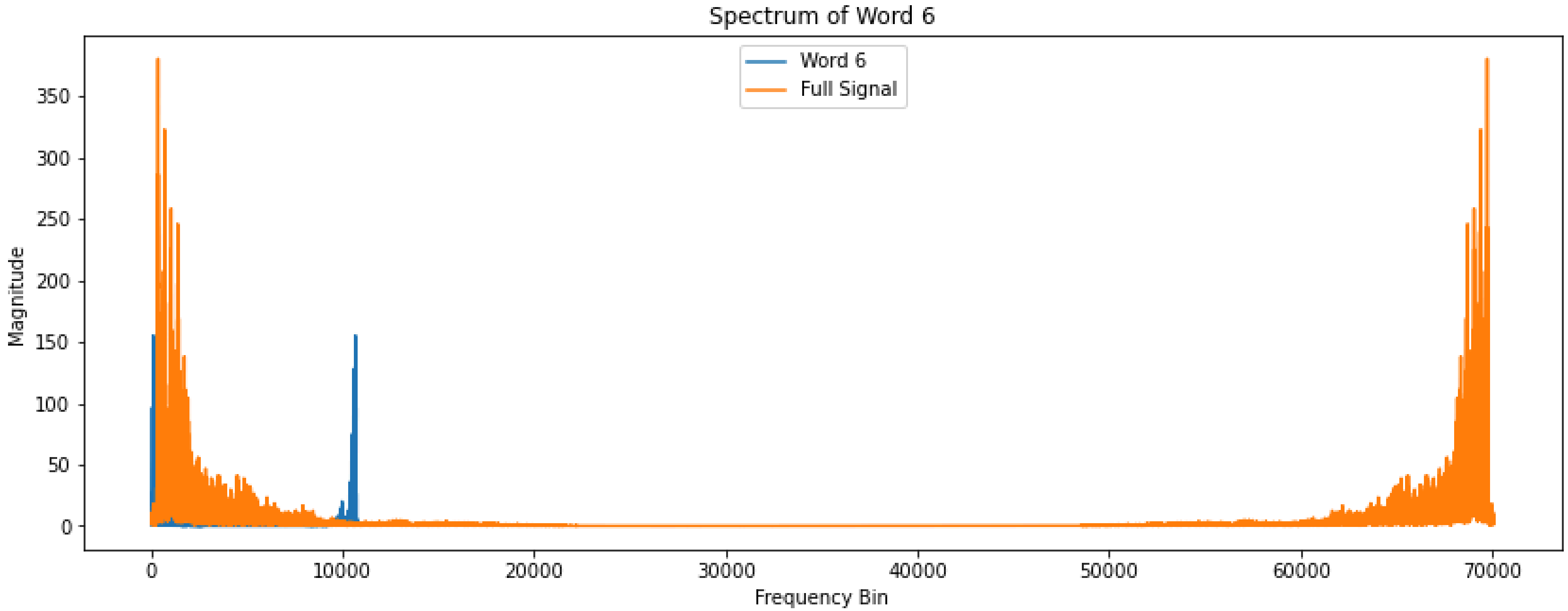



Spectrum of Word 2
Word 2
Full Signal
Magnitude
0
50
100
150
200
250
300
350
0
10000
20000
30000
40000
50000
60000
70000
Frequency Bin


Spectrum of Word 3
Word 3
Full Signal
Magnitude
0
50
100
150
200
250
300
350
0
10000
20000
30000
40000
50000
60000
70000
Frequency Bin


Spectrum of Word 4
Word 4
Full Signal
Magnitude
Frequency Bin
0
50
100
150
200
250
300
350
0
10000
20000
30000
40000
50000
60000
70000


Spectrum of Word 5
Word 5
Full Signal
Magnitude
Frequency Bin
0
50
100
150
200
250
300
350
0
10000
20000
30000
40000
50000
60000
70000



In [ ]:

A4. Take a rectangular window of 20 mili-second sampled at 22.5 KHz. Using FFT, analyse the spectral components

In [ ]:

In [ ]:
```
import numpy as np
import matplotlib.pyplot as plt
from scipy.fft import fft

sr=22500
window_duration = 0.02
window_samples = int(window_duration * sr)
windowed_signal = signal[:window_samples]

# Compute the FFT
X = fft(windowed_signal)
```

```
# Get the one-sided spectrum
n_oneside = window_samples // 2
frequencies = np.arange(n_oneside) * (sr / window_samples)
spectrum = np.abs(X[:n_oneside])

# Plot the frequency spectrum
plt.figure(figsize=(10, 6))
plt.plot(frequencies, spectrum, 'b')
plt.xlabel('Frequency (Hz)')
plt.ylabel('FFT Amplitude ')
plt.title('Frequency Spectrum of Speech Signal')
plt.grid(True)
plt.show()
```

In [ ]:

```
```

A5. Break your speech signal into window lengths of 20 mSec intervals. Evaluate the frequency components using numpy.fft.rfft(). Stack these frequency components as columns in a matrix. Use heatmap plot to display the matrix. You may use librosa.stft() or scipy.signal.stft() as well to achieve this

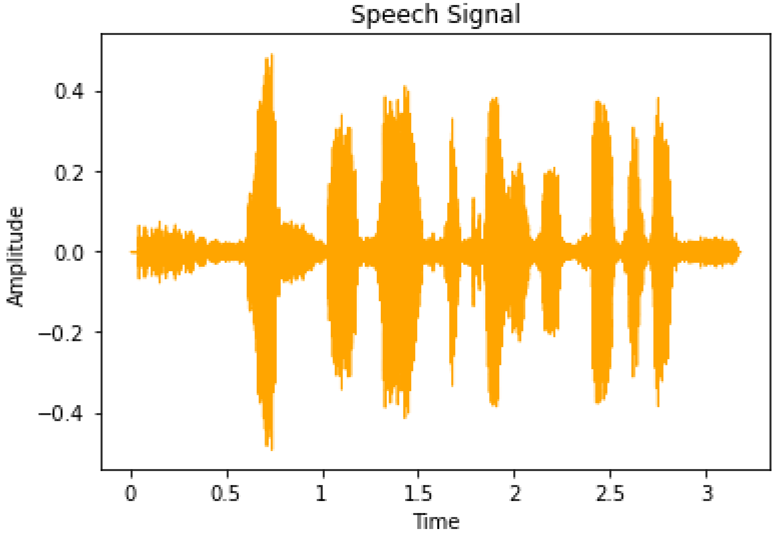
In [ ]:

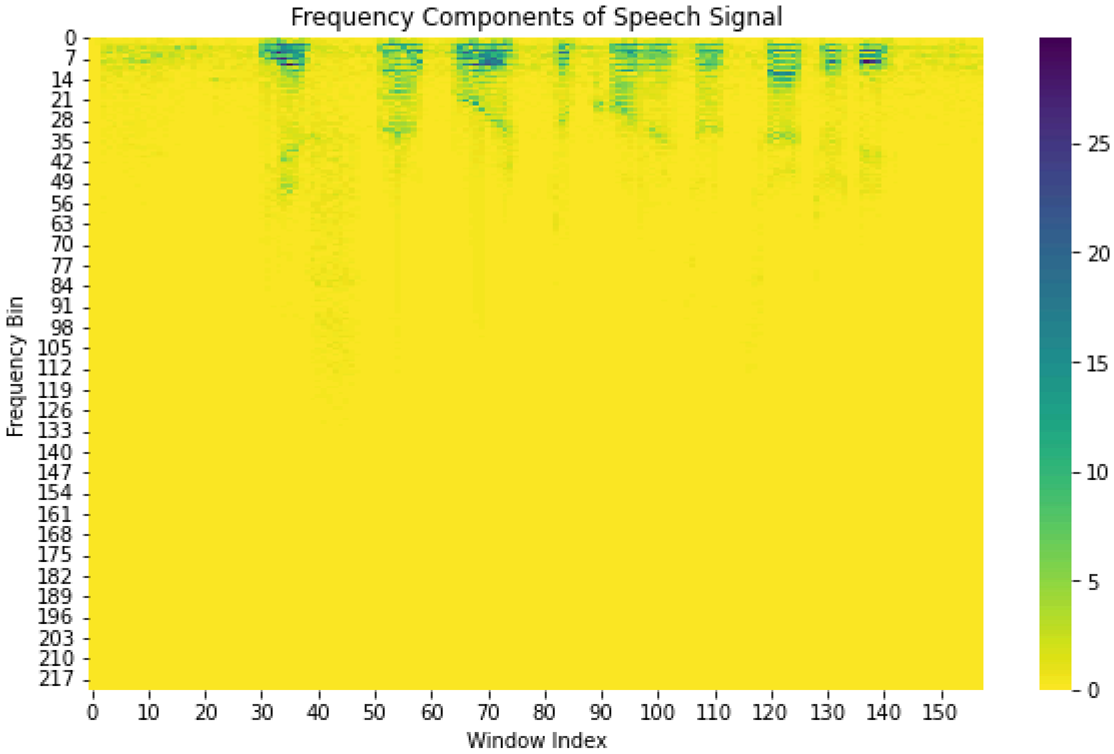
```
```

In [34]:

```
import seaborn as sns

y, rs = librosa.load('statement.wav')
librosa.display.waveshow(y, sr=rs, color = 'orange')
plt.title('Speech Signal')
plt.xlabel('Time')
plt.ylabel('Amplitude')
plt.show()
window_length_sec = 0.02
window_length = int(window_length_sec * sr)
num_windows = len(y) // window_length
freq_matrix = np.zeros((num_windows, window_length // 2 + 1))
```

```
for i in range(num_windows):
    window = y[i * window_length: (i + 1) * window_length]
    fft_result = np.fft.rfft(window)
    freq_matrix[i, :] = np.abs(fft_result)

# Plot the heatmap
plt.figure(figsize=(10, 6))
sns.heatmap(freq_matrix.T, cmap='viridis_r', xticklabels=10)
plt.title('Frequency Components of Speech Signal')
plt.xlabel('Window Index')
plt.ylabel('Frequency Bin')
plt.show()
```





