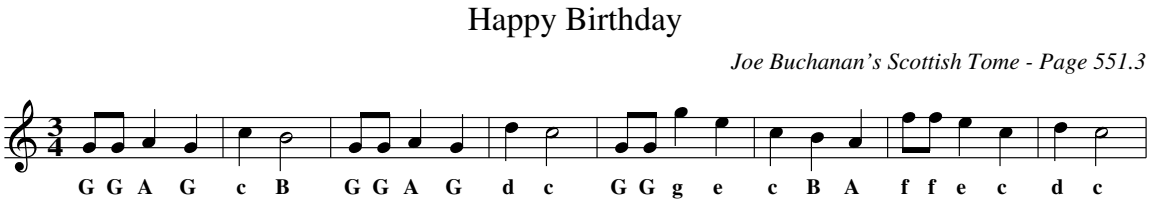
Informática Musical

Procesamiento de audio digital: NumPy, SoundDevice, SciPy.

Los ejercicios marcados con **(Evaluable)** son prácticas de laboratorio evaluables. Deben seleccionarse **al menos 2**, desarrollarlos y subirlos al CV. Todos los archivos entregados tienen que contener el nombre y los apellidos de los miembros del grupo (uno por línea).

En los siguientes ejercicios utilizaremos las librerías de Python estudiadas en clase, así como la clase *KBHit* (basada en *pygame*) para captura no bloqueante de teclado (explicada en los notebooks de clase).

1. **(Entregable)** Consideremos la siguiente partitura:



   Podemos hacer una representación de la secuencia de notas mediante una lista (Python) de pares *(nota,duración)*. Las notas se representan con las letras `A B ...G`; para subir una octava se utilizan las minúsculas `a b ...g`[1]. La duración se representa con un número que indica las *unidades de tiempo* (la duración de una unidad se puede fijar en el programa). De este modo, la partitura anterior quedaría:

$$[(G,0.5),(G,0.5),(A,1),(G,1),(c,1),(B,2),\ldots]$$

   La tabla de frecuencias (para una octava) es la siguiente:

C	D	E	F	G	A	B	c	d . . . g
523,251	587,33	659,255	698,456	783,991	880	987,767	. . .	. . .

   Recordemos que dada la frecuencia de una nota, la de la octava superior se obtiene duplicando dicha frecuencia (por ejemplo, la frecuencia de `c` es $523{,}251 * 2$) y la de la octava inferior, dividiendo entre 2.

   Para reproducir la partitura podría utilizarse el oscilador *osc(nota,volumen,duración)* visto en clase, pero aquí haremos algo más sofisticado. Utilizamos la clase `Osc` extendida con métodos `noteOn` y `noteOff` que permiten activar/desactivar el oscilador. Hay que implementar un pequeño `scheculer` que lance y pare las notas en los instantes temporales oportunos.

   Puede extenderse la implementación para leer de archivo la partitura en notación ABC (Wikipedia, entrada "Notación musical ABC").

2. Implementar una función para *remuestrear* un audio dado con un frame rate a otro especificado, sin alterar el pitch. Esta función servirá para cambiar la frecuencia de muestreo (sample rate). Investigar formas de interpolación para hacerlo.

3. **(Entregable)** Implementar un piano simple utilizando el sample proporcionado *piano.wav*. Utilizaremos KBHit para mapear las teclas *zxc...* a una octava del piano y *qwe...* a la octava superior. A partir de muestra de una nota, por ejemplo un C, el resto de notas pueden obtenerse variando la velocidad de reproducción de esa muestra de acuerdo a estas proporciones:

Nota	C	D	E	F	G	A	B	c	d . . . a
Frec.	1	9/8	5/4	4/3	3/2	5/3	15/8	2	. . .

[1] Para la siguiente octava por arriba podrían utilizarse los apostrofes (`a'`), para la siguiente doble apostrofe (`a''`), etc. Para bajar octava se podrían utilizar las comas (`A, A,,`)

Es decir, para obtener un D (re) podemos reproducir a 1.12 de velocidad. Esto altera el pitch de la muestra en la proporción justa para tener el D.

Tal como hemos visto en clase, dado un array, si generamos otro que contenga solo los samples en posición par (de la mitad de tamaño), obtenemos un sonido una octava por arriba. Y al contrario, si generamos un array del doble de tamaño añadiendo muestras intermedias (como media de la anterior y la superior de una dada) obtenemos una octava menos. Esta idea de interpolación puede generalizarse para obtener cualquier nota, utilizando la tabla de pitch dados. Esta operación de *resampling* puede hacerse con la función `resample` de la librería *SciPy*.

4. **(Entregable)** Implementar un oscilador *chirp(frecIni,frecFin,dur)* que genere una señal sinusoidal de duración *dur*, que comience con una frecuencia *frecIni* y la incremente gradualmente (de manera lineal) hasta alcanzar *frecFin*. Este tipo de osciladores es conocido y utilizado en distintas aplicaciones (`https://en.wikipedia.org/wiki/Chirp`). Para implementarlo se puede partir del oscilador básico visto en clase y modificarlo para variar el parámetro de frecuencia en el argumento de la función *sin*.

5. **(Entregable)** Utilizar el filtro IIR visto en clase para implementar un filtro paso banda (BP) con frecuencia de corte y ancho de banda configurables. Utilizar dos filtros LP y HP en secuencia, calculando los valores de α según se ha explicado en clase.