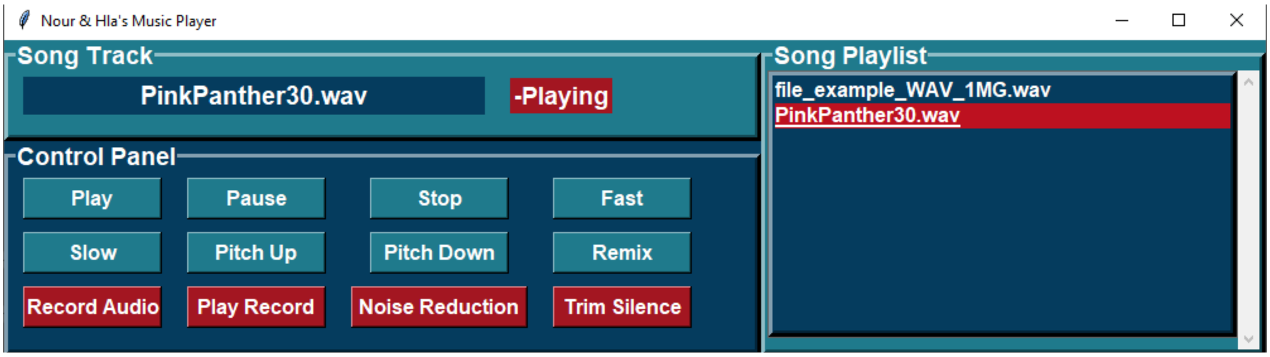
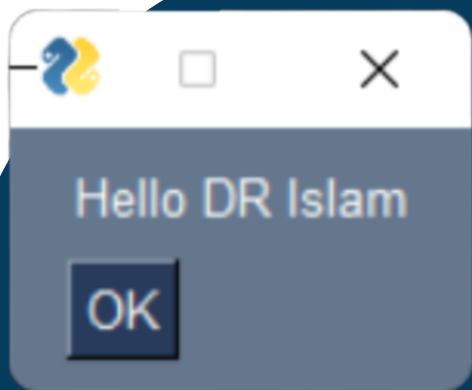
SIGNALS AND SYSTEMS PROJECT REPORT

AUDIO PROCESSING USING PYTHON





Prepared For:

Dr. Islam Shaalan

Prepared By:

Nour Sherif Abouelenin
Hla Essam Dawoud

TABLE OF CONTENTS

INTRODUCTION

Audio signals are signals that vibrate in the audible frequency range. When someone talks, it generates air pressure signals; the ear takes in these air pressure differences and communicates with the brain. That's how the brain helps a person recognize that the signal is speech and understand what someone is saying.

Audio signal processing is a subfield of signal processing that is concerned with the electronic manipulation of audio signals. Audio signals are electronic representations of sound waves—longitudinal waves which travel through air, consisting of compressions and rarefactions.

The sound is typically represented as a waveform: a float or integer (quantized) array representing sound signal A(t) over the discrete time variable t. It can have multiple channels for stereo, 5.1, etc.

In Python, the waveform can be numpy.ndarray or a similar format. The waveform has sampling rate fs, a number of samples per second, e.g. 8k, 16k, 22k, 44k, 48k etc. The highest frequency represented by the waveform is fs/2.

ANACONDA

Anaconda is an open-source distribution for python and R. It is used for data science, machine learning, deep learning, etc. Anaconda offers the easiest way to perform Python/R data science and machine learning on a single machine.

JUPYTER

The Jupyter Notebook App is a server-client application that allows editing and running notebook documents via a web browser. The Jupyter Notebook App can be executed on a local desktop requiring no internet access (as described in this document) or can be installed on a remote server and accessed through the internet.

LIBRARIES

1. Playsound

The playsound module contains only a single function named playsound(). It requires one argument: the path to the file with the sound we have to play. It can be a local file, or a URL.

