T/T/b/h/d/0/w/tip/t/0/6/3개/6번/9/e/h/k/14/1/10

```
a= [[1],[2,3],[4,5,6]]
n=0
for b in a:
  for d in b:
    n+=1
    print(n)
```
->답 6

```
t=0
for p in range(3):
if p<=1:
  for q in range(4):
    t += q
  for r in range(2):
    t +=r
print(t*r)
```

답->14

이미지는 행렬이다. 그거를 길게 쭉 늘여놓는 것을 벡터하기. 라고 말한다.

array는 계산가능하게 바꿔줌.
그러기 위해 numpy를 먼저 import 해주어야함.

```
a = [1,3,5]
b = [2,4,6]
c = a+b
c = [1,3,5,2,4,6]
```

```
A = numpy.array(a)
B = numpy.array(b)
A + B
array([ 3,  7, 11])
```

import numpy as np #numpy 기니까 np로 축약해서

```
x = np.array([[1,2,3],[4,5,6]])
x
array([[1, 2, 3],
       [4, 5, 6]])
x.shape
(2,3)
```

2행 3열

만약 numpy 안에. 이 여러개 존재. A,B,C . A안에 d라는 패키지가 또 있다면,
numpy.A.D에 있는 어떤 함수라고 생각하면됨.
numpy가 젤 위에 있는 상위개념.

만약 우리가 import해서 numpy라고 불렀으면 전부 다 쓸수 있는데, a안에 있는 d안에 있는 function f가 있다고 치자. numpy.A.D.f
from이라는 걸 써서, from numpy. 그 안에 있는 import a를 불러오자.
A.D.f
numpy는 import를 A.d
from numpy import A.d

```
import numpy as np
import matplotlib.pyplot as plt
= from matlpotlib import pyplot as plt
```

numpy는 list 하고 비슷함. 수학적으로 계산도 할 수 있고 앞으로 쓰게 될 모든 데이터처리는 list로 아니라 numpy 처리를 해야 계산이 됨.

np.empty라 하는 순간, empty는 함수. 함수는 괄호로 그 뒤에 input을 받음.
numpy라 하는 젤 큰 라이브러리 속에 상단에 있는 함수가 empty.
입력이 list로 들어감. 2 by 3 로 행렬에 대해서 배웠음. 2줄 3열이 되는 직사각형. 뚱뚱.
내부적으로 들어간느것은 dtype. 그리고 int로 하면 아무 숫자나 막 나옴
np.empty([2,3], dtype='int')

np.zeros([2,3]) zeros라는 함수 사용. 2 by 3를 만드는데 다 0으로 채워짐.

np.array ([[0, 0, 0], [0, 0, 0]])

np.arange(5)
5라고 쓰는 순간 0부터 5개가 만들어짐.

array([0, 1, 2, 3, 4])

np.arange(0, 10, 2)
array([0, 2, 4, 6, 8])

np.linspace(0,10,6)
array([0., 2., 4., 6., 8., 10.])
linspace 하고 범위하고 만들 갯수. 총 6개로 나누어준다. 똑같이. 처음과 끝을 포함해서 6개로 나눠준다.

x = np.array([[1,2],[4,5],[8,9]])
x
array([[1, 2],
       [4, 5],
       [8, 9]])
이건 몇차원일까? 직육면체는 3차원. 3차원 표기가 당연히 가능하겠죠.
마지막에 대괄호가 두 개가 붙어있음. 대괄호가 두 개가 붙어잇는거 보고 2차원이라고 알면 됨. 대괄호가 세 개가 나오면 삼차원.
[[1,2],[4,5],[8,9]] -> 이게 2차원. 대괄호 하나 더 열고 또 하나 적고 대괄호 닫으면 한 차원 늘어나니까 3차원.
x = np.array([[[1,2],[4,5],[8,9]],[[1,3],[4,5],[8,9]]])
x
array([[[1, 2],
        [4, 5],
        [8, 9]],

       [[1, 3],
        [4, 5],
        [8, 9]]])
이게 삼차원.

x.ndim
3
->3차원이라는 것
x.shape
(2,3,2) 2차원짜리 직사각형이 두 개있다.

numpy라는 큰 라이브러리 패키지 속에 random이라는 서브패키지
그 안에 normal 이라는 함수를 쓸 거임.
from numpy import random.~

normal 이라는 함수. 100개의 데이터를 랜덤하게 만들어봐라.
실행안하고 싶을땐 % 앞에다 붙이기.

```
data = np.random.normal(0,1, 100)
print(data)
plt.hist(data, bins=10)
plt.show()
```

바구니 10개에 값 투척.
y축 값 다 합하면 100개.

```
X = np.ones([2, 3, 4])
X
```

2 by 3 by 4 는 곱하면 24개의 element가 있음.
reshape할때는 이 개수를 초과하면 안됨.

```
Y = X.reshape(-1, 3, 2)
Y
```

```
data = np.loadtxt("regression.csv", delimiter=",", skiprows=1, dtype={'names':("X", "Y"), 'formats':('f', 'f')})
data
```

delimeter 은 분리. skiprows=1 첫 번째 줄은 안쓸꺼다. 첫번째껀 x라하고 두번째껀 y라 하겠다. 포맷은 두 개다 float로 하겠다. int 면 I라 쓰면 되고.

```
np.savetxt("regression_saved.csv", data, delimiter=",")
!ls -al regression_saved.csv
```

누구한테 보내구 싶을 때 load가 아니라 save로.

```
arr = np.random.random([5,2,3])
```

```
a = np.arange(1, 5)
b = np.arange(9, 5, -1)
```

이렇게 적으면 1부터 5직전까지
1,2,3,4
9부터 5까지 -1이니까 9,8,7,6까지.

```
print(a - b)
```

print(a * b)
np.array가 된거니까 빼기 곱하기가 됨.

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]]

a는 3 * 3. 2차원임. 첫 번째 차원에 해당하는게
1,2,3/ 4,5,6/ 7,8,9
두 번째 차원은 1,4,7/ 2,5,8 / 3,6,9
a.sum(axis=0), np.sum(a, axis=0)

만약 sampling rate를 10000으로 하면
1초동안 총 10000개의 숫자를 우리가 담을거다.
1초당-hz.

다양한 pure tone들의 합이 복잡한 소리를 내는 걸 배웠음
sinusoidol function을 만들어 내는 것을 phasor라고 함.
sin function 도 phasor 가 되고 cos function도 phasor가 됨.
sin 하고 cos 에 들어가는 입력이 뭐지. sin의 입력값.
ㅠ는 3.141592....
0부터 숫자가 늘어난다고 생각해보자. 2*ㅠ면 6.xxx
degree : 0도 360도
radian : 0 2ㅠ
sin() 에 들어가는 값은 radian이어야 함. 180대신에 ㅠ가 들어가야하는 것.

0부터 100ㅠ까지 계속해서 그리면 50번의 반복이 있음. 2ㅠ씩 반복이 되니까
리드미컬하게~

오일러 공식 -> $e^{ix} = cosx + i\,sinx$
e 는 상수값. e는 2.71....
I는 허수. 루트 -1, 제곱하면 -1되는게 I.
x는 세타.
sin이라고 하면 여기 안에 있는 입력만 바꿔주면 출력으로 뭘 해주는 함수
오일러공식도 마찬가지. e와 I는 픽스되어있고 세타(radian)값만 바뀌는 것.
실수라 하면 허수 포함 못하고, 허수라 하면 실수 포함 못함.
복소수라 하면 a+bi 의 형식. 이걸로 모든 수를 표현 가능.

오일러 공식에 0, 1/2파이, 파이, 2/3 파이, 2파이를 각각 넣으며 1, I, -1, -i 가 반복됨.

sin wave에는 시간의 개념이 들어가 있어야 소리가 나옴.
sin 세타 라 하면 실체의 소리를 만들어 낼 수가 없음.

Phasor

```
# parameter setting
amp = 1            # range [0.0, 1.0]
sr = 10000         # sampling rate, Hz
dur = 0.5          # in seconds
freq = 100.0       # sine frequency, Hz
```

0.5초동안 삐. 1초동안 몇 번을 움직이는지.
sam rate은 frequency처럼 단위가 hz.
sam rate은 음질이 얼마나 고해상도인지. 음질의 해상도가 sampling rate.
거기서 10000이면, 1초에 10000개의 값으로 소리를 표현하겠다.
frequency는 1초에 태극문양이 몇 번 들어가느냐.

t 0.001 0.002 0.003 이렇게 하면 100년 걸리니까 쉽게 만들 수 있는 방법이 있음
1초동안 time이 이렇게하면 몇 개가 들어가지? 0.001, 0.002, 0.003, ...
0.0001, 0.0002, 0.0003, ... 0.5000
sampling rate 과 duration 만 있으면 time을 만들어 낸다.

t = np.arange(1, sr) 라고 하면 1에서부터 9999까지의 숫자 배열이 만들어짐
sr+1 이라고 하면 10000까지의 숫자배열.
t = np.arange(1, sr * dur+1)/sr
10000 곱하기 0.5 는 5000. 여기에 1더해줘야함 마지막수는 포함 안되니까.
이걸 시간으로 하려면 sr 로 나눠주면 1/10000초부터 0.5초까지 만들어짐.

e-04는 1/10000을 뜻함.

각도에 들어가는걸 phase 라고 함.

np.pi 그냥 pi가 아니라 numpy 속에 정의가 되어있어야함
time이 0에서 1초까지 만들어졌다 생각하면 1초에서의 time값은? 1
1초에다가 2pi를 곱하면 2pi지. 2pi 니까 1바퀴 도는거지.
몇바퀴 만드느냐가 frequency의 정의.
2pi가 1초 동안에 몇 개가 있어야 하는지를 frequency에 정의함.

time과 theta. time에 있는 벡터의 사이즈와 sin의 벡터의 사이즈는 같다?
당연히 같음. 1/10000초부터 5000/10000초. 총 5000개 만들었음.
똑같은 개수의 벡터가

11/14

formant-산맥이 다양하게 나올 수 있음. 첫 번째가 어디냐에 따라서 ㅏ가 될수도, l가 될 수도 있음.
1. gradually decreasing하게 그래프 만들어야함.
2. 산맥을 하나하나씩 만들것임. 몇에다가 300하나 만들어줘.

11/7

matplotlib notebook
제일 큰 라이브러리, 그 밑에 서브라이브러리. 제일 크게 matplotlib이 있고
우리가 import 하고 matplotlib 하고. 그속에 있는 from을 쓰고
오일러 function을 쓰면 회전하면서,,

from matplotlib import pyplot as plt

phasor에서 여러 가지 변수들을 미리 정리.
parameter setting이 주는 장점은 나중에 amplitude를 바꾼다, duration을 바꿀 때 이 변수만 바꿔주면 쉽게 나옴. 목적상으로는 parameter 고 파이썬 코딩할땐 variable.
맨처음에 time을 왜 만들까. function들이 받아들이는 값이 정해져있음. 오일러에는 반드시 각도값. 하지만 각도값만 넣어서는 실체의 소리를 만들 수 없기 때문에 time 만듦

각도값만 만들어서 플러팅 했을 때-
theta = np.arange()
arange- 숫자값 만들 때. 각도를 만들어낼때 우리가 0~2π까지.
 theta = np.arange(0, 2*np.pi,)
이렇게 하면 0~2π까지 만들어질텐데.

figure 는 화면전체. 젤 첨에 figure 라는 function 만듦. subplot이라 해서 중간에 여러 개 화면 분리를 해서 그 속에 subplot을 만들 수 있는 것. 화면분리부터 먼저. 그 속에 221. 2 by 2로 나누는데. 이 줄이 두 줄이 열도 두 열로 화면 분리를 한다. 그중에서 첫 번째 껄로 선택한다. 22 하면 바둑판 두 줄 두열로 만들고 첫 번째 조그마한 네모는 1, 두 번째는 2, 세 번째는 3....
subplot 바로 밑에 실제 ploting을 하려고 하는 x와 y 값을 적어주면 됨.
theta, 즉 각도 값을 x값으로. s값은 sin 통과한 거. 총 7개의 세타 값이 s 로 갔으니까 7개의 s값. y축은 sin 함수의 결과.
더 빽빽하게 sin 함수 만들고 싶으면 np.arange이 부분을 (0, 2*np.pi, 0.1) 로 해주면 100배가 됨.

```
from matplotlib import pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
from mpl_toolkits.axes_grid1 import make_axes_locatable
import IPython.display as ipd
import numpy as np
%matplotlib notebook
from scipy.signal import lfilter



theta = np.arange(0, 2*np.pi, 0.1)
theta
```

```
array([0. , 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1. , 1.1, 1.2,
       1.3, 1.4, 1.5, 1.6, 1.7, 1.8, 1.9, 2. , 2.1, 2.2, 2.3, 2.4, 2.5,
       2.6, 2.7, 2.8, 2.9, 3. , 3.1, 3.2, 3.3, 3.4, 3.5, 3.6, 3.7, 3.8,
       3.9, 4. , 4.1, 4.2, 4.3, 4.4, 4.5, 4.6, 4.7, 4.8, 4.9, 5. , 5.1,
       5.2, 5.3, 5.4, 5.5, 5.6, 5.7, 5.8, 5.9, 6. , 6.1, 6.2])
```

```
s = np.sin(theta)
s

fig = plt.figure()
ax = fig.add_subplot(111)
ax.plot(theta, s, '.')
```

-> 빽빽한 싸인그래프 나옴

```
ax.set_xlabel('theta in radians')
ax.set_ylabel('value')
```

추가하면 x축 y축에 이름 붙여짐.

linear한건 y= 2x 같은 함수. x, y의 관계가 1차함수같은거.
non-linear 한 사인 그래프. x와 y의 관계가 line 이 아니다. y=ax^2+bx 같은 함수들.
x들간의 거리는 equi..? 하게 만들었는데, 거리가 좁고 좁지않은 부분이 공존하는 non-linear 함수임.

ax.plot(theta, s, '.')에서'.'을 '-'로 하면 선으로 됨.
10파이까지 나오게 하려면 theta = np.arange(0, 2*np.pi* 5 , 0.1) 로 바꿔주면됨.

시간과 소리는 서로가 있어야함.
시간을 만들 때 t = np.arange(1, sr)
우리가 만약 1초라면 time 틱의 개수는 sampling rate 과 일치함.
dur의 비율이 반영이 되게 t = np.arange(1, sr*dur+1) /sr

**시험문제 나온다면~

```
fig = plt.figure()
ax = fig.add_subplot(111)
ax.plot(t[0:1000], s[0:1000], '.')
ax.set_xlabel('time (s)')
ax.set_ylabel('real')
```

자자 이거 보자~ 이거 보면 t랑 s랑 값이 1000개까지.
각각의 1000개의 값이 짝을 이뤄 상응하면서 그래프에는 총 '1000'개의 점이 찍히게 되는거지. 상응! 해야하니까 만약 t[0:1000]이고 s[1:1000] 이면, 값이 안나오겠지>
같이 0부터 시작해야지 상응하니까!

1j= ij
real만 보면 cos, imag만 보면 sin. 두 얼굴이라 complex phasor.

sin, cos 에 time을 연동시켜야 재생가능.

```
# !pip install sounddevice
import sounddevice as sd
sd.play(c.real, sr)
```

소리 안날 때 sounddevice install.

```
# parameter setting
```

```
amp = 1            # range [0.0, 1.0]
sr = 10000         # sampling rate, Hz
dur = 0.5          # in seconds
freq = 100.0       # sine frequency, Hz
```

dur 얼마나 길게할건지. freq는 sin wave가 몇 번 움직이는지.

complex는 돼지꼬리 소용돌이 치듯이. 그게 얼마나 커지는가. 스프링 내부의 직경이 더 커질 수 있게 하는걸 s = np.sin(theta)
theta = t * 2*np.pi * freq
s = np.sin(theta) 이렇게 정의하면 -1부터 1까지 가는 물결이 실현됨.
-2부터 2까지 만들고 싶으면 전체에다가 곱하기 2 하면되고 이게 amplitude.
우리의 default amp 는 1이었으니까.
s = amp*np.sin(theta)
위에서 parameter에서 amp를 2로 바꾸면 -2부터 2까지의 sin 함수가 만들어지는 것

complex phasor.
c= np.exp(theta*1j)
여기도 똑같이 parameter- amp를 2로 설정해주고 c= amp*np.exp(theta*1j) 실행하면 -2부터 2로 complex 구현됨.
amp 와 반지름의 길이는 2로 같다.
amp 커지니까 소리도 더 커짐.

```
# generate samples, note conversion to float32 array
F0 = 100; Fend = int(sr/2); s = np.zeros(len(t));
for freq in range(F0, Fend+1, F0):
    theta = t * 2*np.pi * freq
    tmp = amp * np.sin(theta)
    s = s + tmp
fig = plt.figure()
```

```
ax = fig.add_subplot(111)
ax.plot(t[0:1000], s[0:1000]);
ax.set_xlabel('time (msec)')
ipd.Audio(s, rate=sr)
```

sampling rate 하고 frequency가 연결되는 부분이 있음.
s.r 이 100hz라고 생각해보자. 그 말은 우리가 표현할 수 있는 숫자의 개수가 1초에 100라고 할 수 있음. 이 100개의 숫자를 가지고 frequency를 1hz의 frequency를 표현할 수 있을까 없을까? 답은 있다! 어떻게 표현하지? 한번의 sin wave 주기가 있으면 되자나. 2hz의 freqeuncy도 가능.
근데, 10000hz가 가능할까? 100개의 숫자가 주어져있는데 1초에 10000번 왔다갔다 움직이게 할 수 있나? 우리가 가진 숫자가 100개 밖에 없기 때문에 안됨. sampling rate 이 넉넉하게 있어야지 그만큼의 주파수를 표현할 수 있음.

주어진 숫자가 10개라고 생각해보자.

0 0 0 0
0 0 0 0 0 0

sr= 10hz
fr= 100hz.
10개의 숫자로 100번 왔다갔다함을 표현할 수 있을까? 없음.
몇 번 왔다갔다 하는 게 frequency.
sr의 반절에 해당하는것만 표현할 수 있음.
sr이 10hz면 fr은 5hz
이걸 nyquist 나이퀴스트 frequency.

sr = 44100hz. 이것의 nyquist freq 는 22050hz.
왜 cd음질을 44100 으로 잡았을까. 사람이 들을 수 있는 가청 주파수가 20000hz.
22050hz면 20000이상으로는 어차피 못들음. 44100이면 충분한 것

유선전화에도 소리가 디지털로 감. 얼마나 빽빽하게 전달하는지가 중요,
어떻게든 아껴서 보내는게 통신사에겐 이득이니까. 예전엔 8000hz였음. 4000이 naquist.

** sr의 half 에 해당되는 naq가 표현할 수 있는 frequency의 maximum
따라서 freq가 22000 hz 소리를 잡고 싶다면 2배는 넉넉하게 sr를 설정해야함.

```
# generate samples, note conversion to float32 array
F0 = 100; Fend = int(sr/2); s = np.zeros(len(t));
for freq in range(F0, Fend+1, F0):
    theta = t * 2*np.pi * freq
```

```
    tmp = amp * np.sin(theta)
    s = s + tmp
fig = plt.figure()
ax = fig.add_subplot(111)
ax.plot(t[0:1000], s[0:1000]);
ax.set_xlabel('time (msec)')
ipd.Audio(s, rate=sr)
```

f0은 100으로 설정. 배수로 올릴 수 있는데, 어디까지 올릴 수 있을까. 무한대까지는 못올림. 그 끝은 어디일까. sr의 반까지만 올라갈 수 있잖아.
fend = int(sr/2) f의 마지막은 sr의 1/2인 naquist frequency. int로 숫자 지저분하지 않게 반올림처리.
time은 위에서 assume 됐다고 치고, 위에서 생각을 해보면 phasor 만들 때 time 만들고, theta값 만들고 그걸 sin 에다가 입력했음. 그 과정을 여러번 반복하는데, freq 만 바꿔준느게 이 코딩의 핵심.

```
for freq in range(F0, Fend+1, F0):
    theta = t * 2*np.pi * freq
    s= s + amp * np.sin(theta)
```

s는 signal이라고 변수 하나 주는데 s= s+~ 여기다가 처음꺼 그다음꺼 그다음다음꺼 더하는 거 뜻함.

for freq in range(F0, Fend+1, F0):
100부터 fend까지 쭉쭉쭉 늘려가는데 increment는 f0만큼 올라가니까 마지막에 f0써줌.
100씩 커지는 것.
+1을 한 이유는 젤 마지막껏도 포함시키기 위해서.
100부터 5000(sr=10000이니까 1/2인 5000까지) 까지 돎.
100, 200, 300 .... 100씩 는 숫자들이 freq에 들어가고 이 루프가 50번정도 돎. (100~5000)

젤 첨의 s값이 뭔지 정의를 해줘야함.
F0 = 100; Fend = int(sr/2); s = np.zeros(len(t));에서 s = np.zeros(len(t));
젤 처음은 0으로 정해져있으면 되잖아. 그냥 0으로 얼만큼 만드냐면 time 벡터의 개수만큼 0을 만들어주면 되는 것. 0000000.,,,, 만들어져있고 sinwave 하나 만들어서 계속 더해지는과정.

11.14
cos 과 sin 은 2/ㅠ 정도의 차이가 있다.
우리 귀가 phase는 인식을 못한다. 전혀 sensitive 하지 않음
phase이동 100번해봤자 못느낀다.

```
ipd.Audio(s, rate=sr)
```
s를 쓰나 c.real 쓰는거나 소리 똑같음

```
def hz2w(F, sr):
    NyFreq = sr/2;
    w = F/NyFreq *np.pi;
    return w

def resonance (srate, F, BW):
    a2 = np.exp(-hz2w(BW,srate))
    omega = F*2*np.pi/srate
    a1 = -2*np.sqrt(a2)*np.cos(omega)
    a = np.array([1, a1, a2])
    b = np.array([sum(a)])
    return a, b
```

def hz2w라는 함수에 F와 sr 이라는 입력 두 개가 들어감.
return 부분이 출력. return없으면 출력하는거 아님.

```
RG = 0  # RG is the frequency of the Glottal Resonator
BWG = 100  # 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)
```

RG에는 frequency 값 줘서 산맥만들기. BWG에는 bandwidth니까 어떤 산을 만들 때 얼마나 이게 뚱뚱한지 뾰족한지 판단.
spectrum에서 크기가 똑같은 100, 200, 300, 400...
gradually decreasing 하는 그래프 만드는 방법은, frequency 0을 중심으로해서 큰 산을 만듦, 가운데가 0임.

rg값 500, 1500, 2500, 3500 으로 주면서 산맥만들기

```
s = lfilter(np.array([1, -1]), np.array([1]), s)
ipd.Audio(s, rate=sr)
```
이거는 위에서 rg값으로 산맥만들더라도 입이 없음.
이거는 입술이 있는 상태

11/19

데이터 기계 데이터
vector 함수 vector
음성이 vector의 형태로 데이터에 들어감. text
text를 쳤는데 음성이 나오면 음성합성
일본어 text가 들어갔는데 한국어 text 가 나오면 기계번역

1*4 행렬이 4*3 행렬을 만나면

5 3 0 1 -1 0 2
          0 1 3   -3 6 23
          3 -5 7
          2 3 4

인공지능이 이런 원리로 돌아감.
선형대수- linear algebra

matrix- mxn행렬
vector- mx1, 1xn 행렬

3
4 칼럼벡터. 열벡터. (3,4)
차원늘리고 점만 찍으면 됨

scalars는 단순한 숫자, vector의 c와 w f

일부분들은 vector spactr 가 될 수 없음.

칼럼벡터의 차원은 2차원. 2차원상에서 어떤 두 점을 찍고 여기에 linear combination 해서 확장시킴. 원래있던 2차원만을 커버함. 그 이상으로 넘어갈 수 없음.
원점과 두 컬럼벡터를 다 연결해서 상상하면 삼각형이 생김

원점과 두 컬럼벡터 연결했을 때 삼각형 나와야하는데, 이거는 원점과 두 컬럼백터. 삼각형이 안돼

dim (whole space)
n rows
dim(column sp

p= plain, 2차원
42번 피핕. 왜 column space 가 2차원이지? 앞에 두 개가 마지막을 만드는 요인이기 때문에 실제로 존재하는건 앞에 두 열, independent 한건 앞에 두 개.

43번 ppt . L은 1차원을 의미하나?

A^T - transpose? = 이런걸 ll 이렇게 변형

컬럼백터를 기준으로 whole space 계산.
1 2 4
3 3 1 이 만들어내는건 무조건 2차원
이게 transpose 했지만 column space는 여전히 2차원

49번 피피티. 두 컬럼이 depent 함. 1차원만 쓰고 나머지 두 차원은 null space
실질적 수학적 정의는 50번. 어떤 행렬이 있을 때 무엇을 곱하면 무조건 0이 되는 값들을 null space라고 함.

대문자- 행렬, 소문자- 벡터
Ax=b
1x4 4x3
linear transfromation 곱하고 더하기햇으니까 linear. 4개짜리를 3개짜리로 바꿧지. 차원과 숫자를 바꿧으니 transformation.
A가 transformation matrix
기계(t.m) 입력 출력

3X2 면 2Xn 으로 와야됨. 두 숫자가 같아야 하는 것.
3Xn으로 새로운 행렬 값이 나옴.

기계함수 - 입력 - 출력
A * x = b
x^T * A^T = b^T

null space = whole- col space

col 이든 row이든 차원은 independent 한게 몇 개 있느냐가 중요함.
col은 세로, row는 가로.

row그래프 a, b, a

벡터는 점이다.
R^3 라 하면 3차원으로 표현ㅇ할 수 있는 벡터들의 합.

column은 어떤 매트릭스가 주어져야됨. 안주어지면 안됨.
null space 가 오른쪽, 왼쪽에 있냐에 따라 right, left로 나뉨.
[x1, x2, x3] 이러면 3차원에 null space 존재.

* null space의 필요성

```
  A          x        b
[ 1 5 3    [ 0       [0
  2 6 1]     0        0]
             0]
```

2 * 3 3 * 1 2*1

whole vector space는 2차원밖에 없다.
row vector space 가 2차원일 때, 차지하지 않는 공간은 null space.
직각을 이루는 선이 null space? Q. 직각을 이뤄야만 하나? O

```
Ax = [ 0
       0]
```

null space 인 선에 있는 값들이 x의 조건에 만족되는 값들.
반드시 선위에 있을필요는 없고 방향쪽으로 따라가서 어떤 점을 선택한다 치더라도 이 결과값에 영향을 미치지 않는게 null space.

원점과 입력값과 출력값이 평행하는 선 有 - 보라색 라인.

벡터라는 것은 매트릭스의 한 형태다Ⅱ. 숫자열이다. 물리적으로 벡터는 방향이다.
보라색 선을 아이겐 벡터들 이라고 부른다. 벡터는 값이 아니라 방향이 중요하다.

아이탄의 전생-

국어와 영어 성적이 비례하는게 상관관계. correlation, r이라는 값으로 표현.
-1 =< r <= 1

inner product (dot product)

a (1,2,3) b(4,5,6)

a의 길이(| al) x cos 세타 x b의 길이

[1 2 3]
 * * *
[4 5 6] = 32
두 벡터를 각각 곱하고 inner로 더하면 32가 나오는 것임.

소리 벡터에다가 사인 웨이브를 여러개 만들어서 하나는 100hz, 하나는 300hz, 1700hz를 만들어서 inner product를 해버림. 그럼 값이 나올텐데 유사한 성분, correlation이 많으면 inner product 값이 높게 나오고, 많이 없으면 낮게 나옴.

11/28

Av= 상수v
를 만족하는 vector가 아이겐벡터.

given A에 대해 어떤 vector를 곱했을 때, transform된 결과가
만약 이 라인이 아이겐 벡터라고 합시다. 원래벡터가 있었다면 어떤 식으로 transform 돼서 Av 가 여기에 와있다고 생각해봅시다. 원점, v, Av가 한 선상에 있음.
어떤 입력에 상수배를 한게, 한 라인에 있다. 이 식이 아이겐 analasys를 하는 식인데.
0 과 v, 0 과 Av, 비율을 재기도 함. 아이겐 value.

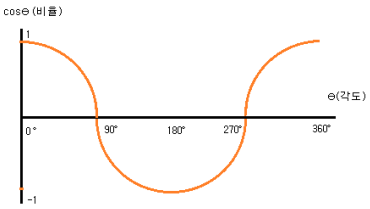
inner product에 대해서 얘기하심. 몇차원이 되든 상관없는데 어떤 두 vector가 있다고 합시다. 아무리 차원이 높다할지라도 두 벡터가 있으면 원점을 중심으로 a, b라는 두 벡터가 있음. 차원이 높다할지라도 원점, a, b 가 2차원의 평면을 이룰 수 밖에 없음.
a- 123 (1x3) 과 b- 247 (1x3) 을 inner product를 한다는 말은, a x b.
둘이 곱하기 위해선 b를 transpose를 해서 3x1 로 만듦.
1 2 2 하면 31나옴. 31이 inner product. a transpose 곱하기 b 하면
 4
 7
a= [1,2,3] b= [2,4,7]
 원점(0,0)
요거 삼각형 만들깅~ OB 에 A에서 90도를 이루는 직선을 내림.
inner product 구하려면 lal (a의 길이) 에다가 cos세타 곱하기 lbl
cos세타 어떻게 measure?
axb = cos세타.

-
lallbl
a의 길이는 루트 1^2+ 2^2 + 3^2

cos90은 0이기 때문에 값이 안나옴. projection 시켜도 직각이라서 0. 0에다가 뭘 곱하든 0.

cos similarity. 한마디로 a와 b가 얼마나 비슷한지를 수치적으로 얘기할 수 있는 방법.
두 개의 벡터 주고 cos similarity 구하라는 문제 나옴.
cos 세타가 cos similarity 임.

wave. 딱 그냥 눈으로 볼 때, 여기에 어떤 성분 두 개가 보여요?
하나의 웨이브는 100hz, 다른 하나는 1000hz.
R^100 (100차원) inner product 값 구하기 위해선 곱하기.



12/3 듣기

12/5

cos(세타) = r
-1부터 1까지 변함.

a= 영어, b= 국어.
학생 1,2,3,4.... 열명. 10차원 벡터 두 개가 생기겠지.

영어	국어
10	0
20	10
30	...
...	
100	90

10차원의 축이 있다고 생각해보면 sub1, sub2, sub3, ... sub 10 10개의 축.
10차원의 한 점은 10개의 숫자를 갖고 있음.
하나는 영어 벡터, 하나는 국어 벡터.
영어와 국어 값의 세타값이 나오고. 그 각도값을 코사인에 취했더니 r=1 과 같은 값이 나오더

라.
데이터를 찍었더니 완전 라인위에 있더라. 그말은 영어벡터와 국어벡터가 완전히 똑같지는 않지만 그 두 개의 각도가 0이 될 것.
r값이 1이면 각도 0, r 값이 -1이면 각도 180도. 데이터 값을 보지 않아도 알 수 있는 정보.

sampling rate 가 0에서 half 까지. 5000까지만 우리가 표현할 수 있는 주파수. 1/2까지만.
7000에 해당되는건 3000과 똑같. half에 해당되는 부분까지만 주파수를 표현하는거다.

```
powspec = 1/nfft * (magspec**2)
plot_spectrogram(powspec);
```

**2는 제곱. 이 작업을 하면 1보다 큰 양의 값은 엄청나게 커지고, 1보다 작은건 엄청나게 0에 가까워질텐데.. 0.0001 에 log10을 치면 4가 나옴. 아주 작았던 숫자가 4, 5로 나옴.
log 처리 하면 너무 먼 숫자를 우리가 다룰 수 있는 숫자로 바꿔줌.
그 값이 나올 때 더하기 10정도 해주면 골고루 나올 수 있다.