Python解析Wav文件并绘制波形的方法

#python #算法

前言

在现在繁忙的生活中，我们经常会听些歌来放松一下自己，我们经常会从各种播放软件中听自己喜欢的歌，并且往往我们会下载一部分歌曲，而现在音频的种类也相当繁多，像是Wav,Mp3,FLAC,AAC等等很多格式，最近由于需要做一个能够分析Wav格式音频的波形来取得一些数据比如获取人录音时是否说完等等用途.本周先对解析Wav并用Python绘制其波形进行了一些探索.

Wav文件格式

我们先来看看wikipedia上对于wav音频格式的解释：

> Waveform Audio File Format（WAVE，又或者是因为扩展名而被大众所知的WAV），是微软与IBM公司所开发在个人电脑存储音频流的编码格式，在Windows平台的应用软件受到广泛的支持，地位上类似于麦金塔电脑里的AIFF.此格式属于资源交换档案格式(RIFF)的应用之一，通常会将采用脉冲编码调制的音频资存储在区块中。也是其音乐发烧友中常用的指定规格之一.由于此音频格式未经过压缩，所以在音质方面不会出现失真的情况，但档案的体积因而在众多音频格式中较为大.

我们可以看到上面提到了两个个关键词RIFF和脉冲编码调制.所以我们接下来先解释一下RIFF「资源交换档案格式」是什么.

RIFF格式

我们同样的来看一下wikipedia对RIFF的解释

> Resource Interchange File Format（简称RIFF），资源交换文件格式，是一种按照标记区块存储数据（tagged chunks）的通用文件存储格式，多用于存储音频、视频等多媒体数据.Microsoft在windows下的AVI、ANI 、WAV等都是基于RIFF实现的.
>
> RIFF是由Microsoft和IBM于1991年，在windows 3.1中引入的，作为windows 3.1默认的多媒体文件格式.RIFF是参考Interchange File Format来的，二者主要的区别是字节序大端、小端的问题。在基于IBM的80x86系列主机下，RIFF的字节序是小端的；而在IFF原有的格式中是按照大端存储整型数据的.

RIFF是由chunk构成的，chunk是RIFF组成的基本单位，每个CHUNK可看作存贮了视频的一帧数

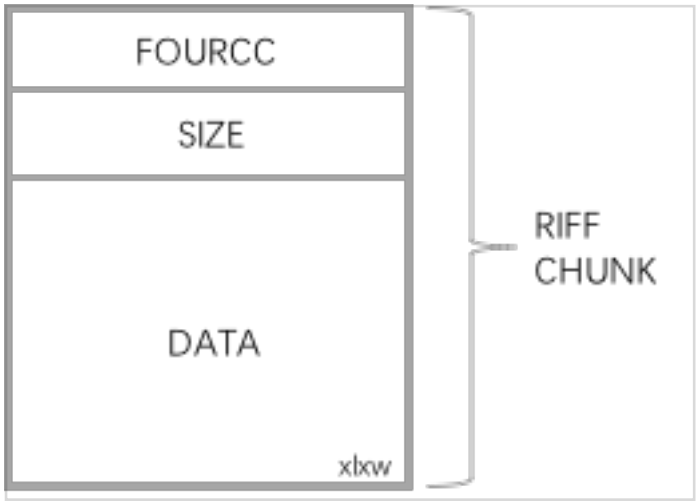
据或者是音频的一帧数据，所以下面我们来讨论一下chunk的结构是怎么样的.

