Sound as a Vector

In [2]:

```
%matplotlib inline

import numpy as np
import matplotlib.pyplot as plt
```

In [5]:

```
from html5_audio import DEFAULT_RATE, get_html5_wave_player #html5_audio.py is a separate python file
```

Here's a function that produces a 'beep' with a parabola as an 'envelope':

In [7]:

```
def make_beep(freq, duration=0.3): #input: frequency and duration (default is 0.3 for duration)
    nsamples = DEFAULT_RATE * duration # 44100 Hz * seconds gives number of samples

    t = np.linspace(0, duration, nsamples)
    data = np.sin(freq*2*np.pi*t) #sound waves are modeled mathematically by sine waves

    ramp = np.linspace(0, 1, nsamples)
    data = data * (1 - ramp)**2 #use parabola 'envelope' to dampen sound

    return(data)
```

Let's put one into the variable beep:

In [8]:

```
beep = make_beep(440) #No input for duration means we use default of 0.3 sec.
```

It's just a numpy array:

In [9]:

```
type(beep)
```

Out[9]:

```
numpy.ndarray
```

A pretty long one though:

In [10]:

```
beep.shape
```

Out[10]:

```
(13230,)
```

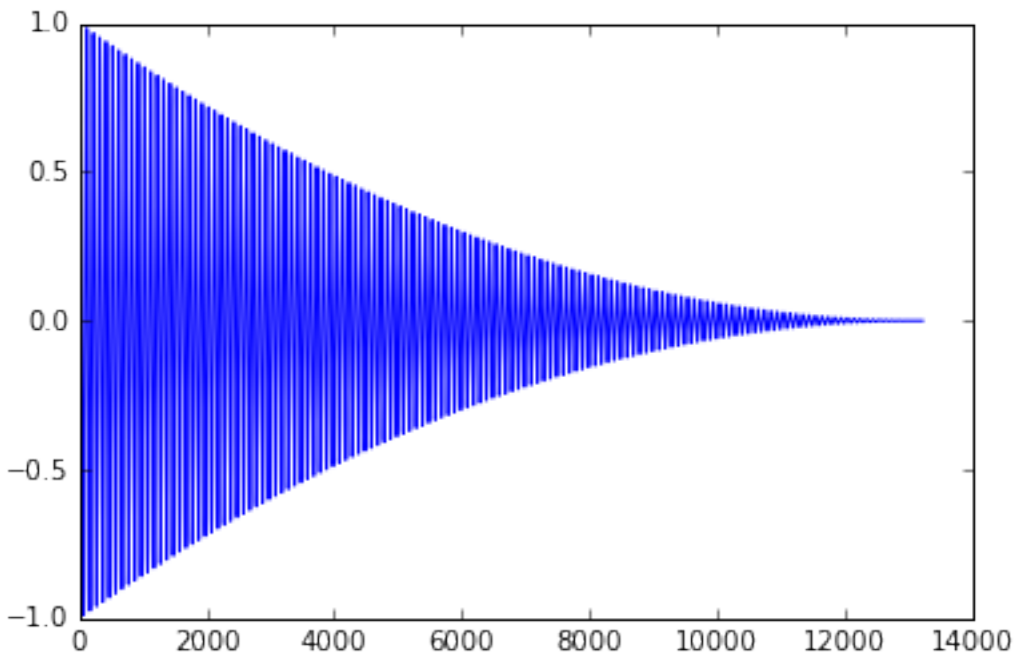
Let's take a look:

In [14]:

```
plt.plot(beep) #should see the parabola envelope above and below sound wave
```

Out[14]:

```
[<matplotlib.lines.Line2D at 0x10faa1a90>]
```



In [15]:

```
get_html5_wave_player(beep)
```

Out[15]:

Can we compose some simple 'music' with this? Let's shoot for three seconds...

In [20]:

```
music = np.zeros(DEFAULT_RATE * 3, dtype = np.float32) #44100 Hz * 3 sec. gives
number of samples.  Using single precision floats (float32)
music.shape
```

Out[20]:

```
(132300,)
```

In [39]:

```
beep1 = make_beep(880, duration = 0.8)
beep2 = make_beep(880 * 1.5, duration = 0.8)
beep3 = make_beep(880 * 2, duration = 0.8) #create beeps of 3 different frequenc
ies
l = len(beep1)
print("length of beep1: {}".format(l))
```

```
length of beep1: 35280
```

In [40]:

```
music.fill(0) #fill array with all zeros
```

In [41]:

```
music[0:l] += beep1 #play beep1 immediately
s = int(0.3 * DEFAULT_RATE) #delay of 0.3 seconds for beep2
music[s:s+l] += 0.1*beep2
s = int(0.6 * DEFAULT_RATE) #delay of 0.6 seconds for beep3
music[s:s+l] += 0.5*beep3
```

In [42]:

```
get_html5_wave_player(music)
```

Out[42]:

Okay, maybe something harder...

