Marwadi University Marwadi Chandarana Group NAAC A+	**Marwadi University** **Faculty of Engineering & Technology** **Department of Information and Communication Technology**	
Subject: Programming With Python (01CT1309)	**Aim:** Audio processing using Python	
Experiment No: 20	**Date:**	**Enrollment No: 92510133028**

Aim: Audio processing using Python

IDE:

Python is a powerful language for audio processing due to its simplicity and ease of use. It provides several libraries for audio processing, including soundfile, librosa, and Pydub, among others. we will explore how to use Python for audio processing and some of its practical applications.

Library Installation

pip install soundfile

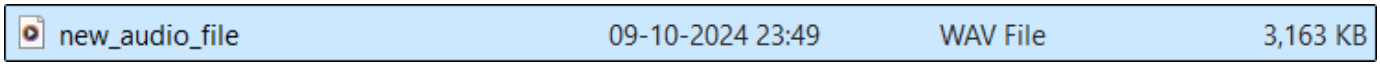
Reading and Writing Audio Files:

The first step in audio processing is reading and writing audio files. In Python, we can use the soundfile library to read and write audio files in various formats, including WAV, FLAC, and MP3. Here's an example of how to read an audio file using the soundfile library:

```
import soundfile as sf
# Load audio file
audio, sample_rate = sf.read('audio_file.wav')
```

Similarly, we can use the same library to write an audio file as follows:

```
import soundfile as sf

# Write audio file
sf.write('new_audio_file.wav', audio, sample_rate)
```



Audio Visualization:

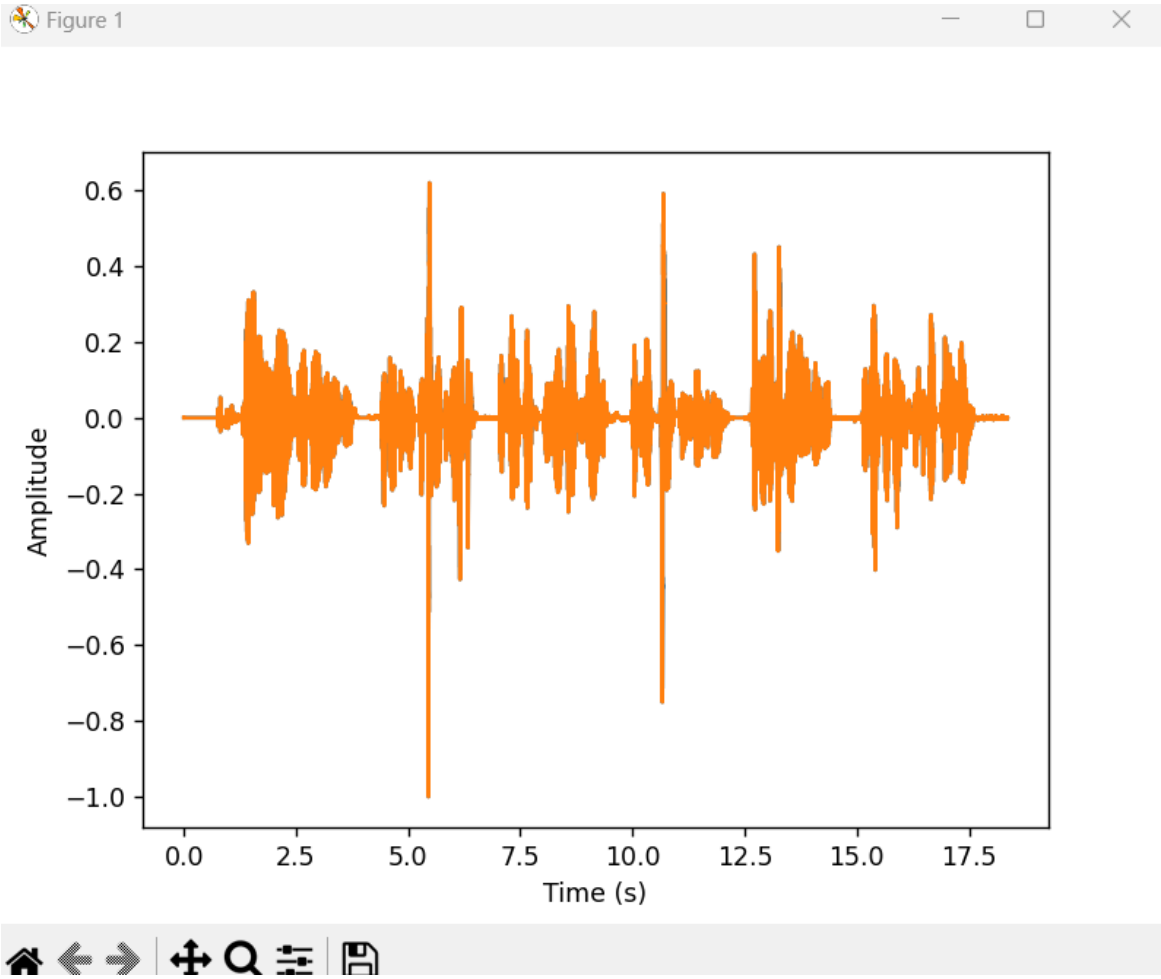
Visualizing audio data is important for analyzing and understanding audio signals. In Python, we can use the matplotlib library to plot audio signals in the time domain or frequency domain. Here's an example of how to

