디지털 신호처리 HW3

컴퓨터과학과/201910976/안선정

In [111]:
```
from __future__ import print_function, division

%matplotlib inline

import thinkdsp
import thinkplot

import numpy as np

from ipywidgets import interact, interactive, fixed
import ipywidgets as widgets
from IPython.display import display
```

EXERCISE 1-2

Go to http:\freesound.org and download a sound sample that includes music, speech, or other sounds that have a well-defined pitch. Select a roughly half-second segment where the pitch is constant. Compute and plot the spectrum of the segment you selected. What connection can you make between the timbre of the sound and the harmonic structure you see in the spectrum?

1. wave 파일 다운로드

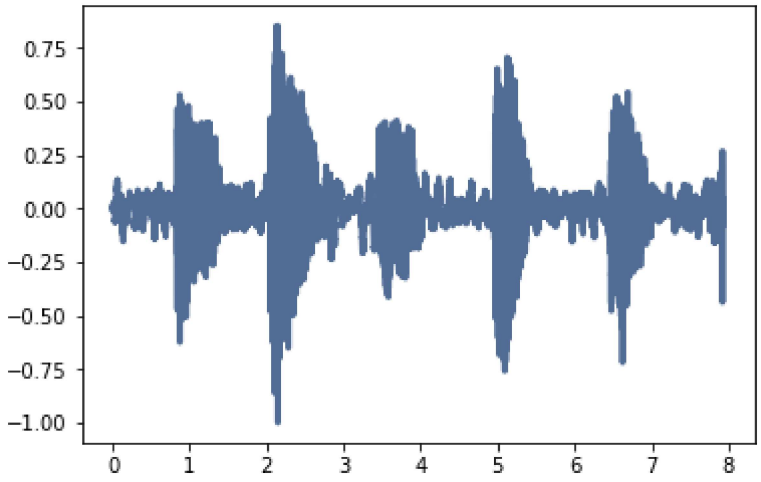
스피치를 진행한 소리를 다운 받아 실습을 진행하였다. **다운로드 링크** : https://freesound.org/people/marcgascon7/sounds/87778/

다음은 wave파일을 읽어들여 모양을 나타내고, make_audio로 음성을 출력하였다.

In [113]:
```
speech_wave = thinkdsp.read_wave('201910976_Ansunjung.wav')
speech_wave.normalize()
speech_wave.plot()
speech_wave.make_audio()
```

Out[113]:

0:00 / 0:07



2. 음 높이가 지속되는 부분 떼어내기

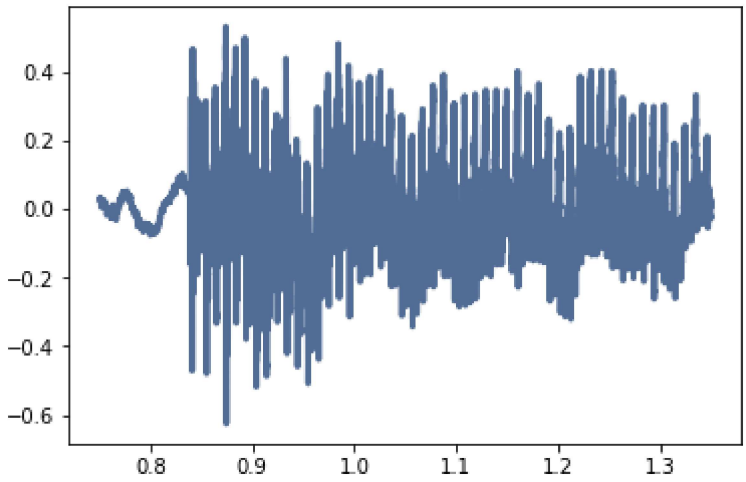
pitch가 지속되는 부분들 중 0.75~1.35까지 segment로 떼어 관찰할 것이다.

다음은 segment에 분할한 wave를 저장하여 모양을 출력한 것이다.

