Generative AI Music Creation

Dayeon Lee

Generative AI Music Generation

1. MusicGen
2. Gemini (Generative AI from Google)
3. YourTTS (Multilingual Multi-Speaker TTS)

Project Goal

1. Music Style Generation
   a. Generate instrumental music in the desired genre and mood.
2. Lyrics Creation
   a. Write lyrics based on the style of the generated music.
3. AI singing Synthesis
   a. Combine the generated music and lyrics.
   b. Use AI to synthesize the singing performance.

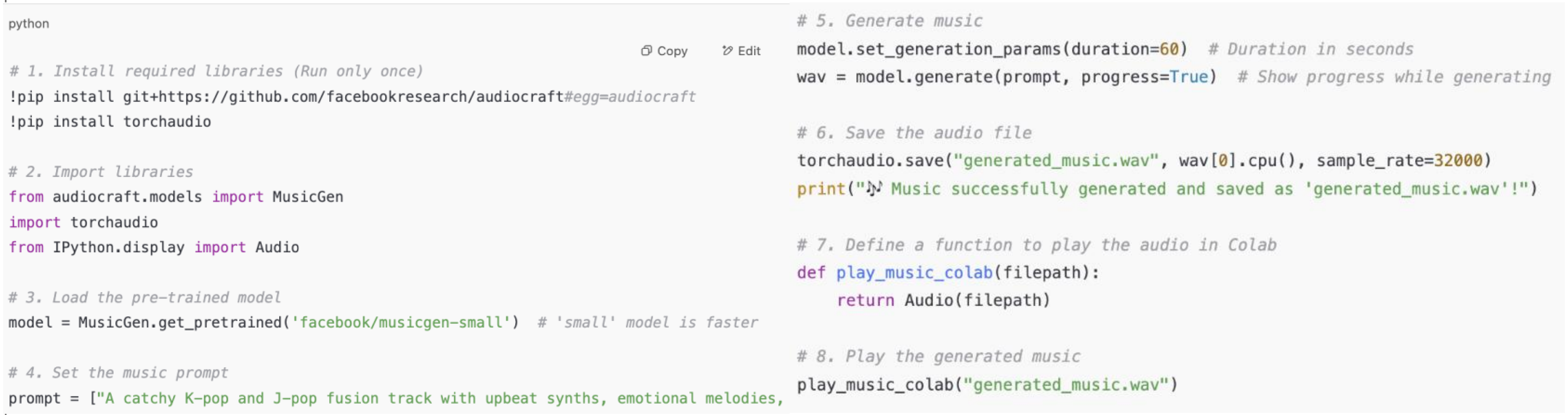
1 MusicGen

- AI-based music generation model developed by Meta AI (formerly Facebook)
- Generates music automatically from text prompts
- Build on a Transformer architecture

1 MusicGen – Pros and Cons

strengths	Limitations
High-quality music from simple text input	Limited to relatively short pieces (seconds to a minute)
Easy to use	Complex music structure can be challenging
Supports a wide range of genres	Bound by its training dataset

1 MusicGen – code structure



```python
# 1. Install required libraries (Run only once)
!pip install git+https://github.com/facebookresearch/audiocraft#egg=audiocraft
!pip install torchaudio

# 2. Import libraries
from audiocraft.models import MusicGen
import torchaudio
from IPython.display import Audio

# 3. Load the pre-trained model
model = MusicGen.get_pretrained('facebook/musicgen-small')  # 'small' model is faster

# 4. Set the music prompt
prompt = ["A catchy K-pop and J-pop fusion track with upbeat synths, emotional melodies,
```

```python
# 5. Generate music
model.set_generation_params(duration=60)  # Duration in seconds
wav = model.generate(prompt, progress=True)  # Show progress while generating

# 6. Save the audio file
torchaudio.save("generated_music.wav", wav[0].cpu(), sample_rate=32000)
print("🎶 Music successfully generated and saved as 'generated_music.wav'!")

# 7. Define a function to play the audio in Colab
def play_music_colab(filepath):
    return Audio(filepath)

# 8. Play the generated music
play_music_colab("generated_music.wav")
```

1 MusicGen – prompt 1

- "A catchy K-pop and J-pop fusion track with upbeat synths, emotional melodies, and bright pop rhythms"

1 MusicGen – prompt 2

- "A gentle and emotional Studio Ghibli-style orchestral piece with soft piano, warm strings, and a touch of fantasy atmosphere"