In [ ]:

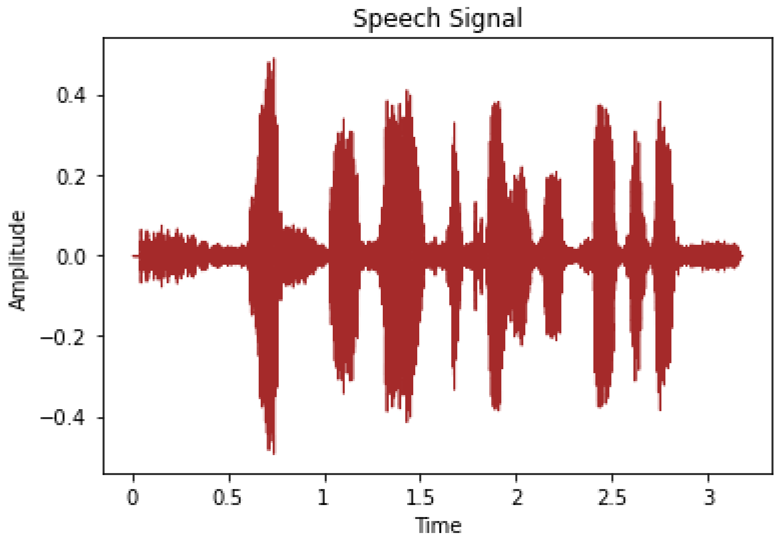
A6. Use scipy.signal.spectrogram() to plot the spectrogram of the speech signal at the same duration. Compare the plots

In [ ]:

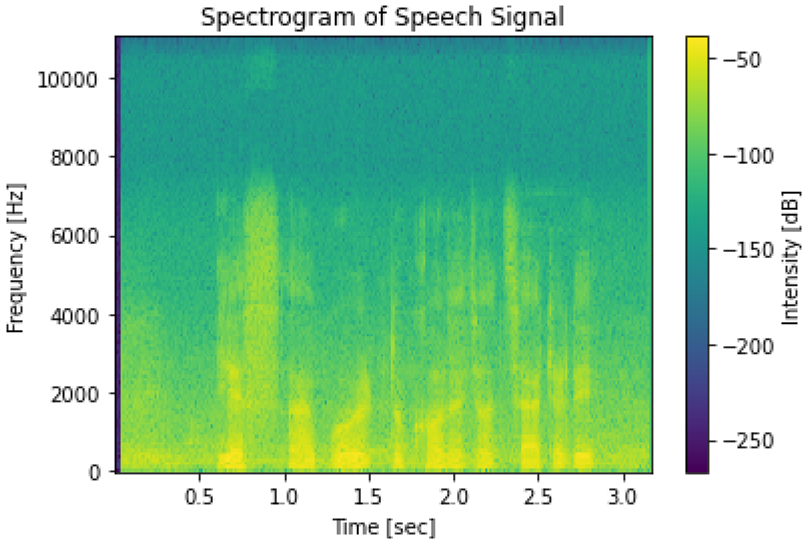
In [38]:

```
from scipy.signal import spectrogram

librosa.display.waveshow(y, color = 'brown')
plt.title('Speech Signal')
plt.xlabel('Time')
plt.ylabel('Amplitude')
plt.show()
```

```
f, t, Sxx = spectrogram(y, sr)
plt.pcolormesh(t, f, 10 * np.log10(Sxx))
plt.ylabel('Frequency [Hz]')
plt.xlabel('Time [sec]')
plt.title('Spectrogram of Speech Signal')
plt.colorbar(label='Intensity [dB]')
plt.show()
```




```
C:\Users\anvit\AppData\Local\Temp\ipykernel_3852\590411667.py:10: RuntimeWarning: divide by zero encountered in log10
  plt.pcolormesh(t, f, 10 * np.log10(Sxx))
```


Spectrogram of Speech Signal
Frequency [Hz]
10000
8000
6000
4000
2000
0
0.5
1.0
1.5
2.0
2.5
3.0
Time [sec]
Intensity [dB]
−50
−100
−150
−200
−250

```
In [ ]:
```