In [114]:
```
segment = speech_wave.segment(start=0.75, duration=0.6)
segment.plot()
segment.make_audio()
```

Out[114]:

0:00 / 0:00



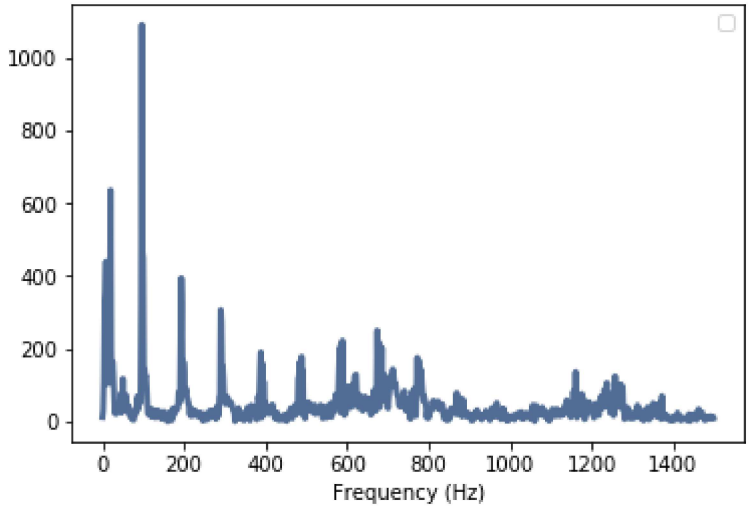
3. spectrum 출력하여 frequency 관찰하기

추출한 segment의 spectrum모양을 출력해 보기 위해 spectrum.plot()을 사용하였다.

처음에 매개변수를 주지않고 실행해보았더니 high = 1500정도 부터 값들의 성분이 거의 없다. 좀 더 자세한 그래프 모양을 위해 high = 1500 을 두고 출력하였다.

In [115]:
```
spectrum = segment.make_spectrum()
spectrum.plot(1500)
```

```
thinkplot.config(xlabel = 'Frequency (Hz)')
thinkplot.show()
```




```
<Figure size 576x432 with 0 Axes>
```

specturm.peaks() : 스펙트럼의 가장 높은 점과 그 주파수를 내림차순으로 출력한다.

In [116]:
```
spectrum.peaks()[:30]
```

Out[116]:
```
[(1090.6058015686515, 96.66666666666667),
 (635.72921475885, 20.0),
 (466.3755331825847, 98.33333333333334),
 (448.3703163204755, 100.0),
 (440.3785195697321, 8.333333333333334),
 (433.27567596801305, 11.666666666666668),
 (394.9237137088834, 193.33333333333334),
 (355.78794873781544, 6.666666666666667),
 (331.4081289994619, 95.0),
 (323.6612746877468, 195.0),
 (323.07008382400306, 5.0),
 (307.4267579601404, 290.0),
 (272.7809463739509, 291.6666666666667),
 (250.37031076026176, 673.3333333333334),
 (222.822868958367, 588.3333333333334),
 (220.09078593590195, 676.6666666666667),
 (215.15805174757116, 191.66666666666669),
 (212.97228499695515, 585.0),
 (206.56112370132251, 681.6666666666667),
 (202.99170076289636, 685.0),
 (201.0270864026436, 581.6666666666667),
 (195.2428541045345, 21.666666666666668),
 (190.8860185299469, 388.33333333333337),
 (190.3617927726597, 18.333333333333336),
 (181.63429287124026, 678.3333333333334),
 (178.51908260271557, 488.33333333333337),
 (175.91565379372918, 771.6666666666667),
 (169.31101529321538, 775.0),
 (164.9189302422054, 198.33333333333334),
 (164.57210707009472, 25.0)]
```

위 출력물을 보아

- dominant peak은 가장 높은 frequency를 보이는 약 97Hz임을 알수있다.
- 193Hz, 290Hz, 388Hz, 488Hz, 581Hz를 보아 fundamental은 약97Hz임을 알 수 있다.

실제로 fundamental을 정수배 한 값 194Hz, 291Hz, 388Hz, 485Hz, and 582Hz.과 비슷한 값을 나타낸 것을 알 수있다.