2 Gemini (Generative AI from Google)

- A next-generation multimodal generative AI model developed by Google DeepMind

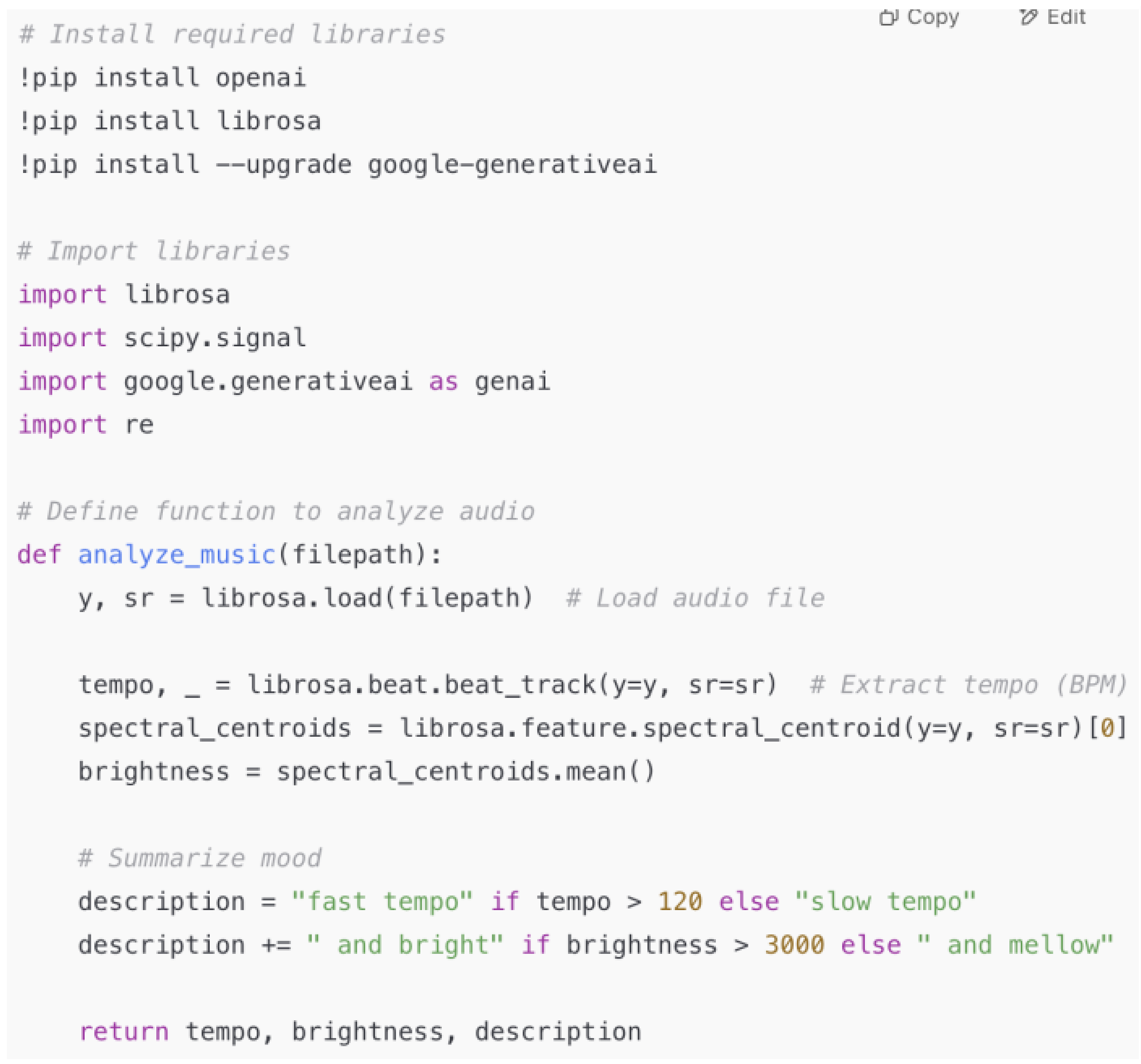
- Gemini is an AI that understands text, images, audio, and video, and responds with natural language explanations.

