

Hands-On Audio Processing

IF4021 - Multimedia Information Processing

Nama: Bagas Andreanto

NIM: 122140017

Link Github Repository: https://github.com/bagas017/Hands-on_Multimedia

Deskripsi Tugas Tugas ini dirancang untuk menguji pemahaman mahasiswa terhadap konsep-konsep fundamental dalam pemrosesan audio digital termasuk manipulasi sinyal audio, filtering, pitch shifting, normalisasi, dan teknik remix audio. Mahasiswa diharapkan dapat menerapkan teori yang telah dipelajari dalam praktik langsung menggunakan Python dan pustaka pemrosesan audio.

Library yang digunakan:

In [3]:

```
import numpy as np
import matplotlib.pyplot as plt
import librosa
import soundfile as sf
import scipy
from scipy.signal import butter, filtfilt, lfilter
from IPython.display import Audio, HTML, display
import os
import pyloudnorm as pyln

print("Library yang digunakan:")
print(f"NumPy      : {np.__version__}")
print(f"Matplotlib : {plt.matplotlib.__version__}")
print(f"Librosa    : {librosa.__version__}")
print(f"SciPy      : {scipy.__version__}")
print(f"SoundFile  : {sf.__version__}")
```

Library yang digunakan:

```
NumPy      : 1.26.4
Matplotlib : 3.9.2
Librosa    : 0.10.2
SciPy      : 1.13.1
SoundFile  : 0.12.1
```

Soal 1: Rekaman dan Analisis Suara Multi-Level

In [4]:

```
Path_Soal_Audio1 = os.path.join(os.getcwd(), 'data', 'Soal1_Audio.wav')

if os.path.exists(Path_Soal_Audio1):
    y, sr = librosa.load(Path_Soal_Audio1, sr=None)
```

```

source_info = f'Lokasi File: {Path_Soal_Audio1}'

print(source_info)
print(f"Shape: {y.shape}")
print(f"Sample rate: {sr:,} Hz")
print(f'Durasi: {len(y)/sr:.2f} detik')

```

Lokasi File: c:\Users\Pongo\OneDrive\Desktop\Git-Upload\Hands-on_Multimedia\Hands-on Pemrosesan Audio\data\Soal1_Audio.wav
 Shape: (1203200,)
 Sample rate: 48,000 Hz
 Durasi: 25.07 detik

Visualisasi Waveform dan Spectrogram

```

In [5]: # Plot waveform
duration = len(y) / sr
time = np.linspace(0, duration, len(y))

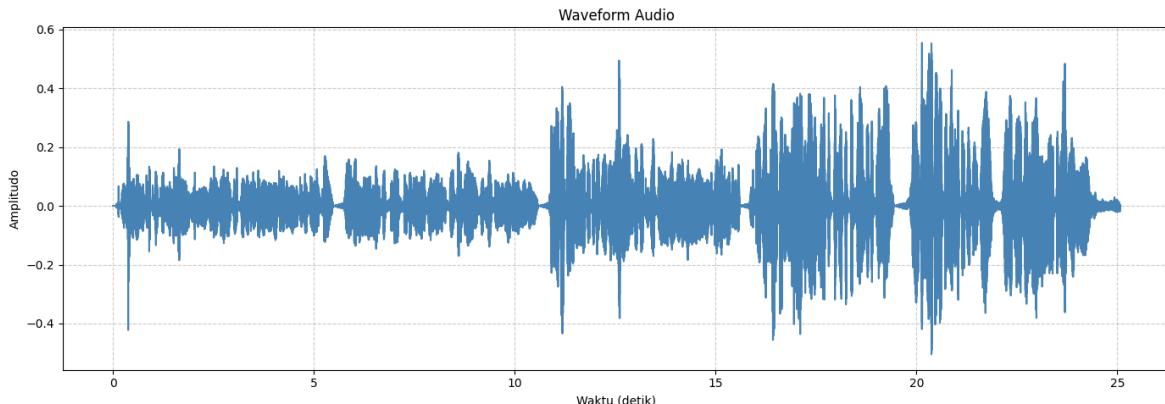
plt.figure(figsize=(14, 5))
plt.plot(time, y, color='steelblue')
plt.title('Waveform Audio')
plt.xlabel('Waktu (detik)')
plt.ylabel('Amplitudo')
plt.grid(True, which='both', linestyle='--', alpha=0.6)
plt.tight_layout()
plt.show()

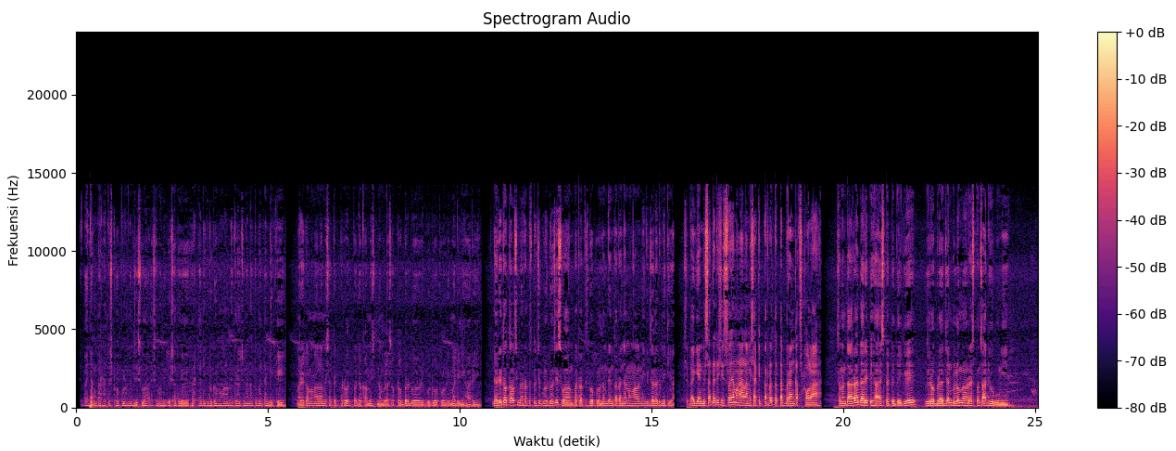
# Plot spectrogram
D = librosa.amplitude_to_db(np.abs(librosa.stft(y)), ref=np.max)

plt.figure(figsize=(14, 5))
librosa.display.specshow(D, sr=sr, x_axis='time', y_axis='hz')
plt.colorbar(format='%+2.0f dB')
plt.title('Spectrogram Audio')
plt.xlabel('Waktu (detik)')
plt.ylabel('Frekuensi (Hz)')
plt.tight_layout()
plt.show()

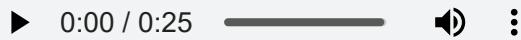
# Audio player
print("Audio Player:")
display(Audio(y, rate=sr))

```





Audio Player:



Penjelasan:

Dapat dilihat pada waveform, untuk di detik awal 0-5 detik memiliki level suara yang sedikit rendah hal ini karena rekaman dilakukan dengan cara berbisik. Kemudian pada detik ke 5-10 suara mulai meningkat karena rekaman dilakukan dengan cara berbicara normal. Selanjutnya pada detik ke 10-15 suara mulai meningkat lagi karena rekaman dilakukan dengan cara berbicara dengan volume yang lebih keras. Pada detik ke 15-20 gelombang sedikit lebih rapat (frekuensi naik) karena audio direkam dengan suara cempreng. Dan pada detik ke 20-25 amplitudo naik karena suara direkam dengan cara berteriak.

Kenapa bentuk visualisasi dari suara berbisik dan normal terlihat mirip?

Mungkin karena saya terlalu dekat dalam merekam suara berbisik, disebabkan situasi di kos tidak kondusif untuk merekam suara berbisik. Sehingga visualisasi yang dihasilkan terlihat mirip meskipun jika di dengar secara langsung suara berbisik akan terdengar sedikit lebih pelan dibandingkan suara normal.