4. high_pass, low_pass, band_stop 적용해보기

다음은 low_pass, high_pass, band_stop을 차례대로 적용한 spectrum plot을 한꺼번에 출력을 하는 코드이다. plot을 출력함과 동시에 audio로 들을 수 있도록 각자 해당하는 wave를 만드는 것까지 포함하였다.

low_wave : low_pass(300)을 적용한 wave이다.

high_wave : high_pass(600)을 적용한 wave이다.

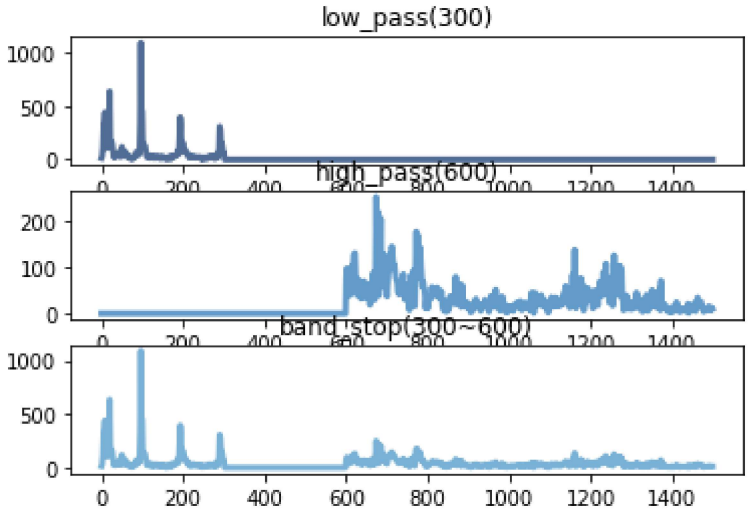
band_wave : band_stop(300, 600)을 적용한 wave이다.

In [117]:
```
import matplotlib.pyplot as plt

plt.subplot(3,1,1)
spectrum = segment.make_spectrum()
spectrum.low_pass(300)
spectrum.plot(1500)
plt.title('low_pass(300)')
low_wave = spectrum.make_wave()


plt.subplot(3,1,2)
spectrum = segment.make_spectrum()
spectrum.high_pass(600)
spectrum.plot(1500)
plt.title('high_pass(600)')
high_wave = spectrum.make_wave()

plt.subplot(3,1,3)
spectrum = segment.make_spectrum()
spectrum.band_stop(300, 600)
spectrum.plot(1500)
plt.title('band_stop(300~600)')
band_wave = spectrum.make_wave()
```



위의 spectrum들을 보아 관찰 할 수 있는 low_pass, high_pass, band_stop의 기능은 다음과 같다.

- low_pass(300) : 300Hz를 기준으로 더 높은 주파수 부분을 제외시킨다.
- high_pass(600) : 600Hz를 기준으로 더 낮은 주파수 부분을 제외시킨다.
- band_stop(300,600) : 300~600Hz의 주파수 부분을 제외시킨다.

5. filter를 적용한 wave소리 듣기

low_pass를 적용한 low_wave 소리

In [118]:
```
low_wave.make_audio()
```

Out[118]:

0:00 / 0:00

low_wave의 소리는 아주 두껍고 낮은 음만 들리는 것을 확인 할 수있다. : 주파수가 0~300Hz에 해당한다.

high_pass를 적용한 high_wave 소리

In [119]:
```
high_wave.make_audio()
```

Out[119]:

0:00 / 0:00

high_wave의 소리는 비교적 얇고 높은 음으로 이루어진 것을 확인 할 수 있다. : 주파수가 600Hz 이상에 해당한다.

band_stop를 적용한 band_wave 소리

In [120]:
```
band_wave.make_audio()
```

Out[120]:

0:00 / 0:00

band_wave 의 소리는 low_wave보다 얇고 높은 음으로, high_wave보다 두꺼운 소리가 나는 것을 확인할 수 있다. : 주파수가 300Hz이하 600Hz 이상에 해당한다.