soundfloat(module)

January 27, 2017

1 Module for sound playback functions for pylab

1.1 using normalized float numbers to access the sound card. Gerald Schuller, July, 2016

- **Importing relevant modules**

```
In [1]: import pyaudio
        from numpy import clip
        import numpy as np

        opened=0
```

- **This function plays out a vector s as a sound at sampling rate FS, like on Octave or Matlab, with: import soundfloat; soundfloat.sound(s,FS)**

```
In [2]: def 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=pyaudio.paFloat32,
                        channels=CHANNELS,
                        rate=RATE,
                        #input=False,
                        output=True,
                        #input_device_index=10,
                        #frames_per_buffer=CHUNK
                        )
            stream.write(s.astype(np.float32))
            """
            for i in range(0, int(len(s) / CHUNK) ):
              #print "i=", i
              #Putting samples into blocks of length CHUNK:
```

```
    samples=s[i*CHUNK:((i+1)*CHUNK)];
    samples=clip(samples,-1,1)
    #print samples[1]
    #print "len(samples)= ", len(samples)
    #Writing data back to audio output stream:
    stream.write(samples.astype(np.float32))
  """

  stream.stop_stream()
  stream.close()

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

- **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')**

```
In [3]: def wavread(sndfile):
            import wave
            import struct

            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
```

- **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')**

```
In [4]: def wavwrite(snd,Fs,sndfile):
            import wave

            WIDTH = 2 #2 bytes per sample
```

```
    #FORMAT = pyaudio.paInt16
    CHANNELS = 1
    #RATE = 22050
    RATE = Fs #32000

    length=len(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()
```

- **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)**

```
In [5]: def record(time, Fs):

            import numpy;
            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=pyaudio.paFloat32,
                    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":
        samples = stream.read(CHUNK).astype(np.float32)
        #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
```

- **Testing the module - Using a sine tone of 400 Hz and sampling frequency of 44.1 kHz and playing it for 1 second.**

```
In [6]: if __name__ == '__main__':
    #Testing:
    s = np.sin(2 * np.pi * 440 * np.arange(0.0, 1.0, 1/44100.0))
    sound(s * 0.3, 44100)
```

```
* done
```