CHUNK的结构

CHUNK总共由三个部分组成：

- FOURCC　使用4字节的ASIIC字符标识类型
- SIZE　　数据的大小
- DATA　　用于存放数据

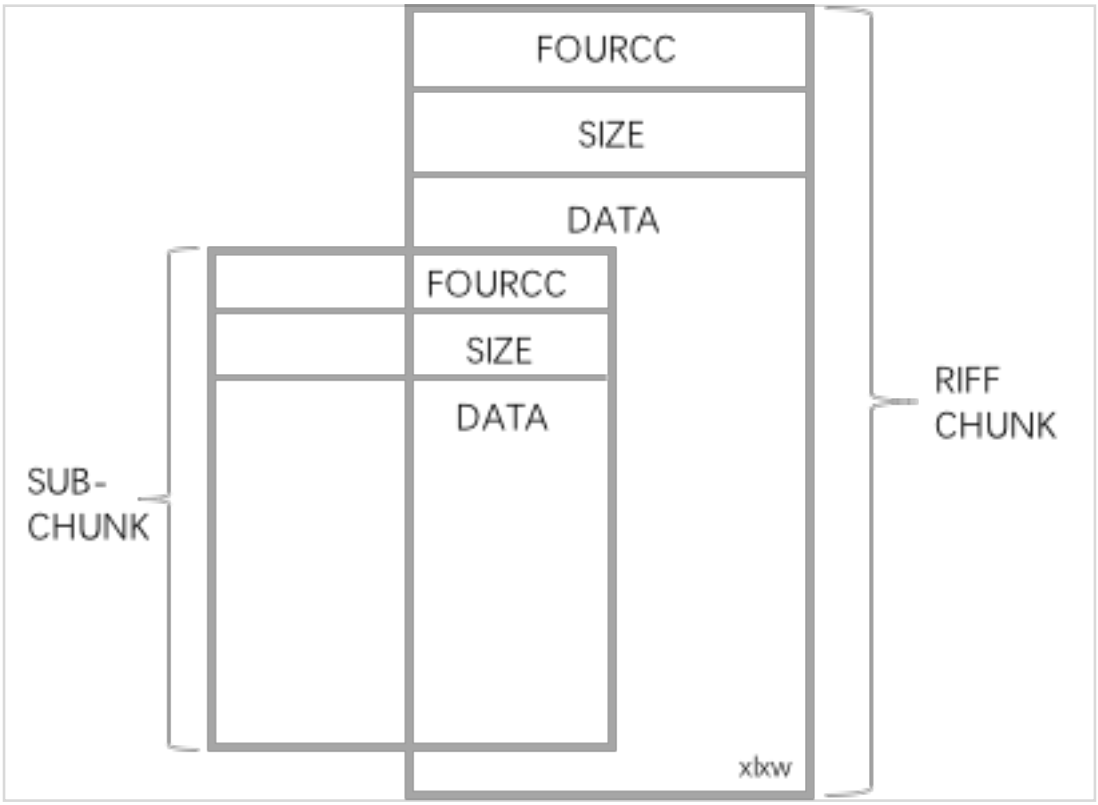
结构示意图如下：



CHUNK的结构

- CHUNK在一般情况下不能嵌套，但是当CHUNK的FOURCC为“RIFF”或者是“LIST”的时候可以嵌套数据.
- “RIFF”的第一个CHUNK的FOURCC一定是“RIFF”，所以LIST为FOURCC的一定是子CHUNK及SUBCHUNK.

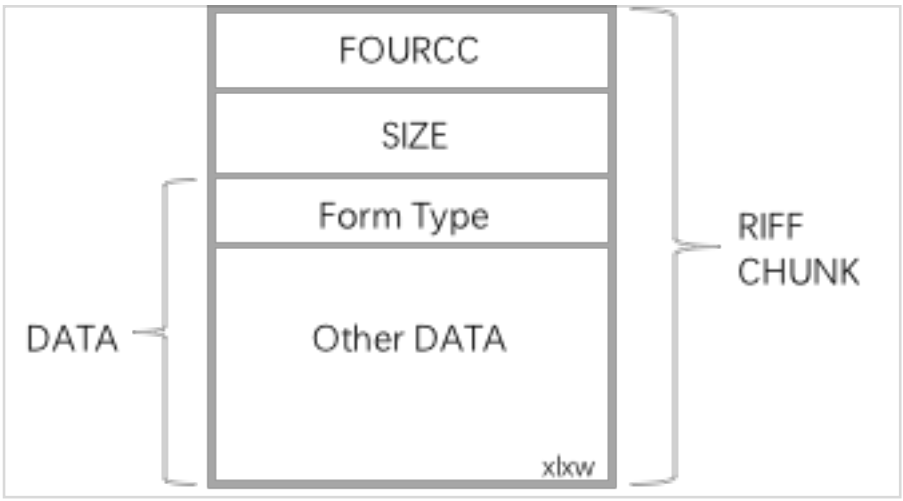
下面是一个包含了子CHUNK的结构示意图：



包含了SUBCHUNK的结构示意图

- “RIFF”的CHUNK在DATA区域的前四个字节称为“Form Type”记录了数据的类型，比如我们的wav文件的Form Type就是“WAV”

  Form Type结构示意图如下：



包含了Form Type的结构示意图

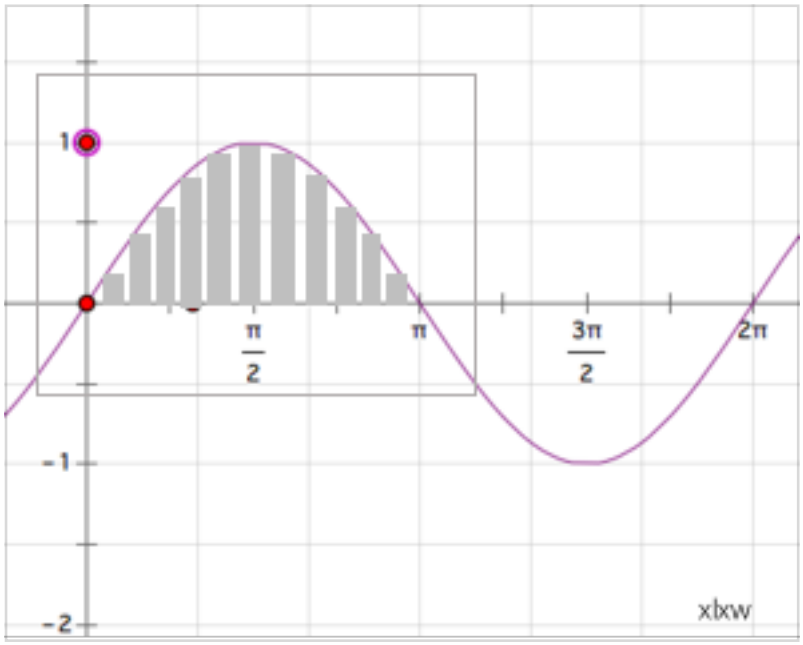
- 同样的FOURCC为LIST的 SUBCHUNK 的DATA区域也包含了LIST Type，用于表示LIST中数据区域格式

脉冲编码调制（PCM）

我们先来看看百度百科对它的解释：

> PCM 脉冲编码调制是Pulse Code Modulation的缩写，脉冲编码调制是数字通信的编码方式之一. 主要过程是将话音、图像等模拟信号每隔一定时间进行取样，使其离散化，同时将抽样值按分层

> 单位四舍五入取整量化，同时将抽样值按一组二进制码来表示抽样脉冲的幅值.
>
> 我们从上面的介绍可以理解为：
>
> 通过三个过程-抽样、量化和编码讲音频的模拟信号转化为数字信号.
>
> **抽样**
>
> 抽样是由于模拟信号是连续的，通过一定频率对模拟信号进行取样，近似得到，如下面的图中灰色框中就是取了一定频率进行抽样的示意图：



抽样过程示意图

- 抽样是要将模拟信号以其信号带宽2倍以上的频率提取样值，变为在时间轴上**离散**的抽样信号，就可获得能取代原来连续音频信号的抽样信号.对一个正弦信号进行抽样获得的抽样信号是一个脉冲幅度调制（PAM）信号，之后对抽样信号进行检波和平滑滤波，即可还原出原来的模拟信号.

量化

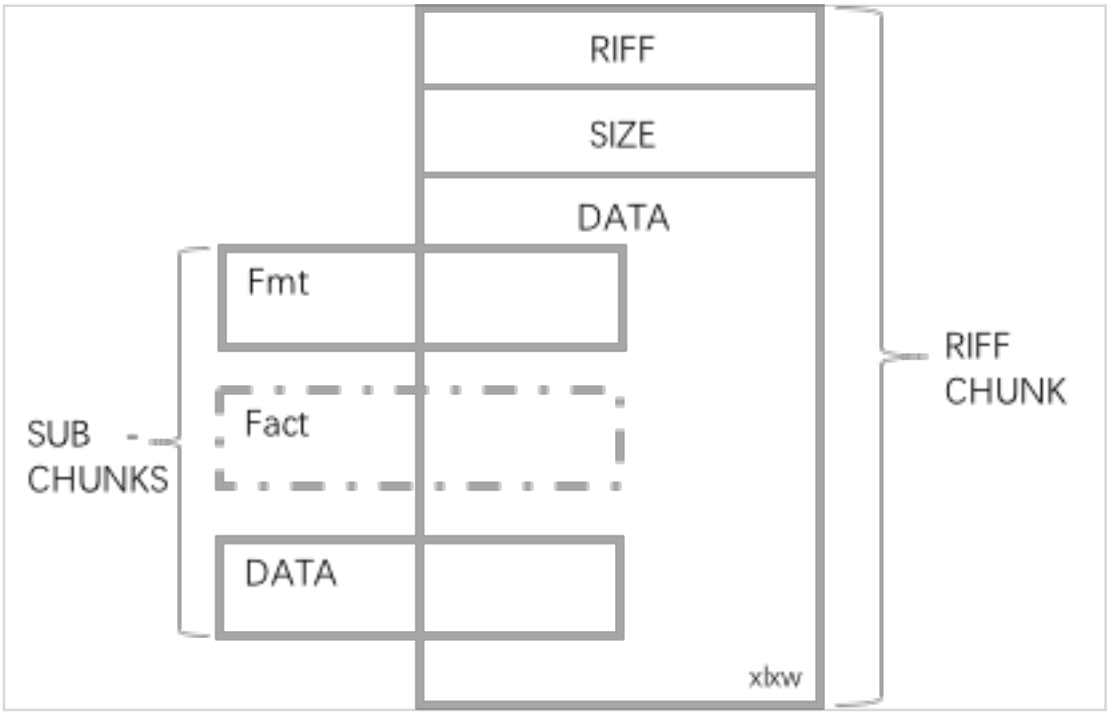
抽样信号离散的模拟信号，其取样的值在一定的取值范围内，由无限多种值可能性存在.为了实现以数字码表示样值，我们采用"四舍五入"的方法把样值分级"取整"，使一定取值范围内的样值由无限多个值变为有限个值.

编码

量化后的抽样信号在一定的取值范围内仅有有限个可取的样值，且信号正、负幅度分布的对称性使正、负样值的个数相等，正、负向的量化级对称分布

WAV文件的CHUNK信息

WAVE文件是由若干个CHUNK组成的.按照文件中CHUNK的出现顺序分别为：RIFF Chunk, Format Chunk, Fact Chunk, Data Chunk,其中的Fact CHUNK为非必要部分，结构具体如下图所示：



WAV文件头文件的CHUNK组成

RIFF是头CHUNK,而Format CHUNK里面记录了WAV的各种参数信息，详细参数信息如下：

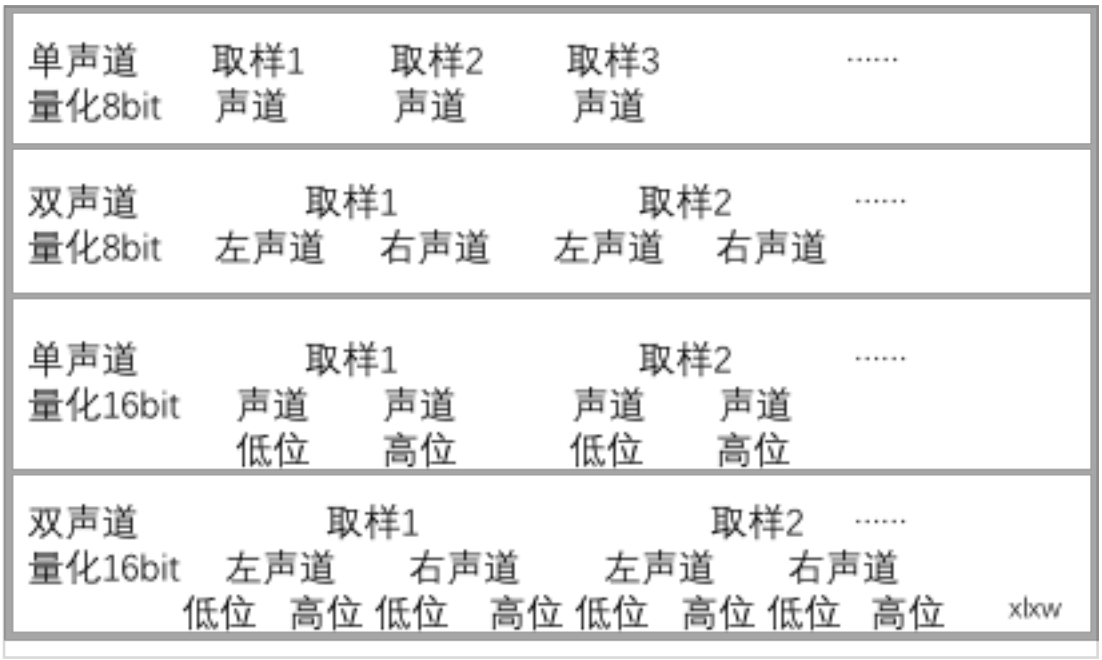
- FormatTag　　音频数据的编码方式，其中PCM方式为1
- Channels　　声道数，单声道为1，双声道为2
- SamplesPerSec　采样率(每秒样本数)
- BytesPerSec*　音频数据传送速率
- BlockAlign*　每次采样的大小
- BitsPerSample*　每个声道的采样精度

而FACT CHUNK的作用是因为有些并没有使用PCM格式，所以需要一个FACT CHUNK记录数据解压缩数据大小.

最后的DATA块中装的是真正的声音数据.一般按照WAVE_FORMAT_PCM的数据格式存贮,即脉冲编码调制PCM.

WAV的DATA部分

DATA块内的信息是根据format chunk内的信息而决定的.由量化位数 / 声道数 / 采样率共同决定，下图为四种情况下DATA区域存储信息的格式：



DATA CHUNK的格式

Python读取WAV文件的信息

在python中，我们可以直接通过许多音频的库对wav文件进行操作，比如自带的标准库wave库，还有如eyeD3,PyAudio,Audacity等等.我们先不介绍这种方式，我们先通过传统的文件操作以二进制的形式读取Wav文件，来分析一下它的头文件来验证一下我们前面有关CHUNK所学的知识.通过二进制操作音频文件并取得前四个字节的代码如下：(我们的测试音频是night.wav，已放在github中，通过我的博客园右上角的绿色图标可以链接到我的github界面，找到lab102下的w8目录即可获取该资源,或者找到本文博客园最上方资源)：

```
#读取wav前四个字节内容  -xlxw

file = open("night.wav", "rb")
s = file.read(4)
print(s)
```

程序运行截图：

```
========= RESTART: /Users/xulvxiaowei/Desktop/Wave Draw/4byteread.py ==
b'RIFF'
>>>
```

我们可以看到最前面的字节和我们认为的没错，是RIFF，那我们来读取44个字节来看看其中的信息是怎么样的：

```
#读取wav前44个字节内容  -xlxw
```

```
file = open("night.wav", "rb")
s = file.read(44)
print(s)
```

