71086032-曾诗仪-第11次作业

当需要计算多段音频统的音高等特征时，串行计算往往需要耗费较长的时间。本周作业要求利用librosa库以及Python多进程实现对多个音频的并行特征计算：

0. import

```
import os
from multiprocessing import Process, Queue
import librosa
```

1. 主进程读取声音数据目录，得到所有声音文件的列表

```
if __name__ == '__main__':
    # 读取声音数据目录，得到所有声音文件的列表
    #读取装载多个音频的文件
    audio_dir = 'C:\\Users\\shiye\\Desktop\\ringtone'
    audio_files = [os.path.join(audio_dir, f) for f in os.listdir(audio_dir)
if f.endswith('.wav')]
```

2. 直接使用Process类构建子进程，利用libraso提供的支持来计算音高和声强，并保存计算结果到文件（一个声音一个文件）。
3. 通过继承Process类来构建子进程，同样利用libraso提供的支持来计算音高和声强，并保存计算结果到文件（一个声音一个文件）。

```
def calculate_features(filename, result_queue):
    # 计算音高和声强
    y, sr = librosa.load(filename)
    pitches, magnitudes = librosa.piptrack(y=y, sr=sr)
    pitch_mean = pitches.mean()
    pitch_std = pitches.std()
    intensity_mean = magnitudes.mean()
    intensity_std = magnitudes.std()

    # 保存计算结果到文件
    result_filename = os.path.splitext(filename)[0] + '.txt'
    with open(result_filename, 'w') as f:
        f.write('Pitch Mean: {}\n'.format(pitch_mean))
        f.write('Pitch Std: {}\n'.format(pitch_std))
        f.write('Intensity Mean: {}\n'.format(intensity_mean))
        f.write('Intensity Std: {}\n'.format(intensity_std))

       # 使用Process类构建子进程并并行计算特征
    processes = []
    for audio_file in audio_files:
```

```
        p = Process(target=calculate_features, args=(audio_file,
result_queue))
        processes.append(p)
        p.start()
```

4. 在主进程中启动利用2，3中构建的子进程，并分发（如通过队列）参数（声音文件名）。

```
    # 将计算结果放入队列
    result_queue.put(result_filename)

    print('Completed processing:', filename)

        result_queue = Queue()
    for p in processes:
        p.join()

    # 从队列中获取计算结果
    results = []
    while not result_queue.empty():
        result = result_queue.get()
        results.append(result)

    print('Feature calculation completed for all audio files.')
    print('Results:', results)
```

5. （附加）. 观察在一定数目的声音文件处理时（建议文件多一些），子进程数目变化与处理总时长的关系。

```
import os
from multiprocessing import Process, Queue
import librosa
import time

def calculate_features(filename, result_queue):
    # 计算音高和声强
    y, sr = librosa.load(filename)
    pitches, magnitudes = librosa.piptrack(y=y, sr=sr)
    pitch_mean = pitches.mean()
    pitch_std = pitches.std()
    intensity_mean = magnitudes.mean()
    intensity_std = magnitudes.std()

    # 保存计算结果到文件
    result_filename = os.path.splitext(filename)[0] + '.txt'
    with open(result_filename, 'w') as f:
        f.write('Pitch Mean: {}\n'.format(pitch_mean))
        f.write('Pitch Std: {}\n'.format(pitch_std))
        f.write('Intensity Mean: {}\n'.format(intensity_mean))
        f.write('Intensity Std: {}\n'.format(intensity_std))
```

```
    # 将计算结果放入队列
    result_queue.put(result_filename)

    print('Completed processing:', filename)

if __name__ == '__main__':
    # 读取声音数据目录，得到所有声音文件的列表
    audio_dir = 'C:\\Users\\shiye\\Desktop\\ringtone'
    audio_files = [os.path.join(audio_dir, f) for f in os.listdir(audio_dir) if
f.endswith('.wav')]

    # 不同的子进程数目
    num_processes_list = [1, 2, 4, 8, 16]

    for num_processes in num_processes_list:
        # 创建队列
        result_queue = Queue()

        # 使用Process类构建子进程并并行计算特征
        processes = []
        start_time = time.time()
        for audio_file in audio_files:
            p = Process(target=calculate_features, args=(audio_file,
result_queue))
            processes.append(p)
            p.start()

            # 限制子进程数目
            if len(processes) >= num_processes:
                for p in processes:
                    p.join()
                processes.clear()

        # 等待剩余子进程完成
        for p in processes:
            p.join()

        # 从队列中获取计算结果
        results = []
        while not result_queue.empty():
            result = result_queue.get()
            results.append(result)

        end_time = time.time()
        total_time = end_time - start_time

        print('Number of Processes:', num_processes)
        print('Total Time:', total_time)
        print('Results:', results)
        print('---')
```

