Report 3 - Automashup

Michael Moreno

31/07/2024

Advancements

We will take a closer look at each file and what changed since the beginning of my internship and what changes were made.

First let's take a look into the main code of the app.

# app.py



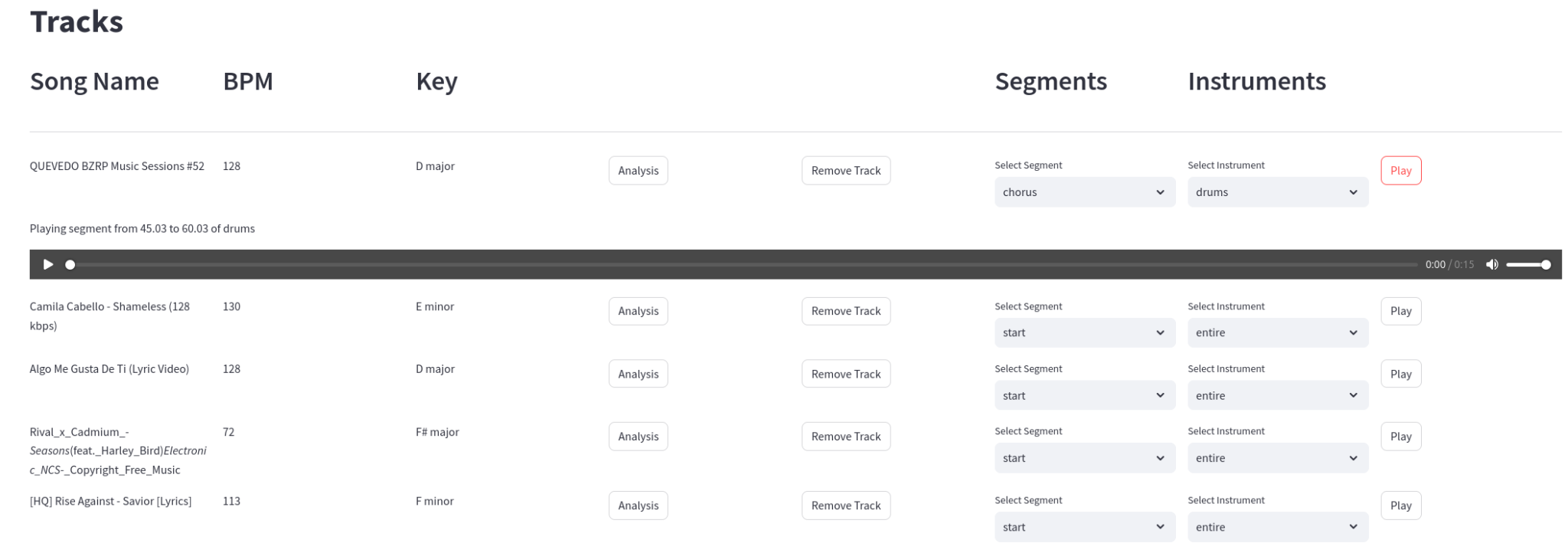
* io: We added io so we could temporarily hold audio data in memory as a binary stream. Then we could download this binary stream without having to create temporary files.
* pickle: handle the serialization and deserialization of object data to be stored and retrieved from disk. We use it to delete the corresponding entry from the schemas' dictionary.
* soundfile: writes the mashup audio data into the audio\_bytes stream in the WAV format without creating a file.
* barfi\_schemas: we profit from this functionality from barfi where we can save the schemas.