程序运行的截图：

```
========== RESTART: /Users/xulvxiaowei/Desktop/Wave Draw/4byteread.py ==========
b'RIFF\x04\xc3[\x01WAVEfmt \x10\x00\x00\x00\x01\x00\x02\x00D\xac\x00\x00\x10\xb1
\x02\x00\x04\x00\x10\x00data\xe0\xc2[\x01'
>>>
```

我们可以看到RIFF后面的字符串为WAVE的Form Type，以及fmt，data这几个FOURCC，而其他用十六进制表示的就是数据大小 / 数据了.所以我们通过二进制读取的WAV文件的信息和我们前面学习的CHUNK中的内容是一致的.

用Wave库提取wav文件信息

在我们前面中的介绍可以知道wav文件的存储方式，并且能够简单的提取其中的信息了，而我们知道wav文件最重要的就是声音信息的存贮，这一部分我们也可以通过对CHUNK的DATA 进行分析，不过我们在python中有更加简单的获得声音的方式，那就是利用python自带的wave库，我们下面就来介绍一下wave库的一些方法，为后文做铺垫.

- 首先引入wave库

  ```
  import wave
  ```

- open()

  打开一个声音文件，使用方法wave.open(声音文件地址，模式)

  其中声音文件地址就是wav文件位置，模式和文件读写差不多，如"wb"-只写；"rb"-只读方式 b代表以二进制模式打开.

- close()

  关闭声音文件

- getparams()

  获取wav文件的参数（以tuple形式输出），依次为(声道数，采样精度，采样率，帧数，.....)

  如下图为本文例子night.wav的getparams()的信息：

  ```
  ========== RESTART: /Users/xulvxiaowei/Desktop/Wave Draw/wave 2bit.py ==========
  The Path is:./night.wav
  _wave_params(nchannels=2, sampwidth=2, framerate=44100, nframes=5697720, comptype='NONE',
  compname='not compressed')
  ```

- readframes()

  得到每一帧的声音数据，返回的值是二进制数据，在python中用字符串表示二进制数据，如下图，所以我们后面要进行转化.

  得到的night.wav的前10帧的数据如下图所示：

  ```
  ========== RESTART: /Users/xulvxiaowei/Desktop/Wave Draw/wave 2bit.py ==========
  The Path is:./night.wav
  b'\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00'
  ```

上面就基本上是wave库的常用方法了.下面会对此进行应用了.

绘制WAV文件的波形

我们经常在许多声音软件，比如CoolEdit，Audition等软件中能看到声音文件的波形，所以我们在这里利用Python以及前面介绍的wave库，辅以numpy，Matplotlib来尝试绘制night.wav文件的波形.

下面我们来简单讲述一下绘制波形的步骤：（以night.wav为例子）

1. 通过wav库获得night.wav的头文件中的信息，如采样率 / 声道数等等.
2. 提取出DATA区域的信息，用numpy将string格式数据转化为数组
3. 通过判定声道数将DATA区域数据进行处理（对数组矩阵进行转换）
4. 得到每个绘制点的时间（x坐标）
5. 用matplotlib库提供的方法绘制出波形图

我们来对这些步骤中的一些部分详细说一下：

对DATA区域的处理

因为night.wav是一个双声道的WAV 文件，我们从上文对DATA区域格式的介绍可以知道储存形式是左声道 / 右声道的形式存贮数据，所以在这里我们要对提取出的数据进行处理，这里numpy库为我们提供了很好的解决方法，我们主要用到了改变形状的Shape方法以及T转置方法，我们在这里来举一个例子：

我们创建一个数组[1,2,3,4,5,6,7,8]里面有8个元素，这时候根据我们的分离方法，应该分为左声道[1,3,5,7]和右[2,4,6,8]，我们可以通过shape先改变矩阵的形状使数据变为两列分别为左右声道，再通过转置得到最终数据，我们的例子可以用如下图所示理解：

```
>>> import numpy as np
>>> n = np.array([1,2,3,4,5,6,7,8])
>>> n.shape = -1,2
>>> n
array([[1, 2],
       [3, 4],
       [5, 6],
       [7, 8]])
>>> n.T
array([[1, 3, 5, 7],
       [2, 4, 6, 8]])
>>>
```

将数据分为左右声道的例子

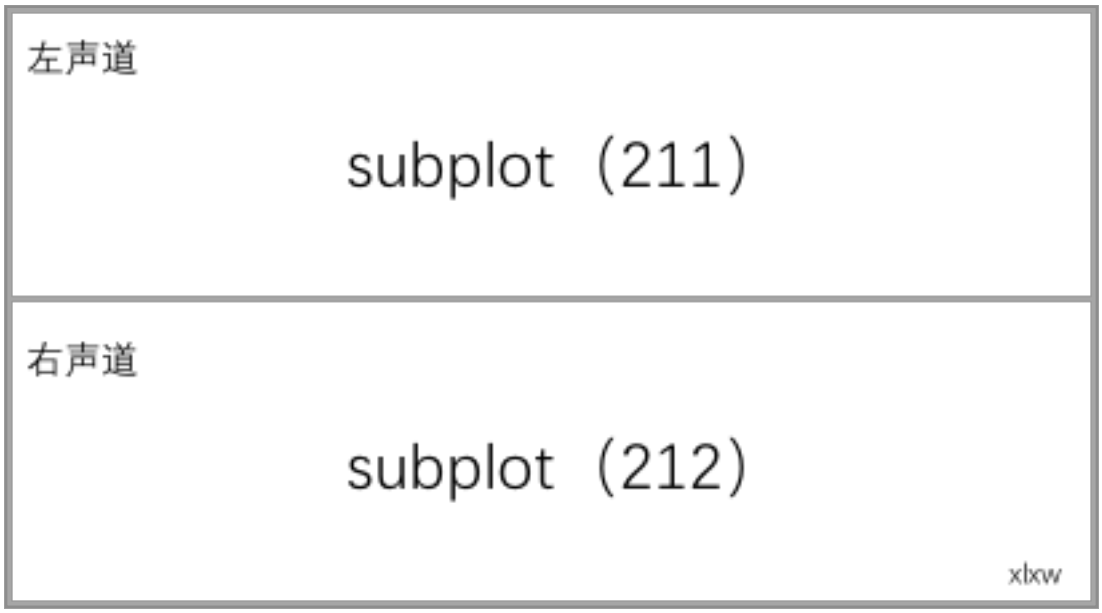
绘制波形图的matplotlib库

这里我们要绘出波形，所以用matplotlib库大大减少我们的绘图难度，我们主要用到plt.subplot和plt.plot这两个方法，所以我们对这两个方法进行解释.

- plt.subplot()

这是用于matplotlib绘制多个子图的方法，因为我们这里音频文件要分为两个部分（左声道 / 右声道）

即分成2X1的形式，所以我们的第一个绘图和第二个绘图分别用plt.subplot(211)和plt.subplot(212)来表示，如下图所示：



matplotlib子图

- plt.plot
  plt.plot()是用于绘制线条的方法，我们用到了其中的三个参数
  （X坐标，y坐标，颜色）
  就能用plot画出最终的波形图了

- 其他的方法这里就不介绍了，但是matplotlib的绘图功能相当强大.

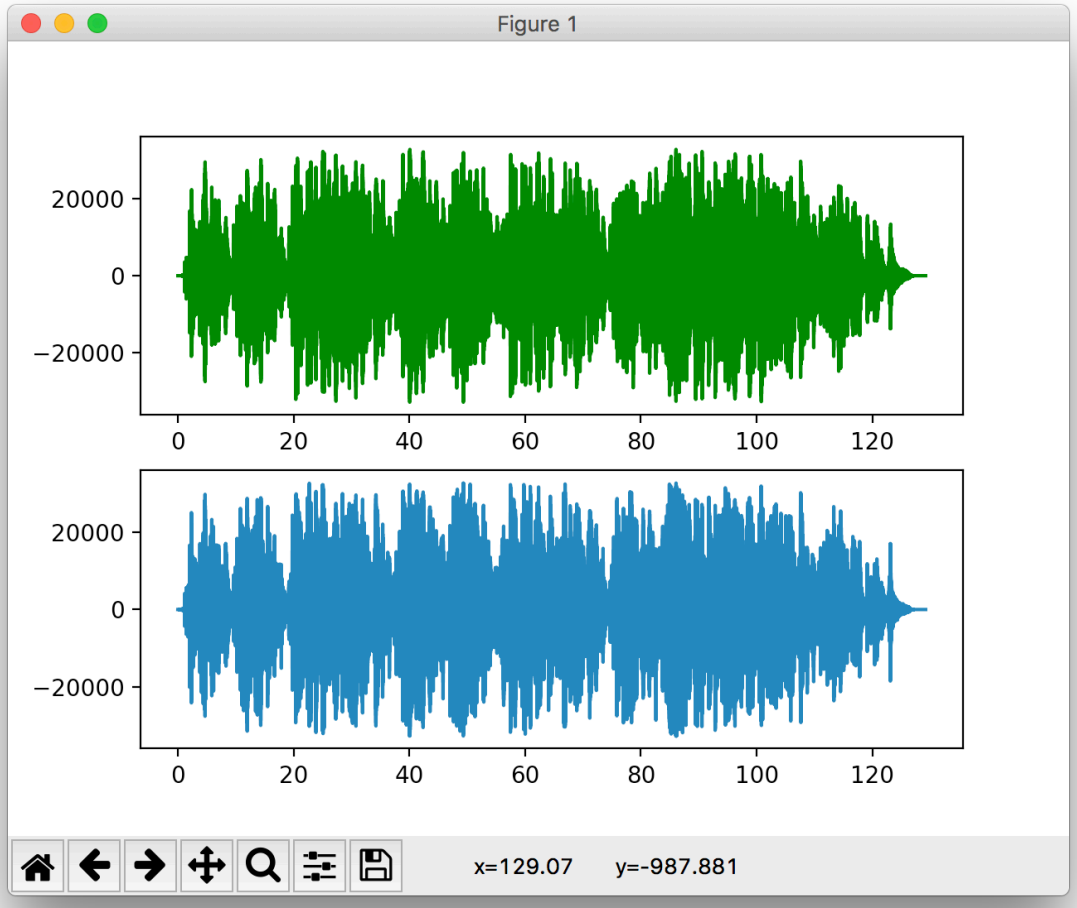
绘制night.wav波形的代码

用于绘制出wav文件波形的代码如下（这里我们还是以night.wav作为例子）

```
#wave data get  -xlxw

#import
import wave as we
import numpy as np
import matplotlib.pyplot as plt

def wavread(path):
    wavfile =  we.open(path,"rb")
    params = wavfile.getparams()
    framesra,frameswav= params[2],params[3]
    datawav = wavfile.readframes(frameswav)
    wavfile.close()
    datause = np.fromstring(datawav,dtype = np.short)
    datause.shape = -1,2
    datause = datause.T
```

```
    time = np.arange(0, frameswav) * (1.0/framesra)
    return datause,time

def main():
    path = input("The Path is:")
    wavdata,wavtime = wavread(path)
    plt.title("Night.wav's Frames")
    plt.subplot(211)
    plt.plot(wavtime, wavdata[0],color = 'green')
    plt.subplot(212)
    plt.plot(wavtime, wavdata[1])
    plt.show()

main()
```

程序绘制出的波形图的截图：



night.wav声音文件的双声道波形

总结&拓展

在本周我学到了wav文件的存贮格式以及怎么用python读取wav文件信息并且绘制出波形图.其中在绘制了波形图后，我们可以对波形多透露出的信息进行分析，比如：

- 得到声音的特征
- 分析录音时被录音者是否停止说话

等等用途，可以用于许多方面，如声音识别，断句（音频分割）等等许多的场景.这里在以后我也会继续进行探索.