具体输出： time.pdf

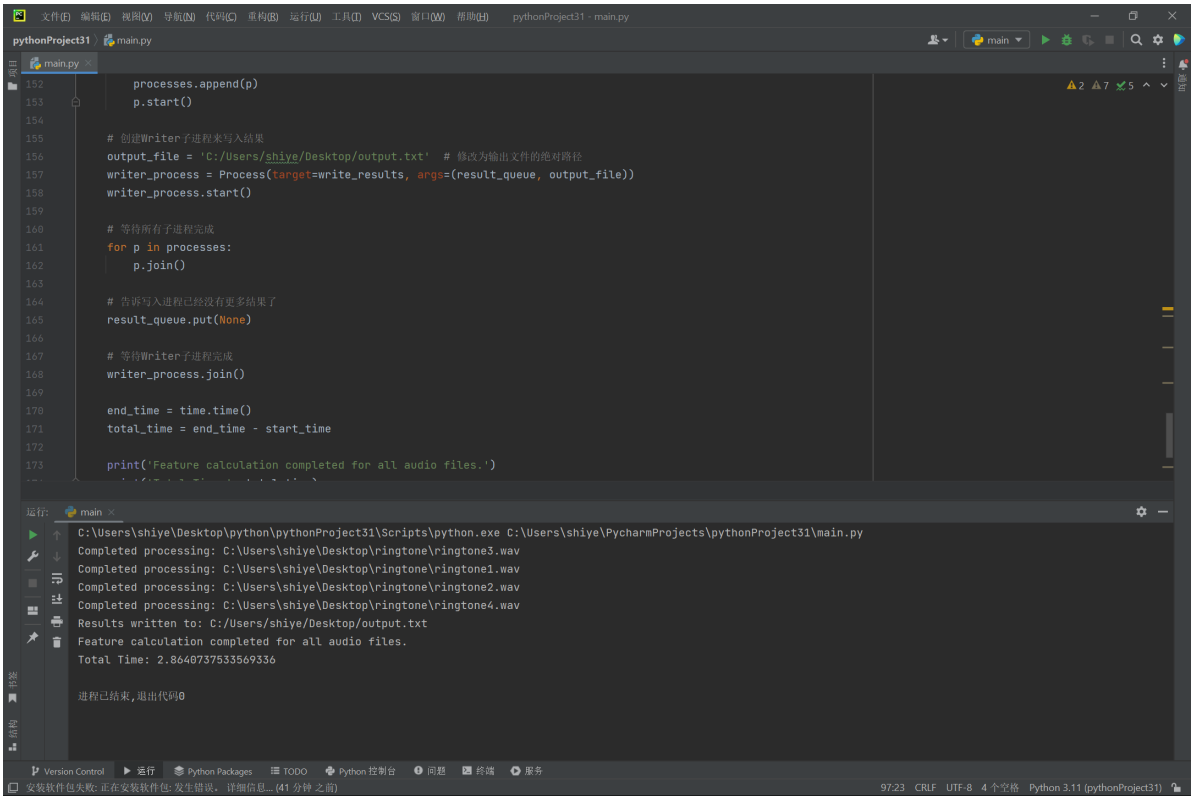
- 随着子进程数目的增加，其处理总时长越短，直到子进程数为16，其其处理总时长呈现出反增长的趋势。

