How to extract the raw data from a mp3 file using python?

I have got homework regarding audio data analysis using Python. I wonder is there any good module for me to use to extract the raw data from a mp3 file. I mean the raw data, not the metadata, id3 tags.

I know how to use the `wave` module to process `.wav` files. I can `readframes` to get the raw data. But I don't know how to do with mp3. I have searched a lot on google and stackoverflow and find `eyeD3` . But unfortunately the documentation is rather frustrating and now the version is 0.7.1, different from most examples I can find on the Internet.

Is there any good module that can extract raw data from a mp3? If there is any good documentation for `eyeD3` , it is also good.

python audio mp3 eyed3

asked May 19 '13 at 11:22

zhangyangyu

6,261 1 18 35

Check this: stackoverflow.com/questions/3049572/… . Apparently, the easiest would be to convert mp3 to wav using an external program – Jakub M. May 19 '13 at 11:25

2 The phrase "raw data" is very confusing. If you say raw data i think you want to get the bytes of the file. (which you get with `open('your.mp3', 'rb')`) But i think you don't want this kind of raw data. – Kritzefitz May 19 '13 at 16:35

I want the kind of raw data - bytes of the file. But no all bytes of the file are the contents of the music. There are still some tags and maybe somethhing others. So I wonder if there is any module can extract it.@IchUndNichtDu – zhangyangyu May 20 '13 at 5:19

3 Answers

If I understand your question, you can try using pydub (a library I wrote) to get the audio data like so:

```
from pydub import AudioSegment

sound = AudioSegment.from_mp3("test.mp3")

# sound._data is a bytestring
raw_data = sound._data
```

edited Sep 23 '13 at 16:07

answered Sep 16 '13 at 14:24

Jiaaro

37.3k 30 127 159

1 Thanks, excellent module. – zhangyangyu Sep 19 '13 at 1:41

Have you tried opening the file in read binary mode?

```
f = open("test.mp3", "rb")
first16bytes = f.read(16)
etc...
```

answered Jul 11 '13 at 4:55

Stephan

5,848 3 22 52

I'm pretty sure the OP wants the audio data which is encoded in the mp3. In that case the mp3 will need to be decoded before it is read. – Jiaaro Sep 11 '13 at 16:41

@Jiaaro you should post an answer explaining that – Stephan Sep 14 '13 at 17:38

There are a few similar questions floating around stackoverflow. There are distinct use cases.

1. The user wants to convert .mp3 files to PCM files such as .wav files.
2. The user wants to access the raw data in the .mp3 file (that is, not treat it as compressed PCM). Here the use case is one of understanding how compression schemes like MP3 and AAC work.

This answer is aimed at the second of these, though I do not have working code to share or point to.

Compression schemes such as MP3 generally work in the frequency domain. As a simplified example, you could take a .wav file 1024 samples at a time, transform each block of 1024 samples using an FFT, and store that. Roughly speaking, the lossy compression then throws away information from the frequency domain so as to allow for smaller encodings.

A pure python implementation is highly impractical if all you want to do is convert from .mp3 to .wav. But if you want to explore how .mp3 and related schemes work, having something which you can easily tinker with, even if the code runs 1000 times slower than what ffmpeg uses, can actually be useful, especially if written in a way which allows the reader of the source code to see how .mp3 compression works. For example see http://bugra.github.io/work/notes/2014-07-12/discre-fourier-cosine-transform-dft-dct-image-compression/ for an IPython workbook that walks through how frequency domain transforms are used in image compression schemes like JPEG. Something like that for MP3 compression and similar would be useful for people learning about compression.

An .mp3 file is basically a sequence of MP3 frames, each of which has a header and data component. The first task then is to write a Python class (or classes) to represent these, and read them from an .mp3 file. First read the file in binary mode (that is, f = open(filename,"rb") and then data = f.read() -- on a modern machine, given that a typical 5min song in .mp3 is about 5MB, you may as well just read the whole thing in in one go).

It may also be worth writing a simpler (and far less efficient) coding scheme along these lines to explore how it works, gradually adding the tricks schemes like MP3 and AAC use as you go. For example, split a PCM input file into 1024 sample blocks, use an FFT or DCT or something, and back again, and see how you get your original data back. Then explore how you can throw data away from the frequency transformed version, and see what effect it has when transformed back to PCM data. Then end result will be very poor, at first, but by seeing the problems, and seeing what e.g. MP3 and AAC do, you can learn *why* these compression schemes do things the way they do.

In short, if your use case is a 'getting stuff done' one, you probably don't want to use Python. If, on the other hand, your use case is a 'learning how stuff gets done' one, that is different. (As a rough rule of thumb, what you could do with optimised assembly on a Pentium 100 from the 90s, you can do at roughly the same performance using Python on a modern Core i5 -- something like that -- there is a factor of 100 or so in raw performance, and a similar slowdown from using Python).

answered Jan 31 '16 at 14:59

John Allsup
190 1 4
