

## 영어음성학 기말 summary

<NUMPY>

data - 숫자화 될 수 있는 정보

데이터의 종류

소리 데이터

이미지 영상 데이터

텍스트 데이터

숫자 데이터

직사각형의 형태로 숫자를 행과 열로 배열 - 이것이 그림이 될 수 있음 - 행열

숫자들을 왜 이렇게 일자로 펼쳐 놓을까 - 숫자의 나열은 벡터 모든 데이터는 벡터의 형태로 되어야 함 행열은 그래서 벡터가 아님 모든 데이터는 벡터의 상태여야 다루기가 쉬움

그 행열이 3층으로 되어 있으면 칼라 1층이면 흑백

영상은 2차원 칼라 이미지는 3차원 시간까지 있으면 4차원

어떻게 소리가 벡터화가 될까

우리의 소리는 wave form으로 이루어져있고 그 웨이브 폼 확대해보면 하나하나의 값들 그 값들을 벡터화 시킬 수 있음

텍스트는 어떻게 벡터화가 될까

0과 1로 벡터화 시킬 수 있음

모든 데이터는 벡터화 시킬 수 있음

제일 중요한 library중에 하나가 numpy: 벡터라이즈된 데이터들은 더하기 곱하기 같은 수학적 계산을 해야함 하지만 그냥 처리해버리면 list로만 처리되고 더하기 곱하기 안됨 그것을 도와주는게 numpy

array 리스트를 계산 가능한 행열 혹은 벡터로 바꿔주는 것 이 전에 import numpy 무조건 먼저 해줘야 함 import numpy as np 라고 해주면 numpy를 그 이후로 np로 바꿀 수 있음 뭐든 가능 n으로 바꿀수도 있음

X.shape 은 차원을 말해주는 것임

-----

<NUMPY2>

import 이제부터 해야함

기본적인 함수들은 그냥 사용했지만 중요한 함수들은 그 함수들이 포함되어 있는 library를 불러와야 함

numpy 가 제일 상위 개념

함수를 불러오는 법 numpy.A.D.f(함수) 이런식임 A안에 D안에 f라는 함수를 불러온다

.(점)은 함수간의 포함 관계를 말해주는 것

import matplotlib.pyplot as plt

똑같이 쓸 수 있는 것이 아닌 것은? 이런식 시험

=FROM matplotlib import pyplot as plt

numpy를 왜 필요한가

list와 아주 비슷하지만 수학적으로 계산을 할 수 있기 때문

앞으로는 모두 numpy처리 해야함

matplotlib.pyplot 는 plot 하는 것과 관련된 것

empty도 numpy 제일 큰 library안에 있는 함수

np.empty([2,3], dtype='int') #이러한 갯수와 종류로 행열을 만들어주어라(랜덤)

np.zeros([2,3]) #np library에 있는 zeros함수를 이용하는데 2 by 3 매트릭스를 만들어내라(0으로 채워진)

np.array([[0, 0, 0],[0, 0, 0]]) # [0, 0, 0]는 그냥 벡터 우리가 이것을 2 by 3로 만들고 싶을 때는 [0, 0, 0] 추가 [[0, 0, 0],[0, 0, 0]], 이것을 계산이 가능한 array로 만드는 방법이 np.array()

np.ones([2,3]) 이렇게 하면 1., 찍힘 아마도 이러한 함수의 디폴트값이 float일 것임 그렇기 때문에 dtype = int로 해주면 점들이 없어짐

float도 종류가 여러가지가 있음 소수점 몇자리까지 해줄까 에 대한 정의가 float64 이런 것임 아주 정교한 일을 하고 싶을 때(아주 정교하지만 단점은 메모리를 많이 차지함)

아주 중요한 두 함수(arange, linspace)

np.arange(5) #5개의 index생성 for loop다음에 range썼던 것과 비슷 거기에 있었던 것들은 계산이 안되는 것 but 이 것들은 계산이 되는 것들

np.arange(0,10) #이런 식으로 range를 정해줄 수도 있음

np.arange(0,10,2, dtype='float64') #중요함 0부터 10까지 2의 간격으로 float~ :뒤에 것은 얼마나 정확도를

높이느냐

`np.linspace(0,10,6)` #0부터 10 포함해서 '6개' 로 나누어 준다

`X = np.array([[1, 2], [4, 5], [8, 9]])` #여기서 대괄호가 두개가 나오기때문에 2차원이다.. 3개면 3차원

`Y.ndim` #차원에 대해서

`Y.shape`

`(2, 3, 2)`

# 3x2(직사각형)이 2개가 있더라 제일 큰 것 부터 생각하면 됨 2개가 있다가 먼저

`Y.dtype` #타입에 대해서 알려줌

`Y.astype(np.float64)` 이렇게 바꿀 수 있음

`np.zeros_like(Y)` #형태를 유지한 채 모든 숫자를 0으로 바꿔주어라 `Y*0` 과 같음

유용하고도 중요한 것 plotting?

numpy 안 subpackage에 random 함수가 있음

`data = np.random.normal(0, 1, 100)`

만약 from numpy 를 썼다면 np쓸 필요 없음

from numpy import random.normal 이런 식

normal는 normal distribution의 약자(정규 분포 데이터를 만들어 주는 것 중 모양이 아니라 데이터!를 랜덤으로 만들어 주는 것 평균값과 표준편차 필요함) (0 - 평균, 1 - 표준편차, 100 - 데이터의 갯수)

`plt.hist(data, bins=10)`

`plt.show()`

#histogram bin이라는 옵션을 항상 달고 다니는데 바구니를 몇 개를 달고 다닐 것인가 x축위에 쌓여서 그래프를 그리는 것 그러한 그래프 y값에 해당하는 값은 항상 0을 포함한 자연수값이 나옴

저 값들을 다 합하면 100개 (시험!)

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

`X`

이렇게 행열을 만들어 낼 수 있음

이러한 행열을 reshape할 수 있음 몇 by 몇 by 몇이라는 shape을 바꿀 수 있음 but 그 안에 있는 element의 갯수의 총합은 같아야 바꿀 수 있음 -1은 알아서 숫자를 채워넣어라

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

`Y`

`np.allclose(X.reshape(-1, 3, 2), Y)` #완전히 일치하느냐 하는 함수 `allclose`

`random`안에 있는 유용한 함수들 중에 방금 배웠던 `normal`함수 말고도 다른 것 많음 그 중 하나 `randint`

`a = np.random.randint(0, 10, [2, 3])` #0부터 10까지 숫자 중에 뽑아서 2 by 3 행열을 만들어라

`b = np.random.random([2,3])` #말그대로 랜덤하게 만들어내라

`np.savez("test", a, b)` #파일을 저장해주는 함수 `a`와`b`라는 variable을 파일로 저장해주는 함수

`!ls -al test*` #저장이 잘 되었는지 확인해주는 함수

`del a, b` #메모리에서 variables 지우고 싶을 때

`who`는 어떤 variables들이 지금 available한지

`npzfiles = np.load("test.npz")` #저장한 값들을 불러오고

`npzfiles.files`

`npzfiles['arr_0']` #그 파일 이름에서 불러올 것 적으면 값이 불러져 나옴

`data = np.loadtxt("regression.csv", delimiter=";", skiprows=1, dtype=('names':("X", "Y"), 'formats':('f','f')))` #csv는  
콤마로 나뉘어진 values들 이라는 파일 타입 엑셀로도 불러오기 할 수 있음 데이터를 파일로 주는 방법 중 하나

`np.savetxt("regression_saved.csv", data, delimiter=";")` #그 데이터를 주고싶거나 저장하고 싶을 때

`!ls -al regression_saved.csv`

`print(arr.size)` #element의 총 갯수

`a = np.arange(1, 10).reshape(3, 3)`

`b = np.arange(9, 0, -1).reshape(3, 3)`

`print(a)`

`print(b)`

`a == b` #이러한 비교를 하려면 dimension이 완전히 똑같아야 함

`a.sum()`, `np.sum(a)` #`a`는 이미 numpy로 만들어진 산물이기 때문에 이대로 쓸 수도 있고 / numpy안에 있는  
`sum`이라는 함수를 사용하겠다 이렇게 쓸 수도 있고

`a.sum(axis=0)`, `np.sum(a, axis=0)` #axis, 몇 번째 차원에서 sum을 할 것인가 0이면 행들을 세로로 합친다고

생각하면 됨

`a = np.arange(1, 25).reshape(4, 6)` #이렇게 `arange`다음에 바로 `reshape` 붙일 수 있다  
`a`

`b = np.arange(6)`  
`b`

`a+b` #차원이 달라도 더할 수 있다

-----

<SOUND>

praat할 때 sound가 pure tone들의 합으로 만들어 진다고 배웠음  
사인의 그래프 형식을 띄는 곡선을 sinusoidal이라고 하고  
그 그래프를 만들어 내는 것을 phasor라고 함  
사인과 코사인 function에 들어가는 입력값?  $\sin(?)$  - 세타(radian값)  
각도 값을  $0^\circ \sim 360^\circ$  -  $0 \sim 2\pi$  radian값으로 바꿔줘야 함

0부터 100파이 까지 사인 코사인 그래프를 그리면 몇번의 반복이 있을까? 50번

사인 코사인 그래프 기본적인 것 알아야 함  $0$   $1/2\pi$   $1\pi$   $2/3\pi$   $2\pi$  2파이 주기임

오일러 공식(euler)

우리가 생각하는 모든 수를 포함하는 것  $a+bi$ (복소수)

$f(\text{세타}) = e^{\text{세타} \times i}$  =  $\cos(\text{세타}) + \sin(\text{세타})i = a+bi$

세타가 0 일 때는  $f(0) = 1$

세타가 2분의파이  $f(2\text{분의}\pi) = i$

세타가 파이  $f(\pi) = -1$

세타가 2분의3파이  $f(2\text{분의}3\pi) = -i$

이 것 4개 반복

이러한 복소수를 어떻게 plot할 것인가 complex plot x축에는 a y축에는 b

projection

x축 혹은 y축만으로 보는 것 실수만 보고싶다면 위에서(x축만) 보면 되고 허수만 보고 싶다면 옆에서(y축만) 보면 됨

위에서 보면 1에서 출발해서 -1까지 왔다갔다 함 = cos 그래프와 같음

옆에서 보면 0에서 출발해서 1과 -1왔다 갔다 = sin 그래프와 같음

input이 다 공통적으로 radian 각도값 세타값.. praat을 이용해서 소리를 만들 때 phasor을 define할 때 넣는 입력값 중 하나가 frequency 1초에 몇번 왔다갔다 하는가

우리가 세타값만 넣었을 때는 시간의 개념이 들어가 있지 않음 시간의 개념이 들어가야 우리는 1초에 몇번 왔다갔다 해서 소리의 높이를 알 수 있는 것임

진폭을 1로 정의하면 그냥 원래대로 x1해서 -1부터 1까지인 사인 코사인 곡선

sampling rate 1초에 몇개의 숫자로 해서 음질상 얼마나 고해상인 것이냐

duration 몇초동안 소리가 나느냐 정의하고

freq는 1초에 몇번 왔다갔다 하느냐

우리가 가장 먼저 해야하는 일 - time을 만드는 것

t 0.0001 0.0002 0.0003...0.5000 이렇게 일일이 만들기 힘들

dur이 0.5니까 0.5까지 만들면 되니까 t의 끝이 0.5로 끝나면 되는 것이고

sr이 10000개니까 1초일 때 만개가 들어가야 한다는 것이고

amp와 freq는 아직 여기에 들어가있지 않음

t = np.arange(1, sr+1) 이렇게 만들면 무엇이 만들어질까

(numpy는 계산을 위해서 쓰는 것이고) 1에서 10000까지 10000개가 만들어짐 dur이 만약 1초였으면 딱 맞는 것인데 지금 여기서 dur이 0.5니까 sr에다가 dur를 곱해줌 +1은 우리가 원래 뭐부터 뭐까지 할 때 하나 빼니까 더해주는 것 (1, sr \* dur + 1) 여기까지 했음

여기까지 하면 5000까지 만든 것 타임의 틱은 만들어낸 것이지만 실제 타임은 아님 여기서 sr로 나누어주면 시간의 개념이 되는 것

1.000e-04, 2.000e-04,, 이런 식으로 나오는데 e-04 = 10000분의 1 따라서 1 \* 만분의 1 = 1/10000 이런 식

# generate time

t = np.arange(1, sr\*dur+1)/sr

그래서 이 식이 되는 것

time이 없으면 실체의 소리를 만들어낼 수 없음 time을 먼저 만들고 그 다음으로 세타값이 필요함 그 각도 값을 phase라고 하는데 time과 연동시켜서 만들어내야 함 그 방법이 아래 것

```
# generate phase  
theta = t * 2*np.pi * freq
```

np.pi는 numpy속에 pi값이 정의되어있기 때문에 그렇게 쓰는 것 그렇기 때문에 그냥 상수값 그래서 time에다가 2pi 를 곱한게 무슨 상태냐면 time이 0에서 1초까지 만들어졌다고 생각해보면 1초에서의 time값은 1임 그것에 2파이를 곱하면 2파이초 가 됨 여기까지가 한바퀴 우리는 몇바퀴씩 도는걸 만들어내야 하기 때문에 여기에다가 freq를 곱해줘야 함

이 두개 가장 중요 그 다음에 세타를 넣어주어서 곡선을 만드는 것임

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

여기서 문제 time과 세타의 당연히 숫자열이니까 벡터일텐데 time의 벡터의 사이즈(숫자가 몇개 있느냐)와 세타의 벡터의 사이즈는 같다?

당연히 같음 우리가 time을 5000개의 타임 벡터를 만들었는데 freq곱하고 2파이 곱하기만 했을 뿐 그 안에 들어있는 벡터의 갯수는 같을 것

그렇게 해서 s 값이 들어가 있는데 이제 여기서부터 plotting하는 방법

```
fig = plt.figure() - plot이라는 라이브러리 가져옴
```

```
ax = fig.add_subplot(111) - 그 figure에 있는 함수를 해서 그림을 만들 준비를 하는 것이고
```

```
ax.plot(t[0:1000], s[0:1000], '.') - 여기서 plot을 하겠다 plot이라는 함수를 써서 밑에 나오는 그림처럼 만드는 것임 이 함수는 두개의 입력값을 받는데 x와 y값을 받음 그래서 그 중 하나로 우리가 이미 만들었고 관심이 있는 time을 x라고 넣었고 sin의 결과 값 s를 y라고 넣었음 그렇게 해서 점으로 표시 해줘라 하는 것
```

```
ax.set_xlabel('time (s)') - label에 이름 붙여주는 것
```

```
ax.set_ylabel('real')
```

우리가 5천개 만들었지만 천개만 plot한 것임 너무 다하면 뻑뻑해서 안보이니까 우리가 sin을 썼으니까 sin곡선이 0부터 시작하는 것을 볼수가 있음

exp는 오일러 공식의 e에 해당하는 것이고 다른 것은 크게 다른 점이 없음

ipd.Audio(s, rate=sr) 이런 공식이 있는데

play를 하는 것인데 s값을 위에서 뽑았음 sin함수로서 그것을 플레이 하는것인데 두가지값이 필요함 하나는 s, 하나는 sr를 적어줘야 함 이것을 안적으면 소리의 실체가 없음

-----  
<SOUND 복습 + 오일러 PHASE>

지난 시간에 cos sin phasor와 오일러 phasor를 배웠음  
오일러 공식은 이왕이면 외우는 것이 좋음  
지난 시간에 했던 것들 빠진 것 보충 복습

library import한 것들

from matplotlib import pyplot as plt - plotting할 때 필요한 것 import 시험문제를 낸다면 똑같은 것을 나타내는  
게 무엇이냐 import matplotlib.pyplot 이라고 쓸 수도 있음

from mpl\_toolkits.mplot3d import Axes3D - 3D Plotting을 해야할 때 필요한 것  
from mpl\_toolkits.axes\_grid1 import make\_axes\_locatable  
import IPython.display as ipd  
import numpy as np - 이것은 우리가 배운 것  
%matplotlib notebook  
from scipy.signal import lfilter

phasor에서 여러가지 변수들을 정의 parameter setting 왜 저렇게 해놓느냐 주는 장점이 무엇이냐 나중에 어떤  
변수를 바꿀 때 이것만 바꾸면 되기 때문에 목적상론 parameter 코딩상론 variable

사인 코사인 오일러 관련한 phasor function들은 애내들이 받아들이는 것이 각도값이지 시간이 아님 각도값만  
넣어서는 실체의 소리를 만들 수 없음 그렇기 때문에 시간을 먼저 만들어야 함

우선 이해를 돕기 위해 시간없이 각도값만 만들어서 phasor를 만들어 보겠음

0에서  $2\pi$ 까지 각도값(0에서  $6.xx$ 까지)

$\theta = \text{np.arange}(0, 2*\text{np.pi})$  세타값을 만들어서 사인함수에 넣을 준비

$s = \text{np.sin}(\theta)$  이 사인함수를 plot하면 사인 곡선이 나올텐데 plot을 어떻게 하느냐

$\text{fig} = \text{plt.figure}()$  위에서 받아온plt라는 함수 속에 들어있는 figure라는 function을 사용해서 figure를 먼저 만들어  
놓음

$\text{ax} = \text{fig.add_subplot}(111)$  그 figure에 들어있는 것 중에 add\_subplot이라는 것이 있고

figure는 이 화면 전체를 말하는 것이고 subplot이라고 중간에 여러가지 해서 만들 수 있는데 화면을 분리함 2  
by 2 로 화면을 분할하는데 그 중 첫번째 것이 221



`ax.plot(theta, s, '.')` 실제 plotting으로 하려는 것을 x와 y값을 적어주면 됨 여기서 점(.)대신 (-)이것을 써주면 직선의 형태로 나타남 아무것도 안적으면 default값은 직선

7개의 세타값이 사인에 들어가서 나온 값이 x값들 똑같이 7개 x값과 y값의 갯수 corresponding

하지만 세타값이 7개밖에 없어서 그래프가 너무 sparse함 그렇기 때문에 `theta = np.arange(0, 2*np.pi)` 여기서 `(0, 2*np.pi, 0.1)` 이렇게 간격의 값을 바꿔주면 됨

여기서 decoration을 좀 해주고 싶은데 x축 y축의 값들의 이름을 label 해주는 것이 `ax.set_xlabel('theta in radians')`  
`ax.set_ylabel('value')`

이 사인 곡선에서 x의 값들이 equidistance한데 과연 y의 값들도 equidistance 할까? - 아님, linear해야지만 x의 값들이 equidistance할 때 y의 값들도 equidistance. 하지만 사인 함수는 linear(라인,  $y=ax+b$ 의 형태로 나타나는 것)이 아니기 때문에 x축이 equidistance라 할지라도 y축은 그렇지 않다 - nonlinear(일차함수를 제외한 모든 것)

이 다음부터 저번 시간의 복습

소리는 시간이 있어야 함 그렇기 때문에 소리를 정의해줘야 함

time tick? 의 갯수를 먼저 만들어 준다 만약 시간이 1초라면 어느 것의 갯수와 같을까? sampling rate와 같음 sampling rate의 갯수만큼 만들어 준다 (1, sr)이렇게 하면 sr의 갯수만큼의 time tick을 만드는 것임 1초가 아닌경우에는 이것보다 작아져야 함 그렇기 때문에 dur만큼 곱해주면 되는 것 그것을 sr로 나눠주면 됨

이 time을 기반으로 한 세타를 만들면 됨 우리가 time은 0부터 1까지인데 곡선 한바퀴 돌려면  $2\pi$  필요 그렇기 때문에 t에다가  $2\pi$ 를 곱해주는 것 여기까지 하면 1초에 한바퀴 도는 것을 만든 것이고 몇바퀴씩 도는 것을 해주려면 freq를 곱해주면 됨

이제 t과 theta, s 값 확보 그런데 우리는 일단 time에 관련해서 그 값들에 관심이 있는거기 때문에 time과 s값 넣어주면 됨 (theta값은 뺄함 한바퀴면 2파이, 두바퀴면 4파이,,)

시험 문제 여기 있는 점들의 갯수는 몇개인가 - 1000개 t와 s의 범위가 correspond해야 몇 콤마 몇 이렇게 점이 찍힘 (범위가 다르면 안됨)

지금까지는 sin혹은 cos관련된 phasor만 배웠는데 오일러 공식을 이용한 phasor를 만들어 볼 것

`np.exp` - 이것이 오일러 공식에서 큰 e라고 생각하면됨  $1j - 1i$ , 이제 이것을 c(complex)라는 변수에 받아볼 건데 c의 모든 값들은  $a+bi$ 의 형식으로 되어있음

실제로 값들을 보면 모두  $a+bi$ 로 되어 있음 (-01 = 10분의 1, -02 = 10의 제곱분의 1 ... - 앞의 소수점을 어디로 옮기느냐를 결정 - 왜 저런 형식으로 표현하느냐 컴퓨터는 정보량을 같게 해줄 필요가 있는데 저렇게 표현하면

큰수든 작은수든 똑같은 형식으로 표현해줄 수 있음  $0.293928e-0.1j$  이런 식)

이것을 어떻게 plotting할까

다른 것 다 똑같고 projection 3D로만 바꿔주면 되고 ax.plot해서 3D답게 변수 3개가 들어가야 함 (t, c.real(실수 부분), c.imag(허수 부분)) 이런 식

복소수는 하나의 값으로 나오지만 사실 거기에 두개의 정보가 있는 것임 a값 b값 하나로 받을수도 있지만 그것을 두개의 정보로 받을 수 있음 그렇기 때문에 나눠서 real - a값, imag - b값 이렇게 받음 c. 이것은 a혹은 b값만 가져오게 해줌

projection에 관한 것 배웠는데 허수부분을 죽이고 실수만 보면 cos그래프가 되고 허수만 보면 sin그래프가 보이는 것을 알 수 있음

ipd.Audio(s, rate=sr) - ipd위에 library에서 가져왔음 오디오라는 function을 가져온 것 s대신 c.real, c.imag이런 것 써도 됨

-----  
<SOUND PURE TONE의 합으로 성대를 표현>

phase는 세타값만 있더라도 곡선은 만들어지기도 하지만 실제 소리를 만들기 위해서는 time이 필요하다는 것을 배웠음

sr은 점들이 숫자들이 1초동안 얼마나 많이 나오냐 고 freq는 shape이 1초동안 몇번 반복 되는가 단위가 같지만 둘이 구분할 줄 알아야 함

# generate signal by cosine-phasor

s = np.sin(theta)

사인 곡선을 만들어 내는데 그 크기를 결정할 수 있음 스프링의 크기가 얼마나 될지 크게 하고 싶으면 전체에다가 x2 그러면 -2 부터 2까지인 사인 곡선이 나올 것임 이것이 amplitude(진폭)

complex phasor도 마찬가지

complex phasor의 반지름과 amplitude의 값은 같다

가장 최소에 해당하는 harmonics를 정하고 frequency를 정하고 f0를 먼저 정하고 sin웨이브 만들고 그런 식

이것을 파이썬으로 어떻게 구현을 할까

\* 너무너무 중요한 개념 \*

sr와 freq 연결되는 부분이 있음

sr이 100hz라 했을 때 우리가 표현할 수 있는 숫자의 갯수가 100개라는 뜻 100개를 가지고 freq를 1hz로

표현할 수 있을까?

-있다 한번의 사인웨이브의 주기가 있으면 됨

2hz 가능할까 -있음

10000hz 가능할까 - 만번을 왔다갔다하게 할 수 있을까 숫자는 100개밖에 없는데.. - 안됨 sr이 충분히 있어야 그만큼의 freq를 표현할 수 있음

sr이 10hz라 쳤을 때 정말 최대로 freq를 만들면 5번임 즉 freq= 5hz가 최대임 주어진 갯수를 가지고 표현할 수 있는 최대의 주파수(freq)는 절반임 - 이것을 nyquist frequency라고 함

그래서 우리가 CD 음질이라고 할 때 sr=44100을 담을 수 있음 이럴 때 22050freq정도까지 표현할 수 있음  
그렇다면 왜 CD음질을 44100으로 잡았을까 - 사람이 들을 수 있는 가청주파수가 20000freq 그렇기 때문에 44100정도면 충분히 다 들을 수 있음

# generate samples, note conversion to float32 array

F0 = 100; Fend(f가 어디까지 갈것인지 f end) = int(sr/2); s = np.zeros(len(t));

time은 이미 위에서 만들어졌음

for freq in range(F0부터, Fend+1까지 제일 마지막것도 포함시키기 위해서 +1, F0만큼씩):

theta = t \* 2\*np.pi \* freq - 세타 생성

tmp = amp \* np.sin(theta) - 사인에다가 세타값 입력

s = s + tmp (s는 signal 변수.)

맨 처음 f0값을 넣었을 때 s값이 정의가 안되어있기 때문에 첫 루프에서 에러가 나게 되어있음 그렇기 때문에 정의해줘야 함 s = np.zeros(len(t)) 맨위에 있는 이것이 그것

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)

다음시간은 vocal tract에 관한 것을 어떻게 만드느냐에 관한 것

-----

지금까지 wave에 대해 배웠음

이것이 여러개 쌓여서 성대에서 나는 소리가 되는 것 praat에서 spectrogram에 대해 배웠는데 이것과 spectrum에 대한 이해 필요

formant라고 하는 것이 성대를 거쳐 아 이 등의 소리가 나는 것인데 산맥에 따라 아 모음도 되고 이 모음도 되고 그러는데 그러한 작업을 오늘 할 것임

지금 우리가 만든 것은 amplitude가 모두 똑같지만 이것을 gradually decreasing하게 만들어야 하고 그 다음에 산맥을 차례대로 만들어야 함 이 작업을 하면 성대 그 자체의 소리에서 입안 구조를 거친 소리로 변하게 됨

sin과 cos그래프는 소리가 서로 다르지 않음 (ipd.Audio(s, rate=sr)) sin과 cos은 2분의 파이 차이임 즉 90도 차이라는 점이 있음 그렇다면 8분의 파이 이동했으면 소리가 다를까? - 아님 이 각도를 phase라고 하는데 phase는 우리의 귀가 인식을 못한다 phase를 shift해도 sensitive하지 않다 우리는 frequency의 변화만 알아차림

c의 값 자체를 plotting하지는 못함(복소수기 때문에) a+bi에서 a,b값을 가져와서 2차원에서 찍을 뿐

그림에서

한 타임 포인트에서의 주파수 성분을 보여주는 것이 spectrum

print와 같은 function을 즉석에서 만들었음 하지만 어떻게 만들었는지는 몰라도 되고 하는 역할 어떻게 쓰지만 알면됨

def 다음 function name 그 다음 괄호 열고 내가 쓰고싶은 입력값을 적으면 됨

F - frequency

sr - sampling rate

BW - bandwidth

resonance 함수

첫번째 입력으로 sr, 산맥의 위치를 적는 것이 RG(Frequency값), 산맥이 얼마나 뚱뚱하냐 뾰족하냐의 정도가 BWG(산맥의 shape)

spectrum에서 크기가 똑같은 100 200 300 400 500... 에서 gradually decreasing을 만들어야 하는데 f0을 가운데로 놓고 큰 산 하나를 만든다고 생각하면 됨

RG = 0

BWG = 100

f0을 가운데에 놓고 100정도 두개의 산을 만들어라

~~해서

RG = 3500

BWG = 200

여기까지 만들면 산맥을 만든 것

여기까지는 입술 없는 소리

```
s = lfilter(np.array([1, -1]), np.array([1]), s)
ipd.Audio(s, rate=sr)
```

이것이 입술이 있는 것을 반영한 것