Exer01

Exercie1

read chap01.ipynb and run examples

```
    Plots the spectrum and displays an Audio widget.

    wave: Wave object
    start: time in s
    duration: time in s
    cutoff: frequency in Hz
    """
    segment = wave.segment(start, duration)
    spectrum = segment.make_spectrum()

    spectrum.plot(color='0.7')
    spectrum.low_pass(cutoff)
    spectrum.plot(color='#045a8d')
    decorate(xlabel='Frequency (Hz)')
    plt.show()

    audio = spectrum.make_wave().make_audio()
    display(audio)
```

Adjust the sliders to control the start and duration of the segment and the cutoff frequency applied to the spectrum.

```
In [31]: from ipywidgets import interact, fixed

         wave = read_wave('92002__jcveliz__violin-origional.wav')
         interact(filter_wave, wave=fixed(wave),
                  start=(0, 5, 0.1), duration=(0, 5, 0.1), cutoff=(0, 10000, 100));
```

start 2.00

duration 2.00

cutoff 5000

(只要讀過且跑過)

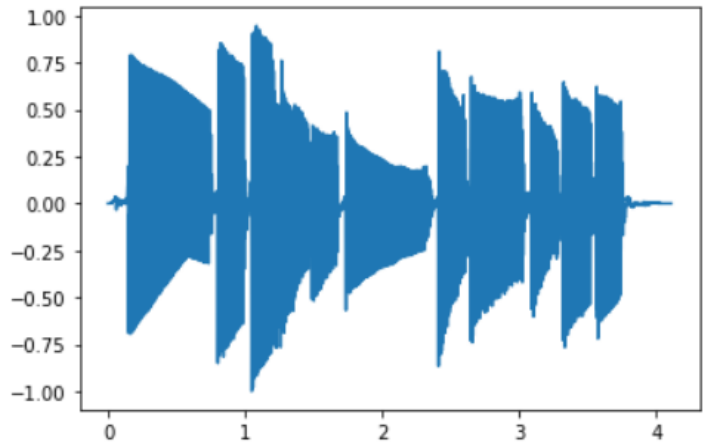
Exercie2

```
In [28]: if not os.path.exists('328878__tzurkan__guitar-phrase-tzu.wav'):
             !wget https://github.com/AllenDowney/ThinkDSP/raw/master/code/170255__dublie__trumpet.wav

In [29]: from thinkdsp import read_wave

         wave = read_wave('328878__tzurkan__guitar-phrase-tzu.wav')
         wave.normalize()
         wave.make_audio()
```

Out[29]: ▶ 0:04 / 0:04

Here's what the whole wave looks like:

```
In [30]: wave.plot()
```



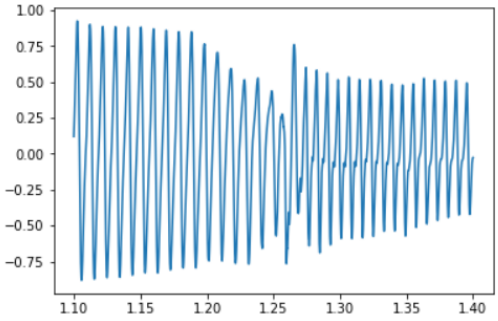
By trial and error, I selected a segment with a constant pitch (although I believe it is a chord played by at least two horns).

```
In [31]: segment = wave.segment(start=1.1, duration=0.3)
         segment.make_audio()
```

Out[31]: 0:00 / 0:00

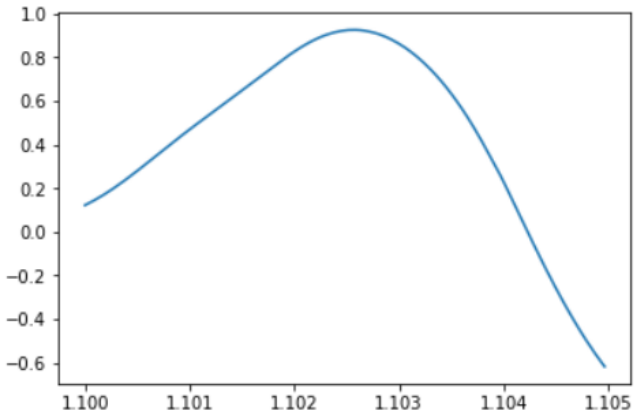
Here's what the segment looks like:

```
In [32]: segment.plot()
```

