In [ ]:
```
# Individual Assignment
# Mason Wheeler
# Creating my own 5-band equalizer
```

Sub-bass: 20 to 60 Hz Bass: 60 to 250 Hz Lower midrange: 250 to 500 Hz Midrange: 500 to 2 kHz Upper midrange: 2 to 4 kHz Presence: 4 to 6 kHz Brilliance: 6 to 20 kHz

In [ ]:
```
import numpy as np
from scipy.io import wavfile
from scipy import signal
import simpleaudio as sa

# Define the cutoff frequencies for the 5 bands in terms of Nyquist rate
cutoffs = [0.01, 0.05, 0.1, 0.4, 0.8, 1.0]  # Adjust as necessary
orders = [5] * 6  # Use 5th order filters

# Create filters for the bands
filters = [signal.butter(order, cutoff, 'low' if i == 0 else 'high' if i == ler
                         btype='butter', analog=False, output='ba')
           for i, (order, cutoff) in enumerate(zip(orders, cutoffs))]

# Function to equalize audio
def equalize_audio(audio, gains):
    assert len(gains) == len(filters), "Number of gains must equal number of fi

    # Apply each filter and gain to the audio and sum the results
    equalized_audio = sum(gain * signal.lfilter(b, a, audio) for (b, a), gain i
    return equalized_audio

# Load audio
fs, audio = wavfile.read('music.wav')
audio = audio.astype(float)

# Equalize the audio
gains = [1.0, 0.8, 1.2, 0.9, 1.1, 0.95]  # Sample gains; adjust as necessary
equalized_audio = equalize_audio(audio, gains)

# Save equalized audio
wavfile.write('equalized_music.wav', fs, equalized_audio.astype(np.int16))

# Play the original and equalized audio for comparison
print("Playing original audio...")
sa.play_buffer(audio.astype(np.int16), 1, 2, fs).wait_done()
print("Playing equalized audio...")
sa.play_buffer(equalized_audio.astype(np.int16), 1, 2, fs).wait_done()
```