세 가지: larynx, velum 에서 비음, constrictor

constrictor: lips toungue tip, tongue body

specify 되면 특정 소리 말할 수 있음

ex)/p/ lips cl: dilavelor cd: stop, velum: raised, larynx: open(voiceless)

/d/ tongue tip, cl: alveolar cd: stop, velum: raised, larynx: closed

/z/ tongue tip cl: alveolar cd: fricative, velum: raised, larynx: closed

/n/ tongue tip cl: alveolar cd: stop, velum: lower, larynx: closed

(velum: lower 은 mn 응밖에 없음)

Praat 사용법

10/1

코딩은 자동화라고 생각하자. 그런데 왜 자동화할까? 사인웨이브의 반복이라.

세상의 언어는 단어와 문법으로 이루어진다.

컴퓨터 언어도 이와 비슷하다.

컴퓨터 언어: 변수에 정보 지정. 조건 if 컨디셔닝, 반복 for 루프, 함수 필요. 함수가 가장중요

정보의 종류는 숫자와, 문자

Jupyter-variable

a=1 이라고 했을 때, 오른쪽에 있는게 정보이고 왼쪽에 있는 것이 변수.

함수() print(a) 일 때 ()가 입력이다.

셀 셀렉하고 b 누르면 below 에 셀 생성, a 누르면 above 에 생성, x 누르면 셀 제거

, 실행은 shift+enter

문자를 정보에 입력하려 할 때 무조건 ''를 붙여줘야 함

변수명만 쳐도(print 안해도) 결괏값을 보여준다.

;입력하면 줄바꿈안해도 한줄에 쓸 수 있음.

한 변수에 여러개의 정보를 넣는게 list []대괄호로 표현,

type 변수 입력하면 list(여러개), int(숫자), float(실수), str(문자) 등 출력 dict 를 표현하려면 중괄호{}

안에 표제어:설명어 의 페어 형식을 가진다.

Q a=[1, 'love', [1, 'bye']] 일 때, a 에는 세 개의 정보가 들어있고, type(a)를 입력하면 list 를 출력한다.

이 안에는 int, str, list 가 들어있고, 세 번째 list 안에는 int 와 float 이 들어있다.

()는 tuple

10/8

a=[1, 2]

b=[3, 4]

c=a[0]+b[0]

c 를 하면 4 가 출력됨. []는 0 번째 값을 retrieve 하는 역할을 함.

같은 원리로

a=[1,2]

b=[3, 4]

c=a[1]+b[1]

c 를 하면 6 이 출력됨.

partial value 값을 가져올 때는 대괄호 [] 사용.

float 는 varible 이 들어오면 float type 으로 바꿔줌.

a 의 type 이 float 인데 int 로 강제하면 int 로 출력 a = 1.2; a = int(a); print(type(a))

어떠한 내부정보로 들어갈 땐 []안에 인덱스를 적으면 내부정보를 가져온다.

a = '123'; a = list(a); print(type(a)); print(a); print(a[2]) 실행하면

<class 'list'>

['1', '2', '3']

3

a 는 1, 2, 3 이라는 문자를 가진 list 가 된 거임. 리스트되고 a 의 두 번째 값인 3 이 문자로서 나온 것. 이 부분 이해.

a = [1,'2', [3, '4']]; print(type(a)); print(a[0]); print(a[1]); print(a[2])

<class 'list'>

1

2

[3, '4']

여기서 1 은 숫자, 2 는 문자로 출력된 것.

dict 에서 access 할 때는 pair 에서 앞부분을 index 로 씀.

list 처럼 숫자를 인덱스로 쓰지 않음.

a = {"a": "apple", "b": "orange", "c": 2014} ;print(type(a)); print(a["a"])

<class 'dict'>

apple

여기서 print(a["a"])는 a 안의 "a"인덱스 값을 가져오는 것.