Resampling Audio

```
In [6]: # Resampling Audio
print("Sample rate asli:", sr)

# Target resampling 16000 Hz
target_sr = 16000
y_resampled = librosa.resample(y=y, orig_sr=sr, target_sr=target_sr)
print("Sample rate baru:", target_sr)

# Menyimpan output resampling
output_dir = os.path.join(os.getcwd(), 'output')
os.makedirs(output_dir, exist_ok=True)
output_filename = 'Soal1_Audio_Resampled.wav'
output_path = os.path.join(output_dir, output_filename)

sf.write(output_path, y_resampled, target_sr)
```

```

# Perbandingan waveform sebelum dan sesudah resampling

# Waveform sebelum before
plt.figure(figsize=(14, 5))
plt.subplot(2, 1, 1)
plt.plot(time, y, color='steelblue')
plt.title('Waveform Sebelum Resampling')
plt.xlabel('Waktu (detik)')
plt.ylabel('Amplitudo')
plt.grid(True, which='both', linestyle='--', alpha=0.6)
plt.tight_layout()
plt.show()

# Waveform sesudah after
Path_Output_Resampled = os.path.join(os.getcwd(), 'output', 'Soal1_Audio_Resampl')

if os.path.exists(Path_Output_Resampled):
    y_resampled, sr_resampled = librosa.load(Path_Output_Resampled, sr=None)
    duration_resampled = len(y_resampled) / sr_resampled
    time_resampled = np.linspace(0, duration_resampled, len(y_resampled))

plt.figure(figsize=(14, 5))
plt.subplot(2, 1, 1)
plt.plot(time, y, color='orange')
plt.title('Waveform Sebelum Resampling')
plt.xlabel('Waktu (detik)')
plt.ylabel('Amplitudo')
plt.grid(True, which='both', linestyle='--', alpha=0.6)
plt.tight_layout()
plt.show()

# Spectrogram sebelum before
D_before = librosa.amplitude_to_db(np.abs(librosa.stft(y)), ref=np.max)

plt.figure(figsize=(14, 5))
librosa.display.specshow(D_before, sr=sr, x_axis='time', y_axis='hz')
plt.colorbar(format='%+2.0f dB')
plt.title('Spectrogram Sebelum Resampling')
plt.xlabel('Waktu (detik)')
plt.ylabel('Frekuensi (Hz)')
plt.tight_layout()
plt.show()

# Spectrogram sesudah after
D_after = librosa.amplitude_to_db(np.abs(librosa.stft(y_resampled)), ref=np.max)

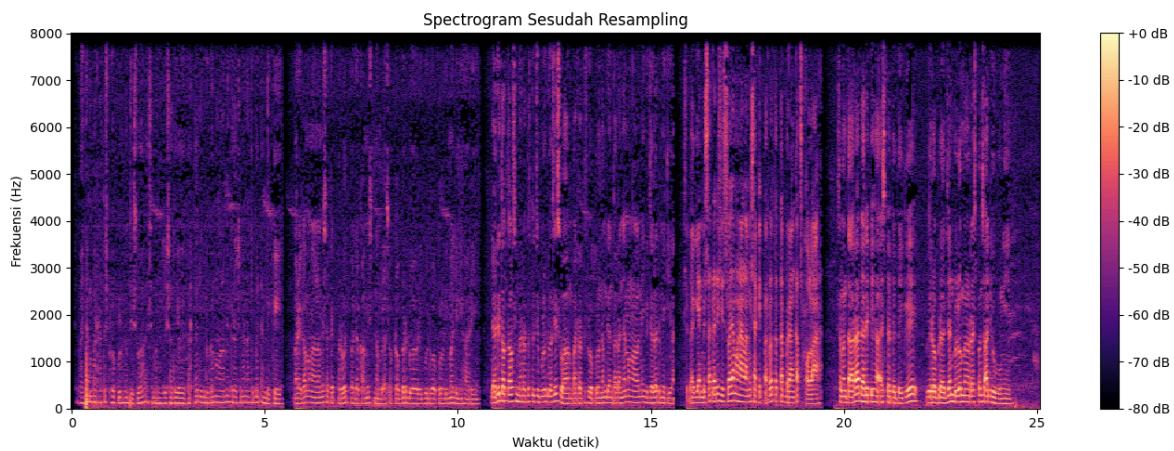
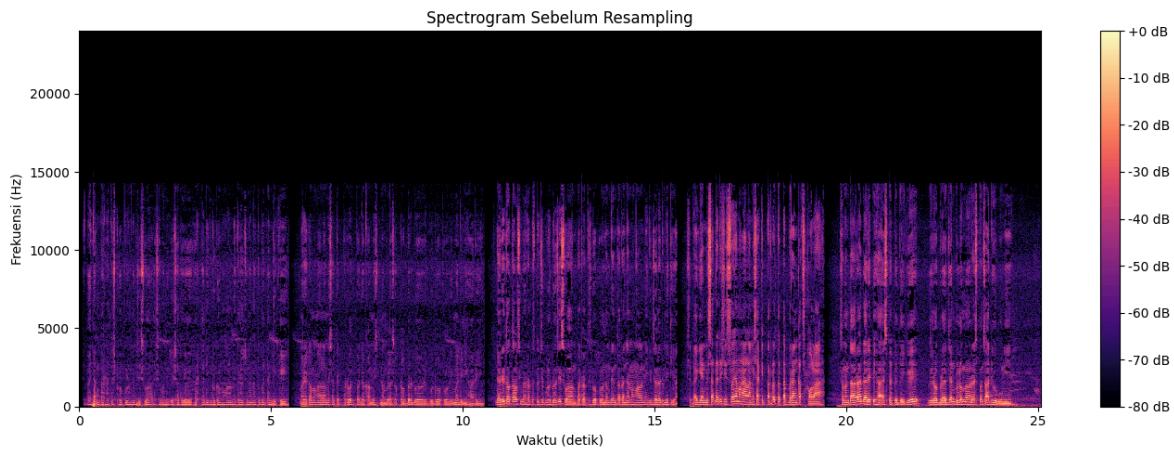
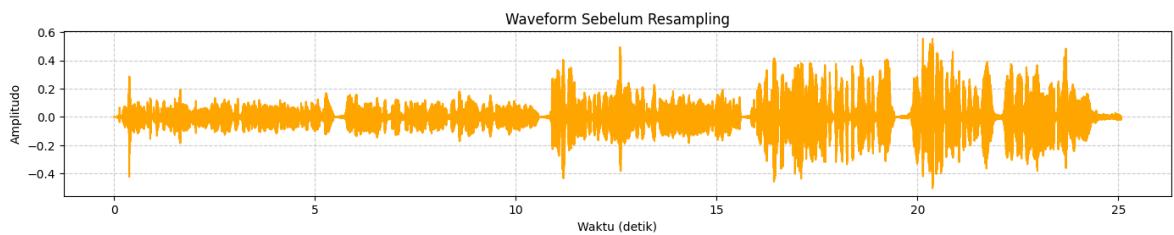
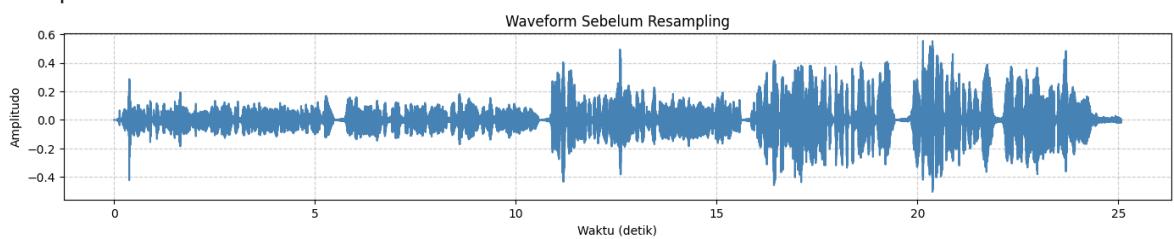
plt.figure(figsize=(14, 5))
librosa.display.specshow(D_after, sr=sr_resampled, x_axis='time', y_axis='hz')
plt.colorbar(format='%+2.0f dB')
plt.title('Spectrogram Sesudah Resampling')
plt.xlabel('Waktu (detik)')
plt.ylabel('Frekuensi (Hz)')
plt.tight_layout()
plt.show()

# Perbandingan Audio Player
print("Audio Player Sebelum Resampling:")
display(Audio(y, rate=sr))

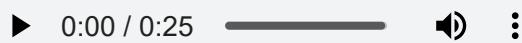
```

```
print("Audio Player Sesudah Resampling:")
display(Audio(y_resampled, rate=sr_resampled))
```

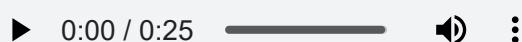
Sample rate asli: 48000
Sample rate baru: 16000



Audio Player Sebelum Resampling:



Audio Player Sesudah Resampling:



Penjelasan:

Untuk dibagian waveform memang tidak terlihat perbedaan secara jelas, karena akan sulit untuk membedakan frekuensi dari waveform.

Namun untuk spectrogram terlihat jelas untuk perbedaan dari hasil setelah di lakukan downsampling. Pada audio asli, frekuensi bisa terlihat hingga ke 15000 Hz, sedangkan setelah di downsampling menjadi 8000 Hz.

Dari hasil perbandingan audio player juga menunjukan kualitas suara yang menurun untuk hasil setelah downsampling, dimana suara menjadi sedikit lebih pecah dan kurang jernih dibandingkan audio asli.

Soal 2: Noise Reduction dengan Filtering

```
In [7]: Path_Soal_Audio2 = os.path.join(os.getcwd(), 'data', 'Soal2_Audio.wav')

if os.path.exists(Path_Soal_Audio2):
    y2, sr2 = librosa.load(Path_Soal_Audio2, sr=None)
    source_info2 = f'Lokasi File: {Path_Soal_Audio2}'

print(source_info2)
print(f'Shape: {y2.shape}')
print(f'Sample rate: {sr2:,} Hz')
print(f'Durasi: {len(y2)/sr2:.2f} detik')
```

```
Lokasi File: c:\Users\Pongo\OneDrive\Desktop\Git-Upload\Hands-on_Multimedia\Hands-on Pemrosesan Audio\data\Soal2_Audio.wav
Shape: (395264,)
Sample rate: 48,000 Hz
Durasi: 8.23 detik
```

Low-Pass Filtering

```
In [8]: # Fungsi Low-pass filter
def butter_lowpass(cutoff, sr, order=4):
    nyq = 0.5 * sr # frekuensi Nyquist
    normal_cutoff = cutoff / nyq
    b, a = butter(order, normal_cutoff, btype='low', analog=False)
    return b, a

# Fungsi untuk menerapkan filter
def apply_lowpass(y, sr, cutoff=8000, order=4):
    b, a = butter_lowpass(cutoff, sr, order)
    y_filtered = filtfilt(b, a, y)
    return y_filtered

# Terapkan filter (6500 Hz)
y2_filtered = apply_lowpass(y2, sr2, cutoff=6500)

# Simpan hasil filtering
output_dir = os.path.join(os.getcwd(), 'output')
os.makedirs(output_dir, exist_ok=True)
output_path = os.path.join(output_dir, 'Soal2_LowPass.wav')

sf.write(output_path, y2_filtered, sr2)
```

```
print("\n== Low-pass filtering selesai! ===")
print(f"Hasil disimpan di: {output_path}")

# Spectrogram sebelum dan sesudah filtering
D2_before = librosa.amplitude_to_db(np.abs(librosa.stft(y2)), ref=np.max)
D2_after = librosa.amplitude_to_db(np.abs(librosa.stft(y2_filtered)), ref=np.max)

plt.figure(figsize=(14, 10))

plt.subplot(2, 1, 1)
librosa.display.specshow(D2_before, sr=sr2, x_axis='time', y_axis='hz')
plt.colorbar(format='%+2.0f dB')
plt.title('Spectrogram Sebelum Filtering')
plt.xlabel('Waktu (detik)')
plt.ylabel('Frekuensi (Hz)')

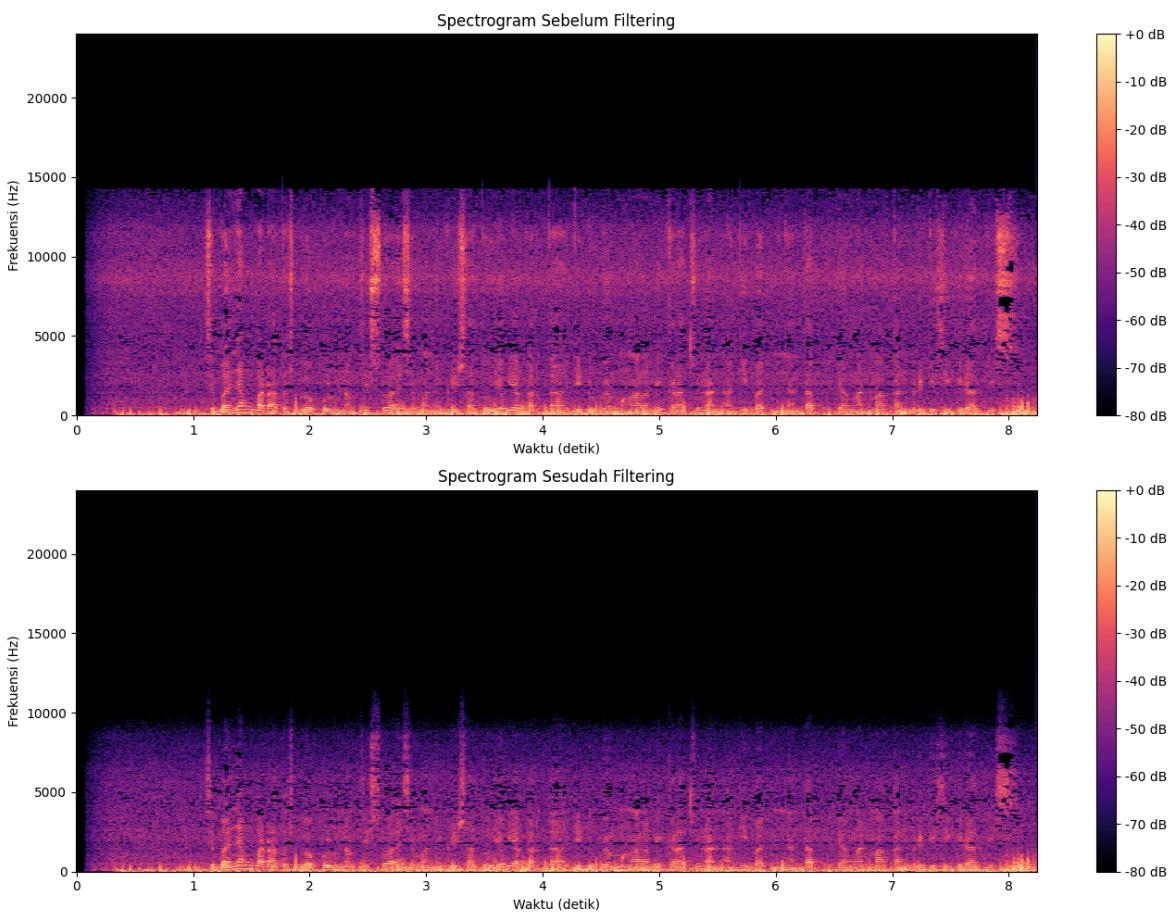
plt.subplot(2, 1, 2)
librosa.display.specshow(D2_after, sr=sr2, x_axis='time', y_axis='hz')
plt.colorbar(format='%+2.0f dB')
plt.title('Spectrogram Sesudah Filtering')
plt.xlabel('Waktu (detik)')
plt.ylabel('Frekuensi (Hz)')

plt.tight_layout()
plt.show()

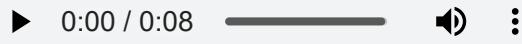
print("Audio Player Sebelum Filtering:")
display(Audio(y2, rate=sr2))

print("Audio Player Sesudah Low-pass Filtering:")
display(Audio(y2_filtered, rate=sr2))
```

