自定义参数生成音频信号

- import numpy as np
- import matplotlib.pyplot as plt
- from scipy.io.wavfile import write
- '''　功能作用 ：自定义参数生成音频信号
- 步骤 :
- 1.定义好指定的参数
  - 时间长度信号 采样频率 音频频率 时间轴的范围
- 2.利用sin函数生成音频信号　sin(2*pi*频率*变量)
- 3.增加必要的噪声 , 还是利用 duration * sampling_freq 随机生成
- 4. 增加完成后,把数据变成16位的整数型
- 5.开始写入文件wav中
- 6.以上完成之后,就开始绘制信号图
- '''# 定义存储音频的输出文件
- output_file = 'output_generated.wav
- '''指定音频生成参数。我们希望生成一个3 s长度的信号，采样频率为44100 Hz，音频的频率为587 Hz。时间轴上的值将从-2×pi到2×pi：'''
- # 指定音频生成的参数
- duration = 3 # 单位秒
- sampling_freq = 44100 # 单位
- Hztone_freq = 587 # 频率
- min_val = -2 * np.pi # 时间轴的最小值
- max_val = 2 * np.pi # 时间轴的最大值
- # 生成音频信号
- t = np.linspace(min_val, max_val, duration * sampling_freq)
- # sin(2*pi*频率*变量) 例如 sin(A*pi*x) ,构建一个正弦波曲线
- audio = np.sin(2 * np.pi * tone_freq * t)
- # 增加噪声
- noise = 0.4 * np.random.rand(duration * sampling_freq)
- audio += noise
- # 将这些数值转为16位整型数
- scaling_factor = pow(2,15) - 1audio_normalized = audio / np.max(np.abs(audio))
- audio_scaled = np.int16(audio_normalized * scaling_factor)
- # 写入输入文件
- write(output_file, sampling_freq, audio_scaled)
- # 提取前100个值
- audio = audio[:100]
- # 生成时间轴X
- x_values = np.arange(0, len(audio), 1) / float(sampling_freq)
- #将时间轴的单位转成毫秒msx_values *= 1000

- # 画出音频信号图
- plt.plot(x_values, audio, color='black')
- plt.xlabel('Time (ms)')
- plt.ylabel('Amplitude')
- plt.title('Audio signal')
- plt.show()