- Designed for advanced reasoning, productivity, and creativity tasks

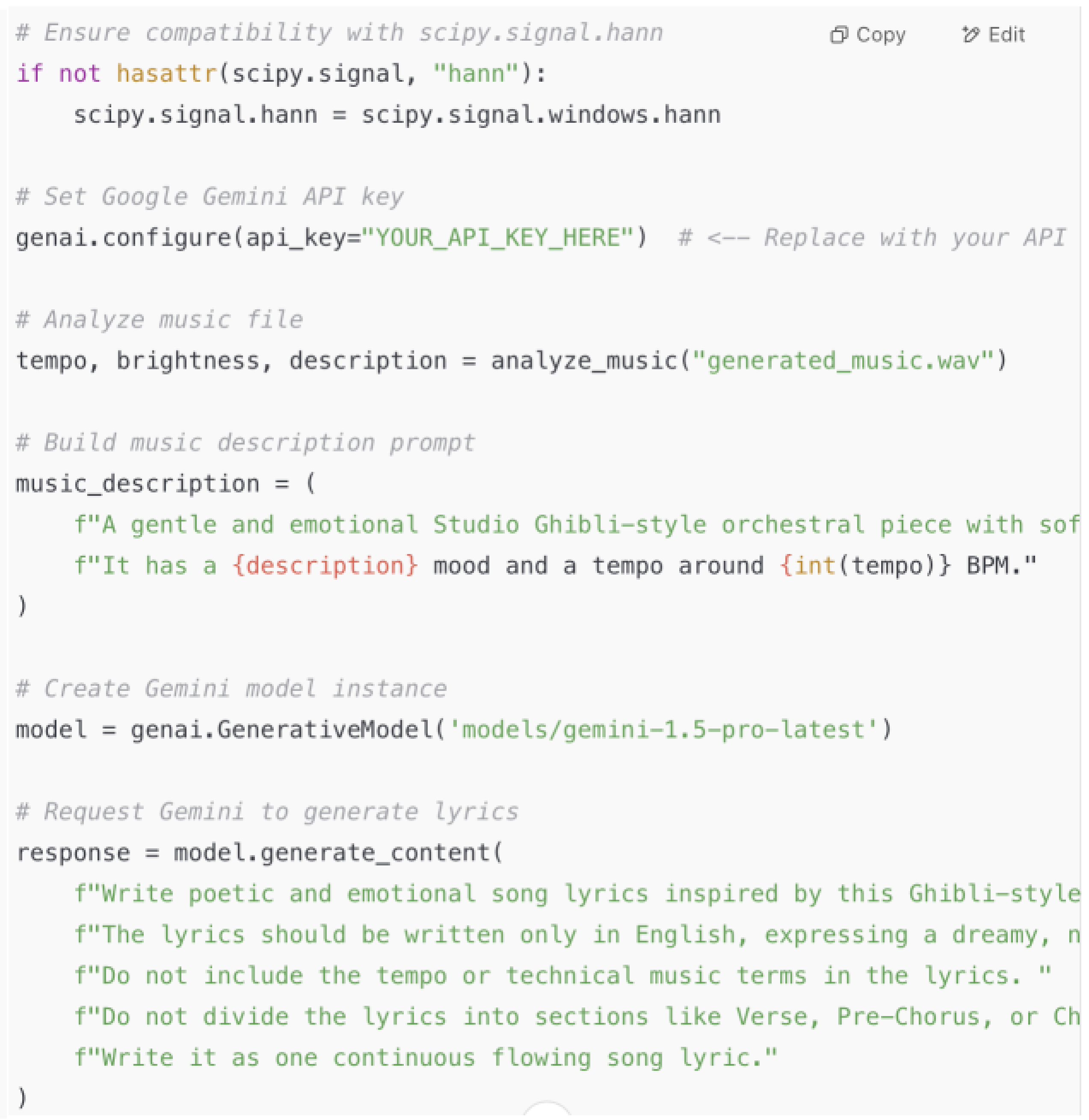
❷ Gemini – Pros and Cons

strengths	Limitations
Multimodal integration	Limited music emotion inference
creative synergy	No deep music theory
Automation-friendly	Requires valid API key

2 Gemini (Generative AI from Google)



```
# Install required libraries
!pip install openai
!pip install librosa
!pip install --upgrade google-generativeai

# Import libraries
import librosa
import scipy.signal
import google.generativeai as genai
import re

# Define function to analyze audio
def analyze_music(filepath):
    y, sr = librosa.load(filepath)  # Load audio file

    tempo, _ = librosa.beat.beat_track(y=y, sr=sr)  # Extract tempo (BPM)
    spectral_centroids = librosa.feature.spectral_centroid(y=y, sr=sr)[0]
    brightness = spectral_centroids.mean()

    # Summarize mood
    description = "fast tempo" if tempo > 120 else "slow tempo"
    description += " and bright" if brightness > 3000 else " and mellow"

    return tempo, brightness, description
```



```
# Ensure compatibility with scipy.signal.hann
if not hasattr(scipy.signal, "hann"):
    scipy.signal.hann = scipy.signal.windows.hann

# Set Google Gemini API key
genai.configure(api_key="YOUR_API_KEY_HERE")  # <-- Replace with your API

# Analyze music file
tempo, brightness, description = analyze_music("generated_music.wav")

# Build music description prompt
music_description = (
    f"A gentle and emotional Studio Ghibli-style orchestral piece with sof
    f"It has a {description} mood and a tempo around {int(tempo)} BPM."
)

# Create Gemini model instance
model = genai.GenerativeModel('models/gemini-1.5-pro-latest')

# Request Gemini to generate lyrics
response = model.generate_content(
    f"Write poetic and emotional song lyrics inspired by this Ghibli-style
    f"The lyrics should be written only in English, expressing a dreamy, n
    f"Do not include the tempo or technical music terms in the lyrics. "
    f"Do not divide the lyrics into sections like Verse, Pre-Chorus, or Ch
    f"Write it as one continuous flowing song lyric."
)
```

2 Gemini (Generative AI from Google) – Results 1

Raindrops on the window pane, blurring the city lights, like my memories of you, fading into the night. We walked along this street, hand in hand, beneath the cherry blossoms, now winter chills my hand. I see your ghost in every crowd, hear your laughter in the silence, a bittersweet symphony, playing in the distance....

2 Gemini (Generative AI from Google) – Results 2

Golden sunlight through the leaves, whispers a forgotten name,
Carried on the gentle breeze, a memory flickering like a flame.
Dust motes dancing in the air, a story waiting to unfold,
Of whispered secrets, silent prayer, and dreams of silver and of gold.........

3 YourTTS (Multilingual Voice Cloning Model)

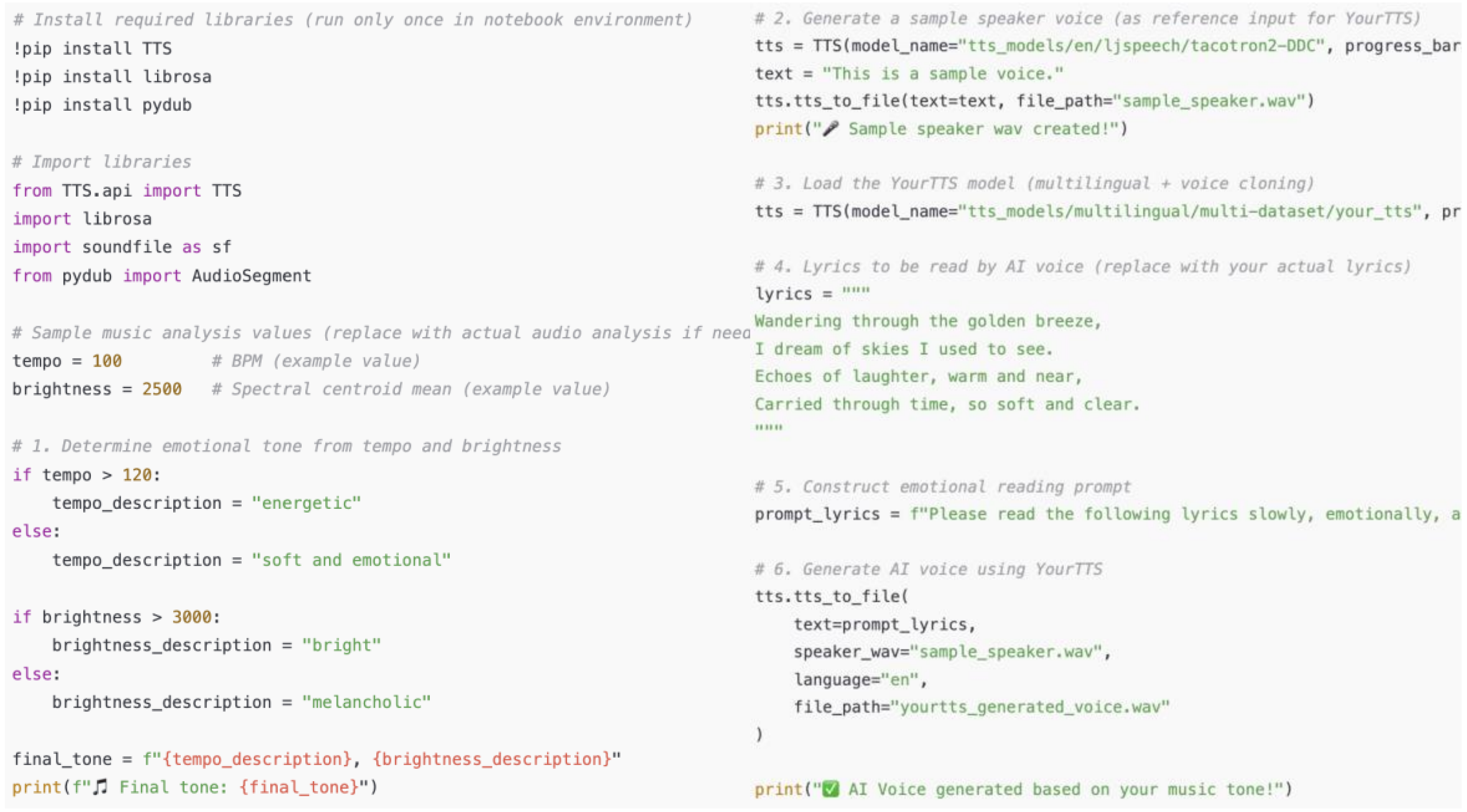
- YourTTS is generative AI that creates speech from text and a voice sample.