2. Librosa

Librosa is a python package developed for music and audio analysis. It provides the building blocks necessary to create music information retrieval systems. It is the easiest and the most used in this project.

3. Sounddevice

Sounddevice package provides bindings for the PortAudio library and a few convenience functions to play and record NumPy arrays containing audio signals.

4. Numpy

NumPy is the fundamental package for scientific computing in Python. It is a Python library that provides a multidimensional array object, various derived objects (such as masked arrays and matrices), and an assortment of routines for fast operations on arrays, including mathematical, logical, shape manipulation, sorting, selecting, I/O, discrete Fourier transforms, basic linear algebra, basic statistical operations, random simulation and much more. It was used in this project to convert the audio array to audio.

5. Matplotlib

Matplotlib is a comprehensive library for creating static, animated, and interactive visualizations in Python. It was used to plot the audio signals.

6. Tkinter

Tkinter is a standard library for GUI creation. The Tkinter library is most popular and very easy to use and it comes with many widgets (these widgets help in the creation of nice-looking GUI Applications).

Also, Tkinter is a very light-weight module and it is helpful in creating cross-platform applications (so the same code can easily work on Windows, macOS, and Linux)

7. Pygame

Pygame is a Python module that works with computer graphics and sound libraries and is designed with the power of playing with different multimedia formats like audio, video, etc. While creating our Music Player application, we will be using Pygame's mixer.music module for providing different functionality to our music player application that is usually related to the manipulation of the song tracks.

8. OS module

There is no need to install this module explicitly, as it comes with the standard library of Python. This module provides different functions for interaction with the Operating System.

We are going to use the OS module for fetching the playlist of songs from the specified directory and make it available to the music player application.

PYTHON CODE PROCESS

1. Processes

a. Record Audio

Firstly, we import the libraries we would be using (Sounddevice, Playsound), the scipy.io which is used to write a Numpy array as WAV file. Then, we constructed a function that would record a 15 second audio and save it to your directory.

```
def recordaudio():
   import sounddevice
   from scipy.io.wavfile import write
   from playsound import playsound
   #44100 or 48000 is used frequently in CDs and computer audio , there are
other more commom frequency samples
   fs = 44100
   #duration of record
   second = 15
   print("recording....")
   #sounddevice.rec records an audio and save it in a form of numpy array
   record_voice = sounddevice.rec(int(second*fs),samplerate=fs,channels=2)
   sounddevice.wait()
   #write(filnename,rate,data) which converts a numpy array into a wav file
   write('record.wav',fs,record_voice)
   print("playing record....")
   #Play record saved as wav file
   playsound('directory')
```

b. Recorded Audio Clarification

Again, initially we import the packages. Then, we load and play the pre-recorded file.

```
#importing main packages
import librosa
import numpy as np
import scipy as sp
import librosa.display
import IPython.display
from IPython.display import display
from scipy import signal
from scipy.io import wavfile
import noisereduce as nr
#loading the pre-recorded WAV file
filename = ('record.wav')
y0, sr0 = librosa.load(filename)
```

Noise Reduction

We work on reducing the background noises by two methods, noise reduction by median and noise reduction by the pre-imported library noisereduce.

```
#reduce noise by median
def reduce_noise_median(y, sr):
   y = sp.signal.medfilt(y,3)
   return (y)
wavfile.write("mywav_reduced_noise1.wav", sr0, reduce_noise_median(y0, sr0))
#loading the 1st noise reduced WAV file
filename = ('mywav_reduced_noise1.wav')
y1, sr1 = librosa.load(filename)
#perform 2nd noise reduction by noisereduce
reduced_noise = nr.reduce_noise(y=y1,sr=sr1,thresh_n_mult_nonstationary=2,
stationary=False,n_jobs=2,)
wavfile.write("mywav_reduced_noise2.wav", sr1, reduced_noise)
#loading the final noise reduced WAV file
filename = ('mywav_reduced_noise2.wav')
y2, sr2 = librosa.load(filename)
#play pre-recorded audio after noise reduction as an array
display(IPython.display.Audio(data=y2, rate=sr2))
```

