Task 3 – Musical Instruments Emphasizer

Introduction: Music industry is one of the major application fields of digital signal processing. Songs consist of music and vocal, and both of them are just mixture of different frequencies. Manipulating different frequency ranges in a signal is a broad family of functionality called "Signal Equalizer". While Ssund equalizer is a basic tool in the music industry, it also serves in several biomedical applications like hearing aid industry

Description: Develop and application for musical instruments emphasizer where:

- The user can open any music file, and play/pause it as a signal (i.e. in a graph) while hearing it. i.e. your application should send the signal to the sound output to be heard.
- The user can control the volume of the music from inside the program (you need to add the proper ui element for this functionality).
- The user can see the song spectrogram while running the signal (similar to task 1).
- Do your reading on the frequency ranges of the different musical instruments (some resources are below) and prepare your signal equalizer where the user can reduce/null/increase the contribution of any of the instruments in the song or music file.
  Some quick arbitrary resources from google search are below:
  - https://www.masteringthemix.com/blogs/learn/understanding-the-different-frequency-ranges
  - https://onlinelibrary.wiley.com/doi/pdf/10.1002/9781119164746.app2
  - http://virtualplaying.com/interactive-frequency-chart
  - https://blog.landr.com/eq-cheat-sheet
- For each instrument, there should be a slider that represent the gain of this specific instrument in the output volume. The slider's default value is 1 (i.e., its default weight in the played file) and the user can move it up or down (from 0 to 10) to increase or decrease the weight of this instrument without affecting the other instruments. This should reflect on the played sound, the signal in your graph, and the displayed histogram.
- Beneath the UI elements, each slider in connected to a certain frequency range. Upon each action, the signal should go to Fourier domain, weight the instrument with the slider gain and then return back to the time domain. Note that you need to do this for all instruments (i.e., frequency ranges). Remember the "*No code repetition*" rule!
- The control of each instrument should be through UI element(s) that show a visual for this instrument (e.g. an image, cartoon figure, etc). i.e., the user can see which instrument s/he is controlling. You will need to think a little about how such UI will look (i.e., with such images and slides, sound player and histogram).
- Using the same information collected from your research above, and in a separate tab in your application, simulate the output of three musical instruments. Pick any three instruments and create a nice visual for them to help your user play this instrument through your application. For each instrument, the application should give the user some of the instrument's basic options/settings through some UI elements. Please, check www.virtualmusicalinstruments.com to get an idea and imagine the experience.
- Put an effort to make an attractive application not just a dull one! Use your extensive software experience as a user to produce your own nice user-friendly software.

Code practice:

- Same practices from Task 2 (i.e., proper variable names & No code repetition) will continue with task 3.
- Logging: Logging is a very important concept you should get used to. It helps you to track how problems happen and figure out their resolution much quicker. Python has a logging library. All you need to do is to import it and start logging the main user interactions and main steps into some text file. I need to see how logging has helped you to debug the problems you faced during your development. i.e. Do NOT just throw any trash variables to the log file to show you did it. You will be asked why you logged this variable or that.