영어음성학 수업 내용 정리

2018130891 영어영문학과 신여진

1.English consonants and vowels

English consonants: p, t, k, b, d, g, m, n, f, v, θ, ð, s, z, ʃ, ʒ, l, w, r, j, h ...

English Vowels: e, æ, i, ɔ, o, u, ə, ʌ, ɑ ei, ai, au, ɔi, ou, iər, uər ...

모든 소리는 voiced(유성음)과 voiceless(무성음)으로 나눌 수 있다. 모든 모음과 일부 자음은 유성음이고 일부 자음은 무성음이다.

또한 소리는 nasal을 통해 나는 소리와 oral을 통해 나는 소리로 구분할 수 있다.

2.Phonetics 음성학

Ex) "가"라는 소리를 10번 낼 때 phonetics에서는 physical하게 분석하여 다 다른 소리라고 하는 반면 phonology에서는 cognitive하게 분석하여 다 같은 소리라고 할 수 있음

Articulatory phonetics 조음음성학 소리를 만드는 방법, Acoustic phonetic 음향음성학 바람의 원리

Auditory phonetics 청각음성학 귀로 들리는 원리

3.Articulation

Vocal tract에는 크게 네 가지가 있다 Nose, ear pharynx, larynx

그중 vocal tract(upper): lip, teeth, hard palate, soft palate(velum), uvula, pharynx wall, alveolar ridge, pharynx wall larynx,

그중 vocal tract(lower): lip tongue tip, tongue blade, tongue front, tongue back, tongue root, tongue center, epiglottis

vocal tract (upper)부분은 움직일 수 없고 vocal tract (lower)을 움직여서 소리를 낸다

4.Speech에서 아주 중요한 세가지!

4-1 oral nasal process

Nasal sound와 아닌 것을 구분하는 process로, Nasal sound 비음의 경우, nasal tract가 열려있고 velum은 lowered되어있다

반대의 경우 velum이 raised되어있고 nasal tract는 닫혀있다

4-2 phonation process

유성음 무성음을 구분하는 process로 larynxf를 확 열면 무성음 확 닫으면 유성음이다

4-3 articulatory process

Constrictors(Lips, tongue tip, tongue body)를 활용해 소리를 만드는 process이다

5.control of constrictors

Constriction location :어디서 장애가 발생하는가?

-Lips가 조금 앞으로가면 bilaibial, 조금 뒤로가면 labiodental

-Tongue body가 조금 앞으로가면 palatal에서, 조금 뒤로가면 velar에서

-Tongue tip이 조금 앞으로 가면 dental에서 뒤로가면 palate-alveolar에서

Constriction degree: 어느정도로 조음과정에서 장애가 발생하는가

stops ex)p,t,b,d / fricatives ex)s,z,f,v... / approximants ex)m,n... /vowels

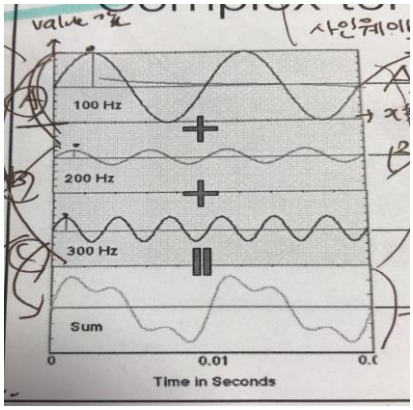
6.praat에서

Intensity, Pitch 남녀에 따라 pitch setting 해주는 것 중요 남자 65-200HZ 여자는 보통 145-275HZ

Forman 모든 사람이 'a'녹음하면 다들 비슷한 Forman임

이 세상의 모든 sound포함 signal은 여러 sine wave의 결합으로 표현된다. 당연히 반대로, 여러 다른 sine wave들의 합은 복잡한 소리가 된다.

-그래프 분석해보기



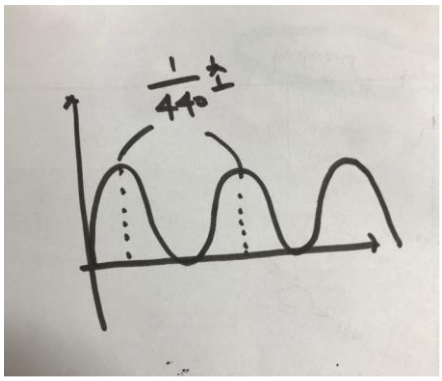
A,B,C모두 simplex tone, 사인웨이브라고 생각하면 됨

-B가 A에 비해 두 배 빠르다

-A를 분석해보자면 magnitude, 진폭 크고 frequency가 작은 것으로 보아 저음에 해당한다

-x축은 시간, y축은어떤 value 값으로 보면 됨

-그래프 분석 후 praat을 키고 440의 tone frequency를 가지고 1의 amplitude를 가진 sine wave를 만들었음



여러 다른 simplex tone중 제일 slow한 simplex tone의 frequency가 우리말의 음높이와 비슷하다

-voice source만들기

100,200,300,400...HZ 점점 올리고 1, 0.95, 0.90, 0.85...amplitude 점점 줄여서 10개를 만든 후 shift 키 이용해서 10개 한꺼번에 선택 combine to stereo – convert to mono 후 반복되는 패턴보기

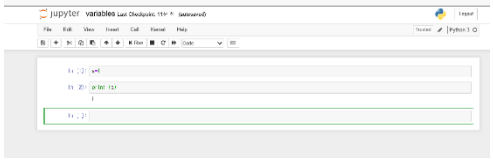
인지심리학적으로 100hz play할 때랑 비슷하게 들림을 알 수 있음

-등차간격으로 보이던 source spectrum이 우리의 입모양을만나 봉우리 모양으로 변하는 것을 볼 수 있었음. 가장 위로 튀어나온게 first formant, 그 다음이 second formant...인 것을 알 수 있었다. 가장 먼저의 막대기는 f0라고 부르는 것이다.

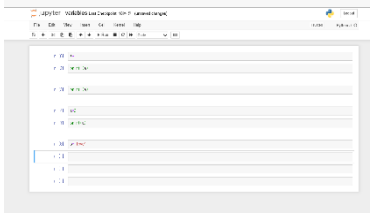
-또한 같은 모음에 대해서 누구나 비슷한 formant를 가지고 있는 것을 볼 수 있었다.

변수라고 하는 그 그릇에다가 정보를 assign해주기 (variable assignment)
자동화 기계화에서 직관적으로 우리가 떠오르는 것이 —할 때 —하라 이런 조건이 붙는 것은 당연히 필요함 이런 conditioning에 대한 문법이 필요함 (if conditioning)
자동화의 가장 중요한 것 중 하나는 여러 번 반복하는 것이다 이것은 (for loop)
문법에서 가장 중요한 것이 함수이다

variable이라는거 하나의 정보, 정보의 종류 하나는 숫자, 하나는 글자
in 옆에 a=1치고 run 눌러보기 (a에 1을 넣은거임) 컴퓨터 language에서 =은 같다의 표시가 아님. 오른쪽에 있는 정보를 왼쪽의 variable로 assign 한다는 뜻임 문자와 숫자의 순서 역시 바뀌면 안됨
그 밑에 print (a)치고 run 누르면 밑에 1 뜬다



여기서 print 라는 함수는 a(입력)라는 변수를 넣으면 1이 나오는 그런 함수임. 즉, 함수이름을 치고 괄호안에 입력값을 넣어주면됨 print 함수, a 입력값1 프린트된 값
(a=1 줄의 in누르면 파란색으로 변하는데 이때 a를 누르면 위에 칸 만들어지고 b누르면 밑에 칸 이만들어진다 줄을 없애고 싶다면 x (그 줄을 cell이라고 함)



변수를 문자를 넣어보자
문자 양옆에 작은따옴표 넣는거 핵심 shift하고 enter누르면 실행이 됨
그다음 print (b) 하고 run누르면 love 나옴

love=2라고 해보자
love라는 변수에 숫자 2를 넣은거야!
그리고 b=love라고 하면
b라는 변수넣으면 b-love-2메커니즘따라 2나옴

대괄호 해서 안에 많은 숫자 넣어줄 수 있다 그건 list라고 부른다
type (a)하고 run해주면 list라고 나온다 이걸 통해 얘가 단순 문자 숫자인지 list인지 구분할 수 있다
숫자 문자 리스트 변수로 넣을 수 있다

이외에도 type함수를 통해 문자는 str로 실수값은 float로 보안이 더 잘되는 tuple이 나오게 할 수

도, 표제어와 설명에 관련된 dict가 나올수도 있다는 것을 알 수 있었음.

-variable 추가와 string

a[0]에서 괄호의 역할은 a의 내부에 들어가서 무언가를 가져오라는 의미임 즉 어떠한 variable의 내부정보에 들어가려면 반드시 []가 필요하다

a라는 int variable이 a=float(a)의 함수에 들어가면 float으로 바뀜

dict에서 pair의 앞부분을 index로 사용한다 즉 []안에 들어올 것으로 앞부분 표제어를 사용한다는 말

s='abcdef'로 놓고

s[1:3]는 첫번째에서 세번째 직전까지라는 말, s[1:] 첫번째에서 끝까지라는 말, s[:3]은 처음부터 세번째 직전까지라는 말 s[:]전부다라는말

.upper()쓰면 해당 variable이 전부 대문자로 바뀐다

rinex라는 함수는 찾고 싶은 것이 그 문장에서 중복으로 있다면 마지막 것을 기준으로 찾아준다

.strip()함수는 string속에 번잡한 것들을 제거하고 순수한 것들만을 남겨준다

Tokens함수는 ' '사이의 것을 기준으로 문장을 쪼개준다

join함수는 ' ' 사이의 것을 이용하여 token함수로 쪼개진 것을 붙여준다

replace함수의 경우는 문장 속의 특정 단어를 내가 지정하는 단어로 바꿔준다

-for loop

```
a=1,2,3,4
for I in a:
```

의 해석 in뒤에 있는 것을 하나씩 돌려서 한번한번 넣을때마다 i로 받아서 무언가를 해라!

-range라는 함수 : 리스트를 만들어주는거임

예를 들어 range(4)이면 함수를 통해 네개의 인덱스를 지닌 리스트를 만들어준 것, i는 0~3까지 차례대로 들어감 결국 a의 0,1,2,3번째가 차례로 나오게 되는 것

-enumerate라는 함수: 번호를 매겨주는 함수, 리스트가 있다면 리스트 안의 것들에 번호를 매겨주는 함수임

-if conditioning

```
a=0
if a == 0:
```

print (“yay”)

out yay

만약 if a!=0 인경우 (이때 !는 아니라면 이라는거임) yay가 나오지 않음

Numpy class 9

-가로는 행 세로는 열

-직사각형에 각각 넣어놓음

-숫자가 쭉 나열되어있는 것 벡터, 모든 데이터는 벡터의 모습으로 되어있어야한다

-영상은 몇차원인가 - 2차원

-컬러 이미지는 몇차원인가- 3차원

-import numpy as np하면 이후부턴 np만 써도 numpy로 역할가능

Numpy class 10

-Numpy 라는 라이브러리 속에는 package A,B,C,,그리고 그 안에 a,b,c더 들어있을 수 있다

Ex)import Numpy.a.d 여기서 .의 의미는 ‘~의 안의’라는 것

From numpy import a 이렇게도 사용가능

```
In [11]: np.arange(5)
Out[11]: array([0, 1, 2, 3, 4])

In [12]: np.arange(0,10)
Out[12]: array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])

In [13]: np.arange(0,10,2)
Out[13]: array([0, 2, 4, 6, 8])
```

뒤에붙은 2는 증가분을얘기하는 것이다

```
In [14]: np.linspace(0,10,6)
Out[14]: array([ 0.,  2.,  4.,  6.,  8., 10.])
```