Trimming Silence

```
#loading the final noise reduced WAV file
filename = ('mywav_reduced_noise.wav')
y2, sr2 = librosa.load(filename)
#trim the beginning and ending silence
yt, index = librosa.effects.trim(y2)
#print the durations
print(librosa.get_duration(y2), librosa.get_duration(yt))
from IPython.display import Audio
wave_audio_trim = np.sin(yt)
display(Audio(wave_audio_trim, rate=20000))
```

c. Time Stretching & Pitching

As usual, we start by importing the required library which in this project is the librosa package. Then, we load the pre-recorded WAV file and print its waveform and play it.

```
#main packages
import librosa
import librosa.display
import IPython.display
import numpy as np
import matplotlib.pyplot as plt
```

```python
from IPython.display import display
#loading the WAV file
filename = librosa.example('nutcracker')
y, sr = librosa.load(filename)
#printing the waveform and the sampling rate
print('waveform')
print(y)
print('\nsampling rate')
print(sr)
#ploting the original wave
fig, ax = plt.subplots(nrows=1, sharex=True)
librosa.display.waveshow(y, sr=sr)
ax.set(title='Envelope view, mono')
ax.label_outer()
#play the audio as an array
display(IPython.display.Audio(data=y, rate=sr))
```

Time Stretching

```python
#speed-up audio (compresses the audio to be twice as fast)
y_fast = librosa.effects.time_stretch(y, rate=2.0)
#slow-down audio (compresses the audio to half the original speed)
y_slow = librosa.effects.time_stretch(y, rate=0.5)
```

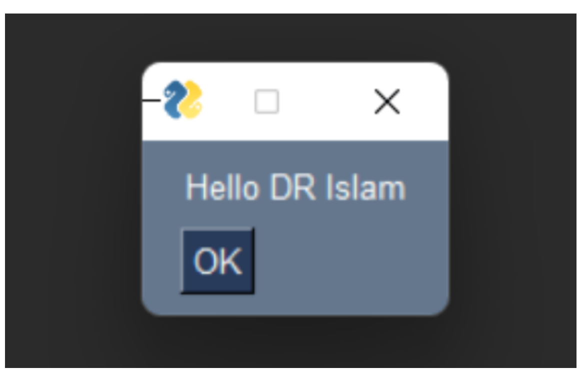
Pitching

```python
#pitch shift by -5, 5 octaves down (frequency and pitch decrease)
y_tritone = librosa.effects.pitch_shift(y, sr=sr, n_steps=-5)
#pitch shift by 5, 5 octaves up (frequency and pitch increase)
y_third = librosa.effects.pitch_shift(y, sr=sr, n_steps=5)
```

2. Graphical User Interface - GUI

The graphical user interface (GUI) is a form of user interface that allows users to interact with electronic devices through graphical icons and audio indicators such as primary notation, instead of text-based UIs, typed command labels or text navigation.

a. Our First Try at GUI



b. The GUI Process

First, we import Tkinter, pygame, and os packages and any other package we will be using.

```
#main packages
from tkinter import *
import pygame
from pygame.locals import *
import os
from os import path
import librosa
import librosa.display
import IPython.display
import numpy as np
from IPython.display import Audio
import sounddevice
from scipy.io.wavfile import write
from playsound import playsound
import scipy as sp
from IPython.display import display
from scipy import signal
from scipy.io import wavfile
import noisereduce as nr
import subprocess
pygame.mixer.pre_init(44100, 16, 2, 4096) #frequency, size, channels,
buffersize
pygame.init() #turn all of pygame on.
```

After importing packages and modules, now it's time to create a basic window where we will add our UI elements or Tkinter widgets to the MusicPlayer Class like our buttons and scroll down list.

1. _init_ Constructor

With the help of this constructor, we set the title for the window and geometry for the window. We also initiate pygame and pygame mixer and declare the track variable and status variable.

```
def __init__(self,root):
    self.root = root
    # Title of the window
    self.root.title("Nour & Hla's Music Player")
    # Window Geometry
    self.root.geometry("1000x250+250+250")
    # Initiating Pygame
    pygame.init()
```

```
    # Initiating Pygame Mixer
    pygame.mixer.init()
    # Declaring track Variable
    self.track = StringVar()
    # Declaring Status Variable
    self.status = StringVar()

    # Creating the Track Frames for Song label & status label
    trackframe = LabelFrame(self.root,text="Song
Track",font=("Helvetica",14,"bold"),bg="#1f7a8c",fg="white",bd=5,relief=RAISED
)
    trackframe.place(x=0,y=0,width=600,height=80)
    # Inserting Song Track Label
    songtrack =
Label(trackframe,textvariable=self.track,width=30,font=("Helvetica",15,"bold")
,bg="#053c5e",fg="white").grid(row=0,column=0,padx=10,pady=5)
    # Inserting Status Label
    trackstatus =
Label(trackframe,textvariable=self.status,font=("Helvetica",14,"bold"),bg="#a3
1621",fg="white").grid(row=0,column=1,padx=10,pady=5)

    # Creating Button Frame
    buttonframe = LabelFrame(self.root,text="Control
Panel",font=("Helvetica",14,"bold"),bg="#053c5e",fg="white",bd=5,relief=RAISED
)
    buttonframe.place(x=0,y=80,width=600,height=170)
    # Inserting Play Button
    playbtn =
Button(buttonframe,text="Play",command=self.playsong,width=10,height=1,font=("
Helvetica",12,"bold"),fg="white",bg="#1f7a8c").grid(row=0,column=0,padx=10,pad
y=5)
    # Inserting Pause Button
    playbtn =
Button(buttonframe,text="Pause",command=self.pausesong,width=10,height=1,font=
("Helvetica",12,"bold"),fg="white",bg="#1f7a8c").grid(row=0,column=1,padx=10,p
ady=5)
    # Inserting Stop Button
    playbtn =
Button(buttonframe,text="Stop",command=self.stopsong,width=10,height=1,font=("
Helvetica",12,"bold"),fg="white",bg="#1f7a8c").grid(row=0,column=2,padx=10,pad
y=5)
    # Inserting Fast Button
    playbtn =
Button(buttonframe,text="Fast",command=self.fast,width=10,height=1,font=("Helv
etica",12,"bold"),fg="white",bg="#1f7a8c").grid(row=0,column=3,padx=10,pady=5)
    # Inserting Slow Button
    playbtn =
Button(buttonframe,text="Slow",command=self.slow,width=10,height=1,font=("Helv
etica",12,"bold"),fg="white",bg="#1f7a8c").grid(row=1,column=0,padx=10,pady=5)
    # Inserting Pitch Shift Down Button
    playbtn = Button(buttonframe,text="Pitch
Down",command=self.pitchshiftdown,width=10,height=1,font=("Helvetica",12,"bold
"),fg="white",bg="#1f7a8c").grid(row=1,column=2,padx=10,pady=5)
```

```
    # Inserting Pitch Shift Up Button
    playbtn = Button(buttonframe,text="Pitch
Up",command=self.pitchshiftup,width=10,height=1,font=("Helvetica",12,"bold"),f
g="white",bg="#1f7a8c").grid(row=1,column=1,padx=10,pady=5)
    # Inserting Remix Button
    playbtn =
Button(buttonframe,text="Remix",command=self.remix,width=10,height=1,font=("He
lvetica",12,"bold"),fg="white",bg="#1f7a8c").grid(row=1,column=3,padx=10,pady=
5)
    # Inserting Record Button
    playbtn = Button(buttonframe,text="Record
Audio",command=self.recordaudio,width=10,height=1,font=("Helvetica",12,"bold")
,fg="white",bg="#a31621").grid(row=2,column=0,padx=10,pady=5)
    # Inserting Noise Reduction Button
    playbtn = Button(buttonframe,text="Noise
Reduction",command=self.noisereduction,width=13,height=1,font=("Helvetica",12,
"bold"),fg="white",bg="#a31621").grid(row=2,column=2,padx=10,pady=5)
    # Inserting Trim Button
    playbtn = Button(buttonframe,text="Trim
Silence",command=self.trimsilence,width=10,height=1,font=("Helvetica",12,"bold
"),fg="white",bg="#a31621").grid(row=2,column=3,padx=10,pady=5)
    # Inserting Play Record Button
    playbtn = Button(buttonframe,text="Play
Record",command=self.playrecord,width=10,height=1,font=("Helvetica",12,"bold")
,fg="white",bg="#a31621").grid(row=2,column=1,padx=10,pady=5)

    # Creating Playlist Frame
    songsframe = LabelFrame(self.root,text="Song
Playlist",font=("Helvetica",14,"bold"),bg="#1f7a8c",fg="white",bd=5,relief=RAI
SED)
    songsframe.place(x=600,y=0,width=400,height=250)
    # Inserting scrollbar
    scrol_y = Scrollbar(songsframe,orient=VERTICAL)
    # Inserting Playlist listbox
    self.playlist =
Listbox(songsframe,yscrollcommand=scrol_y.set,selectbackground="#BD1120",selec
tmode=SINGLE,font=("Helvetica",12,"bold"),bg="#053c5e",fg="white",bd=5,relief=
RAISED)
    # Applying Scrollbar to listbox
    scrol_y.pack(side=RIGHT,fill=Y)
    scrol_y.config(command=self.playlist.yview)
    self.playlist.pack(fill=BOTH)
    # Changing Directory for fetching Songs
    os.chdir("C:/Users/Desktop/Python Project")
    # Fetching Songs
    songtracks = os.listdir()
    # Inserting Songs into Playlist
    for track in songtracks:
     self.playlist.insert(END,track)
```