## 

## Functionality

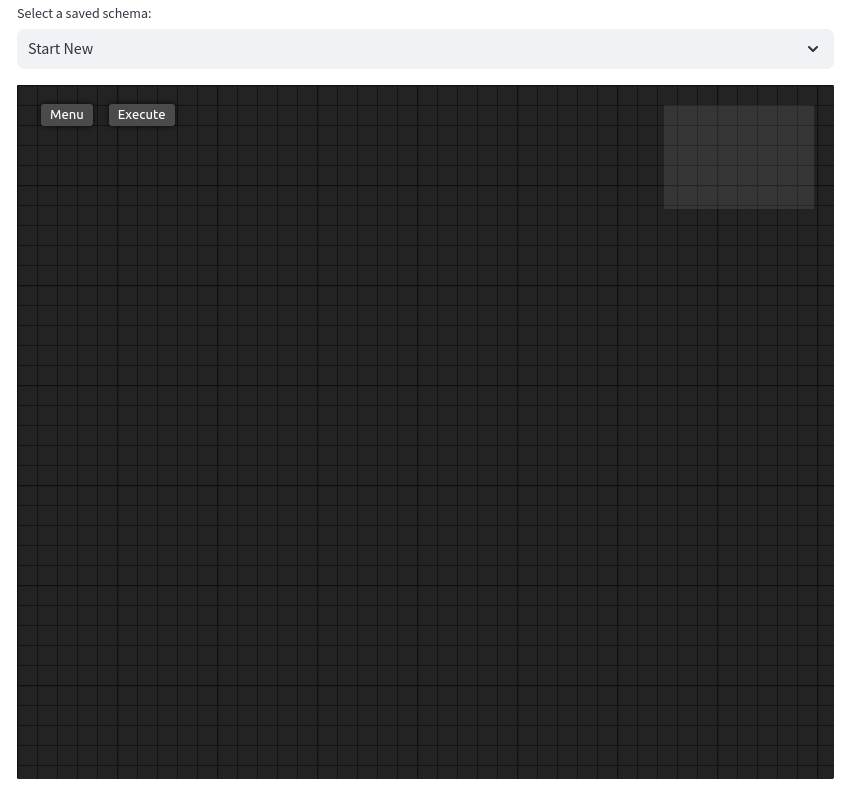
### Segment preview

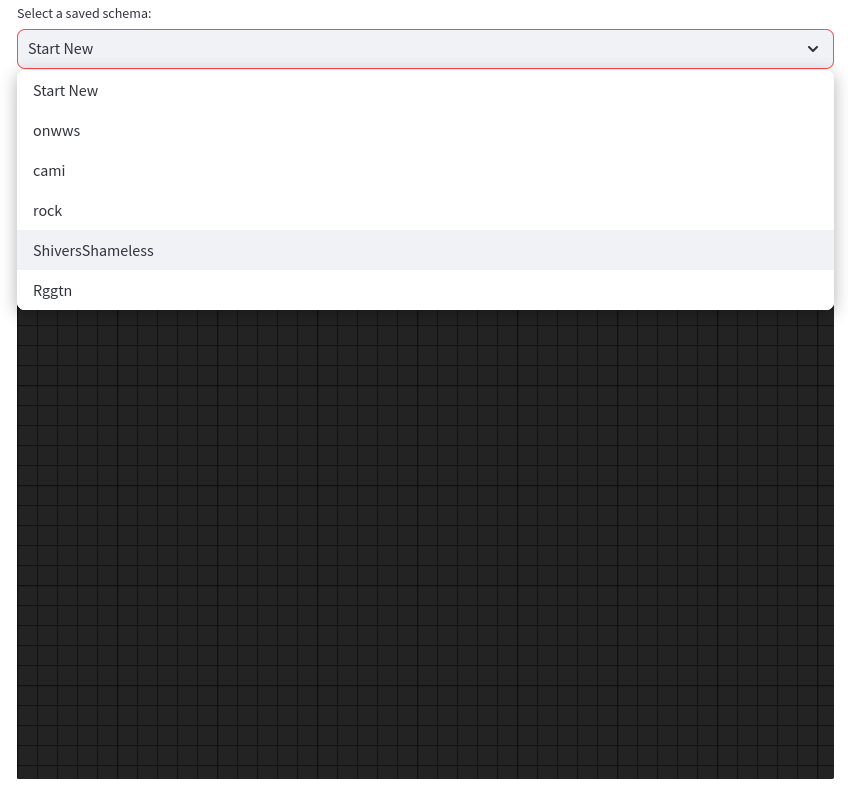
We created more functionality on the interface. Now after uploading songs, you may listen to each segment from it. You may also choose any instrument separately (vocals, bass, drums, other sounds or all together).

