for start we define some function and initializing some variables and arrays witch we need in future:

notes\_base: This line creates an array of base frequencies for different musical notes. It uses NumPy's arange function to generate an array of numbers from 0 to 11, which are divided by 12 and raised to the power of 2. These values are then multiplied by 27.5 to obtain the frequencies for each note.

notes\_duration: This line creates an array representing the duration of each note in milliseconds. It uses NumPy's array function to create an array with specific values, which are then multiplied by 0.7.

notes\_ann: This line creates a list of note names. Each element corresponds to a note in the notes\_base array.

sin\_wave function: This function generates a sinusoidal waveform. It takes three parameters: the frequency of the waveform (f), the number of samples (n), and the sample rate (fs). It first creates a linearly spaced array x from 0 to 2π, representing the time axis. It then creates another linearly spaced array xp from 0 to -1\*(n\*ring/fs), where ring is set to 30. The function generates a sinusoidal waveform y by taking the sine of the product of x, f, and n/fs, and multiplies it by the exponential of xp. Finally, it creates a 2-dimensional array z with shape (n, 2) and assigns y to both columns of z. The function returns z.

play\_note function: This function plays a single note. It takes four parameters: the note ID (note\_id), the octave (octave), the duration (dur), and the sample rate (fs). If the note\_id is less than 3, it increments the octave by 1. It then calls the sin\_wave function to generate the waveform y for the specified note and duration. It uses sd.play from the sounddevice library to play the waveform y at the specified sample rate. Finally, it uses sd.wait() to wait until the playback is finished and returns.

put\_note function: This function is similar to play\_note, but instead of playing the note, it generates and returns the waveform y for the specified note, octave, duration, and sample rate.

get\_music function: This function takes a list of music notes (music\_notes) and the sample rate (fs). It iterates over each item in music\_notes and calls the put\_note function to generate the waveform y for each note. The generated waveforms are concatenated using NumPy's concatenate function and returned as a single waveform m.

fs1: This variable stores the sample rate of the audio, set to 44100.

music: This list represents a musical composition. Each item in the list corresponds to a note and contains three elements: the note ID, the octave, and the duration.

y: This line calls the get\_music function with the music list and the sample rate fs1. It generates the waveform y for the musical composition.

sd.play(y, fs1): This line plays the waveform y using the sounddevice library at the sample rate fs1.

Overall, this code defines functions to generate and play sinusoidal waveforms for different musical notes and durations. It then composes a piece of music using the defined functions and plays it using the sounddevice library.

\begin{lstlisting}[style=mystyle]

# This is a Python comment

print("Hello, world!")

\end{lstlisting}

\lstdefinestyle{mystyle}{

language=Python,

basicstyle=\ttfamily,

keywordstyle=\color{blue},

commentstyle=\color{gray},

stringstyle=\color{green},

showstringspaces=false,

numbers=left,

numberstyle=\tiny,

breaklines=true,

breakatwhitespace=true,

frame=single,

captionpos=b,

caption={Example Python code},

}