2. Playsong() Function

```
def playsong(self):
    # Displaying Selected Song title
    self.track.set(self.playlist.get(ACTIVE))
    # Displaying Status
    self.status.set("-Playing")
    # Loading Selected Song
    pygame.mixer.music.load(self.playlist.get(ACTIVE))
    #sound = pygame.mixer.Sound(self.playlist.get(ACTIVE))
    # Playing Selected Song
    pygame.mixer.music.play()
```

3. Stopsong() Function

```
def stopsong(self):
    # Displaying Status
    self.status.set("-Stopped")
    # Stopped Song
    pygame.mixer.music.stop()
```

4. Pausesong() Function

```
def pausesong(self):
    # Displaying Status
    self.status.set("-Paused")
    # Paused Song
    pygame.mixer.music.pause()
```

5. Fast() Function

```
def fast(self):
    # Displaying Selected Song title
    self.track.set(self.playlist.get(ACTIVE))
    # Displaying Status
    self.status.set("-Speeding Up")
    # Loading Selected Song
    pygame.mixer.music.load(self.playlist.get(ACTIVE))
    y, sr = librosa.load(self.playlist.get(ACTIVE))
    y_fast = librosa.effects.time_stretch(y, rate=0.5)
    wave_audio = np.sin(y_fast)
    Audio(wave_audio, rate=20000)
    import soundfile as sf
    sf.write('wave_audio.wav',wave_audio,48000)
    # Playing Selected Song
    file = "wave_audio.wav"
    pygame.mixer.music.load(file)
    pygame.mixer.music.play()
```