Can you name this tune?

In [69]:

```
name_this_tune = np.zeros(DEFAULT_RATE * 8, dtype = np.float32)
A = make_beep(440, duration = 0.25) #different notes have different frequencies
B = make_beep(493.9, duration = 0.25) #duration = 0.25 is quarter note
Csharp = make_beep(554.4, duration = 0.25)
D = make_beep(587.3, duration = 0.25)
E = make_beep(659.3, duration = 0.5) #duration = 0.5 is half note
qsize = A.shape[0] #quarter note = .25 seconds
hsize = E.shape[0] #half note = .5 seconds
print("name_this_tune.shape: {}".format(name_this_tune.shape))
print("quarter note: {}".format(qsize))
print("half note: {}".format(hsize))
```

```
name_this_tune.shape: (352800,)
quarter note: 11025
half note: 22050
```

In [70]:

```
name_this_tune.fill(0)
s = 0; e = qsize #will fill array with quarter and half notes
# first bar
name_this_tune[s:e] += A
s = e + qsize; e = s + qsize
name_this_tune[s:e] += B
s = e + qsize; e = s + qsize
name_this_tune[s:e] += Csharp
s = e + qsize; e = s + qsize
name_this_tune[s:e] += A
# second bar!
s = e + qsize; e = s + qsize
name_this_tune[s:e] += A
s = e + qsize; e = s + qsize
name_this_tune[s:e] += B
s = e + qsize; e = s + qsize
name_this_tune[s:e] += Csharp
s = e + qsize; e = s + qsize
name_this_tune[s:e] += A
# third....
s = e + qsize; e = s + qsize
name_this_tune[s:e] += Csharp
s = e + qsize; e = s + qsize
name_this_tune[s:e] += D
s = e + qsize; e = s + hsize
name_this_tune[s:e] += E
# fourth!
s = e + hsize; e = s + qsize
name_this_tune[s:e] += Csharp
s = e + qsize; e = s + qsize
name_this_tune[s:e] += D
s = e + qsize; e = s + hsize
name_this_tune[s:e] += E
```

In [71]:

```
get_html5_wave_player(name_this_tune)
```

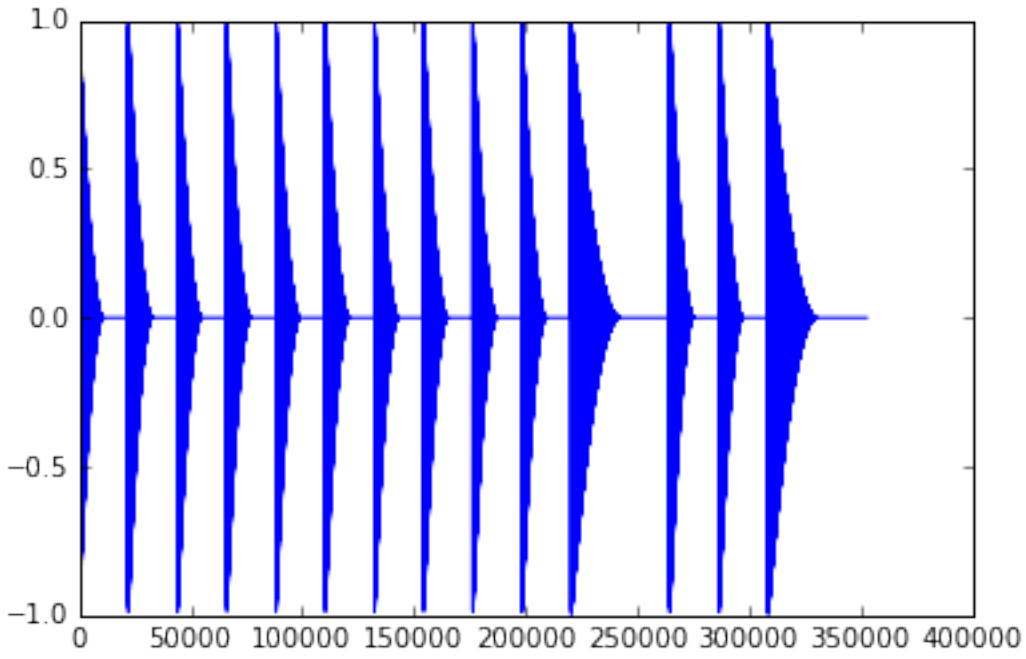
Out[71]:

In [72]:

```
plt.plot(name_this_tune)
```

Out[72]:

```
[<matplotlib.lines.Line2D at 0x10c36e4e0>]
```



In [ ]: