Instituto Tecnológico y de Estudios Superiores de Monterrey
Escuela de Ingeniería y Ciencias
Agosto - Diciembre 2022
Campus Monterrey

Implementación de Métodos Computacionales

Avance de Situación Problema 1
Implementación básica de un DSL para ejecutar música en Python

Profesor:
Roman Martinez Martinez

Grupo: 850

Integrantes:

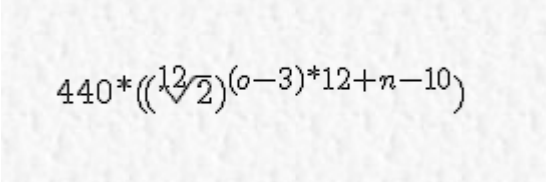
Leonardo González Guerra | A01721434

8 de septiembre de 2022

Empezaremos por definir las librerías que ocuparemos de Python para que nos ayuden a generar los sonidos que queremos. Usaremos la librería re para leer las expresiones regulares, la librería numpy para que nos ayude en las fórmulas matemáticas y la librería sound device que nos ayudará a generar los sonidos.

```
import re
import numpy as np
import sounddevice as sd
```

Con estas librerías podremos pasar a nuestro segundo paso el cual será definir nuestra fórmula para obtener la frecuencia y onda para generar el sonido. Dándole un modelo matemático de una onda sonora va poder reproducirla y transformarla en una onda acústica. La fórmula de la frecuencia es la cantidad de veces que se produce un suceso en determinada cantidad de tiempo. Como hablamos de ondas, será la cantidad de oscilaciones que van a producirse en un segundo que se mide en hertz y mientras más oscilaciones hay más aguda será la nota y al contrario mientras menos oscilaciones tenga por segundo nuestra nota será más grave.



```
def get_freq(note, octave):
    expo = (octave - 4) * 12 + (note - 10)
    return 440 * ((2 ** (1/12)) ** expo)
```

Después queda modelar nuestra onda la cual es la amplitud de la frecuencia. Pero necesitamos pasarle la amplitud a distintos instantes para que vaya formando la onda. Si partimos un segundo en dos, y calcularemos la altura de la onda cada medio segundo y unimos los puntos, obtendremos una onda deforme, lo que menos parecería es una onda. Pero podemos seguir dividiendo un segundo en 4, 8, 50, 100 y así hasta obtener mejores resultados.

Por eso podemos usar como framerate o frecuencia de muestreo 44100 hz, o 48000 hz si ustedes lo prefieren, que son convenciones que se utilizan en audio y son inputs que admite sounddevice

```
framerate = 44100
```

Ahora queda calcular las muestras para el total de la duración de la nota. Suponiendo que dicha duración es un segundo vamos a partir ese segundo en 44100 (hz) partes iguales y como nuestro segundo se expresa en milisegundo. 1 segundo = 1000 milisegundos. Esto nos devolverá un arreglo con 44100 valores por segundo el cual debemos despues pasar la fórmula de la onda.

$$y(t) = A\,\mathrm{sen}(2\pi f t + \varphi)$$

```
time = 1000

def get_wave(note, octave, duration):
    framerate = 44100
    freq = get_freq(note, octave)
    t = np.linspace(0, duration/time, int(framerate*duration/time))
    wave = np.sin(2 * np.pi * freq * t)
```

Para reproducir nuestra onda solo queda usar nuestra librería sounddevice para obtener el sonido acústico.

- La onda que ya modelamos
- La frecuencia de muestreo

```
sd.play(data, frameRate)
sd.wait()
```

El lenguaje MUTRAN utiliza una sintaxis muy sencilla y fácil de leer aunque no tengas mucha experiencia leyendo música utilizas nada mas lo básico para poder generar un arreglo de notas musicales la cual será después interpretado por nuestro código, lexemas, fórmulas y funciones.

```
#Ejemplo MUTRAN que leera la escala musical

#DO   RE   MI FA  SOL  LA   SI
#C C# D D# E F F# G G# A A# B

instrumento C3h C#3h D3h D#3h E3h F3h F#3h G3h G#3h A3h A#3h B3h
```

En este ejemplo tenemos un archivo .txt el cual será leído por nuestro código en python que interpretará de la siguiente forma el lenguaje MUTRAN:

- Los comentarios empiezan con # y serán ignorados ya que son para que un usuario sepa interpretar y no una máquina.
- El “instrumento” colocado antes de la escala es el sonido que escogerá para interpretar.
- Empezamos con la escala musical y empezamos con el Do el cual es también llamado C. Tenemos nuestra escala musical la cual es C, D, E, F, G, A, B pero en esta ocasión usamos sostenidos por eso es C, C#, D, D#, E, F, F#, G, G#, A, A#, B.
- El 3 que le sigue es en la octava que estará del piano la cual es la posición que estuviéramos en un piano. Mientras más grave queramos el sonido tendremos que bajar el número y si lo queremos más agudo obviamente tendremos que subir el número.
- El “w”, “h”, “q”, “e”, “s”, “t” y “f” al final de cada nota es el tiempo que durará la nota. Ya sea una blanca, negra, corchea etc. Esto nos regresará un valor del 16.6 al 1000 que es el máximo tiempo que pusimos como variable el cual es un segundo.

Con esto ya aprendido hay que ver como nuestro código leerá este léxico. Deberemos hacer funciones dentro de nuestro código para que puedan leer nuestros archivos .txt y puedan reconocer según la nota que muestra el archivo el valor que regresaran para después usar dentro de nuestras fórmulas seguidas por el modelo matemático de la frecuencia en una onda. Para la función que leerá nuestra nota y nos regresara el sonido de la nota en cuestión será la siguiente.

```
def get_letter(note):
    if note == "A":
        return 10
    elif note == "B":
        return 12
    elif note == "C":
        return 1
    elif note == "D":
        return 3
    elif note == "E":
        return 5
    elif note == "F":
        return 6
    elif note == "G":
        return 8
    return 0
```

Después viene reconocer en qué octava tenemos posicionadas nuestras notas musicales. Tendremos un rango de 0 a 8 siendo 0 el más grave y 8 el más agudo.

```
def get_octave(octave):
    if octave == "0":
        return 0
    elif octave == "1":
        return 1
    elif octave == "2":
        return 2
    elif octave == "3":
        return 3
    elif octave == "4":
        return 4
    elif octave == "5":
        return 5
    elif octave == "6":
        return 6
    elif octave == "7":
        return 7
```

```
    elif octave == "8":
        return 8
    return 0
```

Para nuestros sostenido o bemoles definiremos una pequeña función que llamaremos accidental la cual nos regresara el valor indicado según la nota a la cual está concatenada.

```
def get_accidental(accidental):
    if accidental == "#":
        return 1
    elif accidental == "b":
        return -1
    return 0
```

En el caso de nuestro tempo tendremos que diferenciar nuestra letra minúscula después de nuestra nota para ser interpretada como cierto valor numérico que será nuestra x en esta fórmula 1000/x.

```
def get_temp(value):
    if value == "w":
        return 1000
    elif value == "h":
        return 500
    elif value == "q":
        return 250
    elif value == "e":
        return 125
    elif value == "s":
        return 62.5
    elif value == "t":
        return 33.3
    elif value == "f":
        return 16.6
    return 0

def get_modtemp(valueTemp):
    if valueTemp == "t":
```

```
    return 2/3
elif valueTemp == "3":
    return 2/3
elif valueTemp == "5":
    return 4/5
elif valueTemp == "7":
    return 6/7
elif valueTemp == "9":
    return 8/9
elif valueTemp == ".":
    return .25
elif valueTemp == "tt" or valueTemp == "33":
    return 4/9
return 1
```