0부터 10까지 사이를 6개로 똑같이 나눠어준다. 처음과 끝을 포함해서 . 리니어라는 말은 똑같이라는 의미를 지니고있음 그래서 차이를 똑같이해서 나눠라 이런것

```
In [15]: np.linspace(0,10,7)
Out[15]: array([ 0.        ,  1.66666667,  3.33333333,  5.        ,  6.66666667,
                 8.33333333, 10.        ])

In [19]: x=np.array([[1,2,3],[4,5,6]])
         x
Out[19]: array([[1, 2, 3],
                [4, 5, 6]])

In [29]: x=np.array([[[1,2],[4,5],[8,9]],[[1,2],[4,5],[8,9]]])
         x
Out[29]: array([[[1, 2],
                 [4, 5],
                 [8, 9]],
```

이건 삼차원임 괄호의 개수에 따라 차원수 차이남

```
In [25]: x=np.array([[1,2],[4,5],[8,9]])

In [26]: x.ndim
Out[26]: 2
```

차원수를 말해주는 것

```
In [27]: x.shape
Out[27]: (3, 2)

In [ ]: 직사각형의 모양을 말해주는 것

In [30]: x.astype(np.float64)
Out[30]: array([[[1., 2.],
                 [4., 5.],
                 [8., 9.]],

                [[1., 2.],
                 [4., 5.],
                 [8., 9.]]])

In [31]: np.zeros_like(x)
Out[31]: array([[[0, 0],
                 [0, 0],
                 [0, 0]],

                [[0, 0],
                 [0, 0],
                 [0, 0]]])
```

0을가지고 아까 x처럼 만들어라

```
In [32]: x*0
Out[32]: array([[[0, 0],
                 [0, 0],
                 [0, 0]],

                [[0, 0],
                 [0, 0],
                 [0, 0]]])
```