Marwadi University Marwadi Chandarana Group NAAC A+	**Marwadi University** **Faculty of Engineering & Technology** **Department of Information and Communication Technology**	
Subject: Programming With Python (01CT1309)	**Aim:** Audio processing using Python	
Experiment No: 20	**Date:**	**Enrollment No: 92510133028**

plot an audio signal in the time domain:

```
import soundfile as sf
# Load audio file

audio, sample_rate = sf.read(r'C:\Users\Mitesh\OneDrive\Desktop\test.wav')
# Write audio file

sf.write('new_audio_file.wav', audio, sample_rate)



import matplotlib.pyplot as plt

import numpy as np

import soundfile as sf

# Load audio file

#audio, sample_rate = sf.read('audio_file.wav')

# Create time axis

time = np.arange(0, len(audio)) / sample_rate

# Plot audio signal

plt.plot(time, audio)

plt.xlabel('Time (s)')

plt.ylabel('Amplitude')

plt.show()
```

Marwadi University Marwadi Chandarana Group NAAC A+	**Marwadi University** **Faculty of Engineering & Technology** **Department of Information and Communication Technology**	
Subject: Programming With Python (01CT1309)	**Aim:** Audio processing using Python	
Experiment No: 20	**Date:**	**Enrollment No: 92510133028**



Audio Effects:

Python provides several libraries for adding audio effects to audio files. One such library is Pydub, which can be used to add effects such as fade in/out, change speed/pitch, and add equalization (EQ). Here's an example of how to add a fade-in effect to an audio file using Pydub:

