EE120 Fall 2014 PS 6, upsampling music

To output python notebook, use ``ipython nbconvert file.ipynb" from command prompt. You may need to install pandoc first.

In [1]:
```
print 4+5  # check to see if iPython is running...
```

```
9
```

In [2]:
```
%pylab
```

```
Using matplotlib backend: Qt4Agg
Populating the interactive namespace from numpy and matplotlib
```

In [3]:
```
import numpy as np
import scipy as sp
import matplotlib.pyplot as plt
%matplotlib inline
print 'Done importing'
```

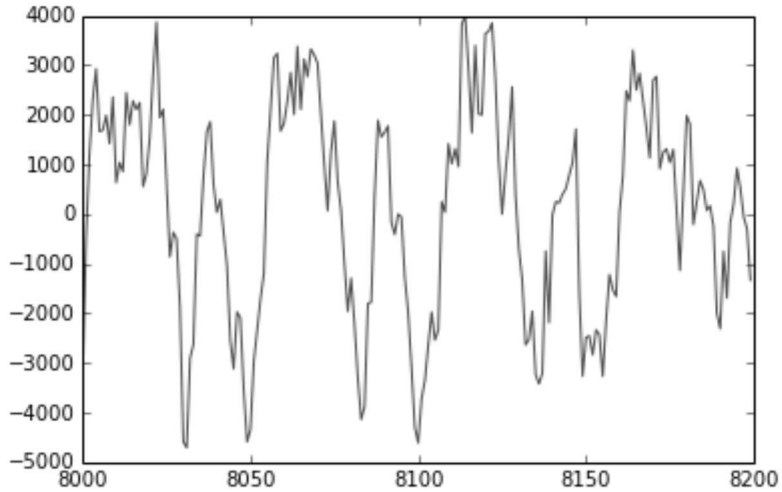
```
Done importing
```

In [4]:
```
# Graphing helper function
def setup_graph(title='', x_label='', y_label='', fig_size=None):
    fig = plt.figure()
    if fig_size != None:
        fig.set_size_inches(fig_size[0], fig_size[1])
    ax = fig.add_subplot(111)
    ax.set_title(title)
    ax.set_xlabel(x_label)
    ax.set_ylabel(y_label)
```

In [5]:
```
# import file
from scipy.io import wavfile
rate1,data1= wavfile.read('music.wav')  # 16 bit data if from Audacity
print 'rate1 =', rate1   ### downsampled is actually at 8820 Hz
#print 'data1 =', data1
length = np.size(data1)/2
print 'length = ', length
t1 = np.linspace(0, float(length)/float(rate1), length)
#
#
```

```
rate1 = 44100
length =  104857
```

In [7]:
```
# plot data
sound = np.zeros(length)
for i in range(0,length):
   sound[i] = data1[i][0] # copy left channel
# Choose N
sound = np.concatenate( (sound, np.zeros(2**17 - length)))
_ = plt.plot(range(8000,8200), sound[8000:8200])
```



Add necessary upsampling steps to cells below

In [14]:
```
# calculate FFT for a segment to see filter before downsample
Ne = np.size(sound)
print 'Ne=', Ne
X = np.fft.fft(sound)  # calculate X[k]
n1 = np.linspace(0,Ne-1, Ne)
Y = np.copy(X)  ### change this line to do FFT of upsampled signal
```

```
Ne= 131072
```