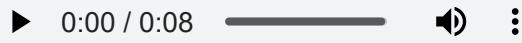
```
== Low-pass filtering selesai!
Hasil disimpan di: c:\Users\Pongo\OneDrive\Desktop\Git-Upload\Hands-on_Multimedia
\Hands-on Pemrosesan Audio\output\Soal2_LowPass.wav
```



Audio Player Sebelum Filtering:



Audio Player Sesudah Low-pass Filtering:



High-pass Filtering

```
In [9]: # Fungsi high-pass filter
def butter_highpass(cutoff, sr, order=4):
    nyq = 0.5 * sr # frekuensi Nyquist
    normal_cutoff = cutoff / nyq
    b, a = butter(order, normal_cutoff, btype='high', analog=False)
    return b, a

# Fungsi untuk menerapkan filter
def apply_highpass(y, sr, cutoff=100, order=4):
    b, a = butter_highpass(cutoff, sr, order)
    y_filtered = filtfilt(b, a, y)
    return y_filtered

# Terapkan filter
y2_filtered = apply_highpass(y2, sr2, cutoff=200)

# Simpan hasil ke folder output
output_dir = os.path.join(os.getcwd(), 'output')
os.makedirs(output_dir, exist_ok=True)
output_path = os.path.join(output_dir, 'Soal2_HighPass.wav')
```

```

sf.write(output_path, y2_filtered, sr2)

print("\n==== High-pass filtering selesai! ===")
print(f"Hasil disimpan di: {output_path}")


# Spectrogram sebelum dan sesudah filtering
D2_before = librosa.amplitude_to_db(np.abs(librosa.stft(y2)), ref=np.max)
D2_after = librosa.amplitude_to_db(np.abs(librosa.stft(y2_filtered)), ref=np.max)

plt.figure(figsize=(14, 10))

plt.subplot(2, 1, 1)
librosa.display.specshow(D2_before, sr=sr2, x_axis='time', y_axis='hz')
plt.colorbar(format='%+2.0f dB')
plt.title('Spectrogram Sebelum High-pass Filtering')
plt.xlabel('Waktu (detik)')
plt.ylabel('Frekuensi (Hz)')

plt.subplot(2, 1, 2)
librosa.display.specshow(D2_after, sr=sr2, x_axis='time', y_axis='hz')
plt.colorbar(format='%+2.0f dB')
plt.title('Spectrogram Sesudah High-pass Filtering')
plt.xlabel('Waktu (detik)')
plt.ylabel('Frekuensi (Hz)')

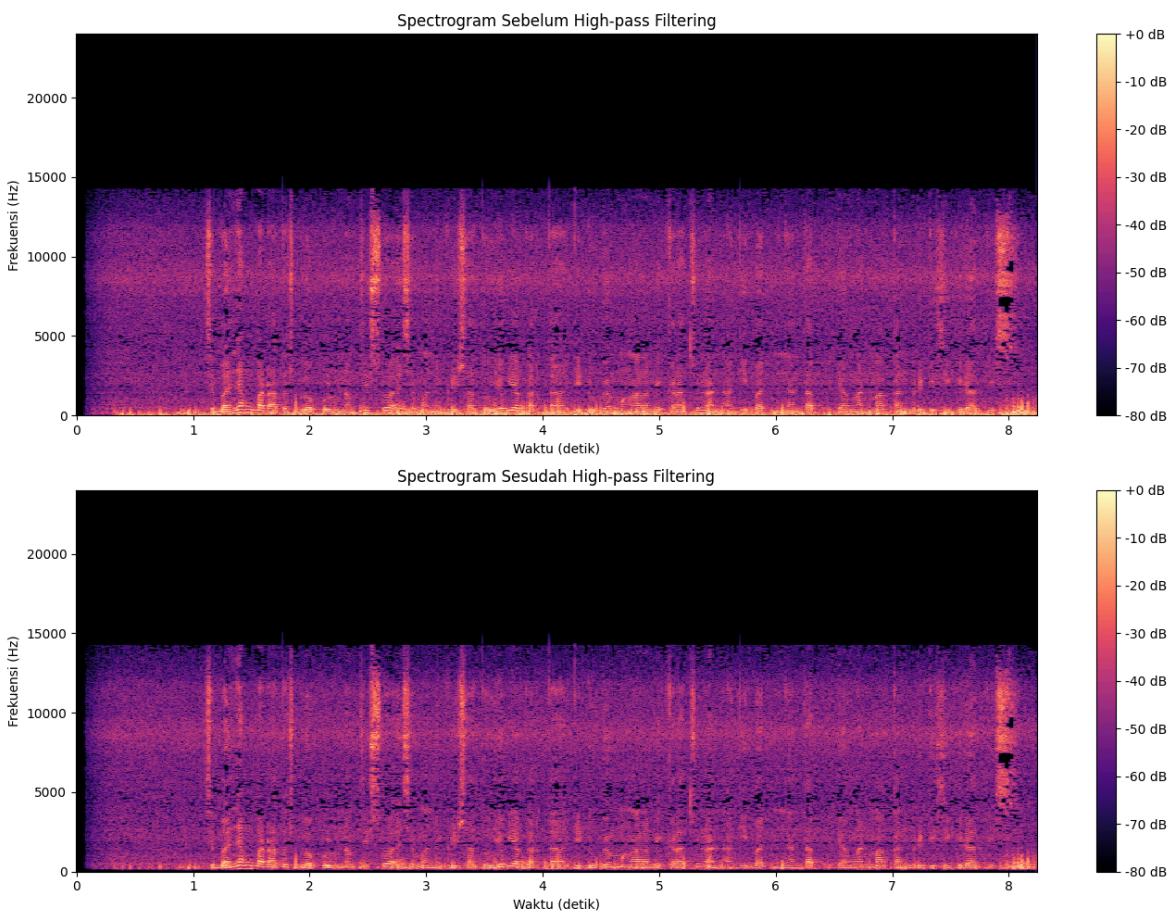
plt.tight_layout()
plt.show()

print("Audio Player Sebelum Filtering:")
display(Audio(y2, rate=sr2))

print("Audio Player Sesudah High-pass Filtering:")
display(Audio(y2_filtered, rate=sr2))

```

==== High-pass filtering selesai! ===
 Hasil disimpan di: c:\Users\Pongo\OneDrive\Desktop\Git-Upload\Hands-on_Multimedia\Hands-on Pemrosesan Audio\output\Soal2_HighPass.wav



Audio Player Sebelum Filtering:



Audio Player Sesudah High-pass Filtering:



Band-pass Filtering

```
In [10]: # Fungsi band-pass filter
def butter_bandpass(lowcut, highcut, sr, order=4):
    nyq = 0.5 * sr # frekuensi Nyquist
    low = lowcut / nyq
    high = highcut / nyq
    b, a = butter(order, [low, high], btype='band', analog=False)
    return b, a

# Fungsi untuk menerapkan filter
def apply_bandpass(y, sr, lowcut=100, highcut=8000, order=4):
    b, a = butter_bandpass(lowcut, highcut, sr, order)
    y_filtered = filtfilt(b, a, y)
    return y_filtered

# Terapkan filter
y2_filtered = apply_bandpass(y2, sr2, lowcut=100, highcut=7500)

# Simpan hasil ke folder output
output_dir = os.path.join(os.getcwd(), 'output')
os.makedirs(output_dir, exist_ok=True)
```

```

output_path = os.path.join(output_dir, 'Soal2_BandPass.wav')

sf.write(output_path, y2_filtered, sr2)

print("\n== Band-pass filtering selesai! ===")
print(f"Hasil disimpan di: {output_path}")

# Spectrogram sebelum dan sesudah filtering
D2_before = librosa.amplitude_to_db(np.abs(librosa.stft(y2)), ref=np.max)
D2_after = librosa.amplitude_to_db(np.abs(librosa.stft(y2_filtered)), ref=np.max)

plt.figure(figsize=(14, 10))

plt.subplot(2, 1, 1)
librosa.display.specshow(D2_before, sr=sr2, x_axis='time', y_axis='hz')
plt.colorbar(format='%.2f dB')
plt.title('Spectrogram Sebelum Band-pass Filtering')
plt.xlabel('Waktu (detik)')
plt.ylabel('Frekuensi (Hz)')

plt.subplot(2, 1, 2)
librosa.display.specshow(D2_after, sr=sr2, x_axis='time', y_axis='hz')
plt.colorbar(format='%.2f dB')
plt.title('Spectrogram Sesudah Band-pass Filtering')
plt.xlabel('Waktu (detik)')
plt.ylabel('Frekuensi (Hz)')

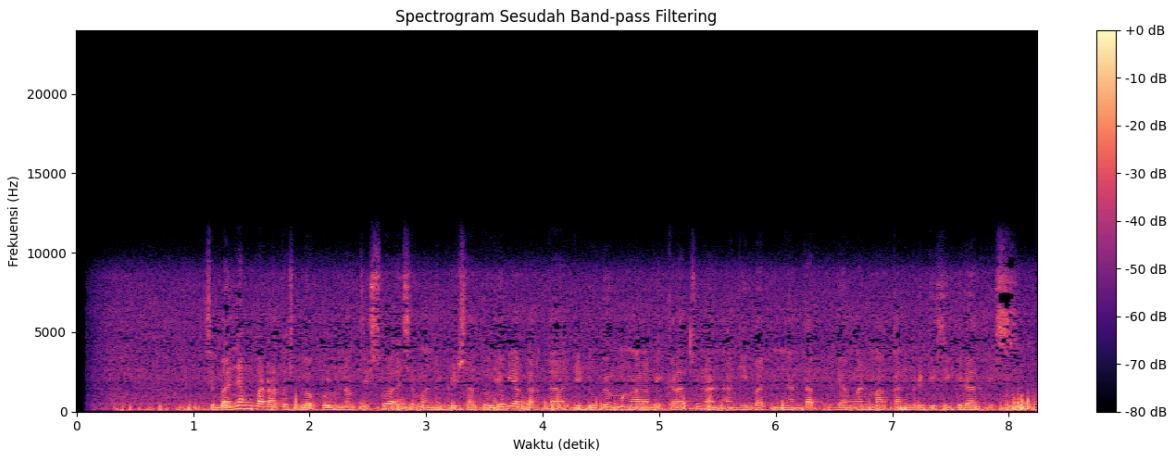
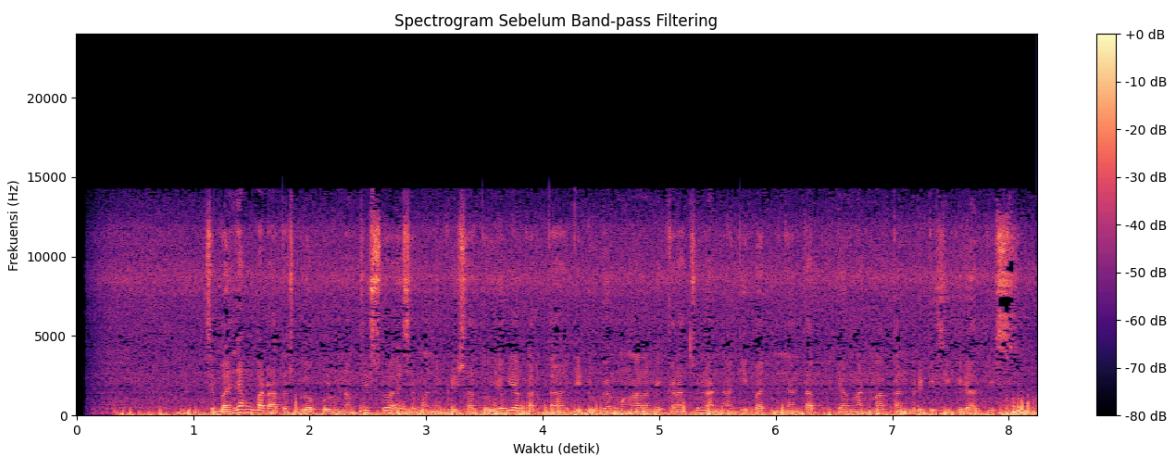
plt.tight_layout()
plt.show()

print("Audio Player Sebelum Filtering:")
display(Audio(y2, rate=sr2))

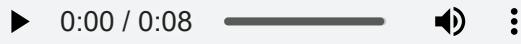
print("Audio Player Sesudah Band-pass Filtering:")
display(Audio(y2_filtered, rate=sr2))

```

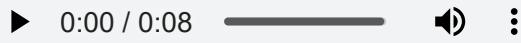
=== Band-pass filtering selesai! ===
 Hasil disimpan di: c:\Users\Pongo\OneDrive\Desktop\Git-Upload\Hands-on_Multimedia\Hands-on Pemrosesan Audio\output\Soal2_BandPass.wav



Audio Player Sebelum Filtering:



Audio Player Sesudah Band-pass Filtering:



Penjelasan:

Audio sebelum dilakukan filtering (low-pass, high-pass, band-pass) memiliki banyak sebaran frekuensi tinggi dari rentang 0 sampai 15000 Hz. Sebaran tersebut memiliki intensitas beragam mulai dari 40dB sampai 60dB. Suara noise yang dihasilkan berupa suara dengan frekuensi tinggi (Hisssss) yaitu suara kipas laptop dan kipas angin yang menyebabkan seperti ada suara nafas di dalam audio.

Perbandingan setelah dilakukan filtering:

1. Low-pass: Mengurangi frekuensi tinggi di atas 6500 Hz, sehingga suara noise (Hisssss) berkurang signifikan. Suara kipas laptop dan kipas angin menjadi tidak terdengar.
2. High-pass: Mengurangi frekuensi rendah di bawah 200 Hz, sehingga sedikit mengurangi suara nafas/angin di dalam audio. Namun, suara kipas tetap terdengar.

3. Band-pass: Mempertahankan frekuensi antara 100 Hz dan 7500 Hz, sehingga suara nafas dan suara kipas terdengar seimbang. Noise frekuensi tinggi di atas 7500 Hz berkurang, tetapi suara kipas masih sedikit terdengar.

Jelaskan:

- Jenis noise yang muncul pada rekaman Anda: **Suara Frekuensi Tinggi (Hisssss dari kipas laptop dan kipas angin)**
- Filter mana yang paling efektif untuk mengurangi noise tersebut: **Band-pass Filter**
- Nilai cutoff yang memberikan hasil terbaik: **100 Hz - 7500 Hz**
- Bagaimana kualitas suara (kejelasan ucapan) setelah proses filtering: **Kualitas suara menjadi lebih jelas, meskipun masih tetap ada sedikit noise**

Soal 3: Pitch Shifting dan Audio Manipulation

```
In [11]: Path_Soal_Audio3 = os.path.join(os.getcwd(), 'data', 'Soal1_Audio.wav')

if os.path.exists(Path_Soal_Audio3):
    y3, sr3 = librosa.load(Path_Soal_Audio3, sr=None)
```

```
In [12]: # Pitch shifting +7 dan +12 semitone
y_chipmunk_7 = librosa.effects.pitch_shift(y3, sr=sr3, n_steps=7)
y_chipmunk_12 = librosa.effects.pitch_shift(y3, sr=sr3, n_steps=12)

# Simpan hasil ke folder output
output_dir = os.path.join(os.getcwd(), 'output')
os.makedirs(output_dir, exist_ok=True)

output_path_7 = os.path.join(output_dir, 'Soal3_PitchShift_+7.wav')
output_path_12 = os.path.join(output_dir, 'Soal3_PitchShift_+12.wav')

sf.write(output_path_7, y_chipmunk_7, sr3)
sf.write(output_path_12, y_chipmunk_12, sr3)

print("\n==== Pitch shifting selesai! ===")
print(f"Hasil +7 semitone disimpan di: {output_path_7}")
print(f"Hasil +12 semitone disimpan di: {output_path_12}")
```

```
# Perbandingan waveform asli dan pitch-shifted

# Asli
plt.figure(figsize=(14, 10))
plt.subplot(3, 1, 1)
plt.plot(time, y3, color='orange')
plt.xlabel('Waktu (detik)')
plt.ylabel('Amplitudo')
plt.title('Waveform Asli')

# +7 semitone
plt.subplot(3, 1, 2)
plt.plot(time, y_chipmunk_7, color='green')
plt.xlabel('Waktu (detik)')
plt.ylabel('Amplitudo')
```

```

plt.title('Waveform +7 Semitone')

# +12 semitone
plt.subplot(3, 1, 3)
plt.plot(time, y_chipmunk_12, color='blue')
plt.xlabel('Waktu (detik)')
plt.ylabel('Amplitudo')
plt.title('Waveform +12 Semitone')

plt.tight_layout()
plt.show()

# Perbandingan Spectrogram asli dan pitch-shifted

# Asli
D3_asli = librosa.amplitude_to_db(np.abs(librosa.stft(y3)), ref=np.max)
D3_7 = librosa.amplitude_to_db(np.abs(librosa.stft(y_chipmunk_7)), ref=np.max)
D3_12 = librosa.amplitude_to_db(np.abs(librosa.stft(y_chipmunk_12)), ref=np.max)

plt.figure(figsize=(14, 10))

plt.subplot(3, 1, 1)
librosa.display.specshow(D3_asli, sr=sr3, x_axis='time', y_axis='hz')
plt.colorbar(format='%+2.0f dB')
plt.title('Spectrogram Asli')
plt.xlabel('Waktu (detik)')
plt.ylabel('Frekuensi (Hz)')

plt.subplot(3, 1, 2)
librosa.display.specshow(D3_7, sr=sr3, x_axis='time', y_axis='hz')
plt.colorbar(format='%+2.0f dB')
plt.title('Spectrogram +7 Semitone')
plt.xlabel('Waktu (detik)')
plt.ylabel('Frekuensi (Hz)')

plt.subplot(3, 1, 3)
librosa.display.specshow(D3_12, sr=sr3, x_axis='time', y_axis='hz')
plt.colorbar(format='%+2.0f dB')
plt.title('Spectrogram +12 Semitone')
plt.xlabel('Waktu (detik)')
plt.ylabel('Frekuensi (Hz)')

plt.tight_layout()
plt.show()

# Perbandingan Audio Player
print("Audio Player Asli:")
display(Audio(y3, rate=sr3))

print("Audio Player +7 Semitone:")
display(Audio(y_chipmunk_7, rate=sr3))

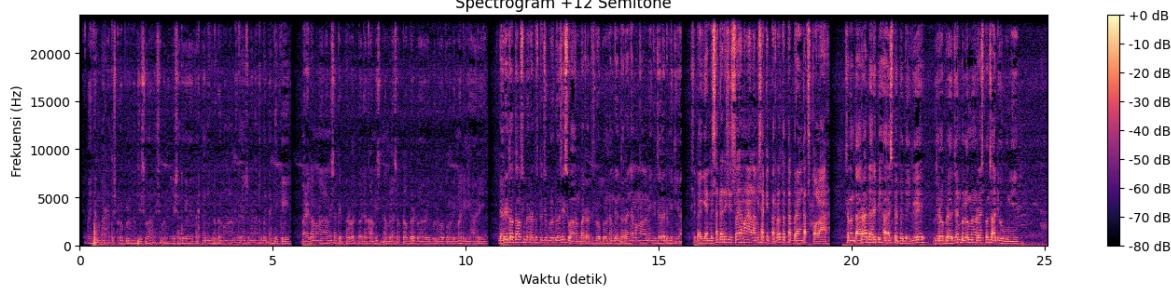
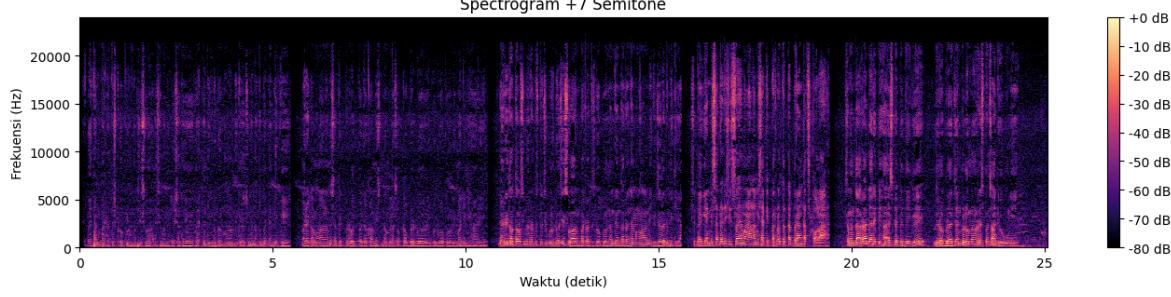
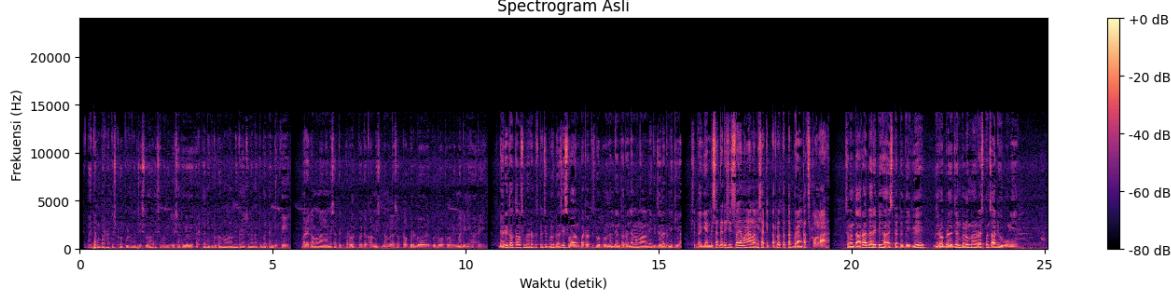
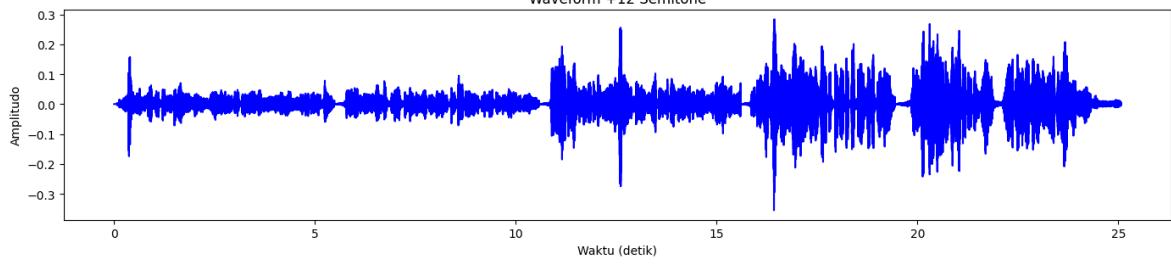
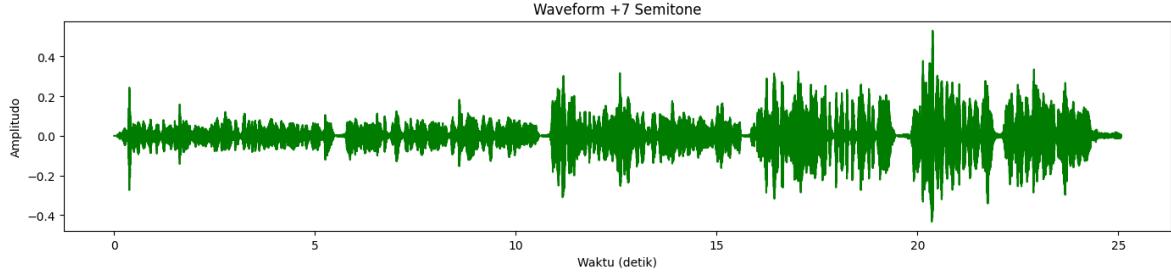
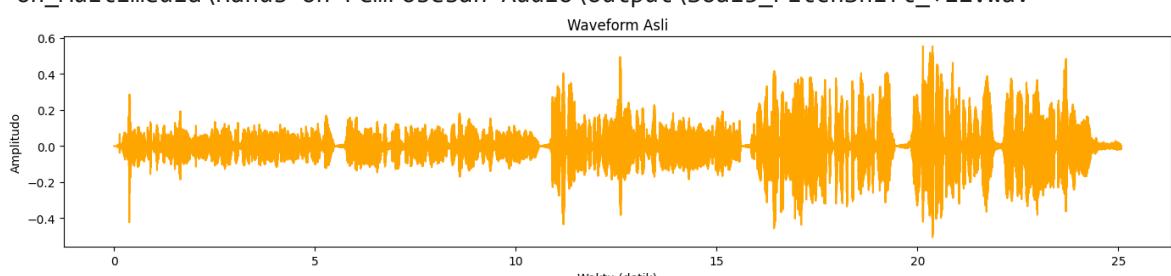
print("Audio Player +12 Semitone:")
display(Audio(y_chipmunk_12, rate=sr3))