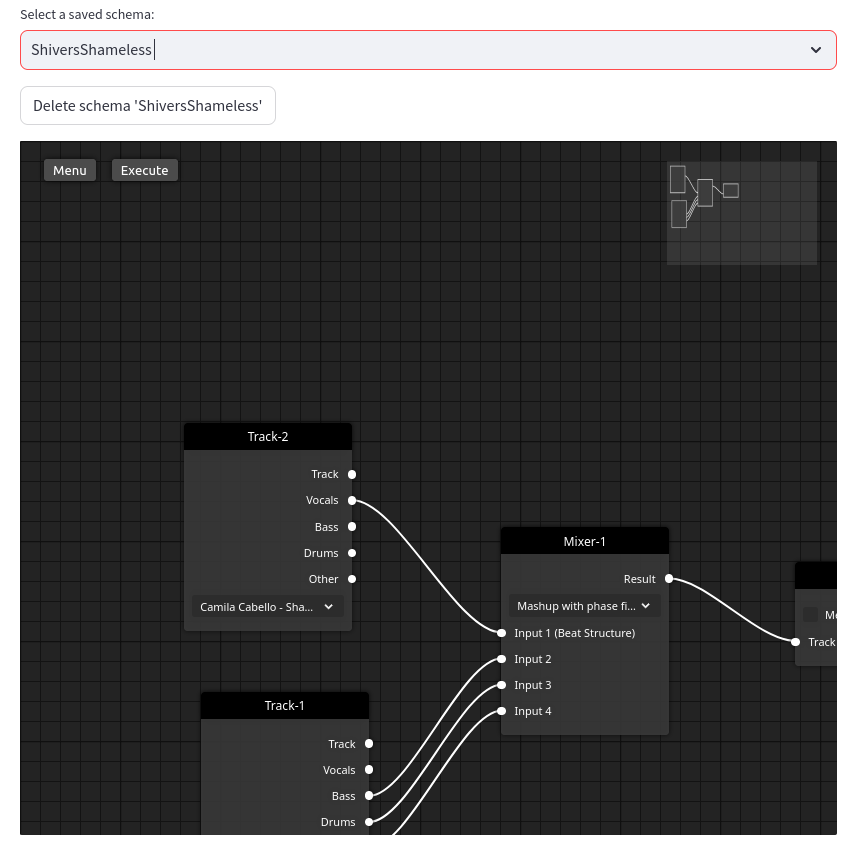


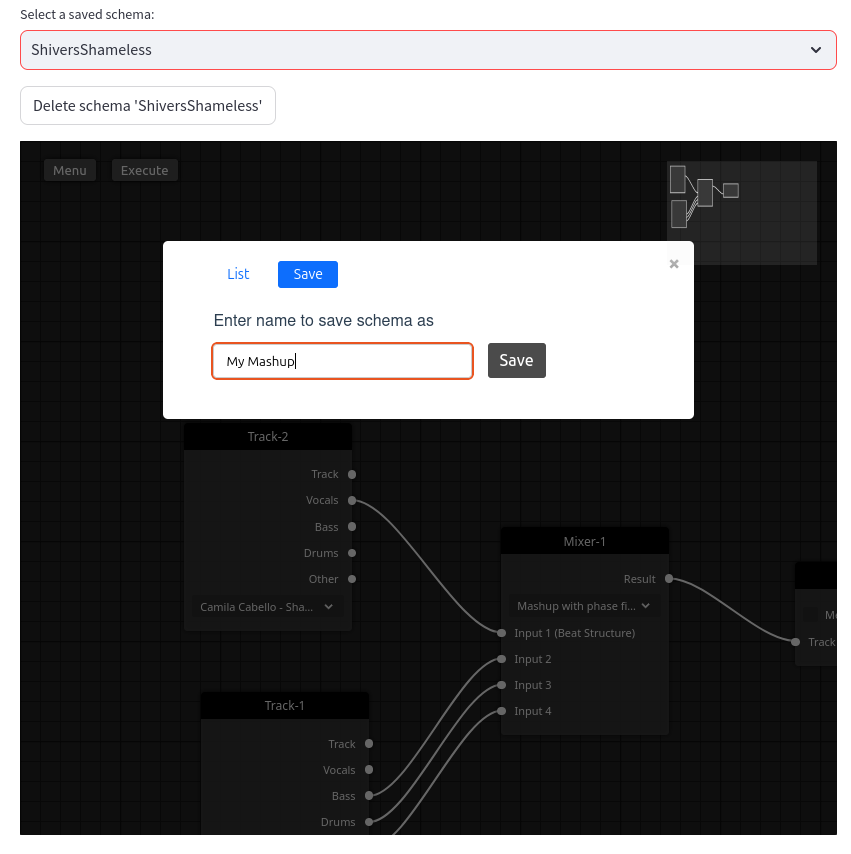
### Save and load

Now you can save and upload your schemas so you don’t have to redo them each time.









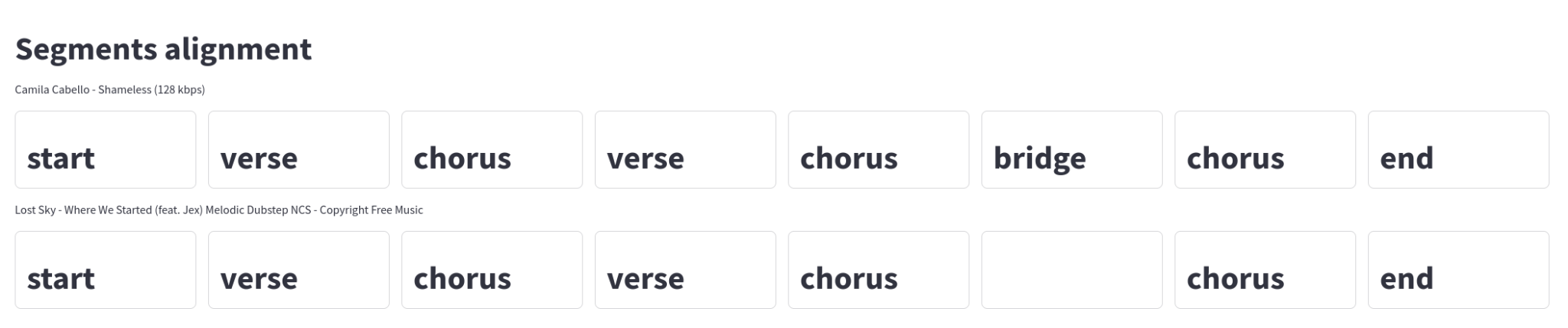
### Download results

After creating your own mashup, you may download it right away!



### Segment summary

Whenever the result mashup uses the phase fit technique, a segment summary will show up after the mashup. This way we can take a closer look at possible silences when the complementary songs don’t have a segment the main track does.