6. Slow() Function

```
def slow(self):
    # Displaying Selected Song title
    self.track.set(self.playlist.get(ACTIVE))
    # Displaying Status
    self.status.set("-Slowing Down")
    # Loading Selected Song
    pygame.mixer.music.load(self.playlist.get(ACTIVE))
    y, sr = librosa.load(self.playlist.get(ACTIVE))
    y_slow = librosa.effects.time_stretch(y, rate=0.2)
    wave_audio1 = np.sin(y_slow)
    Audio(wave_audio1, rate=20000)
    import soundfile as sf
    sf.write('wave_audio1.wav',wave_audio1,48000)
    # Playing Selected Song
    file = "wave_audio1.wav"
    pygame.mixer.music.load(file)
    pygame.mixer.music.play()
```

7. Pitchshiftdown() Function

```
def pitchshiftdown(self):
    # Displaying Selected Song title
    self.track.set(self.playlist.get(ACTIVE))
    # Displaying Status
    self.status.set("-Pitch Down")
    # Loading Selected Song
    pygame.mixer.music.load(self.playlist.get(ACTIVE))
    y, sr = librosa.load(self.playlist.get(ACTIVE))
    #pitch shift by -10, 10 octaves down (frequency and pitch decrease)
    y_tritone = librosa.effects.pitch_shift(y, sr=sr, n_steps=-10)
    wave_audio2 = np.sin(y_tritone)
    Audio(wave_audio2, rate=20000)
    import soundfile as sf
    sf.write('wave_audio2.wav',wave_audio2,48000)
    # Playing Selected Song
    file = "wave_audio2.wav"
    pygame.mixer.music.load(file)
    pygame.mixer.music.play()
```

8. Pitchshiftup() Function

```
def pitchshiftup(self):
    # Displaying Selected Song title
    self.track.set(self.playlist.get(ACTIVE))
    # Displaying Status
    self.status.set("-Pitch Up")
    # Loading Selected Song
    pygame.mixer.music.load(self.playlist.get(ACTIVE))
    y, sr = librosa.load(self.playlist.get(ACTIVE))
```

```
    #pitch shift by 10, 10 octaves up (frequency and pitch increase)
    y_third = librosa.effects.pitch_shift(y, sr=sr, n_steps=10)
    wave_audio3 = np.sin(y_third)
    Audio(wave_audio3, rate=20000)
    import soundfile as sf
    sf.write('wave_audio3.wav',wave_audio3,48000)
    # Playing Selected Song
    file = "wave_audio3.wav"
    pygame.mixer.music.load(file)
    pygame.mixer.music.play()
```

9. Remix() Function

```
def remix(self):
    # Displaying Selected Song title
    self.track.set(self.playlist.get(ACTIVE))
    # Displaying Status
    self.status.set("-Remix")
    # Loading Selected Song
    pygame.mixer.music.load(self.playlist.get(ACTIVE))
    y, sr = librosa.load(self.playlist.get(ACTIVE))
    #compute beats
    _, beat_frames = librosa.beat.beat_track(y=y, sr=sr,
                                             hop_length=512)
    #convert from frames to sample indices
    beat_samples = librosa.frames_to_samples(beat_frames)

    #generate intervals from consecutive events
    intervals = librosa.util.frame(beat_samples, frame_length=2,
                                   hop_length=1).T

    #reverse the beat intervals
    y_out = librosa.effects.remix(y, intervals[::-2])
    wave_audio4 = np.sin(y_out)
    Audio(wave_audio4, rate=20000)
    import soundfile as sf
    sf.write('wave_audio4.wav',wave_audio4,48000)
    # Playing Selected Song
    file = "wave_audio4.wav"
    pygame.mixer.music.load(file)
    pygame.mixer.music.play()
```

10. Recordaudio() Function

```
def recordaudio(self):
    # Displaying Status
    self.status.set("-Recording Done")
    #44100 or 48000 is used frequently in CDs and computer audio , there are
other more commom frequency samples
    fs = 44100
    #duration of record
```

```
    second = 15
    #sounddevice.rec records an audio and save it in a form of numpy array
    record_voice = sounddevice.rec(int(second*fs),samplerate=fs,channels=2)
    sounddevice.wait()
    #write(filnename,rate,data) which converts a numpy array into a wav file
    write('record.wav',fs,record_voice)
    write('record.mp3',fs,record_voice)
    # Displaying Selected Song title
    self.track.set("record.wav")
```