이곱하기를 활용해서도 할수있음

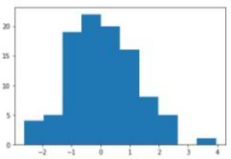
```
In [33]: data=np.random.normal(0,1,100)
         print(data)
```

```
[ 1.6097714e-02 -8.7664533e-01  1.3135180e+00  5.1552704e-04
  6.9745784e-01 -1.1301545e+00 -7.6607452e-02 -5.3303826e-01
  3.2851427e-01  5.2013180e-01  4.7760327e-01 -1.6008323e+00
 -3.8725260e+00  2.2115818e-01  8.0435074e-01 -8.5818550e-01
  7.5322524e-01 -1.7888998e+00 -4.1492302e-01  1.1587402e+00
 -1.4952647e-01 -1.0831991e+00  4.5102259e-01  5.1696219e-01
 -6.0873492e-01 -1.4485877e+00  7.1399750e-01 -6.9084542e-01
 -5.8049273e-01  1.4234407e+00 -2.0560333e-02  1.3159956e+00
 -6.9737239e-02 -1.8685348e+00  9.4245862e-01 -1.2287856e+00
  1.1851783e+00 -1.1000323e+00 -5.7394570e-01 -1.7143656e+00
  1.2626075e+00  6.5589492e-01 -1.3745218e-01  1.8905782e+00
 -6.1369387e-01 -1.3612016e+00  1.2110024e+00 -1.0020827e+00
  5.1212557e-01 -1.0148404e-01  5.8197720e-01 -5.3318103e-01
  3.3493790e-01 -1.3297714e+00 -6.0076380e-01  2.8698625e+00
  3.6718846e-01  7.0443626e-01  4.6493644e-01  1.1488001e+00
  4.9724757e-01  6.9346112e-03  1.4634626e+00 -2.5499986e-01
 -7.5301904e-01 -1.3136505e+00  3.9802594e-01 -1.4818790e-01
 -1.4894102e+00  7.3074239e-01  6.6325602e-01  8.0068441e-01
 -4.4344659e-01  1.1741691e+00  3.8970133e-01 -5.8545954e-01
  1.9600389e+00 -1.6259957e+00  1.5700131e+00  7.1203041e-01
 -1.5880705e-01  6.7406490e-01 -6.0309459e-01 -1.0257497e+00
  8.0271597e-01  1.4678868e+00 -1.1696089e+00 -5.7627549e-01
  5.9516919e-01 -1.5376639e+00  1.1520357e+00  1.7299539e+00
  3.4505210e-01  2.6515249e-01 -1.6990596e+00  1.1131948e-01
  3.6387384e-03 -6.3013213e-01 -2.0508754e-01 -1.3811239e+001
```

```
In [34]: data.shape
Out[34]: (100,)

In [35]: data=np.random.normal(0,1,100)
         print(data)
         plt.hist(data, bins=10)
         plt.show()
```

```
[-1.03437417  0.12782066  0.59194459 -0.85487658 -0.74762451 -1.00006256
  0.63592498 -0.65099497 -1.31423015 -0.60587934  1.13433662  1.20924875
 -0.82749147 -0.97072013  1.61915832 -2.62432454  1.43013872 -2.11146122
 -0.42175861  0.52799329  1.73712487  1.26407967 -0.94721281 -1.08860789
 -0.09114118  0.45040471  3.95164108  1.28672979 -0.42470043 -1.06745064
 -0.54046628  0.30699613  0.06599612  0.32454811  2.36388367  0.08423376
 -0.46386621  1.01792987  0.15540762 -0.71193566  0.8892813  -1.73254548
  2.33482018 -0.97130642  2.04514719 -0.07723695 -0.05603595  1.26472103
 -0.17526774 -0.82796345  1.59365723  0.97462118  0.03594349  0.87682116
  0.91864435  1.39436511  0.23213123 -0.32748278  1.10015099  0.91229147
 -0.13723115  0.63589112  0.64156681 -0.93871245 -0.7034705   1.32160222
 -0.56315752 -1.44517369 -0.44659316  1.257145   -1.19046137  1.82054352
 -0.23142295 -0.32599174 -0.77850018 -0.90820686 -1.0920982  -0.63611956
  0.24864977 -1.73306705 -0.99429442 -1.11612187 -0.04393893  2.40161037
  0.57231754 -1.43419167  1.12159147  0.17608516 -0.08736883  0.23253677
  1.33851777  1.25040818  0.32835116 -2.55913347 -0.60492091  0.57342463
  0.74803038 -2.34675569 -0.6066341   2.2211763 ]
```



```
In [36]: x=np.ones([2,3,4])
         x
Out[36]: array([[[1., 1., 1., 1.],
                 [1., 1., 1., 1.],
                 [1., 1., 1., 1.]],

                [[1., 1., 1., 1.],
                 [1., 1., 1., 1.],
                 [1., 1., 1., 1.]]])

In [38]: y=x.reshape(-1,3,2)
         y
Out[38]: array([[[1., 1.],
                 [1., 1.],
                 [1., 1.]],

                [[1., 1.],
                 [1., 1.],
                 [1., 1.]],

                [[1., 1.],
                 [1., 1.],
                 [1., 1.]],

                [[1., 1.],
                 [1., 1.],
                 [1., 1.]]])

In [40]: np.allclose(x.reshape(-1, 3, 2), y)
Out[40]: True
```

assert 돌라두돼

```
In [41]: a=np.random.randint(0,10,[2,3])
         b=np.random.random([2,3])
         np.savez("test",a,b)
```

```
In [41]: a=np.random.randint(0,10,[2,3])
         b=np.random.random([2,3])
         np.savez("test",a,b)

In [42]: del a,b
         %who  # Print all interactive variables
No variables match your requested type.

In [43]: npzfiles=np.load("test.npz")
         npzfiles.files
Out[43]: ['arr_0', 'arr_1']

In [44]: arr=np.random.random([5,2,3])

In [45]: print(type(arr))
         print(len(arr))
         print(arr.shape)
         print(arr.ndim)
         print(arr.size)
         print(arr.dtype)
<class 'numpy.ndarray'>
5
(5, 2, 3)
3
30
float64
```

comparison위의 세줄 패스

```
In [46]: a = np.arange(1, 10).reshape(3,3)
         b = np.arange(9, 0, -1).reshape(3,3)
         print(a)
         print(b)
[[1 2 3]
 [4 5 6]
 [7 8 9]]
[[9 8 7]
 [6 5 4]
 [3 2 1]]

In [47]: a==b
Out[47]: array([[False, False, False],
                [False,  True, False],
                [False, False, False]])

In [48]: a.sum()
Out[48]: 45

In [ ]: a.sum(axis=0)

In [49]: a.sum(axis=0)
Out[49]: array([12, 15, 18])
```