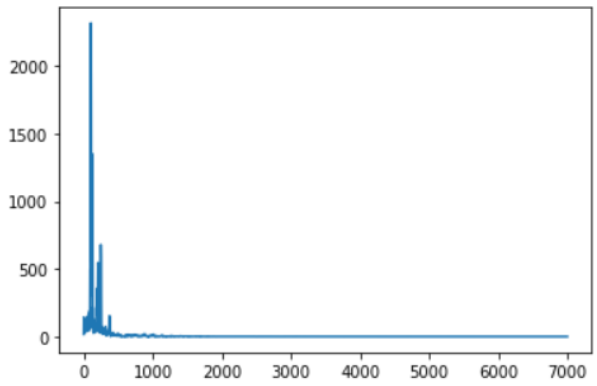


And here's an even shorter segment so you can see the waveform:

In [33]:
```
segment.segment(start=1.1, duration=0.005).plot()
```

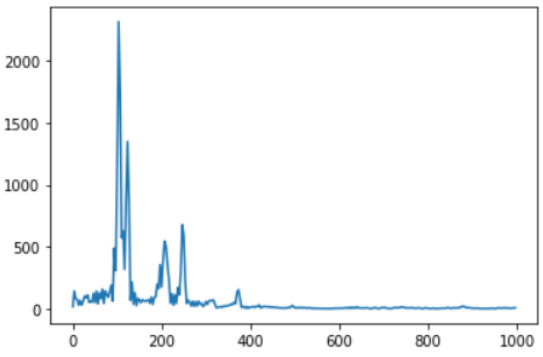


In [34]:
```
spectrum = segment.make_spectrum()
spectrum.plot(high=7000)
```



It has lots of frequency components. Let's zoom in on the fundamental and dominant frequencies:

In [35]:
```
spectrum = segment.make_spectrum()
spectrum.plot(high=1000)
```



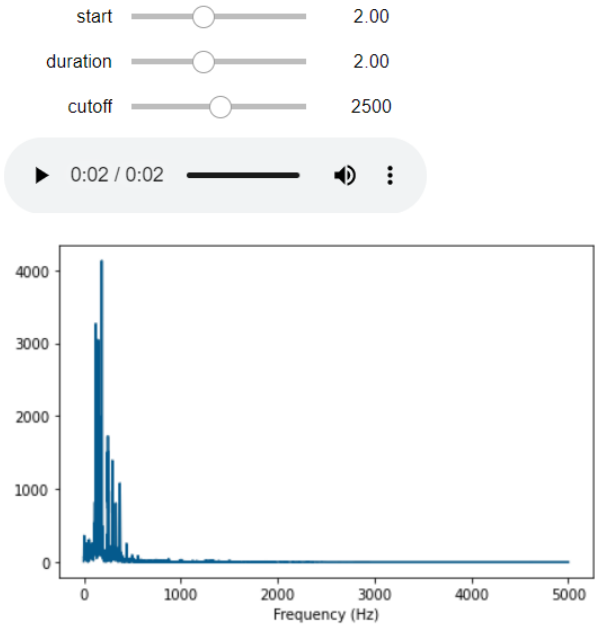
In [37]:
```
spectrum.low_pass(2000)
```

And here's what it sounds like:

In [38]:
```
spectrum.make_wave().make_audio()
```

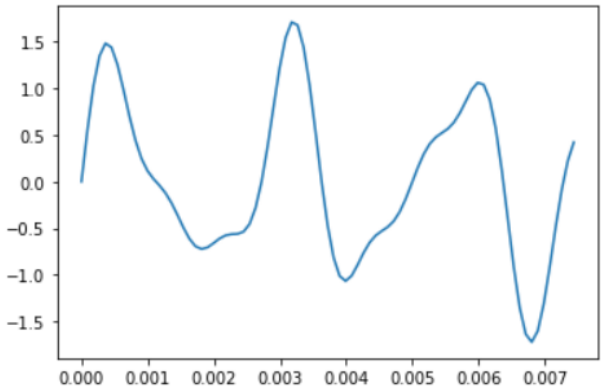
Out[38]:

In [40]:
```
from ipywidgets import interact, fixed

interact(filter_wave, wave=fixed(wave),
         start=(0, 5, 0.1), duration=(0, 5, 0.1), cutoff=(0, 5000, 100));
```



Exercie3

In [52]:
```
from thinkdsp import SinSignal

signal = (SinSignal(freq=400, amp=1.0) +
          SinSignal(freq=700, amp=0.5) +
          SinSignal(freq=1000, amp=0.25))
signal.plot()
```



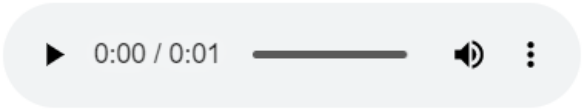
We can use the signal to make a wave:

In [53]:
```
wave2 = signal.make_wave(duration=1)
wave2.apodize()
```

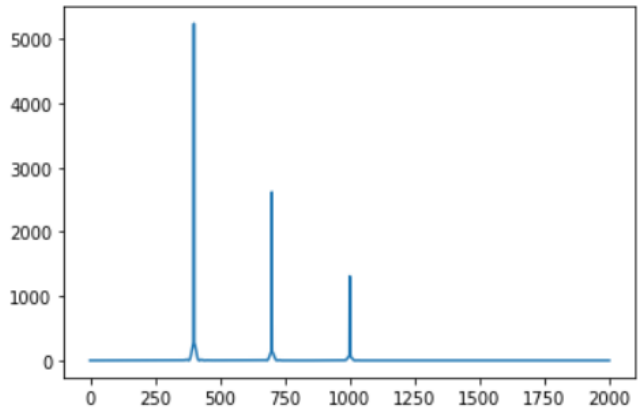
And here's what it sounds like:

In [54]:
```
wave2.make_audio()
```

Out[54]:



In [55]:
```
spectrum = wave2.make_spectrum()
spectrum.plot(high=2000)
```



If we add a component that is not a multiple of 200 Hz, we hear it as a distinct pitch.

In [56]:
```
signal += SinSignal(freq=450)
signal.make_wave().make_audio()
```

Out[56]:

Exercie4

In [48]:
```
wave3 = read_wave('328878__tzurkan__guitar-phrase-tzu.wav')
wave3.normalize()
wave3.make_audio()
```

Out[48]:

Here's my implementation of `stretch`

In [49]:
```
def stretch(wave, factor):
    wave.ts *= factor
    wave.framerate /= factor
```

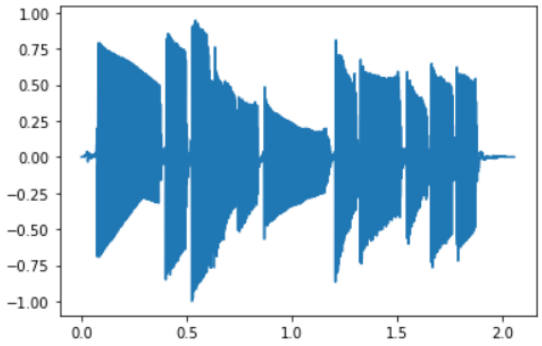
And here's what it sounds like if we speed it up by a factor of 2.

In [50]:
```
stretch(wave3, 0.5)
wave3.make_audio()
```

Out[50]:

Here's what it looks like (to confirm that the `ts` got updated correctly).

In [51]:
```
wave3.plot()
```



I think it sounds better speeded up. In fact, I wonder if we are playing the original at the right speed.