11. Playrecord() Function

```
def playrecord(self):
    # Displaying Status
    self.status.set("-Playing Record")
    # Displaying Selected Song title
    self.track.set("record.mp3")
    # Loading Selected Song
    file = "record.mp3"
    sound = pygame.mixer.Sound(file)
    sound.play()
```

12. Noisereduction() Function

```
def noisereduction(self):
    # Displaying Status
    self.status.set("-Noise Reducing")
    #loading the pre-recorded WAV file
    filename = ('record.wav')
    y0, sr0 = librosa.load(filename)
    #reduce noise by median
    def reduce_noise_median(y, sr):
        y = sp.signal.medfilt(y,3)
        return (y)
    wavfile.write("mywav_reduced_noise1.wav", sr0, reduce_noise_median(y0,
sr0))
    #loading the 1st noise reduced WAV file
    filename = ('mywav_reduced_noise1.wav')
    y1, sr1 = librosa.load(filename)
    #perform 2nd noise reduction by noisereduce
    reduced_noise = nr.reduce_noise(y=y1,sr=sr1,
thresh_n_mult_nonstationary=2,stationary=False,n_jobs=2,)
    wavfile.write("mywav_reduced_noise2.wav", sr1, reduced_noise)
    write("mywav_reduced_noise2.mp3",sr1,reduced_noise)
    #loading the final noise reduced WAV file
    file = "mywav_reduced_noise2.mp3"
    sound = pygame.mixer.Sound(file)
    sound.set_volume(1.0)
    sound.play()
```

13. Trimsilence() Function

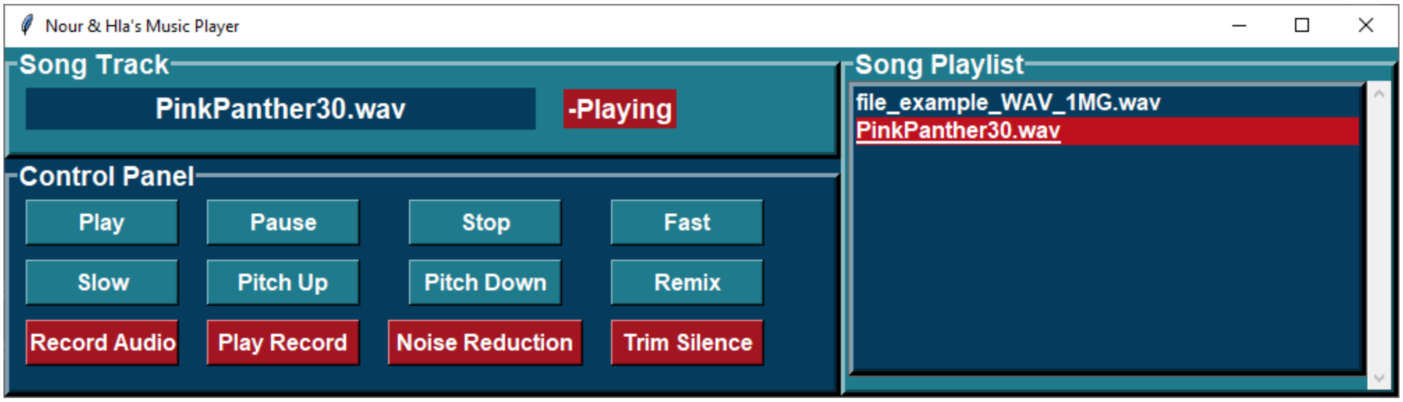
```
def trimsilence(self):
    # Displaying Status
    self.status.set("-Trimming Silence")
    #loading the WAV file
    filename = ('mywav_reduced_noise2.wav')
    y2, sr2 = librosa.load(filename)
    #trim the beginning and ending silence
    yt, index = librosa.effects.trim(y2)
    from IPython.display import Audio
    wave_audio_trim = np.sin(yt)
    Audio(wave_audio_trim, rate=20000)
    import soundfile as sf
    wavfile.write('wave_audio_trim.wav',sr2,wave_audio_trim)
    # Playing Selected Song
    file = "wave_audio_trim.wav"
    sound = pygame.mixer.Sound(file)
    sound.set_volume(1.0)
    sound.play()
```