그래서 a = {"1": "apple", "b": "orange", "c": 2014}; print(type(a)); print(a["1"])

해도 같은 값 나옴.

a=[(1,2,3), (3,8,0)]; print(type(a)); a

<class 'list'>

[(1, 2, 3), (3, 8, 0)]

type(a[0])

tuple

string

s = 'abcdef'; print(s[0], s[5], s[-1], s[-6])

a f f a

인덱스가 음수일때도 확인.

Q. 첫 번째 인덱스는 항상 0 이고 마지막 인덱스는 –1 인 것.

s = 'abcdef'; print(s[1:3], s[1:], s[:3], s[:])

bc bcdef abc abcdef

인덱스가 1:3 이면 1 위치에서 3 위치 전까지, 1:이면 1 위치에서 끝까지.

:3 이면 처음부터 2 위치까지, :이면 전체

len()은 정보의 길이를 준다고 생각.

s = 'abcdef'; len(s)

6

s = 'abcdef'; s.upper()

'ABCDEF'

s = ' this is a house built this year.₩n'; result = s.find('house'); result

11

띄어쓰기 포함 11 번째에서 house 가 시작된다는 의미.

result = s. find('this'); result; result = s. rindex('this'); result;

1

23

s = s.strip(); s

'this is a house built this year.'

space 나 불필요한 기호를 제거해주는 함수

tokens = s.split(' '); tokens

['this', 'is', 'a', 'house', 'built', 'this', 'year.']

은 split 괄호 내부에 있는 것으로 자르라는 의미.

s = ' '.join(tokens); s

'this is a house built this year.'

은 다시 붙이는 것

s = s.replace('this', 'that'); s

'that is a house built that year.'

은 this 를 that 으로 바꿔주는 것.

10/10

variable assign 은 =해서 하는 것. string 하고 list 하고 비슷한 성격을 가진다. varible 종류.

funciton 에 대해 배웠다. 오늘은 for loop 와 if 조건에 대해 배울 것.

Jupyter 에 노트남기는 법: #쓰고 쓰면 명령어 실행 안됨. 셀종류를 markdown 으로 바꾸면 실행 x.

syntax

```
a = [1, 2, 3, 4]
for i in a:
    print(i)
```

1

2

3

4

in 뒤에 있는 것을 하나씩 돌려서 한 번할 때마다 i 로 받아서 뭔가를 하라

range 뒤에 어떤 함수가 나오면 list 를 만들어줌. 인덱스를 0 부터 만들어주는것

```
a=[1,2,3,4]
for i in range(4):
    print(a[i])
```

1

2

3

4

print (i)

0

1

2

3

range(4): 0 부터 4 개 (0 1 2 3)

a = [1, 2, 3, 4]

for i in range(len(a)):

    print(a[i])

1

2

3

4

첫 번째 루프에서 i 는 0

len(a)=1

len(a)=4

```
a = ['red', 'green', 'blue', 'purple']

for i in a:

    print(i)
```

랑

```
a = ['red', 'green', 'blue', 'purple']

print (a[0])

print (a[1])

print (a[3])

print (a[4])
```

의 결괏값은

```
red

green

blue

purple
```

로 같다

```
a=[red, green, blue, purple]

for s in a:
```

```
    print (s)
```

하면 네 번 루프를 돌고 s 는 매번 달라진다.

```
a = ['red', 'green', 'blue', 'purple']
for s in range(len(a)):
    print(s)
```

하면

0

1

2

3

```
a = ['red', 'green', 'blue', 'purple']
for s in range(len(a)):
    print(a[s])
```

enumerate 는 번호를 매기는 것.

```
a = ["red", "green", "blue", "purple"]

b = [0.2, 0.3, 0.1, 0.4]

for i, s in enumerate(a):
```

첫 번째 루프에서 i 에는 0, s 에는 red 가 들어감. for 뒤에서 앞(i)에는 번호, 뒤(s)에는 element.

```
a = ["red", "green", "blue", "purple"]

b = [0.2, 0.3, 0.1, 0.4]

for i, s in enumerate(a):

    print("{}: {}%".format(s, b[i]*100))red: 20.0%

green: 30.0%

blue: 10.0%

purple: 40.0%
```

i=0 s=red 이런식으로 네 번 돌아간다. format 함수 안 두 개가 {}에 각각 들어간다.

```
a = ["red", "green", "blue", "purple"]

b = [0.2, 0.3, 0.1, 0.4]

for s, i in zip(a, b):

    print("{}: {}%".format(s, i*100))
```

a 와 b 의 길이는 같아야. a 와 b 를 페어로 만들어줌

```
a = 0
if a == 0:
    print("yay")
```

yay

a 가 0 이면 yay 를 print 해라

```
a = 0
if a != 0:
    print("yay")
```

a 가 0 이 아니면 yay 를 print 해라. 그래서 프린트 안됨.

```
    print("no")
a = 0
if a != 0:
    print("yay")
else:
    print("no")
```

no

요부분 유튜브 볼 것

```
for i in range(1,3):
    for j in range(3,5):
        print(i*j)
```

3

4

6

8

두 번씩, 1 일 때 두 번째 줄 3,4 돌고 2 일 때 3,4 해서 총 네 번 돈다.

```
for i in range(1,3):
    for j in range(3,5):
        if j >=4:
            print(i*j)
```

4

8

이때 indent 가 없으면 다 에러뜬다!

Q. 특정 컴비네이션이 있을 때 없을수 있음. 그런게 뭘까? 존재한다 안한다로 답.

Q. lips 가 cl 로 velar 를 가질 수 있을까?이거는 영어에서 못하는걸까 사람이 못하는걸까?

사람이 못하는 거임

Q. lips 가 cl 로 alveolar 로 가지는 언어가 있을까?

이론상으로 가능. 인체 구조상 가능.

기말

숫자화되지 않은 데이터들이 어떻게 숫자화되는가. 숫자열로 표현되는가.

이미지: 모든데이터는 벡터의 상태일 때 다루기 쉽고 벡터화된다.

소리: 소리도 벡터화 된다.

텍스트: 5000 번째 단어라고 했을 때 0000....1(5000 개) 이런식으로 표현. 역시 벡터화

모든 벡터화된 데이터는 수학적 처리를 해야하는데, 그냥 더하기로 처리하면 안된다.

이걸 해주는게 numpy.

```
import numpy

A = numpy.array(a)

B = numpy.array(b)
```

array 는 list 를 계산가능하게 만들어주는 것. import 반드시 해줘야.

```
A+B

array([ 3, 7, 11])
```

```
type(A)

numpy.ndarray
```

```
import numpy as np

A = np.array(a)

B = np.array(b)
```

as np 해주면 이렇게 바꿔쓸 수 있다.

numpy 라는 패키지 안에는 여러 패키지가 포함될 수 있다. import numpy 하면 numpy 안에 있는 모든 것들을 쓸 수 있다. A 안에 D 안에 f 가 있다고 하자. 부르려면 numpy.A.D.f 라고 하면 됨.

from numpy import A 를 하면 그다음부터는 A.D.f 로 불러올 수 있다.

Q. 불러올수 있는 다양한 방법 중 틀린 것은? import numpy as np

import matplotlib.pyplot as plt = from matplotlib import pyplot as plt

이렇게 하면 plt 만 해도 불러올 수 있음.

import numpy as np

import matplotlib.pyplot as plt

np.empty([2,3], dtype='int')

array([[8, 0, 0],

[0, 0, 0]])

숫자는 랜덤. [행,렬]이 만들어짐.

np.zeros([2,3])

array([[0., 0., 0.],

[0., 0., 0.]])

zeros, ones 함수는 기본적으로 dtype 이 float 인 arrary 를 만든다.

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

array([[0., 0., 0.],

[0., 0., 0.]])

array 는 리스트를 계산할 수 있게 만들어줌.