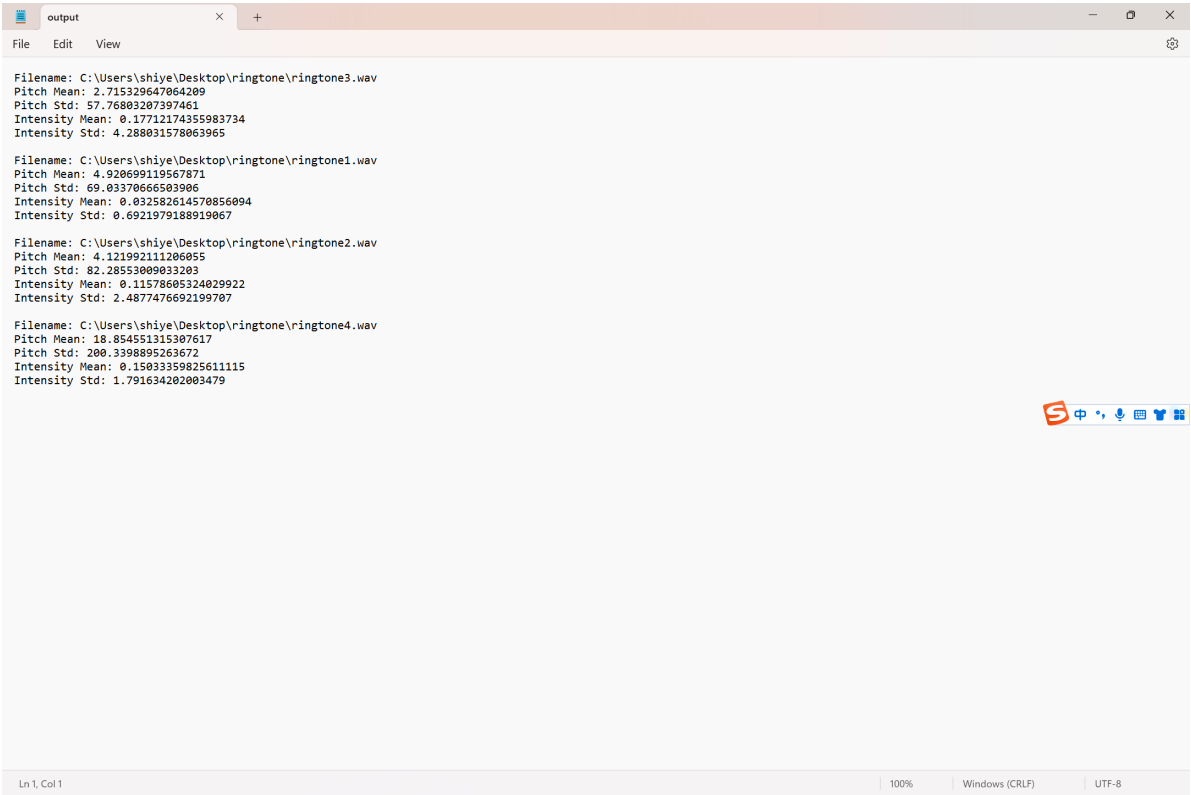
6. （附加）. 如果修改2，3中的要求，不再为每个声音保存一个文件，而是将所有声音的结果保存到一个文件中，应该如何处理？比如，计算子进程会将结果写入一个队列，而一个新的Writer子进程将负责从队列里读取内容并写入一个特定的汇总文件。

```
import os
import librosa
import time
import queue
from multiprocessing import Process, Queue
from multiprocessing import Process, Manager

def calculate_features(filename, result_queue):
    # 计算音高和声强
    y, sr = librosa.load(filename)
    pitches, magnitudes = librosa.piptrack(y=y, sr=sr)
    pitch_mean = pitches.mean()
    pitch_std = pitches.std()
    intensity_mean = magnitudes.mean()
    intensity_std = magnitudes.std()

    # 将计算结果放入队列
    result = (filename, pitch_mean, pitch_std, intensity_mean, intensity_std)
    result_queue.put(result)

    print('Completed processing:', filename)

def write_results(result_queue, output_file):
    with open(output_file, 'w') as f:
        while True:
            result = result_queue.get()  # 从队列中获取计算结果
            if result is None:
                break

            filename, pitch_mean, pitch_std, intensity_mean, intensity_std =
result
            f.write('Filename: {}\n'.format(filename))
            f.write('Pitch Mean: {}\n'.format(pitch_mean))
            f.write('Pitch Std: {}\n'.format(pitch_std))
            f.write('Intensity Mean: {}\n'.format(intensity_mean))
            f.write('Intensity Std: {}\n'.format(intensity_std))
            f.write('\n')

    print('Results written to:', output_file)

if __name__ == '__main__':
    # 读取声音数据目录，得到所有声音文件的列表
    audio_dir = 'C:\\Users\\shiye\\Desktop\\ringtone'
    audio_files = [os.path.join(audio_dir, f) for f in os.listdir(audio_dir) if
f.endswith('.wav')]

    # 创建Manager对象和共享的队列
    manager = Manager()
    result_queue = manager.Queue()
```

```python
    # 使用Process类构建子进程并并行计算特征
    processes = []
    start_time = time.time()
    for audio_file in audio_files:
        p = Process(target=calculate_features, args=(audio_file, result_queue))
        processes.append(p)
        p.start()

    # 创建Writer子进程来写入结果
    output_file = 'C:/Users/shiye/Desktop/output.txt'  # 修改为输出文件的绝对路径
    writer_process = Process(target=write_results, args=(result_queue,
output_file))
    writer_process.start()

    # 等待所有子进程完成
    for p in processes:
        p.join()

    # 告诉写入进程已经没有更多结果了
    result_queue.put(None)

    # 等待Writer子进程完成
    writer_process.join()

    end_time = time.time()
    total_time = end_time - start_time

    print('Feature calculation completed for all audio files.')
    print('Total Time:', total_time)
```

输出结果：





```
Filename: C:\Users\shiye\Desktop\ringtone\ringtone3.wav
Pitch Mean: 2.715329647064209
Pitch Std: 57.76803207397461
Intensity Mean: 0.17712174355983734
Intensity Std: 4.288031578063965

Filename: C:\Users\shiye\Desktop\ringtone\ringtone1.wav
Pitch Mean: 4.920699119567871
Pitch Std: 69.03370666503906
Intensity Mean: 0.032582614570856094
Intensity Std: 0.6921979188919067

Filename: C:\Users\shiye\Desktop\ringtone\ringtone2.wav
Pitch Mean: 4.121992111206055
Pitch Std: 82.28553009033203
Intensity Mean: 0.11578605324029922
Intensity Std: 2.4877476692199707

Filename: C:\Users\shiye\Desktop\ringtone\ringtone4.wav
Pitch Mean: 18.854551315307617
Pitch Std: 200.3398895263672
Intensity Mean: 0.15033359825611115
Intensity Std: 1.791634202003479
```

参考资料

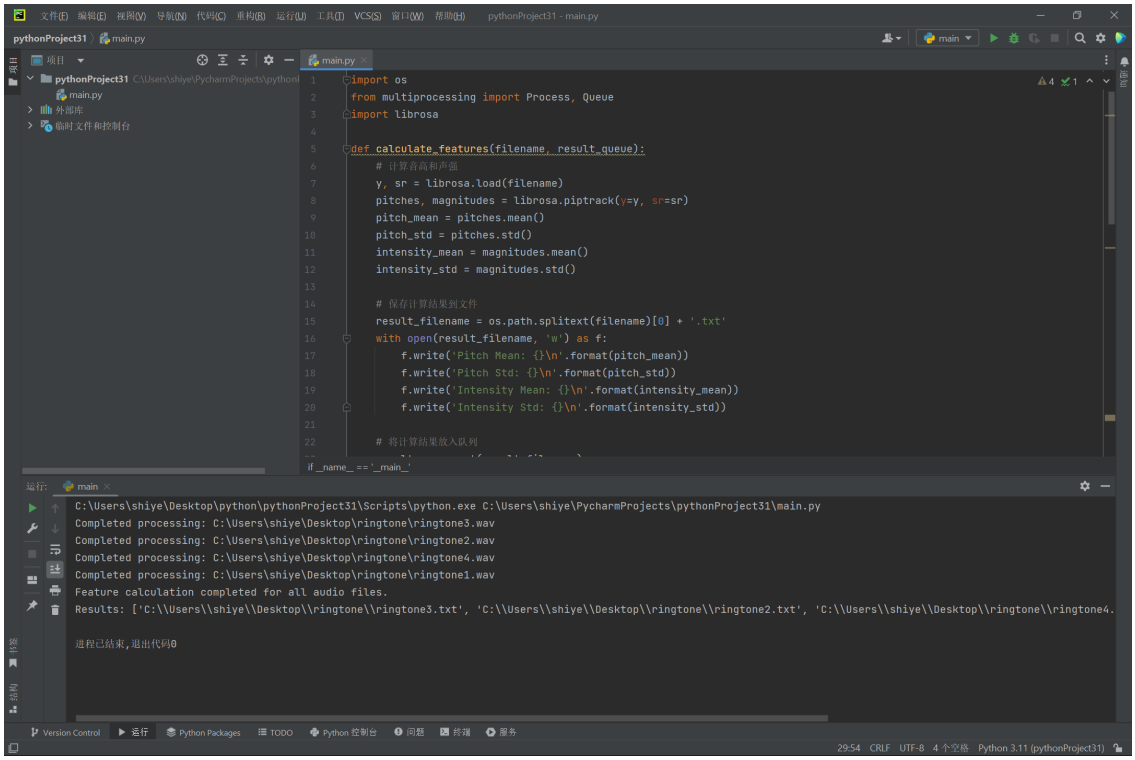
https://librosa.org/doc/latest/generated/librosa.yin.html#librosa.yin
https://librosa.org/doc/latest/generated/librosa.amplitude_to_db.html#librosa.amplitude_to_db

1-4题目总代码：

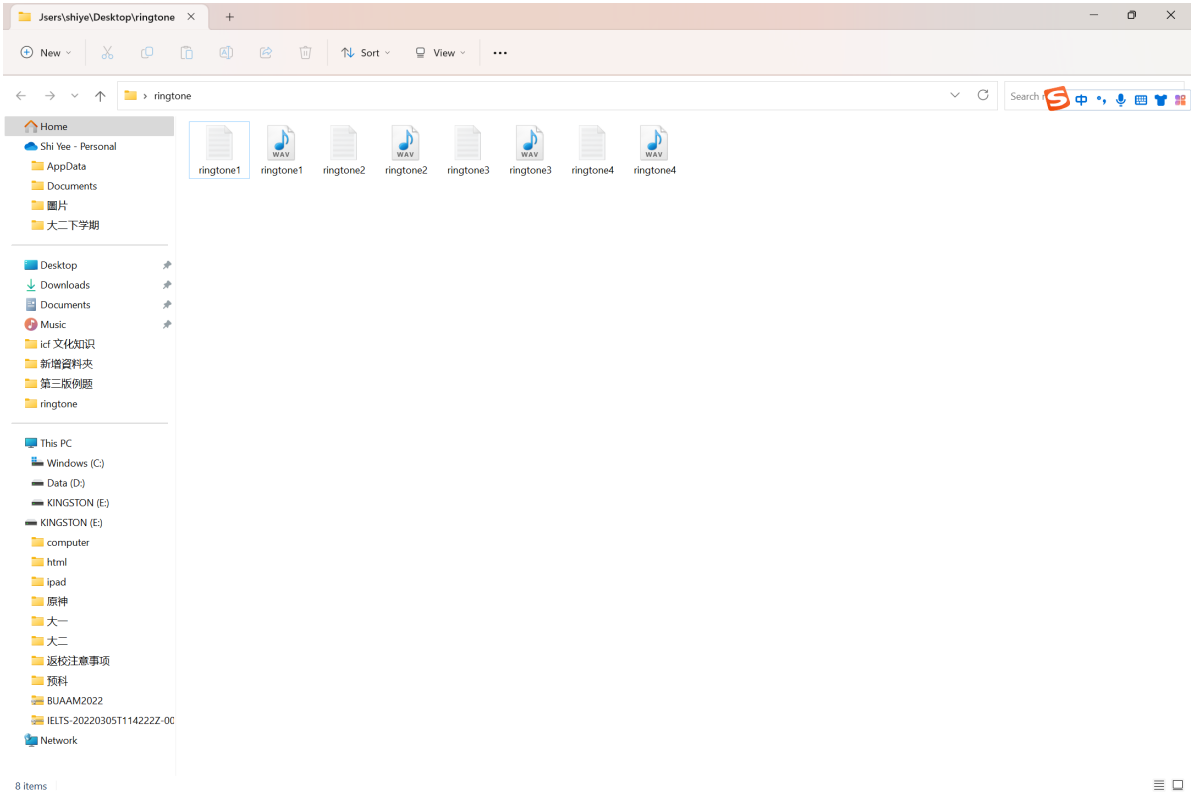
```python
import os
from multiprocessing import Process, Queue
import librosa

def calculate_features(filename, result_queue):
    # 计算音高和声强
    y, sr = librosa.load(filename)
    pitches, magnitudes = librosa.piptrack(y=y, sr=sr)
    pitch_mean = pitches.mean()
    pitch_std = pitches.std()
    intensity_mean = magnitudes.mean()
    intensity_std = magnitudes.std()

    # 保存计算结果到文件
    result_filename = os.path.splitext(filename)[0] + '.txt'
    with open(result_filename, 'w') as f:
        f.write('Pitch Mean: {}\n'.format(pitch_mean))
        f.write('Pitch Std: {}\n'.format(pitch_std))
        f.write('Intensity Mean: {}\n'.format(intensity_mean))
        f.write('Intensity Std: {}\n'.format(intensity_std))

    # 将计算结果放入队列
    result_queue.put(result_filename)

    print('Completed processing:', filename)
```

```python
if __name__ == '__main__':
    # 读取声音数据目录，得到所有声音文件的列表
    audio_dir = 'C:\\Users\\shiye\\Desktop\\ringtone'
    audio_files = [os.path.join(audio_dir, f) for f in os.listdir(audio_dir) if f.endswith('.wav')]

    # 创建队列
    result_queue = Queue()

    # 使用Process类构建子进程并并行计算特征
    processes = []
    for audio_file in audio_files:
        p = Process(target=calculate_features, args=(audio_file, result_queue))
        processes.append(p)
        p.start()

    # 等待所有子进程完成
    for p in processes:
        p.join()

    # 从队列中获取计算结果
    results = []
    while not result_queue.empty():
        result = result_queue.get()
        results.append(result)

    print('Feature calculation completed for all audio files.')
    print('Results:', results)
```