```

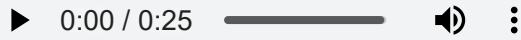
== Pitch shifting selesai! ==

Hasil +7 semitone disimpan di: c:\Users\Pongo\OneDrive\Desktop\Git-Upload\Hands-on_Multimedia\Hands-on Pemrosesan Audio\output\Soal3_PitchShift_+7.wav

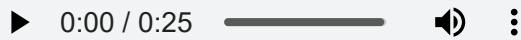
Hasil +12 semitone disimpan di: c:\Users\Pongo\OneDrive\Desktop\Git-Upload\Hands-on_Multimedia\Hands-on Pemrosesan Audio\output\Soal3_PitchShift_+12.wav



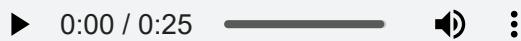
Audio Player Asli:



Audio Player +7 Semitone:



Audio Player +12 Semitone:



Penjelasan:

1. Parameter yang digunakan:

`n_steps` = jumlah semitone (nada) yang digeser.

Jumlah semitone yang akan mempengaruhi pitch suara, jadi semakin besar nilai `n_steps`, maka pitch suara akan semakin tinggi (jika positif) atau semakin rendah (jika negatif).

- Semakin tinggi = semakin cempreng suara (mendekati chipmunk)
- Semakin rendah = semakin berat suara (mendekati suara raksasa atau monster)

`librosa.effects.pitch_shift()`

Library dari librosa yang akan otomatis menjaga durasi tetap sama meskipun pitch diubah.

2. Perbedaan dalam representasi visual antara suara asli dan suara yang telah dimodifikasi:

Untuk informasi yang di dapat dari waveform, terlihat bahwa nilai amplitudonya berkurang atau semakin kecil seiring dengan meningkatnya pitch.

sedangkan untuk informasi yang di dapat dari spectrogram, terlihat bahwa frekuensi pada audio yang telah dimodifikasi (pitch dinaikkan) menjadi lebih rapat atau lebih tinggi dibandingkan dengan audio asli. Hal ini karena pitch yang dinaikkan menyebabkan frekuensi suara menjadi lebih tinggi.

- Frekuensi audio asli: 0 - 15000 Hz
- Frekuensi audio pitch dinaikkan +7: 0 - 20000 Hz
- Frekuensi audio pitch dinaikkan +12: > 20000 Hz

3. Bagaimana perubahan pitch memengaruhi kualitas dan kejelasan suara:

Semakin tinggi pitch, maka suara akan terdengar semakin cempreng dan kurang jelas. Suara terdengar seolah-olah lebih cepat meskipun durasi tetap sama.

Gabungan 2 Pitch Shifting +7 dan +12

```
In [13]: # Samakan panjang kedua sinyal (ambil yang terpendek agar aman)
min_len = min(len(y_chipmunk_7), len(y_chipmunk_12))
y7_aligned = y_chipmunk_7[:min_len]
y12_aligned = y_chipmunk_12[:min_len]

# Gabungkan secara overlay (campur dua sinyal)
combined_overlay = y7_aligned * 0.5 + y12_aligned * 0.5 # normalisasi biar gak

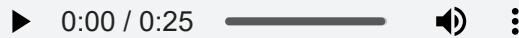
# Simpan hasil overlay
output_overlay_path = os.path.join(output_dir, 'Soal3_PitchShift_Combined_Overlay.wav')
sf.write(output_overlay_path, combined_overlay, sr3)

print(f"\n✓ Gabungan overlay disimpan di: {output_overlay_path}")

# Audio Player gabungan overlay
print("Audio Player Gabungan Overlay (+7 dan +12 Semitone):")
display(Audio(combined_overlay, rate=sr3))
```

✓ Gabungan overlay disimpan di: c:\Users\Pongo\OneDrive\Desktop\Git-Upload\Hands-on_Multimedia\Hands-on Pemrosesan Audio\output\Soal3_PitchShift_Combined_Overlay.wav

Audio Player Gabungan Overlay (+7 dan +12 Semitone):



Soal 4: Audio Processing Chain

```
In [14]: Path_Soal_Audio4 = os.path.join(os.getcwd(), 'output', 'Soal3_PitchShift_Combine.wav')

if os.path.exists(Path_Soal_Audio4):
    y4, sr4 = librosa.load(Path_Soal_Audio4, sr=None)
    print(f"File ditemukan: {Path_Soal_Audio4}")
else:
    print(f"File tidak ditemukan")
```

File ditemukan: c:\Users\Pongo\OneDrive\Desktop\Git-Upload\Hands-on_Multimedia\Hands-on Pemrosesan Audio\output\Soal3_PitchShift_Combined_Overlay.wav

Equalizer (High-pass 100 Hz, Low-pass 7500 Hz)

```
In [15]: # Fungsi filter
def butter_filter(y, sr, cutoff, btype='low', order=6):
    nyquist = 0.5 * sr
    normal_cutoff = cutoff / nyquist
    b, a = butter(order, normal_cutoff, btype=btype)
    return lfilter(b, a, y)

# Terapkan high-pass di 100 Hz dan low-pass di 6000 Hz
y_eq = butter_filter(y4, sr4, 100, btype='high')
y_eq = butter_filter(y_eq, sr4, 6000, btype='low')
print("✓ EQ applied (High-pass 100 Hz, Low-pass 6000 Hz)")

# Audio Player setelah EQ
print("Audio Player Setelah EQ:")
display(Audio(y_eq, rate=sr4))
```

EQ applied (High-pass 100 Hz, Low-pass 6000 Hz)
Audio Player Setelah EQ:

▶ 0:00 / 0:25 ━━ 🔊 ⋮

Fade-in/out 0.5 detik

```
In [16]: fade_len = int(0.5 * sr4) # 0.5 detik fade
fade_in = np.linspace(0, 1, fade_len)
fade_out = np.linspace(1, 0, fade_len)

y_fade = y_eq.copy()
y_fade[:fade_len] *= fade_in
y_fade[-fade_len:] *= fade_out
print("✓ Fade-in/out applied")

# Audio Player setelah fade
print("Audio Player Setelah Fade-in/out:")
display(Audio(y_fade, rate=sr4))
```

Fade-in/out applied
Audio Player Setelah Fade-in/out:

▶ 0:00 / 0:25 ━━ 🔊 ⋮

Normalisasi ke -0 dBFS

```
In [17]: y_norm = y_fade / np.max(np.abs(y_fade))
print("✓ Normalized to peak 0 dBFS")

# Audio Player setelah normalisasi
print("Audio Player Setelah Normalisasi:")
display(Audio(y_norm, rate=sr4))
```

Normalized to peak 0 dBFS
Audio Player Setelah Normalisasi:

▶ 0:00 / 0:25 ━━ 🔊 ⋮

Compression untuk menyeimbangkan dinamika audio

```
In [18]: def simple_compressor(y, threshold=0.1, ratio=4):
    y_compressed = np.copy(y)
    mask = np.abs(y) > threshold
    y_compressed[mask] = np.sign(y[mask]) * (threshold + (np.abs(y[mask]) - threshold) / ratio)
    return y_compressed

y_comp = simple_compressor(y_norm)
print("✓ Compression applied (Threshold=0.2, Ratio=4:1)")

# Audio Player setelah kompresi
print("Audio Player Setelah Kompresi:")
display(Audio(y_comp, rate=sr4))
```

Compression applied (Threshold=0.2, Ratio=4:1)
Audio Player Setelah Kompresi:

▶ 0:00 / 0:25 ━━━━ 🔊 ⋮

Noise Gate

```
In [19]: threshold = 0.001
y_gate = np.where(np.abs(y_comp) < threshold, 0, y_comp)
print(" Noise gate applied (threshold=0.001)")

# Audio Player setelah noise gate
print("Audio Player Setelah Noise Gate:")
display(Audio(y_gate, rate=sr4))
```

Noise gate applied (threshold=0.001)
Audio Player Setelah Noise Gate:

▶ 0:00 / 0:25 ━━━━ 🔊 ⋮

Silence Trimming

```
In [20]: y_trimmed, index = librosa.effects.trim(y_gate, top_db=25)
print(f" Trimmed silence (Start={index[0]/sr4:.2f}s, End={index[1]/sr4:.2f}s)

# Audio Player setelah trim
print("Audio Player Setelah Trim:")
display(Audio(y_trimmed, rate=sr4))
```

Trimmed silence (Start=0.23s, End=24.50s)
Audio Player Setelah Trim:

▶ 0:00 / 0:24 ━━━━ 🔊 ⋮

Normalisasi Loudness ke -16 LUFS

```
In [21]: meter = pyln.Meter(sr4)
loudness = meter.integrated_loudness(y_trimmed)
print(f"Loudness sebelum normalisasi: {loudness:.2f} LUFS")

y_loudnorm = pyln.normalize.loudness(y_trimmed, loudness, -16.0)
print(" Loudness distandarkan ke -16 LUFS")

# Audio Player setelah normalisasi Loudness
print("Audio Player Setelah Normalisasi Loudness:")
display(Audio(y_loudnorm, rate=sr4))
```

Loudness sebelum normalisasi: -25.49 LUFS
 Loudness distandarkan ke -16 LUFS
Audio Player Setelah Normalisasi Loudness:

▶ 0:00 / 0:24 ━━━━ 🔊 ⋮

Visualisasi Waveform dan Spectrogram sebelum dan sesudah pemrosesan

```
In [22]: # Visualisasi Waveform dan Spectrogram sebelum dan sesudah pemrosesan
def plot_waveform_and_spectrogram(y, sr, title_prefix=''):
    D4 = librosa.amplitude_to_db(np.abs(librosa.stft(y)), ref=np.max)

    plt.figure(figsize=(14, 5))
    plt.subplot(2, 1, 1)
    librosa.display.waveshow(y, sr=sr)
    plt.title(title_prefix + 'Waveform')
    plt.xlabel('Waktu (detik)')
    plt.ylabel('Amplitudo')

    plt.subplot(2, 1, 2)
    librosa.display.specshow(D4, sr=sr, x_axis='time', y_axis='hz')
    plt.colorbar(format='%+2.0f dB')
    plt.title(title_prefix + 'Spectrogram')
    plt.xlabel('Waktu (detik)')
    plt.ylabel('Frekuensi (Hz)')
    plt.tight_layout()
    plt.show()

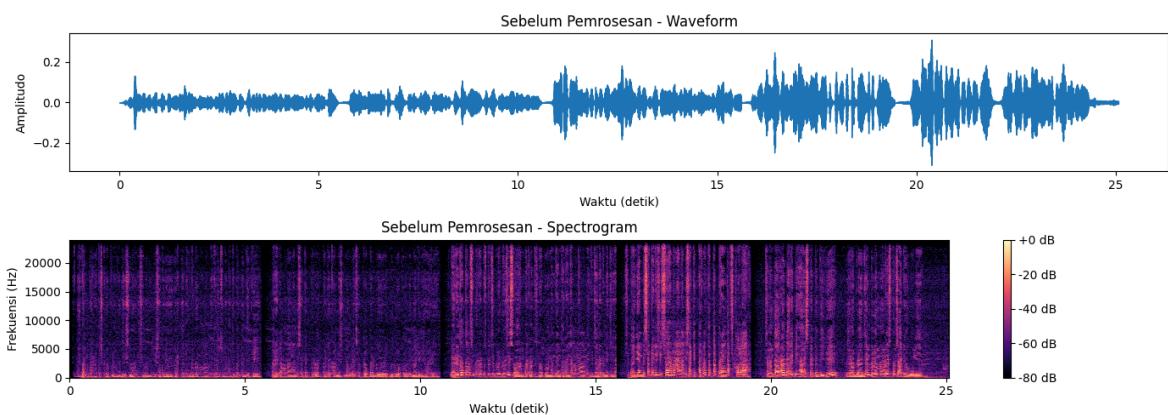
print("Sebelum Pemrosesan:")
plot_waveform_and_spectrogram(y4, sr4, title_prefix='Sebelum Pemrosesan - ')

print("Sesudah Pemrosesan:")
plot_waveform_and_spectrogram(y_loudnorm, sr4, title_prefix='Sesudah Pemrosesan')

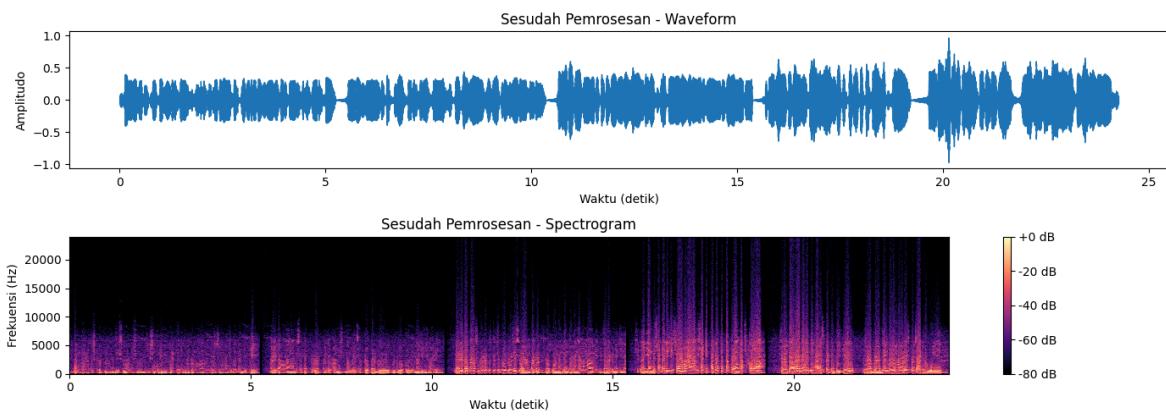
# Perbandingan Audio Player awal dan akhir
print("Audio Player Awal (Sebelum Pemrosesan):")
display(Audio(y4, rate=sr4))

print("Audio Player Akhir (Sesudah Pemrosesan):")
display(Audio(y_loudnorm, rate=sr4))
```

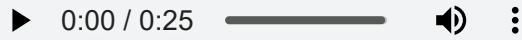
Sebelum Pemrosesan:



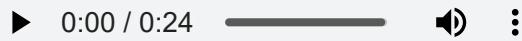
Sesudah Pemrosesan:



Audio Player Awal (Sebelum Pemrosesan):



Audio Player Akhir (Sesudah Pemrosesan):



Penjelasan:

1. Perubahan dinamika suara yang terjadi

- **Equalizer:** Menghilangkan noise di frekuensi tinggi (hissss) yang terdengar jelas setelah proses pitch shifting. Juga mengurangi frekuensi rendah yang tidak diinginkan. (High-pass 100 Hz, Low-pass 7500 Hz)
- **Fade-in/out:** Membuat transisi suara lebih halus di awal dan akhir audio.
- **Normalisasi ke -0 dBFS:** Meningkatkan volume keseluruhan audio tanpa distorsi, jadi untuk suara keras yang masih di bawah -0 dBFS akan dinaikkan ke -0 dBFS.
- **Compression:** Mengurangi perbedaan antara bagian paling keras dan paling pelan sehingga suara lebih konsisten.
- **Noise Gate:** Menghilangkan suara dengan amplitudo rendah (noise kresek-kresek seperti tv rusak).
- **Silence Trimming:** Menghilangkan bagian diam di awal dan akhir audio sehingga durasi menjadi lebih singkat.
- **Normalisasi Loudness ke -16 LUFS:** Menyesuaikan loudness audio ke standar -16 LUFS sehingga lebih nyaman didengar oleh telinga manusia.

2. Perbedaan antara normalisasi peak dan normalisasi LUFS

- **Normalisasi Peak (-0 dBFS):** Menyesuaikan puncak tertinggi sinyal audio ke nilai 0 dBFS, agar tidak melebihi batas maksimum digital. (tidak memperhatikan persepsi pendengaran manusia)
- **Normalisasi LUFS (-16 LUFS):** Menyesuaikan loudness rata-rata audio ke standar -16 LUFS, yang lebih memperhatikan persepsi pendengaran manusia terhadap loudness. (lebih nyaman didengar)

3. Bagaimana kualitas suara berubah setelah proses normalisasi dan loudness optimization

Suara terdengar lebih jernih, noise berkurang, volume lebih konsisten (tidak terlalu keras atau terlalu pelan), dan lebih nyaman didengar.

4. Kelebihan dan kekurangan dari pengoptimalan loudness dalam konteks rekaman suara

- **Kelebihan:**
 - Meningkatkan kenyamanan mendengarkan.
 - Mengurangi kelelahan pendengaran.
 - Memastikan konsistensi volume audio.
- **Kekurangan:**
 - Suara terdengar seperti monoton (kurang dinamis).

Soal 5: Music Analysis and Remix

```
In [23]: # Path Audio Happy
Path_Soal_Audio5_Happy = os.path.join(os.getcwd(), 'data', 'Happy_Music-cut.wav')

if os.path.exists(Path_Soal_Audio5_Happy):
    y5, sr5 = librosa.load(Path_Soal_Audio5_Happy, sr=None)
    print(f"File ditemukan: {Path_Soal_Audio5_Happy}")
else:
    print(f"File tidak ditemukan")

# Deteksi tempo dari audio Happy
tempo_happy, beats_happy = librosa.beat.beat_track(y=y5, sr=sr5)
tempo_happy = tempo_happy[0] if isinstance(tempo_happy, np.ndarray) else tempo_happy

# Audio Player
print("Audio Player Happy Music:")
display(Audio(y5, rate=sr5))

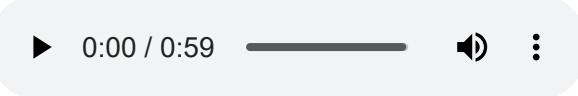
# Path Audio Sad
Path_Soal_Audio5_Sad = os.path.join(os.getcwd(), 'data', 'Sad_Music-cut.wav')

if os.path.exists(Path_Soal_Audio5_Sad):
    y6, sr6 = librosa.load(Path_Soal_Audio5_Sad, sr=None)
    print(f"File ditemukan: {Path_Soal_Audio5_Sad}")
else:
    print(f"File tidak ditemukan")

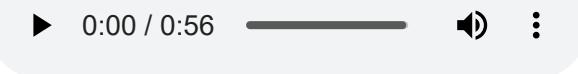
# Deteksi tempo dari audio Sad
tempo_sad, beats_sad = librosa.beat.beat_track(y=y6, sr=sr6)
tempo_sad = tempo_sad[0] if isinstance(tempo_sad, np.ndarray) else tempo_sad

# Audio Player
print("Audio Player Sad Music:")
display(Audio(y6, rate=sr6))
```

File ditemukan: c:\Users\Pongo\OneDrive\Desktop\Git-Upload\Hands-on_Multimedia\Hands-on Pemrosesan Audio\data\Happy_Music-cut.wav
Audio Player Happy Music:



File ditemukan: c:\Users\Pongo\OneDrive\Desktop\Git-Upload\Hands-on_Multimedia\Hands-on Pemrosesan Audio\data\Sad_Music-cut.wav
Audio Player Sad Music:



Deteksi Tempo (BPM):

```
In [24]: # Tampilkan hasil deteksi tempo Happy
print(f"HASIL DETEKSI TEMPO HAPPY MUSIC:")
print(f"• BPM Terdeteksi: {tempo_happy:.1f} BPM")
print(f"• Jumlah beat: {len(beats_happy)} beats")
print(f"• Durasi audio: {len(y5)/sr5:.2f} detik")

# Konversi beat ke waktu (detik)
beat_times_happy = librosa.frames_to_time(beats_happy, sr=sr5)

# Visualisasi tempo detection
fig, ax = plt.subplots(figsize=(15, 6))

ax.plot(np.linspace(0, len(y5)/sr5, len(y5)), y5, alpha=0.6, color='blue', linewidth=1)
ax.vlines(beat_times_happy, -1, 1, color='red', alpha=0.8, linewidth=2, label=f'Beat')
ax.set_title(f'Audio Waveform dengan Beat Detection (BPM: {tempo_happy:.1f})', fontweight='bold')
ax.set_xlabel('Waktu (detik)')
ax.set_ylabel('Amplitudo')
ax.legend()
ax.grid(True, alpha=0.3)
plt.tight_layout()
plt.show()

# Tampilkan hasil deteksi tempo Sad
print(f"HASIL DETEKSI TEMPO SAD MUSIC:")
print(f"• BPM Terdeteksi: {tempo_sad:.1f} BPM")
print(f"• Jumlah beat: {len(beats_sad)} beats")
print(f"• Durasi audio: {len(y6)/sr6:.2f} detik")

# Konversi beat ke waktu (detik)
beat_times_sad = librosa.frames_to_time(beats_sad, sr=sr6)

# Visualisasi tempo detection
fig, ax = plt.subplots(figsize=(15, 6))

ax.plot(np.linspace(0, len(y6)/sr6, len(y6)), y6, alpha=0.6, color='blue', linewidth=1)
ax.vlines(beat_times_sad, -1, 1, color='red', alpha=0.8, linewidth=2, label=f'Beat')
ax.set_title(f'Audio Waveform dengan Beat Detection (BPM: {tempo_sad:.1f})', fontweight='bold')
ax.set_xlabel('Waktu (detik)')
ax.set_ylabel('Amplitudo')
ax.legend()
ax.grid(True, alpha=0.3)
plt.tight_layout()
plt.show()
```

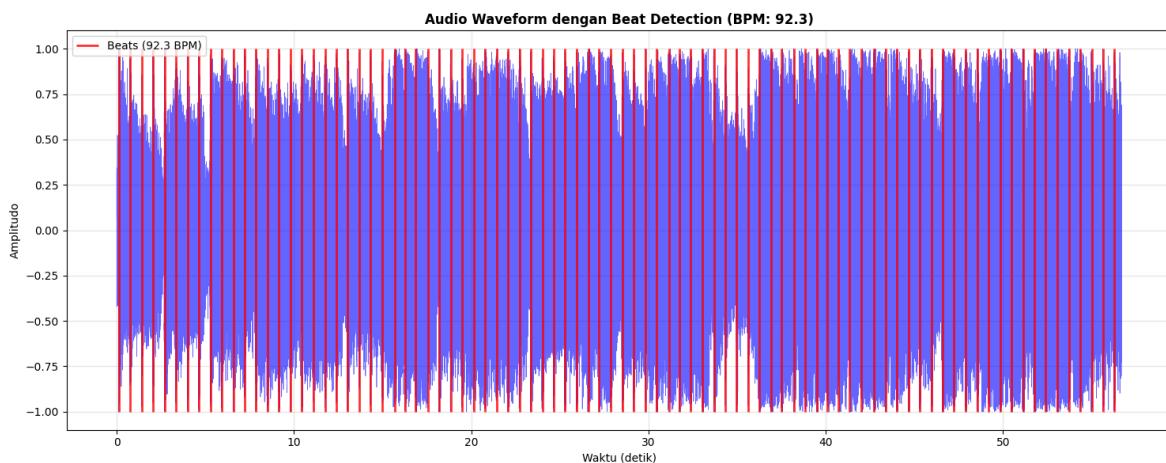
HASIL DETEKSI TEMPO HAPPY MUSIC:

- BPM Terdeteksi: 143.6 BPM
- Jumlah beat: 130 beats
- Durasi audio: 59.20 detik



HASIL DETEKSI TEMPO SAD MUSIC:

- BPM Terdeteksi: 92.3 BPM
- Jumlah beat: 88 beats
- Durasi audio: 56.70 detik



Estimasi Kunci (Key):

```
In [25]: # Extract chroma features Happy
chroma_happy = librosa.feature.chroma_stft(y=y5, sr=sr5)
chroma_happy_mean = np.mean(chroma_happy, axis=1)

# Extract chroma features Sad
chroma_sad = librosa.feature.chroma_stft(y=y6, sr=sr6)
chroma_sad_mean = np.mean(chroma_sad, axis=1)

# Key templates (Krumhansl-Schmuckler profiles)
major_template = np.array([6.35, 2.23, 3.48, 2.33, 4.38, 4.09,
                           2.52, 5.19, 2.39, 3.66, 2.29, 2.88])
minor_template = np.array([6.33, 2.68, 3.52, 5.38, 2.60, 3.53,
                           2.54, 4.75, 3.98, 2.69, 3.34, 3.17])

# Normalize templates
major_template /= np.sum(major_template)
minor_template /= np.sum(minor_template)

# Key names
keys = ['C', 'C#', 'D', 'D#', 'E', 'F', 'F#', 'G', 'G#', 'A', 'A#', 'B']
```

```

def detect_key(chroma_mean):
    major_scores = []
    minor_scores = []

    for i in range(12):
        major_rotated = np.roll(major_template, i)
        minor_rotated = np.roll(minor_template, i)

        major_corr = np.corrcoef(chroma_mean, major_rotated)[0, 1]
        minor_corr = np.corrcoef(chroma_mean, minor_rotated)[0, 1]

        major_scores.append(major_corr)
        minor_scores.append(minor_corr)

    best_major = np.argmax(major_scores)
    best_minor = np.argmax(minor_scores)

    if major_scores[best_major] > minor_scores[best_minor]:
        return f"{keys[best_major]} Major", major_scores, minor_scores
    else:
        return f"{keys[best_minor]} Minor", major_scores, minor_scores

# Deteksi key untuk masing-masing Lagu
key_happy, major_scores_happy, minor_scores_happy = detect_key(chroma_happy_mean)
key_sad, major_scores_sad, minor_scores_sad = detect_key(chroma_sad_mean)

# Tampilkan hasil deteksi kunci
print("HASIL DETEKSI KUNCI LAGU")
print(f"Happy Music Key : {key_happy}")
print(f"Sad Music Key : {key_sad}")
print(f"Tempo Happy : {tempo_happy:.2f} BPM")
print(f"Tempo Sad : {tempo_sad:.2f} BPM")

```

HASIL DETEKSI KUNCI LAGU
 Happy Music Key : A# Major
 Sad Music Key : F Minor
 Tempo Happy : 143.55 BPM
 Tempo Sad : 92.29 BPM

Penjelasan:

Analisis singkat: Lagu dengan vibes happy memiliki tempo yang cepat dan Beats per minute (BPM) yang lebih tinggi yaitu 143.55 dibandingkan lagu vibes sedih yang memiliki tempo lebih lambat dengan BPM 92.29.

Kunci lagu juga berpengaruh terhadap vibes lagu, dimana untuk kunci A# terdengar lebih ceria dan hidup, jika dibandingkan dengan kunci F Minor yang lebih santai dan tenang.

Time Stretch

```
In [26]: # Hitung rasio tempo (Sad → Happy)
rate_sad_to_happy = tempo_happy / tempo_sad

print("INFO TEMPO:")
print("=" * 40)
```

```

print(f"Tempo Happy : {tempo_happy:.2f} BPM")
print(f"Tempo Sad   : {tempo_sad:.2f} BPM")
print(f"Rasio (Sad → Happy): {rate_sad_to_happy:.3f}")

# Samakan tempo Sad agar setara dengan Happy
y6_stretched = librosa.effects.time_stretch(y6, rate=rate_sad_to_happy)

# Analisis durasi
dur_happy = len(y5) / sr5
dur_sad = len(y6) / sr6
dur_stretch = len(y6_stretched) / sr6

print("\nDURASI AUDIO:")
print("=" * 40)
print(f"Happy Music       : {dur_happy:.2f} s")
print(f"Sad Music (Original): {dur_sad:.2f} s")
print(f"Sad (Stretched)    : {dur_stretch:.2f} s ({dur_stretch/sr6:.2f}x dari durasi asli)")

# Audio previews
print("\nPERBANDINGAN AUDIO:")
print("Happy Music:")
display(Audio(y5, rate=sr5))

print("Sad Music (Original):")
display(Audio(y6, rate=sr6))

print(f"Sad Music (Stretched agar tempo ≈ Happy, rate={rate_sad_to_happy:.3f}):")
display(Audio(y6_stretched, rate=sr6))

```

INFO TEMPO:

Tempo Happy : 143.55 BPM

Tempo Sad : 92.29 BPM

Rasio (Sad → Happy): 1.556

DURASI AUDIO:

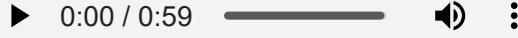
Happy Music : 59.20 s

Sad Music (Original): 56.70 s

Sad (Stretched) : 36.45 s (0.64x durasi Sad asli)

PERBANDINGAN AUDIO:

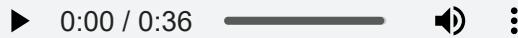
Happy Music:



Sad Music (Original):



Sad Music (Stretched agar tempo ≈ Happy, rate=1.556):



Pitch Shift

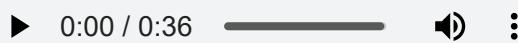
```
In [27]: # Pitch shifting Sad naik 5 semitone
y6_shifted = librosa.effects.pitch_shift(y6_stretched, sr=sr6, n_steps=5)

# Audio Player setelah pitch shifting
print("Sad Music (Setelah Pitch Shifting +5 Semitone):")
display(Audio(y6_shifted, rate=sr6))

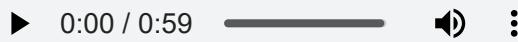
# Happy tidak shift
y5_shifted = y5

# Audio Player Happy
print("Happy Music (Tanpa Pitch Shifting):")
display(Audio(y5_shifted, rate=sr5))
```

Sad Music (Setelah Pitch Shifting +5 Semitone):



Happy Music (Tanpa Pitch Shifting):



Crossfading

```
In [28]: # Gabungan crossfade 2 lagu (Sad + Happy)

def crossfade(y1, y2, sr, fade_duration=5.0):
    fade_len = int(fade_duration * sr)

    fade_in = np.linspace(0, 1, fade_len)
    fade_out = np.linspace(1, 0, fade_len)

    y1_fade = y1[-fade_len:] * fade_out
    y2_fade = y2[:fade_len] * fade_in

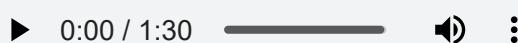
    y_crossfaded = np.concatenate((y1[:-fade_len], y1_fade + y2_fade, y2[fade_len:]))

    return y_crossfaded

# Gabungkan kedua Lagu dengan crossfade
y_combined = crossfade(y6_shifted, y5_shifted, sr6, fade_duration=5.0)

# Audio Player Gabungan
print("Gabungan Lagu (Sad + Happy):")
display(Audio(y_combined, rate=sr6))
```

Gabungan Lagu (Sad + Happy):



Filter Tambahan (Opsional)

```
In [29]: # Tambahan Effect

# BONUS: Track audio dengan tambahan suara "click" di setiap beat detected
```

```

tempo, beats = librosa.beat.beat_track(y=y_combined, sr=sr6)
beat_times = librosa.frames_to_time(beats, sr=sr6)

# Create click track for detected beats
click_duration = 0.5
click_samples = int(click_duration * sr6)
click_freq = 90

# Generate a simple click sound (sine wave with exponential decay)
t_click = np.linspace(0, click_duration, click_samples)
click_sound = np.sin(2 * np.pi * click_freq * t_click) * np.exp(-t_click * 10)
y_with_clicks = y_combined.copy()

for beat_time in beat_times:
    beat_sample = int(beat_time * sr6)
    # Ensure we don't go beyond array bounds
    if beat_sample + click_samples < len(y_with_clicks):
        y_with_clicks[beat_sample:beat_sample + click_samples] += click_sound *

print("Audio with click track:")
display(Audio(y_with_clicks, rate=sr6))

# Low-pass filter di 15000 Hz
y_final = butter_filter(y_with_clicks, sr6, 15000, btype='low')
print("✓ Final low-pass filter applied (15000 Hz)")

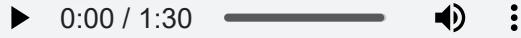
# Normalisasi ke -1 dBFS
y_final = y_final / np.max(np.abs(y_final)) * 0.8913 # -1 dBFS
print("✓ Final normalization to -1 dBFS applied")

# Fade in/out 0.5 detik
fade_duration = int(0.5 * sr6)
y_final[:fade_duration] *= np.linspace(0, 1, fade_duration)
y_final[-fade_duration:] *= np.linspace(1, 0, fade_duration)
print("✓ Final fade-in/out applied (0.5 detik)")

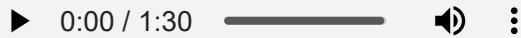
# Audio Player Final
print("Final Audio Player (Setelah Semua Proses):")
display(Audio(y_final, rate=sr6))

```

Audio with click track:



- Final low-pass filter applied (15000 Hz)
 - Final normalization to -1 dBFS applied
 - Final fade-in/out applied (0.5 detik)
- Final Audio Player (Setelah Semua Proses):



Penjelasan:

1. Perubahan durasi audio setelah time-stretching:

Durasi dari Sad Music berubah menjadi lebih pendek setelah dilakukan time-stretching untuk menyesuaikan agar tempo yang awalnya lambat menjadi lebih cepat dan berirama dengan vibes Happy Music. Dengan Rasio tempo yang didapat dari Tempo Happy (143.55 BPM) dibagi Tempo Sad (92.29 BPM) yaitu sekitar 1.556, maka durasi audio Sad yang awalnya 56 detik menjadi sekitar 36 detik setelah di-stretching.

