plot_audio(*sample_rate, impulse_response, title = ‘’*)
Given *sample_rate* and *impulse_response* as returned by **wavfile.read**( ), this function generates a plot of amplitude over time and a spectrogram of amplitude at each frequency over time.* The plot title can be changed through string variable *title*.

perform_FFT(*sample_rate, impulse_response, num_partials = 10, threshold = 0.05, start = 2000*)
Given *sample_rate* and *impulse_response* as returned by **wavfile.read**( ), this function performs fast Fourier transform (FFT) on audio data and extracts prominent frequencies.

num_partials is an integer that specifies the number of partials selected to characterize the sound in FFT. *start* is a floating point number that determines the time in the audio file at which the FFT starts; altering this variable can help circumvent the interference of the striking sound of a percussion instrument in a recording. *threshold* is a floating point number that sets the minimum amplitude at which a frequency is deemed as non-noise and capable of being selected.

This function returns a list of Numpy n-dimensional arrays (ndarray), respectively of all extracted frequencies (*freqs*), their corresponding amplitudes (amp), *num_partials* of the most prominent frequencies sorted by amplitudes (*peak_freqs*), and the corresponding amplitudes of prominent frequencies sorted in decreasing order (*peak_amp*).

plot_FFT(*freqs, amp, peak_freqs, peak_amp, scale = ‘linear’, title = ‘’, x_lim = 6000*)
Given *freqs*, *amp*, *peak_freqs*, and *peak_amp* as returned by **performFFT**( ), this function plots the frequencies extracted by fast Fourier transform. Prominent frequencies generated from **performFFT**( ) are marked by crosses. By default, the y-axis is scaled linearly; this could be changed through *scale* in the parameters. A logarithmic scale is more representative of perception of the human ear. The title and limit of the x-axis of the figure can be changed through variables *title* and *x_lim*.

prom_freq(*peak_freqs, peak_amp*)
Given *peak_freqs* and *peak_amp* as returned by **performFFT**( ), this function finds the most prominent partial (often the fundamental frequency) of the audio input. A floating point number is returned.

diss_measure(*peak_freqs, peak_amp, high_ratio = 4, title = ‘’, show_ratios = True*)
Given *peak_freqs* and *peak_amp* as returned by **performFFT**( ), this function calculates the dissonance at each frequency interval, finds the local minima, and plots the dissonance curve for the given audio input. It returns a list consisting of two Numpy n-dimensional arrays (ndarray), respectively of frequency ratios at local minima on the dissonance curve (*ratios*) and their corresponding sensory dissonance values (*dissonances*).

high_ratio is an optional floating point number that specifies the highest frequency ratio to which the dissonance curve is generated. The title of the figure can be changed through string variable *title*. show_ratios is a Boolean value; if set to True, ratios at each local minima are displayed on the graph.

write_file(*peak_freqs, peak_amp, ratios, dissonances, filename, savepath*)
Given *peak_freqs* and *peak_amp* as returned by **performFFT**( ), and *ratios* and *dissonances* as returned by **diss_measure**( ), this function creates and writes in a .txt file that is saved as *filename*.txt in the designated *savepath*. This text file is generated such that it is compatible with the Max patch interface.†

write_file_direct(*sample_rate, impulse_response, filename, savepath, num_partials = 10, threshold = 0.05, start = 2000, high_ratio = 4*)
This function serves a similar purpose as **write_file**( ), which creates and writes in a .txt file that is saved as *filename*.txt in the designated *savepath*. However, unlike **write_file**( ), this function generates a file directly given *sample_rate* and *impulse_response* as returned by **wavfile.read**( ). The .txt file is saved as *filename*.txt in the designated *savepath*.

num_partials is an integer that specifies the number of partials selected to characterize the sound in FFT. *start* is a floating point number that determines the time in the audio file at which the FFT starts. *threshold* is a floating point number that sets the minimum amplitude at which a frequency is deemed as non-noise and capable of being selected. *high_ratio* is an optional floating point number that specifies the highest frequency ratio to which the dissonance curve is generated.