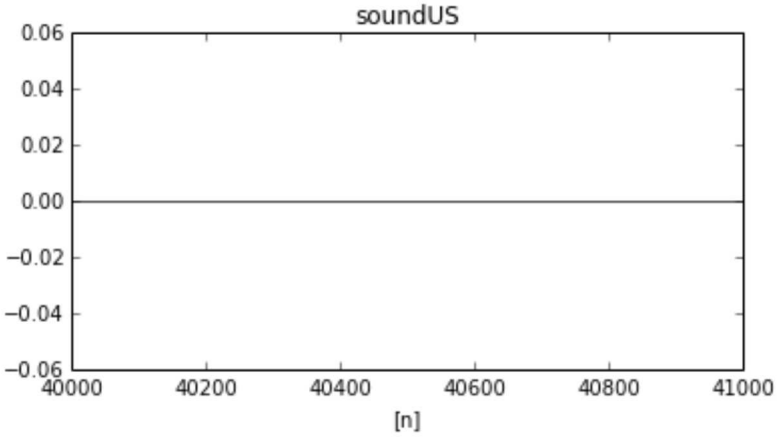
In [18]:
```
#  upsample
soundUS = np.zeros((Ne*5,2))# setup array for stereo
print 'size soundUS =', np.size(soundUS)

for i in range(0,Ne*5):
    soundUS[i,0] = np.real(0.0)
    soundUS[i,1] = np.real(0.0)  ### change to

setup_graph(title='soundUS', x_label='[n]', fig_size=(6,3))
_ = plt.plot(range(40000,41000), soundUS.real[40000:41000])

# now write  data file
wavfile.write('music-up.wav', rate1, soundUS.astype(int16)) # 16 bit int
eger
```

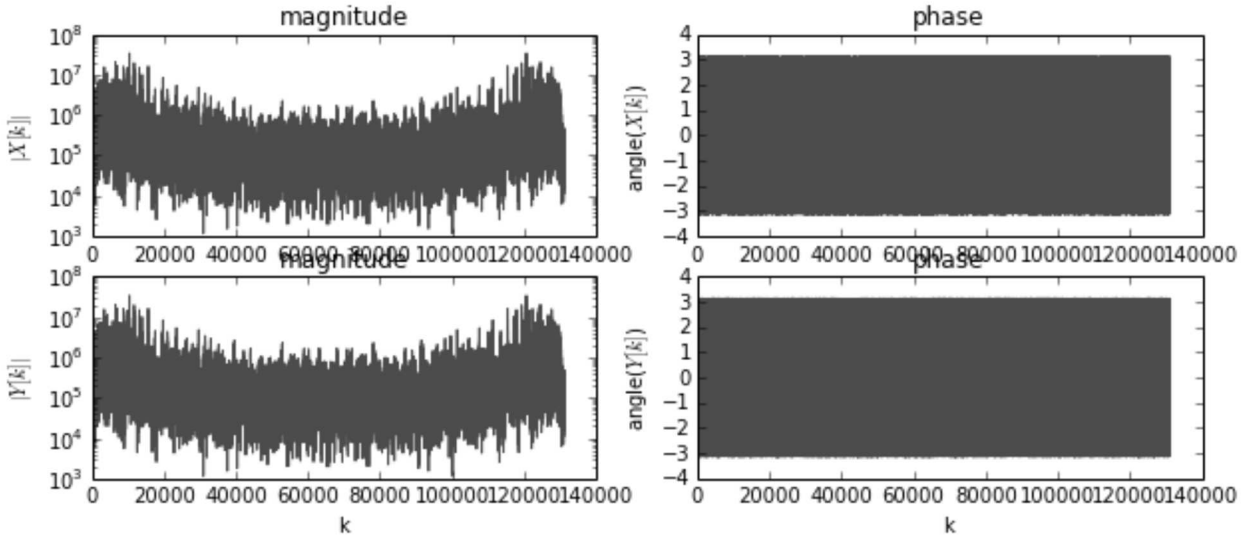
```
size soundUS = 1310720
```



In [15]:
```
# plot FFT in convenient format
# provided to save time in formatting graphs

setup_graph(title='Real', x_label='$k$', fig_size=(10,4))
plt.subplot(2,2,1)
xlabel('k')
ylabel('$|X[k]|$')
title('magnitude')
_ = plt.plot(range(0,Ne), np.abs(X[0:Ne]))
plt.yscale('log')
plt.subplot(2,2,2)
xlabel('k')
ylabel('angle($X[k]$)')
title('phase')
_ = plt.plot(range(0,Ne), np.angle(X))

plt.subplot(2,2,3)
xlabel('k')
ylabel('$|Y[k]|$')
title('magnitude')
_ = plt.plot(range(0,Ne), np.abs(Y[0:Ne]))
plt.yscale('log')
plt.subplot(2,2,4)
xlabel('k')
ylabel('angle($Y[k]$)')
title('phase')
_ = plt.plot(range(0,Ne), np.angle(Y))
```



In []:

In []: