读取和绘制音频数据

- import numpy as np
- import matplotlib.pyplot as plt
- from scipy.io import wavfile
- ''' 功能作用： 读取和绘制音频数据
- 注释：
  - 语音通常以44100 Hz的频率进行采样，这就意味着每秒钟信号被分解成 44 100 份，然后这些抽样值被保存。换句话说，每隔1/44100 s都会存储一次值。如果采样率很高， 用媒体播放器收听音频时，会感觉到信号是连续的 音频信号处理的图形
- 步骤：
  - 1.读取wav文件 ，返回 采样率 和 音频文件数据
  - 2.查看音频文件返回的参数,再继续下一步
  - 3.对音频数据进行标准化
    - 怎么标准化?
      - 得到了音频数据的数据类型,是16进制,然后标准化的分母就设置为 2^(16-1)
  - 4.抽取部分样本,进行画图显示
  - 5.构建时间轴X
    - 怎么构建时间轴?
      - 随机生成一个列表数据,数据长度是 0--到--(部分显示的音频数据长度) 除以采样率 ,最后把数据转化成毫秒* 1000 ms
- '''print('#'*5+'使用wavfile包从input_read.wav中读取音频文件'+'#'*5)
- # 读取输入文件, 返回采样率(样本/秒)和来自WAV文件的数据s
- ampling_freq, audio = wavfile.read('input_read.wav')
- # 打印参数print ('\nShape:', audio.shape)
- print ('Datatype:', audio.dtype)
- print ('Duration:', round(audio.shape[0] / float(sampling_freq), 3), 'seconds')
- # 该音频信号被存储在一个16位有符号整型数据中。标准化这些值 ,2.**15 代表2^15 次方,其中小数点保留一位小数
- audio = audio / (2.**15)
- # 提取前30个值，并将其画出audio = audio[:30]
- # 建立时间轴,X轴为时间轴
- x_values = np.arange(0, len(audio), 1) / float(sampling_freq)
- # 将单位转换为秒ms
- x_values *= 1000
- # 画出声音信号图形plt.plot(x_values, audio, color='black')
- plt.xlabel('Time (ms)')
- plt.ylabel('Amplitude')
- plt.title('Audio signal')
- plt.show()

幕布 - 思维概要整理工具