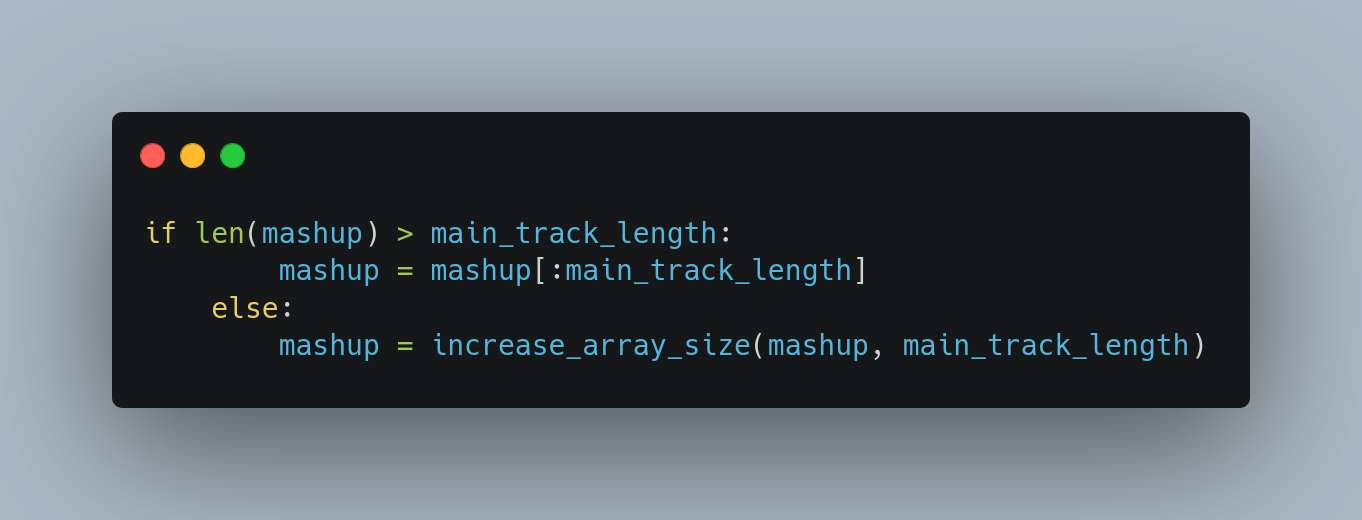


# mashup.py

The main changes rely on the organization of the functions. In case there is no phase fit as a mashup technique, the time stretch will be done to the whole song.

  
In case there is repitch as a mashup technique, we do it first in any case. There is no specific reason for this.



In case there is phase fit as a mashup technique, the time stretch will be done for each segment depending on the main track’s segment bpm. The order will , if selected, repitch, then phase fit, and finally we make sure that the mashup is as long as the main song.

# Track.py

Reconstructed function ‘fit\_phase’.

* Now segments get aligned in order of appearance. For example, if there are multiple chorus segments present on the songs, we align the first two, then we align the second two. In case of having more on the main song than on the other songs, we take the last one found.
* Recalculate the BPM when no segment is found. This was an issue before since we were having shorter or longer silence segments when no match was found.
* New arguments for the function ‘get\_audio\_beat\_fitted’. Now we send the target segment’s bpm and duration.
* We reset the first beat position per segment so all start at the same time.
* Created functions to recover all segments of a Track. This is used on app.py

# Segment.py

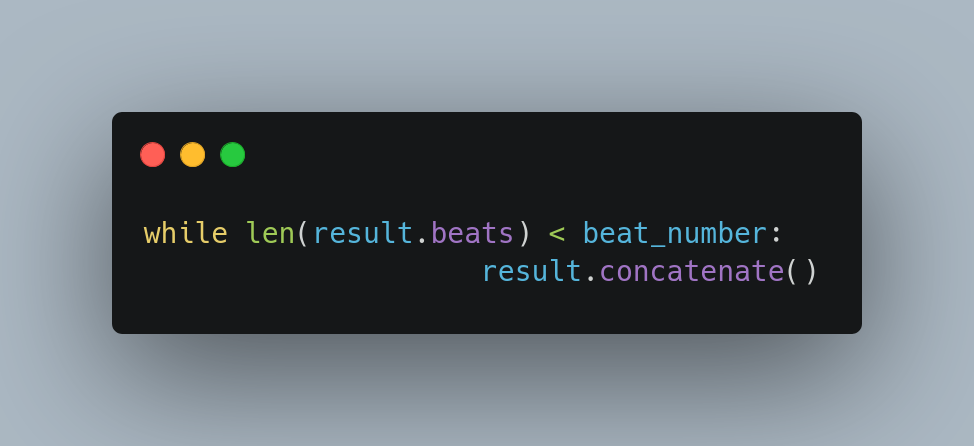
* Included new attribute to the class Segment; duration. This saves the duration of a segment in minutes.
* Modified function ‘concatenate’. It was redundant.
* Now the order for the ‘get\_audio\_beat\_fittet’ function is the next one
  + Compare bpm
  + In case of a significant difference, half or double the target bpm. This is proportional for the amount of beats.