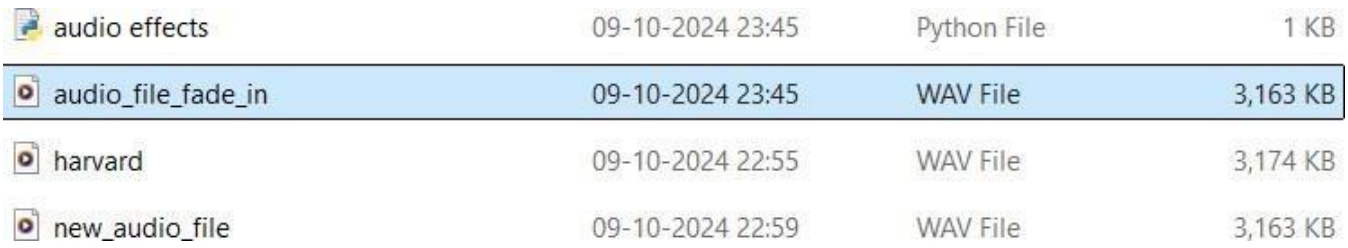
Installation

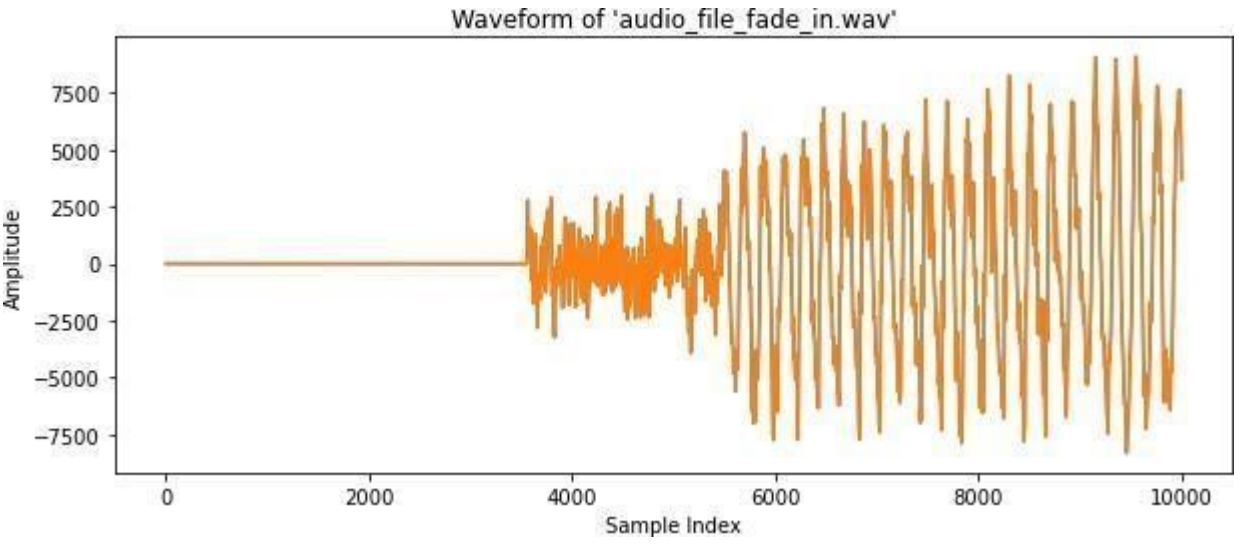
pip install pydub

rom pydub import AudioSegment

Load audio file

```
audio = AudioSegment.from_file('audio_file.wav')

# Add fade in effect
audio_fade_in = audio.fade_in(2000)  # 2 seconds

# Export audio file with fade in effect
audio_fade_in.export('audio_file_fade_in.wav', format='wav')
```





Reference Link: https://www.kaggle.com/code/vuppalaadithyasairam/audio-pre-processing-tutorial-in-python

AUDIO CONVOLUTION of two audio files

```
import numpy as np
import scipy.io.wavfile as wavfile
```

Marwadi University Marwadi Chandarana Group NAAC A+	**Marwadi University** **Faculty of Engineering & Technology** **Department of Information and Communication Technology**	
Subject: Programming With Python (01CT1309)	**Aim:** Audio processing using Python	
Experiment No: 20	**Date:**	**Enrollment No: 92510133028**

```
from scipy.signal import convolve


file1 = "D:\SEM 3 Subjects\Python\audio processing\BAK.wav"
file2 = "D:\SEM 3 Subjects\Python\audio processing\harvard.wav"

rate1, data1 = wavfile.read(file1)


rate2, data2 = wavfile.read(file2)


if rate1 != rate2:
    raise ValueError("Sample rates do not match between the two files")


convolved_audio = convolve(data1, data2, mode='full')


output_file = "D:\SEM 3 Subjects\Python\audio processing\covolved_audio.wav"
wavfile.write(output_file, rate1, convolved_audio.astype(np.int16))
```

https://github.com/keshvi1234/PWP_experiment