

Musicalizador

Pensamiento Computacional

Trabajo Práctico Final



Profesores: Patricio Moreno, Débora Copa, Álvaro Gaona

Alumnos: Manuel Valiente, Marcos Vollert, Mateo Constantini, Santiago Bunge

Mail de los alumnos:

mvaliante@udesa.edu.ar

vollertm@udesa.edu.ar

constantinim@udesa.edu.ar

sbunge@udesa.edu.ar

Objetivos:

El objetivo de este trabajo es hacer un programa en Python que permita a partir de una partitura escrita en un archivo del tipo .txt sintetizar las notas predefinidas a su vez que éste se pueda conectar a un metalófono físico y que mediante el mismo se toquen las notas indicadas en la partitura.

Diseño:

El programa está diseñado a partir de diez archivos: main.py, analyze_files.py, functions.py, harmonics.py, mid2score.py, module.py, notes.py, partiture.txt, piano.txt y audio.wav.

main.py es el archivo principal el cual debe ser ejecutado para hacer funcionar el programa y es quien llama a las otras funciones y archivos.

analyze_files es un archivo que se ocupa de leer los archivos .txt del instrumento y de la partitura y los devuelve en forma de lista para poder ser utilizados en python.

functions.py lo que hace es traducir todas las funciones de los moduladores de amplitud a python para luego ser utilizados.

harmonics.py

module.py

notes.py

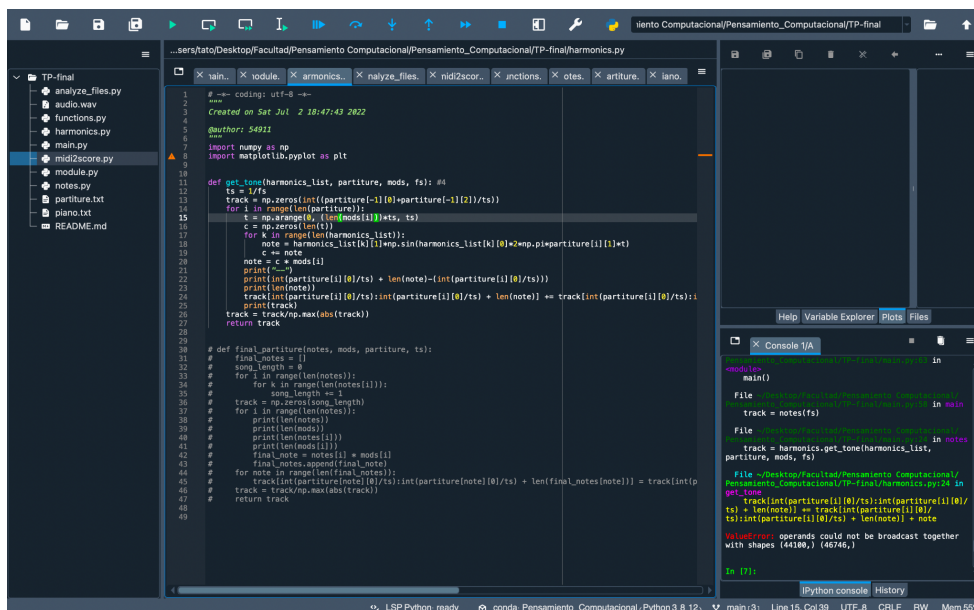
piano.txt

partiture.txt

audio.wav

Reseña sobre problemas encontrados y soluciones:

Los principales problemas fueron de lógica y errores que daba el código al ir avanzando con el mismo.

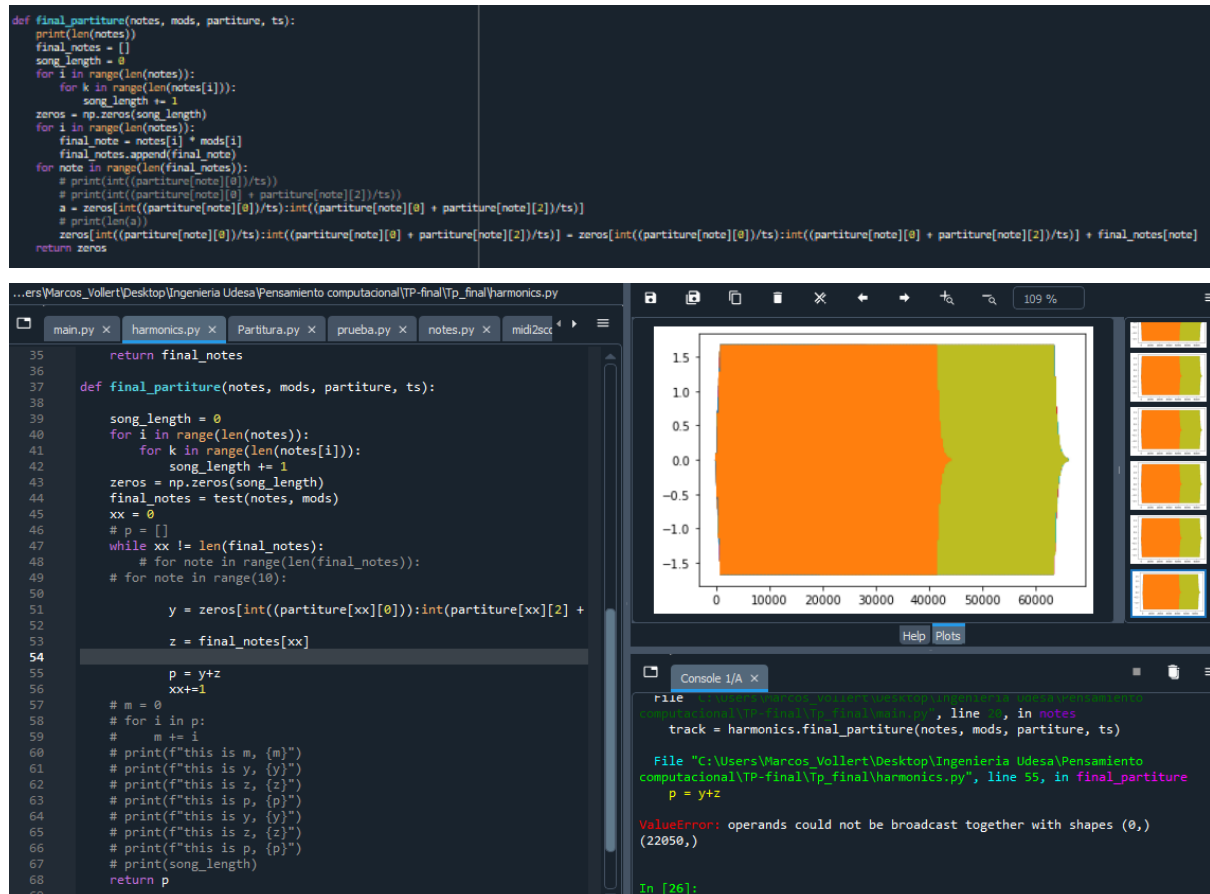


```
1 # -*- coding: utf-8 -*-
2
3 Created on Sat Jul 2 18:47:43 2022
4 @author: 54911
5
6 import numpy as np
7 import matplotlib.pyplot as plt
8
9
10 def get_tone(harmonics_list, partiture, mods, fs): #4
11     ts = 1/fs
12     track = np.zeros(int((partiture[-1][0]-partiture[-1][2])/ts))
13     for i in range(len(partiture)):
14         t = np.arange(0, (len(mods[i])*ts), ts)
15         c = np.zeros(len(t))
16         for k in range(len(harmonics_list)):
17             Note = harmonics_list[k][1]*np.sin(harmonics_list[k][0]*np.pi*partiture[i][1]*t)
18             c += Note
19         Note = c * mods[i]
20         print(len(mods))
21         print(int(partiture[i][0]/ts) + len(note)-(int(partiture[i][0]/ts)))
22         print(len(note))
23         track[int(partiture[i][0]/ts):int(partiture[i][0]/ts) + len(note)] += track[int(partiture[i][0]/ts):
24             int(partiture[i][0]/ts) + len(note)]
25         print(track)
26         track = track/np.max(abs(track))
27     return track
28
29
30 # def final_partiture(notes, mods, partiture, ts):
31 #     final_notes = []
32 #     song_length = 0
33 #     for i in range(len(notes)):
34 #         for k in range(len(mods[i])):
35 #             song_length += 1
36 #     track = np.zeros(song_length)
37 #     for i in range(len(notes)):
38 #         print(len(notes))
39 #         print(len(mods[i]))
40 #         print(len(mods[i]))
41 #         final_note = notes[i] * mods[i]
42 #         final_notes.append(final_note)
43 #     for note in range(len(final_notes)):
44 #         track[int(partiture[note][0]/ts):int(partiture[note][0]/ts) + len(final_notes[note])] = track[int(p
45 #             track = track/np.max(abs(track))
46 #     return track
47
48
49
```

```
File ~/Desktop/Facultad/Pensamiento Computacional/TP-final/harmonics.py:24 in
get_tone
    track[int(partiture[i][0]/ts):int(partiture[i][0]/
ts) + len(note)] += track[int(partiture[i][0]/
ts):int(partiture[i][0]/ts) + len(note)] + note
ValueError: operands could not be broadcast together
with shapes (43180,) (40740,)
```

Este error surgió ya que la longitud de dos vectores que se estaba buscando sumar era distinta, .

```
File "C:\Users\54911\Documents\AA_UdeSa\Cursada\Pensamiento computacional\TP final\harmonics.py",  
line 45, in final_partiture  
zeros[int((partiture[note][0])/ts):int((partiture[note][0] + partiture[note][2])/ts)] =  
zeros[int((partiture[note][0])/ts):int((partiture[note][0] + partiture[note][2])/ts)] +  
final_notes[note]  
ValueError: operands could not be broadcast together with shapes (30869,) (30870,)
```



Indicaciones para ejecutar correctamente el programa y las pruebas:

Primero deberías abrir el terminal (ingresando "cmd" en el buscador de windows en windows o haciendo clic en el launchpad en el dock e ingresando "Terminal" en mac). Luego de hacer eso, deberias ingresar "cd" seguido por el directorio donde se encuentre el archivo "main.py", y una vez que hiciste eso, tenes que ingresar los argumentos necesarios para correr el código en el siguiente orden (con la frecuencia siendo "8000 - 9600 - 11025 - 12000 - 16000 - 22050 - 24000 - 32000 - 41000 - 48000 - 88200 - 96000", y si este paso se saltea, la frecuencia queda como 48000)

-f (frecuencia) -i piano.txt -p Partitura.py -o audio.wav

Resultados de ejecuciones:

Bibliografía:

```
import numpy as np
import matplotlib.pyplot as plt
import module

def get_tone(harmonics_list, partiture, mods, fs): #4
    ts = 1/fs
    song_length = 0
    notes_duration = []
    for i in range(len(partiture)):
        length = (partiture[i][0] + partiture[i][2])/ts
        notes_duration.append(length)
    track = np.zeros(int(max(notes_duration))) #int((partiture[-1][0]+partiture[-1][2])/ts)
    print(len(track))
    for i in range(len(partiture)):
        t = np.arange(0, (len(mods[i])/fs, ts)
        c = np.zeros(len(t))
        for k in range(len(harmonics_list)):
            note = harmonics_list[k][1]*np.sin(harmonics_list[k][0]*2*np.pi*partiture[i][1]*t)
            print(c)
            c += note
```