11..5

Sound를 직접 만들어보자, Sinusoidal phasal

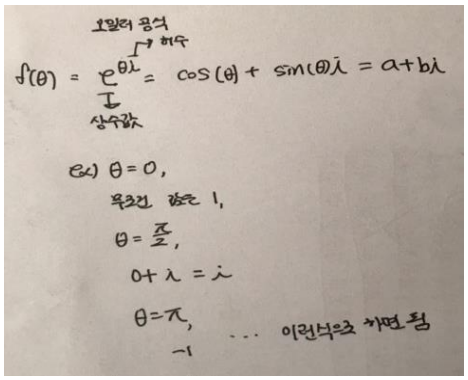
싸인하고 코사인에 들어가는 입력값은 degree가 아니라 radian이 들어가야한다

Degree 0 180 360.....720

Radian 0 π 2π 4π (2파이랑 4π는 똑같음)

Ex) cos (1.5 π)=0

X값에 따라	0	0.5π	π	1.5π	2π
Cos값	1	0	-1	0	1
Sine값	0	1	0	-1	0



-오일러 공식

-가장 많은 수를 포함하는 수체계는 복소수 a+bi

-지금까지 sine(θ)값은 전부 실수였기에 표기에 문제가 없었지만 이런경우 허수가 있는 복소수는 어떻게 표시해줄까?

복소평면으로 표시해줄 수 있음. X축을 a, y축을 b로 놓고 보자.

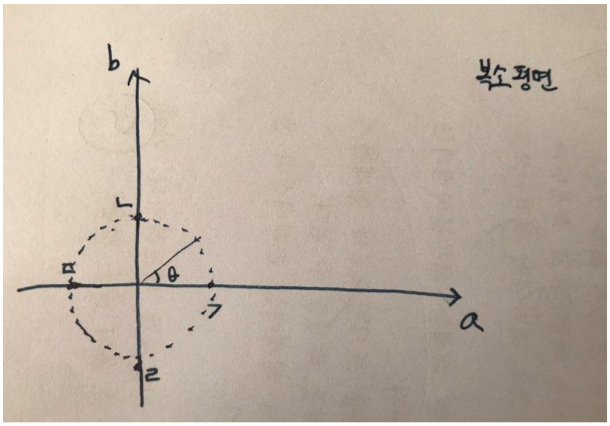
f(θ),

ㄱ. θ=0일때 1, a=1, b=0

ㄴ. θ=0.5π일 때 I, a=0, b=1

ㄷ. θ=π일 때 -1 a=-1, b=0

ㄹ. θ=1.5 π일 때 -i a=0, b=-1



벡터의 정의: 숫자열! a+bi도 결국 다 벡터임 위의 복소평면의 그래프. θ값의 증가에 따라 시계반대방향으로 뱅뱅 돈다

프로젝션! 위에서 보면 엑스축에서의 실수의변화만을 볼 수 있고

오른쪽에서 와이축만 보면 허수의 변화만을 볼 수 있음

실수만 볼때는 코사인만 보면되고 허수만 볼때는 사인만 보면 됨

(b축이 0부터 올라갔다가 내려가는 것 보면 딱 사인그래프 a축이 1부터 내려가는 것을 보면 딱 코사인그래프)

11.12

(지난시간 복습)

Sine wave 위해서 시간먼저 생성하기, theta값만으로는 만들기 어렵다

-진폭은 y축 0을 기점으로 세기

-2부터 2까지 굴곡진것의 진폭은 2임

-!pip install sounddevice 를 해서 sounddevice library를 설치해주는 것임

그리고 나서 sd.play(c.real.sr)해주면 소리가 나올 것임

-sampling rate이 100hz라고 생각해보자 우리가 표현할 수 있는 숫자의 개수가 1초에 100개

이걸가지고 1hz를 우리가 표현할 수 있을까?Yes

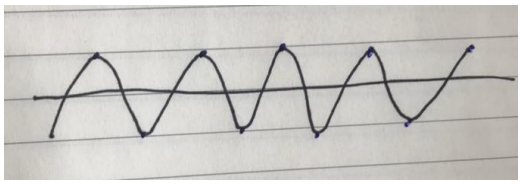
2hz도 가능할까?Yes

10000hz도 가능할까? No

Sampling rate이 충분히 있어야 그만큼의 숫자도 표현할 수 있는 것임

주어진 개수의 표현할 수 있는 주파수는 maxim up의 반밖에 안됨

예를들어 점 열개면 수직선 기준 위에 다섯개 아래 찍는거 생각해보기



Sr=10hz/2,Fr=5 hz, Nyquist hz, frequency는 무조건 sr의 반절임

-사람이 들을 수 있는 가청주파수 20000hz임

-s=s+amp+np.sin(theta)

에서 전항의 s는 후항의 s값에 이후의 식 다 실행해준 새로운 s값이다 라는 뜻임

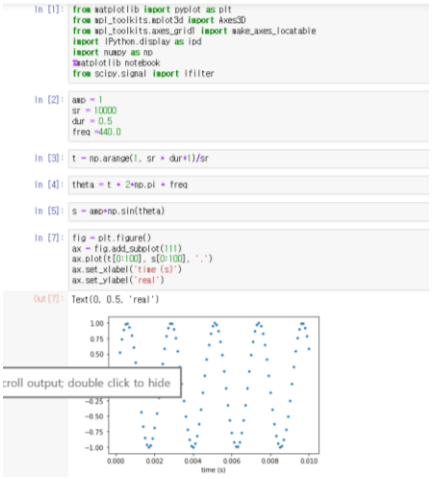
11.14일

-formant라고 하는게 모음의 종류를 정해주는 것 상기하기

-지금까지는 amplitude들이 똑같았는데 이제 그 차이를 달리해줄거임

-산맥을 직접 만들어 볼 것임

-기본설정할때 *표시 실행을 말해주는 것



여기까지의 과정

기본으로 세팅해준 후,

Ipd. Audio(s, rate=sr)하면

두-하는 소리가 나옴, 라소리임

잘못된 원본으로 표시되는 경우 일단은 그냥 넘기기

-이후 440hz를 880hz로, 1760hz로 바꿔보기 그러면 소리가 한결 높은 라가 나옴, 옥타브만 뛰는 것을 알 수 있음

-s=~이후의 식에서 sin을 cos로 바꾸면 그래프의 시작점 달라짐 그러나 그거에 따라 소리가 달라지지는 않음

-sin과 cos는 shape자체는 비슷하나 좌우 이동을 한 느낌임 90도의 차이가 있음

-sin cos의 phase shift의 차이는 우리는 인식하지 못한다

우리가 sensitive한 것은 frequency이다

-c = amp*np.exp(theta*1j)

C 설정해두고 , projection '3d'로 설정해주고 z축설정도 해주면 3차원 나옴을 알 수 있음.

```
def hz2w(F, sr):
```

에서 return은 출력을 의미한다

```
RG = 500  # RG is the frequency of the Glottal Resonator
BWG = 60  # BWG is the bandwidth of the Glottal Resonator
a, b=resonance(sr,RG,BWG)
s = lfilter(b, a, s, axis=0)
ipd.Audio(s, rate=sr)
```

sr은 우리가 아는 sampling rate, rg산맥의 위치를 적어주면 됨. BWG는 산맥의 뚱뚱함과 뾰족함을 정해주는 것임

-rg, bwg 여러쌍 만들어서 초반거와 비교해보니 위로갈수록 옅어짐을 볼 수있었다

-우리의 입술을 통해서 소리가 공명되어 나가는 것