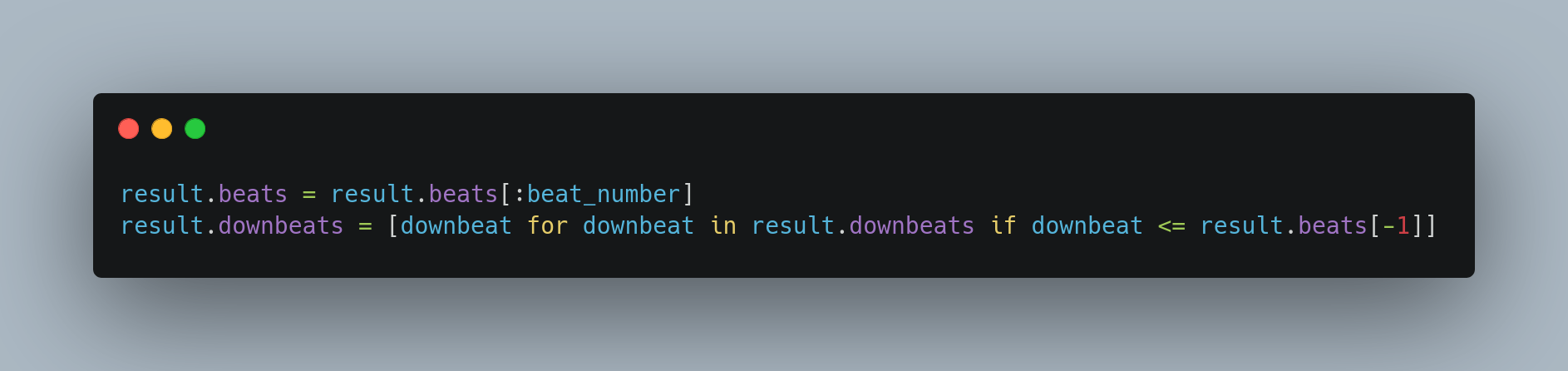
* + Equalize both bpm with ‘time\_stretch’.



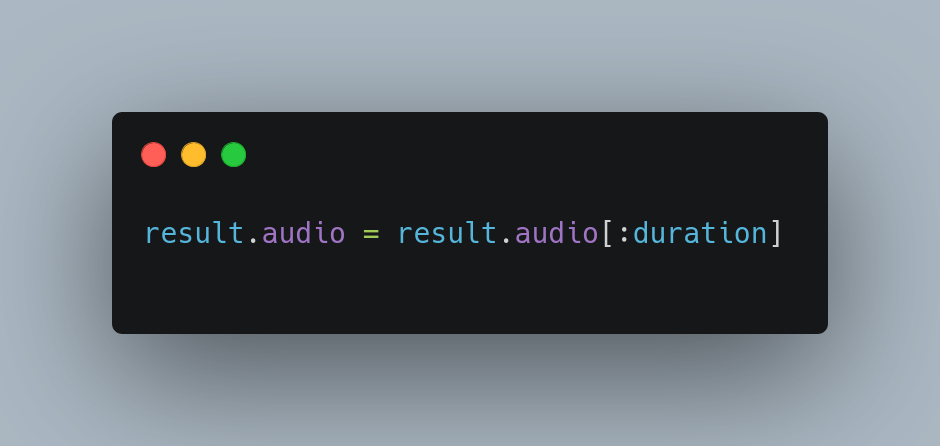
* + Concatenate the segment in case the number of beats is lower than the target segment. This will double the amount of beats of the current segment.



* + Cut the amount of beats and downbeats to have the same amount of beats as the target track



* + Change the audio duration to the same as the target segment’s duration.



# Utils.py

Added the function ‘get\_unique\_ordered\_list’ so the list of segments will show in order of appearance at the dropdown menu on the segment preview interface.

Added the function ‘get\_segment\_times’ for QoL. It returns a segment beginning and ending moment.

Added the function ‘extract\_audio\_segment’ which allows us to get the audio of just the segment to play it on the segments preview

Added the function ‘merge\_segments’ which merge the segments with the same label that are one after the other. It made no sense having them separated.