

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, ɪ, ʊ, o, u, ə, ʌ, ɑ ei, ai, au, ɔi, ou, iə, uə ...

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 larynx를 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 bilabial, 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

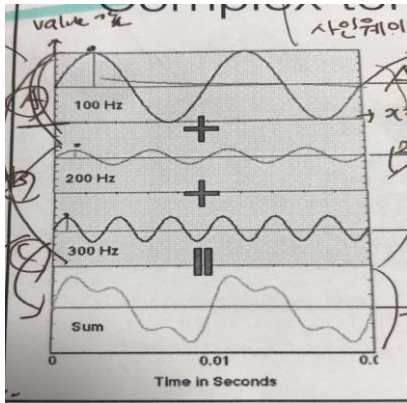
6.praat에서

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

Formant 모든 사람이 'a'녹음하면 다들 비슷한 Formant임

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

-그래프 분석해보기



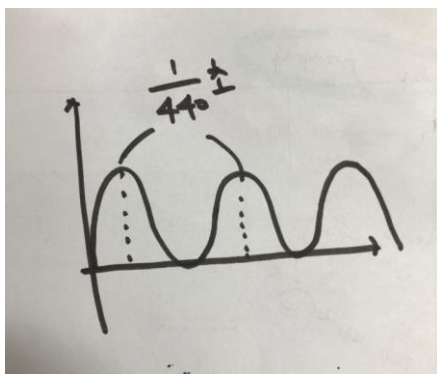
A,B,C모두 simplex tone, 사인웨이브라고 생각하면 됨

-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 모임에 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)

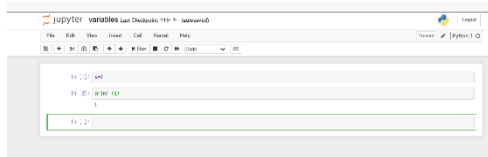
자동화의 가장 중요한 것 중 하나는 여러 번 반복하는 것이다 이것은 (for loop)

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 갈다의 표시가 아님. 오른쪽에 있는 정보를 왼쪽의 variable로 assign 한다는 뜻임 문자와 숫자의 순서 역시 바뀌면 안됨

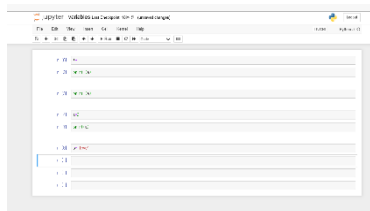
그 밑에 print (a)치고 run 누르면 밑에 1 뜬다



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

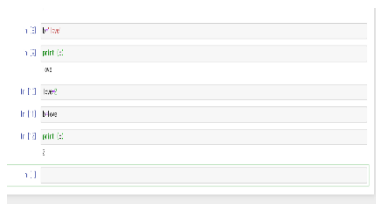
이만들어진다 줄을 없애고 싶다면 x (그 줄을 cell이라고 함)



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 애가 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 l 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 != 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 이후부터 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

```
numpy

In [2]: import numpy as np
import matplotlib.pyplot as plt

이렇게 as를 사용해서 간단히 표현해줄 수 있다는 것 from matplotlib import pyplot as plt 로도 표현가능하다는 것

In [3]: np.empty([2,3],dtype='int')
Out[3]: array([[ 0, 1072168960,  0],
               [1072168960,  0,  0]])

좀 크긴하지만 data type int로 정해놔기에 int가 data값으로 나옴

In [6]: np.zeros([2,3])
Out[6]: array([[0.,  0.,  0.],
               [0.,  0.,  0.]])

zeros로 안에 채울거 이미 정해져있음 함수를 해석하자면 np안에 zeros를 활용해서 2열 3행 만들어라

In [7]: np.array([[0,0,0],[0,0,0]])
Out[7]: array([[0, 0, 0],
               [0, 0, 0]])

In [8]: np.ones([2,3])
Out[8]: array([[1.,  1.,  1.],
               [1.,  1.,  1.]])

In [9]: np.ones([2,3], dtype=int)
Out[9]: array([[1, 1, 1],
               [1, 1, 1]])

In [ ]: dtype=int해줌으로써 위에 .을배준것이다
```

```
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 증가분을제거하는 것이다

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

0부터 10까지 사이를 6개로 똑같이 나누어준다. 처음과 끝을 포함해서. 리니어라는 말은 똑같이하는 의미를 지니고있음 그래서 차이를 똑같이해서 나눠라 이
런것

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 삼자원일 물포의 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 각각 x처럼 만들어라

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

어금하거를 활용해서도 할수있음

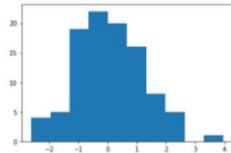
In [33]: data=np.random.normal(0,1,100)
print(data)

[ 1.60977114e-02 -0.76649533e-01  1.31351803e+00  5.19627045e-04
  6.97467784e-01 -1.13016451e+00 -7.66074622e-02 -5.33030331e-01
  3.26914327e-01  5.20131805e-01  4.76027291e-01 -1.60032330e+00
 -3.87252090e-00  2.21159198e-01  8.04302074e-01 -9.59185009e-01
  7.33323444e-01 -1.79889693e-01 -1.49920302e-01  1.15074035e+00
 -1.49925472e-01 -1.08319910e-01  4.51022596e-01  5.16962195e-01
 -6.08754821e-01 -1.44659776e-00  7.13397900e-01 -5.90645432e-01
 -5.90467739e-01 -4.24344074e-00 -2.15603333e-02 -1.31939956e+00
 -6.97373956e-02 -1.86653466e+00  9.42458639e-01 -1.22879556e+00
  1.16517493e+00 -1.00033230e-00 -5.73946701e-01 -1.71439551e+00
  1.26307950e+00  6.55949339e-01 -1.27462189e-01  1.89676709e+00
 -6.13959387e-01 -1.36120916e+00  1.21100243e+00 -1.00202787e+00
  5.12125570e-01 -1.01494049e-01  5.01977207e-01 -5.33118103e-01
  3.34854780e-01 -1.32977114e+00 -6.07963300e-01  2.86692356e+00
  3.67189452e-01  7.04432956e-01  4.64936044e-01  1.14800191e+00
  4.97247576e-01  6.93461212e-03  1.46346295e+00 -2.54999999e-01
 -7.33019430e-01 -1.31360549e+00  3.98029342e-01 -1.49197956e-01
 -1.49941024e+00  7.30742329e-01  6.63296630e-01  8.00694441e-01
 -4.43446959e-01  1.17416914e+00  3.69701335e-01 -5.85149554e-01
  1.60038995e+00 -1.62596539e+00  1.57001319e+00  7.12933041e-01
 -1.59807955e-01  6.74064900e-01 -6.03094959e-01 -1.02574873e+00
  8.02712597e-01  1.46788620e-01 -1.16990099e+00 -5.76275493e-01
  5.46169198e-01 -1.53766392e+00  1.15205265e+00  1.72966539e+00
  3.45621070e-01  2.65152491e-01 -1.69905999e+00  1.11319484e-01
  3.63679494e-03 -6.30322131e-01 -2.05097544e-01 -1.38112932e+00]
```

```
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  0.12762066  0.59194459 -0.85487959 -0.74762451 -1.00006256
  0.62962498 -0.65099497 -1.31423016 -0.60697934  1.13420662  1.20624875
 -0.62749147 -0.97072013  1.61919932 -0.62435454  1.43013972 -2.11140122
 -0.42175884  0.52798029  1.73712487  1.26407367 -0.94717291 -0.08607999
 -0.09141118  0.45940471  3.95164108  1.29672979 -0.42470043 -1.06745064
 -0.54048629  0.30999613  0.09599612  0.32454811  2.36398397  0.09423376
 -0.46396321  1.01752997  0.19540782 -0.71192666  0.89289139 -1.73264648
  2.33482018 -0.97130642  2.04514719 -0.07723695 -0.06603595  1.26472103
 -0.17535774 -0.02796345  1.55966723  0.97462118  0.03994349  0.67682116
  0.91644495  1.39436511  0.23219123 -0.52748279  1.10015099  0.91239147
 -0.13729115  0.63599112  0.64156981 -0.93871245 -0.7034705  1.32160222
 -0.56315752 -1.44517369 -0.44693916  1.257145 -1.1946137  1.82064362
 -0.23142295 -0.23599174 -0.77950019  0.99326688 -1.05020982 -0.63611966
  0.24964977 -1.73006706 -0.91429442 -1.11612187 -0.04939993  2.40161037
  0.57231754 -1.43419167  1.12159147  0.17698616 -0.08796893  0.23263677
  1.33951777  1.26448018  0.32051116 -2.55912347 -0.60482991  0.57342463
  0.74802038 -2.34679599 -0.6060341  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.],
                 [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.]])])
```

```
In [40]: no.allclose(x.reshape(-1, 3, 2), y)
Out[40]: True

assert 통과무됨
```

```
In [41]: a=np.random.randint(0,10,(2,3))
b=np.random.random([2,3])
no.savez('test',a,b)
```

```
In [41]: a=np.random.randint(0,10,(2,3))
b=np.random.random([2,3])
no.savez('test',a,b)

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

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

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

In [45]: print(npz(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

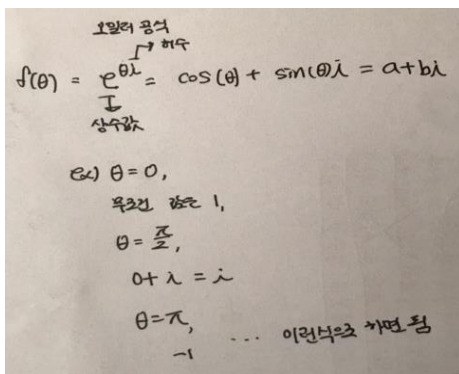
싸인하고 코사인에 들어가는 입력값은 **degree**가 아니라 **radian**이 들어가야한다

Degree 0 180 360.....720

Radian 0 π 2π 4π (2파이랑 4π 는 똑같은)

Ex) $\cos(1.5\pi) = 0$

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 $\sin(\theta)$ 값은 전부 실수였기에 표기에 문제가 없었지만 이런경우 허수가 있는 복소수는 어떻게 표시해줄까?

복소평면으로 표시해줄 수 있음. X축을 a , y축을 b 로 놓고 보자.

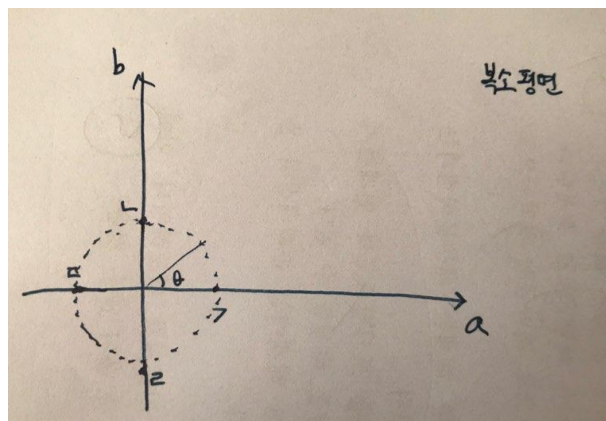
$f(\theta)$,

ㄱ. $\theta=0$ 일때 1, $a=1$, $b=0$

ㄴ. $\theta=0.5\pi$ 일 때 i , $a=0$, $b=1$

ㄷ. $\theta=\pi$ 일 때 -1 $a=-1$, $b=0$

ㄹ. $\theta=1.5\pi$ 일 때 $-i$ $a=0$, $b=-1$



벡터의 정의: 숫자열! $a+bi$ 도 결국 다 벡터임 위의 복소평면의 그래프. θ 값의 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

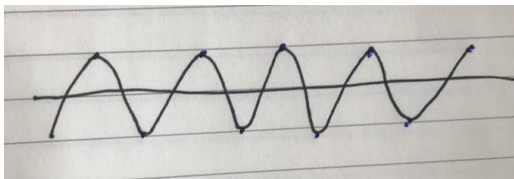
2hz도 가능할까?Yes

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



$Sr=10\text{hz}/2, Fr=5\text{hz}$, Nyquist hz, frequency는 무조건 sr 의 반절임

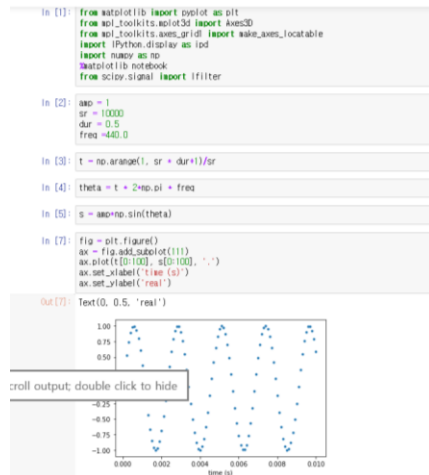
-사람이 들을 수 있는 가청주파수 20000hz임

- $s=s+\text{amp}+\text{np.sin}(\text{theta})$

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,

lpd. 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)
lpd.Audio(s, rate=sr)
```

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

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

11.18

-인공지능이라하는 것

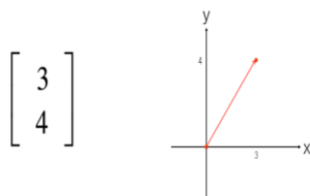
음성을 쳤는데 텍스트가 나오면 - 음성합성, 일본어 텍스트 넣었는데 한국어나오면 - 기계번역, 인공지능= 행렬의 곱, 어떤 입력벡터를 출력벡터로 만들어줌. 입력구, 출력구 둘다 음성, 텍스트, 사람이를 뭐든지 될 수 있다

-선형대수란 무엇인가 (linear algebra)

행렬의 차원 dimension,

M by n 행렬 여기서 by는 곱하기,데이터를 숫자로 (벡터로) 길게 표현가능 하다

앞으로 길게생긴 벡터를 칼럼벡터라고 할 것



왼쪽이 벡터, 오른쪽이 기하

왼쪽에 *2되면 6,8로 그래프의 점 원점에서 멀어질 것

-Linear combinations

$C*v + d*w$, V와 w는 벡터 ,c하고 d는 단순 숫자,위의 식을 실행한게 linear combination

단순히 곱하고 더하고만한게 linear combination

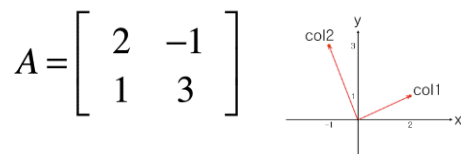
-벡터 스페이스

무수하게 많은 여러 벡터들이 만드는 것 벡터스페이스

-벡터 스페이스 내의 점 linear combination하고 나면 어디로 튈지 모른다는 것

Linear combination 하고 나서 어디로 튈지 모르는 것까지 포함한 것이 벡터 스페이스이다

-column space



2,1찍혀있고 -1,3이찍혀있음, 여기에 linear combination을 무한대로 하면 모든 공간이 채워짐

그것이 colum vector로 인한 colum space,col1 col2는 같은선상에 있지않기에 independent하다

(원점을 기준으로 뺀 줄을 기준으로 independent dependent 고려)

따라서 컬럼벡터가 차원을 넘나들 수는 없음.

$$A = \begin{bmatrix} 1 & 1 & 2 \\ 2 & 1 & 3 \\ 4 & 1 & 5 \end{bmatrix}$$

Whole space: \mathbb{R}^3
Column space: P

어떻게 서로서로 independent 하다고 할 수 있을까?

맨오른쪽은 앞의 두개의 합으로만 들어지잖아 그런식으로라고 맨오른쪽은 독립적이지 않은거임

-transpose

$$A = \begin{bmatrix} 1 & 3 \\ 2 & 3 \\ 4 & 1 \end{bmatrix} \quad A^T = \begin{bmatrix} 1 & 2 & 4 \\ 3 & 3 & 1 \end{bmatrix}$$

Whole space: \mathbb{R}^3
Column space: P

-row 벡터 Column space를 제외하고 null space라고 한다 (기하학적 해석)

$$Ax = \begin{bmatrix} 1 & 2 \\ 2 & 4 \\ 3 & 6 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

$$\begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 2 \\ -1 \end{bmatrix}, \begin{bmatrix} -2 \\ 1 \end{bmatrix}, \begin{bmatrix} 1 \\ -\frac{1}{2} \end{bmatrix}, \dots$$

null space

행렬에 뭘곱해도 0이 되게하는 모든 x값들을 의미함

$Ax=b$, X의 입력벡터의 차원과 b라는 출력벡터의 차원은 반드시 같지는 않아도된다

X를 b로 바꿔주는 transformation matrix는 a이다

11.21

(복습위주)

-한 벡터는 점이다. 숫자가 아무리 많아져도 하나의 점일뿐

-벡터 앞에 곱해지는 숫자 : 스케일러 라고 함

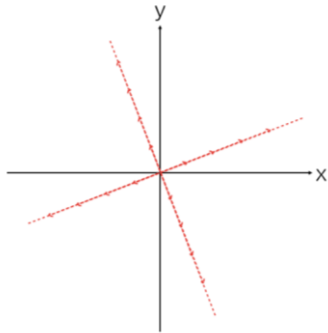
스케일링을 하면 기존의 벡터가 이동하는 것

-벡터 에디션은 두 벡터가 있을 때 기하적으로 이동하는 것으로 볼 수도 있음

$$A = \begin{bmatrix} 2 & -1 \\ 1 & 3 \end{bmatrix}$$

Col1(2,1), Col2(-1,3) 두개는 당연히 independent

스패닝된 column space = 2



-리니얼 콤비네이션, 칼럼
스패닝을 잘 보여주는 그
림

-wholespace 기반 matrix 기반 m by n 디멘션은 row가 몇 개이나

Column space의 디멘션은 그것보다 적을 것임

$$xA = \begin{bmatrix} x_1 & x_2 & x_3 \end{bmatrix} \begin{bmatrix} 1 & 2 \\ 2 & 4 \\ 3 & 6 \end{bmatrix} = \begin{bmatrix} 0 & 0 \end{bmatrix}$$

$$\begin{bmatrix} 0 & 0 & 0 \end{bmatrix},$$

$$\begin{bmatrix} 1 & 1 & -1 \end{bmatrix},$$

$$\begin{bmatrix} -1 & -1 & 1 \end{bmatrix}, \dots$$

left null space

1 by 3

3차원으로 표현되는 null space를

아래에 적은 것이 left null space라고 한다

(right null space도 이것과 똑 같은 원리)

A에 x가 왼쪽에 붙냐 오른쪽에 붙냐의

차이로 봐도 됨

Ax=b 대문자면 matrix 소문자면 matrix중에서도 벡터임. Y=ax+b와 같은 기본공식

행렬에서의 전형적인 표현임

(1,0) (0,1)의 벡터를 transformation하는 것도 연습해봄. 'L' 모양의 그래프가 'V'로 변함 (90도는

유지)

Transformation의 역 detransformtion, 완전히 변형시켜버리면 원상태를 되찾지 못함을 기억