I. Dependencies

- Anaconda: Alternative , but recommended
- Librosa≥ 0.6.1: `pip install librosa`
- NumPy ≥ 1.14.3 & SciPy≥ 1.1.0
- pydub(≥ 0.22.0) `pip install pydub` .

this is needed only when you would like to generate melody wav file with singletone audio.

- pretty-midi(≥ 0.2.8) `pip install pretty_midi`

this is needed only when you would like to generate melody midi file.

II. Application Mode

I. How to use

1.Put your audio file into *'wav'* folder

2.open your command line tool, (in Windows use cmd, in Linux use terminal)

cd into folder where exist file *'app.py'*

3.in command line tool, type : `>python app.py filename [mode]`

***filename**: the audio filename you would like to process*

***mode**: three options , 'wav2wav' , 'wav2midi', or no input*

II. Example

if I have a file **'meiruoliming.wav'** in the **/wav** folder. default mode:

```
(tensorflow) C:\Users\t-yic1i\Documents\GitHub\Melody-Extraction>python app.py meiruoliming.wav
Collecting modules Successfully
default mode, check ./wav2wavmix  and ./wav2midi folder
smooth the music
pitch contouring...
final freq1,freq2,freq3 generated
final freq4 generated OK
modified 27 notes
stereo sound file generated!
```

Figure 1: *no input in mode*

wav2wav mode:

```
(tensorflow) C:\Users\t-yic1i\Documents\GitHub\Melody-Extraction>python app.py meiruoliming.wav wav2wav
Collecting modules Successfully
check ./wav2wavmix folder
smooth the music
pitch contouring...
final freq1,freq2,freq3 generated
final freq4 generated OK
modified 27 notes
stereo sound file generated!
```

Figure 2: *wav2wav mode*

wav2midi mode:

```
(tensorflow) C:\Users\t-yicli\Documents\GitHub\Melody-Extraction>python app.py meiruoliming.wav wav2midi
Collecting modules Successfully
check ./wav2midi folder
smooth the music
pitch contouring...
final freq1,freq2,freq3 generated
final freq4 generated OK
modified 27 notes
```

Figure 3: *wav2midi mode*

III. Developer mode

some important functions you may need for further use.

I. PitchInfo

I.1 audio_notes, audio = PitchInfo.getNotesInfo()

audio_notes: a list of notes. each notes is in the form of start frame, end frame, estimated pitch.
audio: a numpy array showing the waveform data of the audio.

- **Example**

```
from getPitchInfo import PitchInfo
pitch_info = PitchInfo('./wav/meiruoliming.wav')
notes,_ = pitch_info.getNotesInfo()
for note in notes:
        print(note)
#==========result==========#
(47, 117, 57.82826990552989)
(117, 152, 57.650180126963406)
(152, 187, 59.89729843803559)
(187, 228, 59.89729843803559)
(256, 291, 60.66030413979112)
(291, 326, 61.391096674104446)
(326, 396, 62.76617800585667)
(396, 536, 62.09228621547197)
(536, 594, 62.281299902377036)
(686, 771, 59.89729843803559)
(834, 1044, 54.69532293693169)
(1044, 1079, 56.92606818447992)
........
#============================#
```

I.2 audio_freq, audio = PitchInfo.getFreqInfo()

audio_freq: a numpy array aligned with the audio wave form data.
audio: a numpy array showing the waveform data of the audio

- **Example**

```
import matplotlib.pyplot as plt
freq,audio = pitch_info.getFreqInfo()
#illustrate part of the audio and the corresponding frequency
plt.plot(audio[:audio.shape[0]//4] * 400,label = 'audio wave')
plt.plot(freq[:freq.shape[0]//4],label = 'frequency')
plt.legend()
```

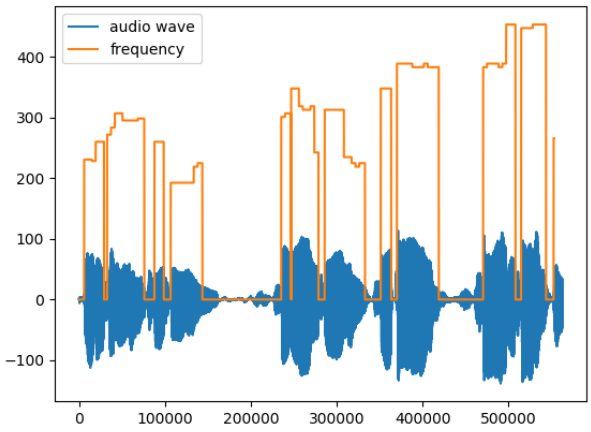

Figure 4: *Frequency And Audio Wave*