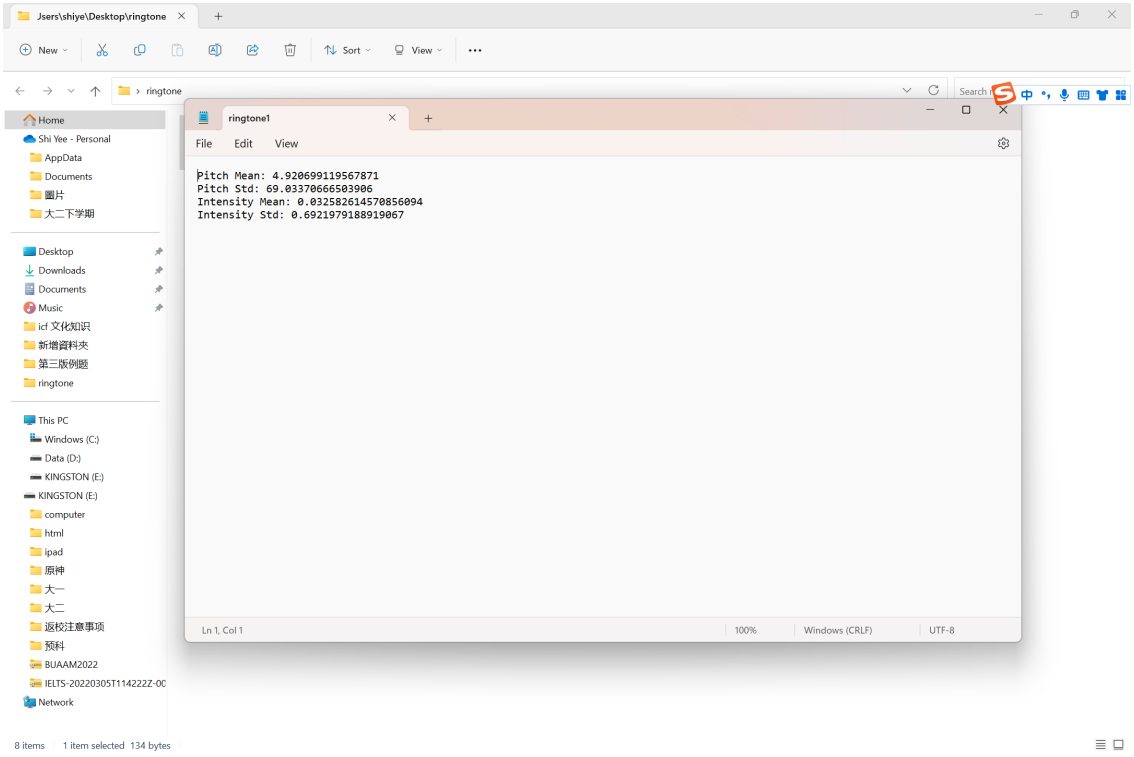
输出结果：

1. 

