Sound(Module)

May 3, 2017

1 This is a module which provides some of the functions for reading, writing, recording and playing a wave file.

1.0.1 import relevant modules and define variables.

```
In [ ]: import pyaudio
        import struct
        from numpy import clip

        opened=0;
        stream=[]
```

1.1 The functions are as follows(description followed by the code):

1.1.1 sound(s, FS)

This function plays the samples vector 's' with the sampling rate of 'FS', where, 's' is the samples vector for a signal to be played through the output & 'fs' is the sampling rate at which the signal has to be played.

```
In [ ]: def sound(s, FS):
            """
            This function plays out a vector s as a sound at sampling rate FS,
            like on Octave or Matlab, with:
            import sound; sound.sound(s,FS)
            """

            CHUNK = 1024 #Blocksize
            WIDTH = 2 #2 bytes per sample
            CHANNELS = 1 #2
            RATE = FS  #Sampling Rate in Hz
            p = pyaudio.PyAudio()

            stream = p.open(format=p.get_format_from_width(WIDTH),
                        channels=CHANNELS,
                        rate=RATE,
                        input=False,
                        output=True,
                        #input_device_index=0,
                        frames_per_buffer=CHUNK)
        for i in range(0, int(len(s) / CHUNK) ):
            #converting from short integers to a stream of bytes in data:
            samples=s[i*CHUNK:((i+1)*CHUNK)];
            samples=clip(samples,-2**15,2**15-1)
            data=struct.pack('h' * CHUNK, *samples);
            #print data[1]
            #Writing data back to audio output stream:
            stream.write(data, CHUNK)


        stream.stop_stream()
        stream.close()

        p.terminate()
        print("* done")
```

1.1.2 wavread(sndfile)

This function reads the file and returns back two values i.e., the vector of samples and the sampling rate, where 'sndfile' is the file name or file path which has to read. E.g., wavread('Track01.wav') will return the series of samples and sampling rate.

```
In [ ]: def wavread(sndfile):
            """
            This function implements the wavread function of Octave or Matlab
            to read a wav sound file into a vector s and sampling rate info at
            its return, with: import sound; [s,rate]=sound.wavread('sound.wav');
            or s,rate=sound.wavread('sound.wav')
            """

            import wave
            wf=wave.open(sndfile,'rb');
            nchan=wf.getnchannels();
            bytes=wf.getsampwidth();
            rate=wf.getframerate();
            length=wf.getnframes();
            print("Number of channels: ", nchan);
            print("Number of bytes per sample:", bytes);
            print("Sampling rate: ", rate);
            print("Number of samples:", length);
            data=wf.readframes(length);
            if bytes==2:
                shorts = (struct.unpack( 'h' * length, data ));
            else:
                shorts = (struct.unpack( 'B' * length, data ));
            wf.close;
```

```
        return shorts, rate;
```

1.1.3 wavwrite(snd, Fs, sndfile)

This function writes the samples vector 'snd' with sampling rate 'Fs' to the file provided in sndfile(string value of filename or filepath). ** 'snd' is the samples vector 'Fs' is the sampling rate 'sndfile' is the desired filename or filepath** E.g., wavwrite(snd, Fs, 'Track22.wav')

```
In [ ]: def wavwrite(snd,Fs,sndfile):
            """
            This function implements the wawwritefunction of Octave or Matlab
            to write a wav sound file from a vector snd with
            sampling rate Fs, with:
            import sound;
            sound.wavwrite(snd,Fs,'sound.wav');"""

            import wave
            import pylab

            WIDTH = 2 #2 bytes per sample
            #FORMAT = pyaudio.paInt16
            CHANNELS = 1
            #RATE = 22050
            RATE = Fs #32000

            length=pylab.size(snd);

            wf = wave.open(sndfile, 'wb')
            wf.setnchannels(CHANNELS)
            wf.setsampwidth(WIDTH)
            wf.setframerate(RATE)
            data=struct.pack( 'h' * length, *snd )
            wf.writeframes(data)
            wf.close()
```

1.1.4 record(time, Fs)

This function records in the audio from the selected input for a given period with desired sampling rate. ** 'time' is desired time of recording in seconds. 'Fs' is the sampling rate for recording.** E.g., record(10, 32000)

```
In [ ]: def record(time, Fs):
            """
            Records sound from a microphone to a vector s, for instance
            for 5 seconds and with sampling rate of 32000 samples/sec:
            import sound; s=sound.record(5,32000)
            """

            import numpy;
```

```python
global opened;
global stream;
CHUNK = 1000 #Blocksize
WIDTH = 2 #2 bytes per sample
CHANNELS = 1 #2
RATE = Fs  #Sampling Rate in Hz
RECORD_SECONDS = time;

p = pyaudio.PyAudio()

a = p.get_device_count()
print("device count=",a)

#if (opened==0):
if(1):
    for i in range(0, a):
        print("i = ",i)
        b = p.get_device_info_by_index(i)['maxInputChannels']
        print("max Input Channels=", b)
        b = p.get_device_info_by_index(i)['defaultSampleRate']
        print("default Sample Rate=", b)

    stream = p.open(format=p.get_format_from_width(WIDTH),
            channels=CHANNELS,
            rate=RATE,
            input=True,
            output=False,
            #input_device_index=3,
            frames_per_buffer=CHUNK)
opened=1;

print("* recording")
snd=[]

#Loop for the blocks:
for i in range(0, int(RATE / CHUNK * RECORD_SECONDS)):
    #Reading from audio input stream into data with block length "CHUNK":
    data = stream.read(CHUNK)
    #Convert from stream of bytes to a list of short integers (2 bytes here) in "sam
    #shorts = (struct.unpack( "128h", data ))
    shorts = (struct.unpack( 'h' * CHUNK, data ));
    #samples=list(shorts);
    samples=shorts;
    snd=numpy.append(snd,samples);
return snd;
```