- Key Features
  - Voice Cloning
  - Multilingual Support
  - Zero-shot Learning
  - Expressive Output

❸ YourTTS (Multilingual Voice Cloning Model) – Pros and Cons

strengths	Limitations
High-quality voice cloning with few seconds of audio	Sensitive to background noise in voice samples
Multilingual and cross-language voice synthesis	Requires a GPU for real-time or high-performance inference
Zero-shot learning	Limited number of supported languages

3 YourTTS (Multilingual Voice Cloning Model)



```python
# Install required libraries (run only once in notebook environment)
!pip install TTS
!pip install librosa
!pip install pydub

# Import libraries
from TTS.api import TTS
import librosa
import soundfile as sf
from pydub import AudioSegment

# Sample music analysis values (replace with actual audio analysis if need
tempo = 100         # BPM (example value)
brightness = 2500   # Spectral centroid mean (example value)

# 1. Determine emotional tone from tempo and brightness
if tempo > 120:
    tempo_description = "energetic"
else:
    tempo_description = "soft and emotional"

if brightness > 3000:
    brightness_description = "bright"
else:
    brightness_description = "melancholic"

final_tone = f"{tempo_description}, {brightness_description}"
print(f"🎵 Final tone: {final_tone}")
```

```python
# 2. Generate a sample speaker voice (as reference input for YourTTS)
tts = TTS(model_name="tts_models/en/ljspeech/tacotron2-DDC", progress_bar
text = "This is a sample voice."
tts.tts_to_file(text=text, file_path="sample_speaker.wav")
print("🎤 Sample speaker wav created!")

# 3. Load the YourTTS model (multilingual + voice cloning)
tts = TTS(model_name="tts_models/multilingual/multi-dataset/your_tts", pr

# 4. Lyrics to be read by AI voice (replace with your actual lyrics)
lyrics = """
Wandering through the golden breeze,
I dream of skies I used to see.
Echoes of laughter, warm and near,
Carried through time, so soft and clear.
"""

# 5. Construct emotional reading prompt
prompt_lyrics = f"Please read the following lyrics slowly, emotionally, a

# 6. Generate AI voice using YourTTS
tts.tts_to_file(
    text=prompt_lyrics,
    speaker_wav="sample_speaker.wav",
    language="en",
    file_path="yourtts_generated_voice.wav"
)

print("✅ AI Voice generated based on your music tone!")
```

3 YourTTS (Multilingual Voice Cloning Model) – Result 1

- Final Result (Voice + Music):

3 YourTTS (Multilingual Voice Cloning Model) – Result 2

- Final Result (Voice + Music):

4 Conculsion

- Analyzed AI-generated music to detect emotional tone (tempo + brightness)
- Created an expressive AI voice using YourTTS, guided by lyrics and mood
- Mixed voice and background music to produce a complete AI song — no human vocals needed

Thank you