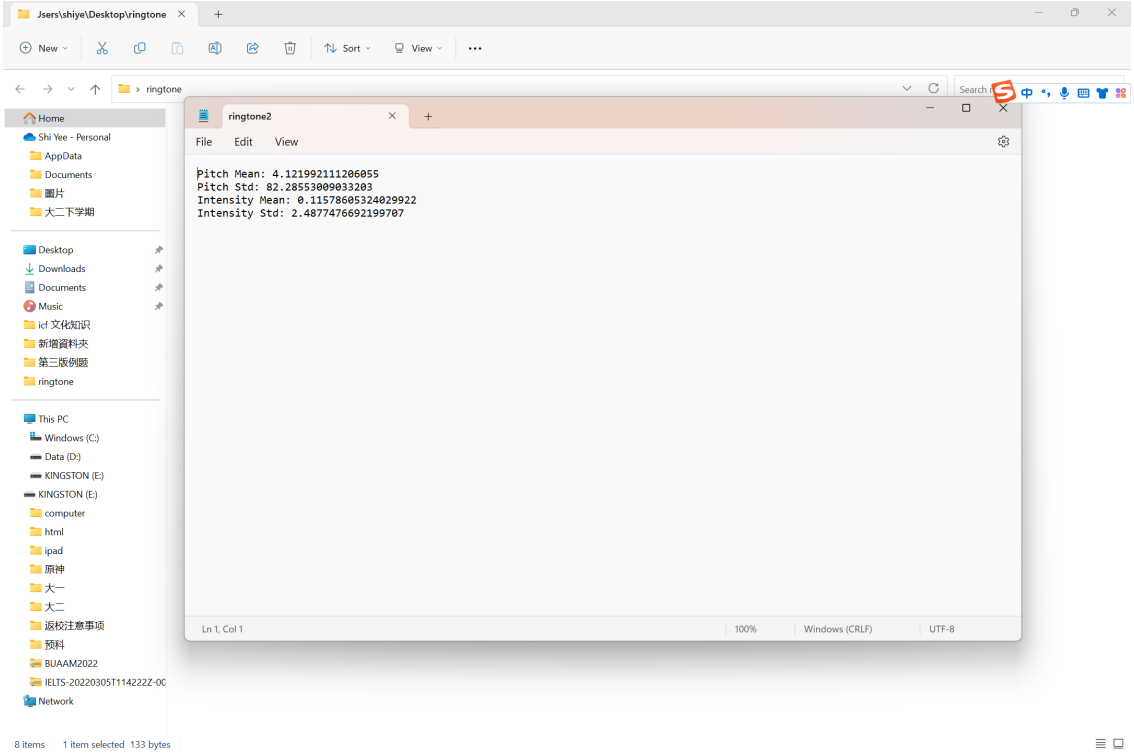
2. 一个声音一个文件：



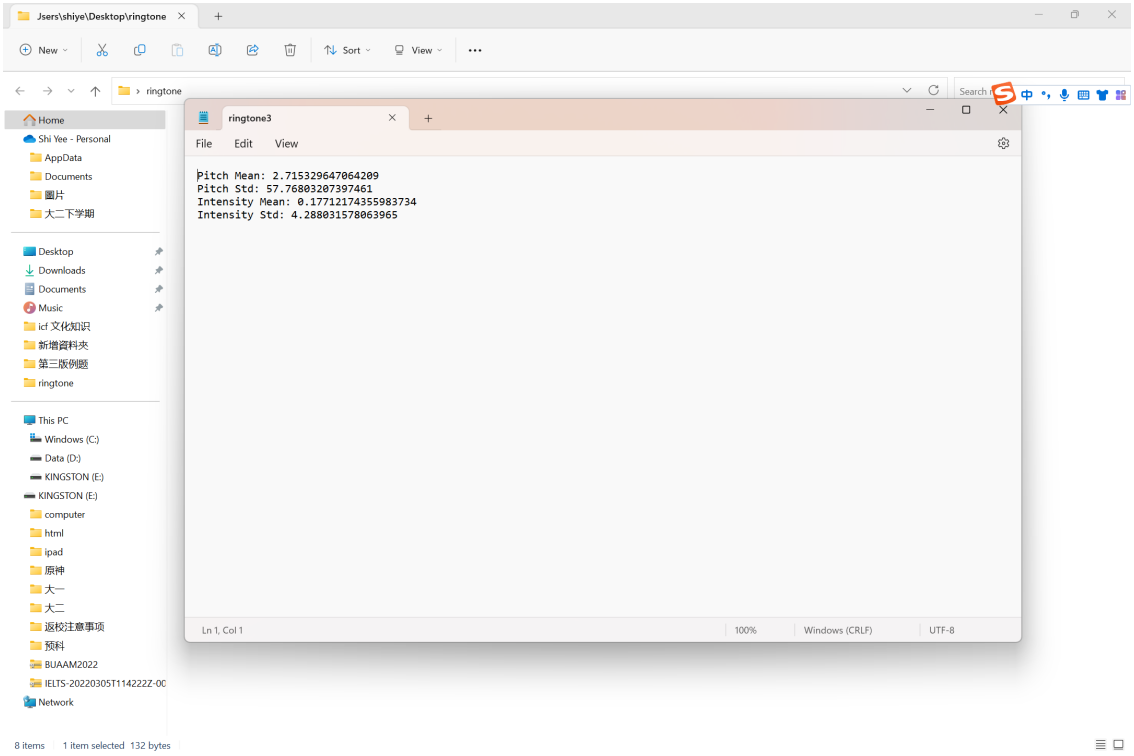
3. ringtone1：



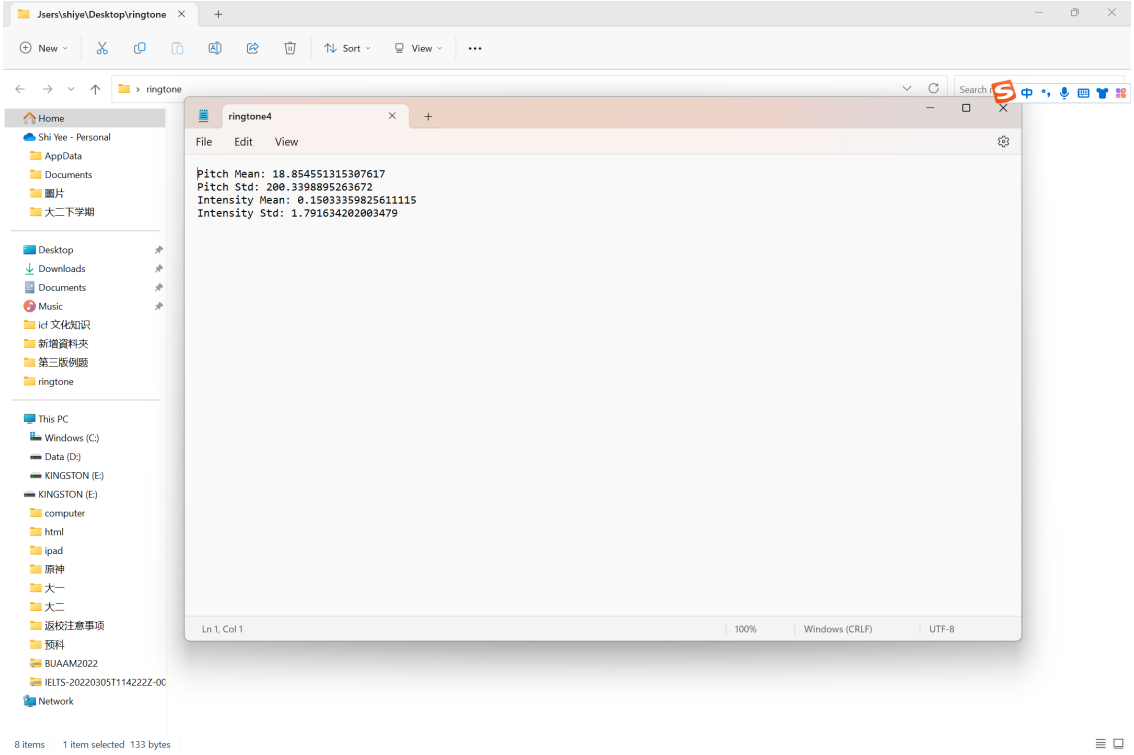
4. ringtone2：



5. ringtone3：



6. ringtone4：


Jsers\shiye\Desktop\ringtone
New
Sort
View
ringtone
Home
Shi Yee - Personal
AppData
Documents
圖片
大二下学期
Desktop
Downloads
Documents
Music
icf 文化知识
新增資料夹
第三版例题
ringtone
This PC
Windows (C:)
Data (D:)
KINGSTON (E:)
KINGSTON (E:)
computer
html
ipad
原神
大一
大二
返校注意事项
预科
BUAAM2022
IELTS-20220305T114222Z-00
Network
8 items
1 item selected 133 bytes
ringtone4
File
Edit
View
Pitch Mean: 18.854551315307617
Pitch Std: 200.3398895263672
Intensity Mean: 0.15033359825611115
Intensity Std: 1.791634202003479
Ln 1, Col 1
100%
Windows (CRLF)
UTF-8