14. The Root Window Looping

```
root = Tk()
# Passing Root to MusicPlayer Class
MusicPlayer(root)
root.mainloop()
```

c. GUI Color Palette

d. Final GUI



REFERENCES

1. *Matplotlib — Visualization with Python*, Matplotlib.
2. *Anaconda | The World's Most Popular Data Science Platform*, Anaconda.
3. "API Documentation — python-sounddevice, version 0.3.12." *Python sounddevice*, API Documentation — python-sounddevice, version 0.3.12.
4. Driscoll, Mike. "PySimpleGUI: The Simple Way to Create a GUI With Python – Real Python." *Real Python*, PySimpleGUI: The Simple Way to Create a GUI With Python.
5. "Effects — librosa 0.9.1 documentation." *Librosa*, Effects — librosa 0.9.1 documentation.
6. Jaroch, Pawel. "High Quality Audio with Python and PyAudio." *Dolby.io*, 29 July 2020, High Quality Audio with Python and PyAudio - Dolby.io.
7. "jiaaro/pydub: Manipulate audio with a simple and easy high level interface." *GitHub*, GitHub - jiaaro/pydub: Manipulate audio with a simple and easy high level interface.
8. "Manipulate Audio File In Python With 6 Powerful Tips." *CODE FORESTS*, 2 October 2021, Manipulate Audio File In Python With 6 Powerful Tips | CODE FORESTS.
9. Murphy, Caleb J. "Pitch Shifter: What It Is and How to Use It." *Musician on a Mission*, 25 May 2021, Pitch Shifter: What It Is and How to Use It.
10. "Music Player Application using Tkinter (Python Project)." *Studytonight*, Music Player Application using Tkinter (Python Project) - Studytonight.
11. "noise_reduction/noise.py at master · dodiku/noise_reduction · GitHub." *GitHub*, noise_reduction/noise.py at master.
12. "Noise reduction using spectral gating in python." *Tim Sainburg*, 7 July 2018, Tim Sainburg – Noise reduction using spectral gating in python.

13. "NumPy documentation — NumPy v1.22 Manual." *NumPy*, NumPy documentation — NumPy v1.22 Manual.
14. "1. What is the Jupyter Notebook? — Jupyter/IPython Notebook Quick Start Guide 0.1 documentation." *Jupyter Notebook Quick Start Guide*, 1. What is the Jupyter Notebook?.
15. "Play sound in Python." *GeeksforGeeks*, 13 January 2021, Play sound in Python - GeeksforGeeks.
16. "Python - GUI Programming (Tkinter)." *Tutorialspoint*, Python - GUI Programming (Tkinter) .
17. "scipy.io.wavfile.write — SciPy v1.8.0 Manual." *Numpy and Scipy Documentation*, scipy.io.wavfile.write — SciPy v1.8.0 Manual .
18. Sikkema, Kelly. "Top 3 Python Packages to learn Audio Data Science Project." *Towards Data Science*, 18 December 2021, Top 3 Python Packages to learn Audio Data Science Project | by Cornellius Yudha Wijaya.
19. Tagliaferri, Lisa. "How To Install pygame and Create a Template for Developing Games in Python 3." *DigitalOcean*, 15 June 2017, How To Install pygame and Create a Template for Developing Games in Python 3 | DigitalOcean.