2. Perubahan pitch audio setelah pitch-shifting:

Berdasarkan representasi not music dalam semitone, Lagu Happy yang berada di kunci A# memiliki nilai 10 semitone, sedangkan Lagu Sad yang berada di kunci F Minor memiliki nilai 5 semitone.

Untuk menyamakan kunci antara kedua lagu, maka perlu dilakukan pitch-shifting pada audio Sad dengan menaikkan pitch sebanyak 5 semitone yang menyebabkan frekuensi audio jadi lebih tinggi, sehingga vibes yang dihasilkan menjadi lebih ceria dari pada versi aslinya.

3. Efek crossfading pada transisi antara dua lagu:

Crossfading membuat transisi antara lagu "Happy" dan "Sad" menjadi lebih halus. Bagian akhir lagu "Happy" secara bertahap memudar (fade out) sementara bagian awal lagu "Sad" secara bertahap muncul (fade in). Dengan memberikan efek 0.5 detik pada awal dan akhir lagu, maka transisi antar lagu tidak terasa tiba-tiba atau mengganggu pendengaran.

4. Filter tambahan dari Bonus Materi 6:

Menambahkan efek yang awalnya clicky dengan frekuensi tinggi di 1000 Hz menjadi, saya ubah menjadi bass di frekuensi 90 Hz untuk setiap ketukan beat. Sehingga menambahkan kesan lebih seru di dalam lagu remix yang dihasilkan.

saya juga melakukan low-pass filter pada frekuensi 15000 Hz karena setelah saya tampilkan spetrogram diakhir, terlihat ada noise di bagian frekuensi tinggi diatas 15000 Hz.

dan juga saya melakukan normalisasi ke -1 dBFS agar volume audio tidak terlalu keras dan tidak terjadi clipping.

saya juga menambahkan efek fade-in di awal dan fade-out di akhir lagu agar transisi audio lebih halus.

```
In [30]: # Tampilan waveform sebelum remix (Sad Music Original)
plt.figure(figsize=(14, 3))
plt.plot(np.linspace(0, len(y6)/sr6, len(y6)), y6, color='orange', linewidth=0.5
plt.title('Waveform Audio Sebelum Remix (Sad Music Original)')
plt.xlabel('Time (s)')
plt.ylabel('Amplitudo')

# Tampilan waveform sebelum remix (Happy Music Original)
```

```

plt.figure(figsize=(14, 3))
plt.plot(np.linspace(0, len(y5)/sr5, len(y5)), y5, color='green', linewidth=0.5)
plt.title('Waveform Audio Sebelum Remix (Happy Music Original)')
plt.xlabel('Time (s)')
plt.ylabel('Amplitudo')

# Tampilan waveform sesudah remix final
plt.figure(figsize=(14, 3))
plt.plot(np.linspace(0, len(y_final)/sr6, len(y_final)), y_final, color='blue',
plt.title('Waveform Audio Sesudah Remix (Final)')
plt.xlabel('Time (s)')
plt.ylabel('Amplitudo')

plt.tight_layout()
plt.show()

D_sad = librosa.amplitude_to_db(np.abs(librosa.stft(y6)), ref=np.max)
D_happy = librosa.amplitude_to_db(np.abs(librosa.stft(y5)), ref=np.max)
D_final = librosa.amplitude_to_db(np.abs(librosa.stft(y_final)), ref=np.max)

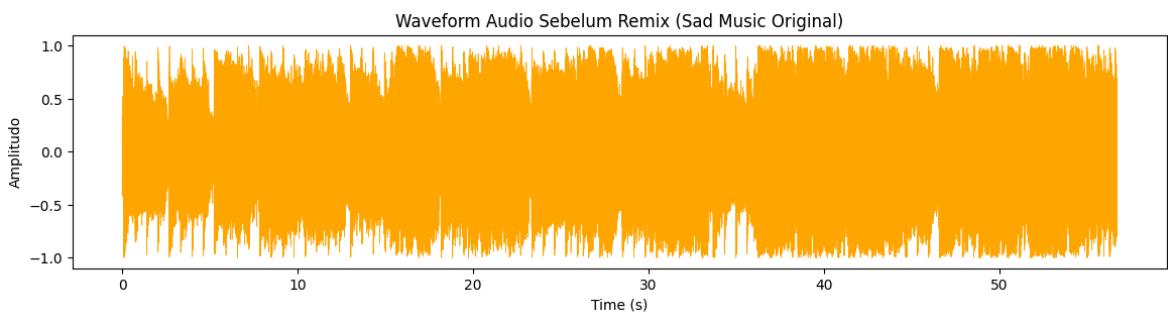
# Tampilan spectrogram sebelum remix (Sad Music Original)
plt.figure(figsize=(14, 3))
librosa.display.specshow(D_sad, sr=sr6, x_axis='time', y_axis='linear')
plt.colorbar(format='%.2f dB')
plt.title('Spectrogram Audio Sebelum Remix (Sad Music Original)')
plt.xlabel('Time (s)')
plt.ylabel('Frequency (Hz)')

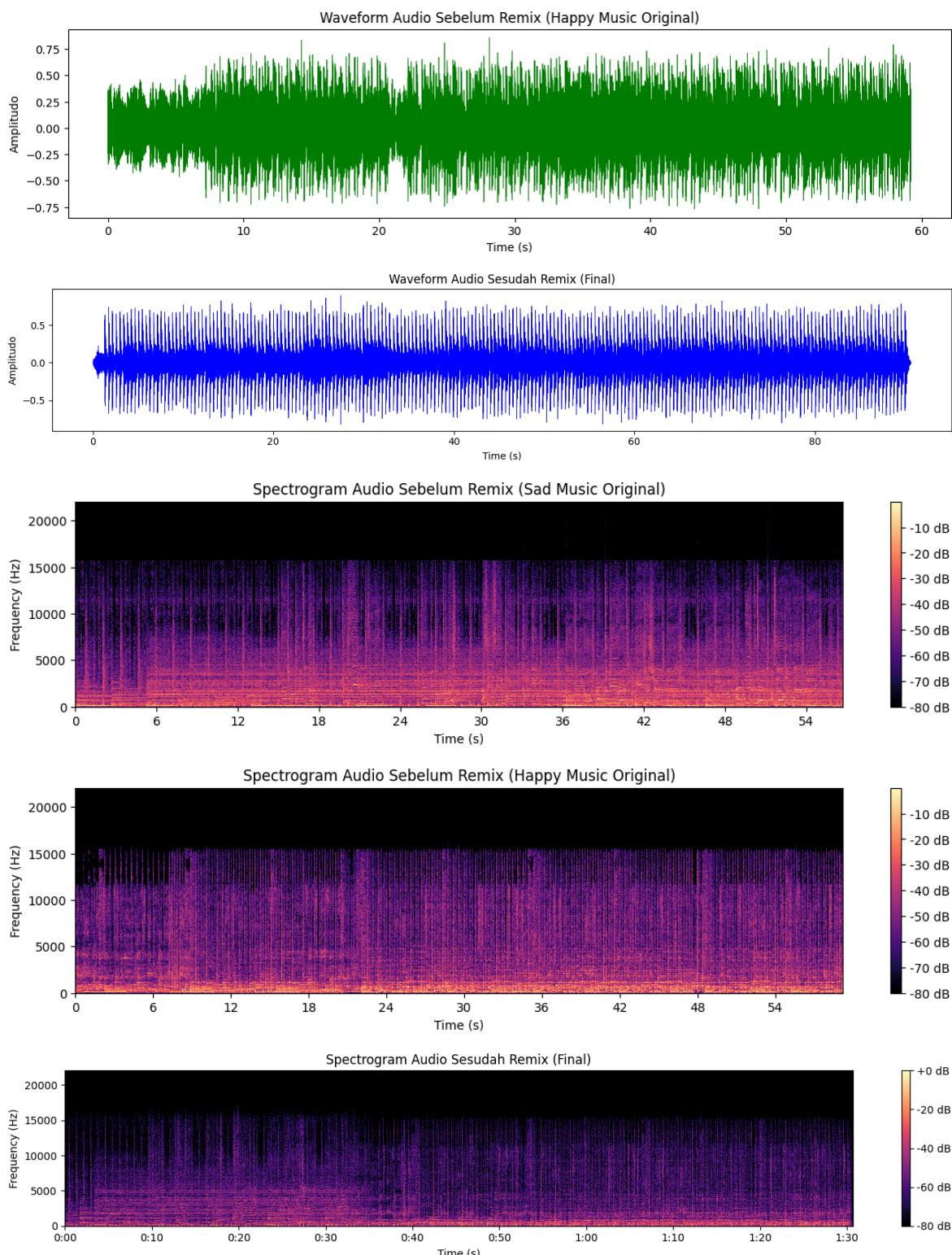
# Tampilan spectrogram sebelum remix (Happy Music Original)
plt.figure(figsize=(14, 3))
librosa.display.specshow(D_happy, sr=sr5, x_axis='time', y_axis='linear')
plt.colorbar(format='%.2f dB')
plt.title('Spectrogram Audio Sebelum Remix (Happy Music Original)')
plt.xlabel('Time (s)')
plt.ylabel('Frequency (Hz)')

# Tampilan spectrogram sesudah remix final
plt.figure(figsize=(14, 3))
librosa.display.specshow(D_final, sr=sr6, x_axis='time', y_axis='linear')
plt.colorbar(format='%.2f dB')
plt.title('Spectrogram Audio Sesudah Remix (Final)')
plt.xlabel('Time (s)')
plt.ylabel('Frequency (Hz)')

plt.tight_layout()
plt.show()

```





Penjelasan:

Hasil Remix yang telah dibuat

1. Menyamakan tempo lagu, untuk lagu Sad dibuat lebih cepat mengikuti lagu Happy agar vibes yang di dapat lebih energic, namun tempo yang diubah menjadi cepat, durasi dari lagu juga ikut berkurang.
2. Kemudian melakukan pitch shifting dari lagu sad yang memiliki dominasi di suara frekuensi rendah, maka di ubah dengan menaikannya +5 semitone. Karena untuk lagu Happy berada di not A# (10 semitone), dan Sad ada di F (5 semitone).

3. Menambahkan efek crossfade agar transisi antar lagu lebih smooth
4. Memberikan filter tambahan yang ada di modul perkuliahan (6) yaitu memberikan efek click pada setiap beat yang kemudian saya ubah frekuensinya menjadi lebih rendah (90 Hz) sehingga suara bass yang muncul untuk setiap ketukan beat.

Setelah di visualkan dalam bentuk spectrogram saya melihat masih banyak noise di lagu sad untuk range lebih dari 15000 Hz. Maka saya melakukan Low-cut di range tersebut serta melakukan normalisasi dBFS -1 suara audio tidak terlalu keras.

dan juga tambahan sedikit efek fade in dan fade out agar transisi masuk dan keluar lagu lebih halus.

Credit dan Referensi

Materi Pembelajaran

1. [Audio Processing Week 3](#)

Bantuan AI (Chat GPT)

1. [Chat GPT 1](#)
2. [Chat GPT 2](#)

Referensi Berita

- [Berita CNN](#)

Link Music yang dipakai

1. [Malam Pagi Remix](#)
2. [sombr - back to friends](#)