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

X

array([[1, 2, 3],

       [4, 5, 6]])
```

```
np.arange(0,10,2, dtype='float64')

array([0., 2., 4., 6., 8.])
```

dtype=뭐라고 하냐에 따라 무슨 숫자가 나올지 정할 수 있다.

float 다음 숫자는 몇 번째 소수점까지 표시할 것인가. 얼마나 정확하냐와 연관되어있음.

arrange(0,10,2)는 0 부터 10 개 2 씩 뛰면서

```
np.arange(5)

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

인덱스 0 부터 다섯 개를 만든 것.

```
np.arange(0,10,2)

array([0, 2, 4, 6, 8])
```

이렇게 하면 0 부터 10 전까지 2 간격으로 만듦

```
np.linspace(0,10,6, dtype=float)

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

0 부터 10 까지 6 개로 나눠주는 것(끝 포함). 그 간격이 다 같음.

X = np.array([[1, 2],[3, 4],[5, 6]])

array([[1, 2],

[3, 4],

[5, 6]])

X = np.array([[[1, 2],[3, 4],[5, 6]], [1, 2],[3, 4],[5, 6]]])

array([[[1, 2],

[3, 4],

[5, 6]],

[[1, 2],

[3, 4],

[5, 6]]])

[]대괄호가 두 개 나오면 이차원 세 개 나오면 삼차원

X.ndim 하면 차원확인 가능

X.shape

(2, 3, 2)

제일 큰 괄호 속 2 개 두 번째 괄호 속 2 개 제일 작은 괄호 속 2 개

X.astype(np.float64)

array([[1., 2., 3.],

[4., 5., 6.]])

type 바꿔주는 것.

np.eros_like(X)

array([[0, 0, 0],

[0, 0, 0]])

다 0 으로 바꿔줌. X*0 해도 됨

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

print(data)

plt.hist(data, bins=10)

plt.show()

100 개의 랜덤한 데이터를 정규분포로 만들어라. 무조건 0 을 포함한 자연수값.

Q. 요 블록 다 합하면 몇 개일까? 100 개

여기서 data.ndim 하면 1(1 차원이라는 의미) data.shape 하면 100 (100 개의 벡터라는 의미)

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

X

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.]]])

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

x 의 엘리먼트는 2*3*4=24 개로 동일하다. 무슨값인지 모르면 –1 적으면 되고 4 적어도됨.

동일한 차원으로만 reshape 가능

np.allclose(X.reshape(-1, 3, 2), Y)

True

X 랑 Y 랑 같냐

a = np.random.randint(0, 10, [2, 3])

b = np.random.random([2, 3])

np.savez("test", a, b)

0 과 10 사이에서 숫자를 픽해서 [2,3]모양을 만들어라.

파일로 저장해주는 것. test.npz 생겨났을 것.

ls –al test*

있는지 확인

del a, b

who

뭐가 지금 available 한지. 근데 delete 해서 없을 것.

npzfiles = np.load("test.npz")

```
npzfiles.files
```

```
['arr_0', 'arr_1']
```

```
npzfiles['arr_0']
```

```
array([[1, 5, 2],
       [1, 7, 0]])
```

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

```
data
```

csv 는 컴마로 분리된 variable. csv 로부터 data 로 들어온 것.

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

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]]

a == b

array([[False, False, False],

[False, True, False],

[False, False, False]])

a > b

array([[False, False, False],

[False, False, True],

[True, True, True]])

a.sum(), np.sum(a)

(45, 45)

요부분 이해.

a.sum(axis=0), np.sum(a, axis=0)

(array([12, 15, 18]), array([12, 15, 18]))

a.sum(axis=1), np.sum(a, axis=1)

(array([6, 15, 24]), array([6, 15, 24]))

몇 번째 차원에서 sum 을 할건가. 첫 번째 차원에서 sum 을 해라.

11/5

Phoasor

sinusoidal을 만들어내는게 phasor. sin의 입력값은 radians(2파이 등). degree(0도-360도)가 아니라.

쉽게 말해 sin, cos이 phasor.

오일러 공식: e^θi = cos(θ)+sin(θ)i

cos가 real, sin imaginary

그냥 e가 숫자값이구나..만 알면됨. f(θ)=e^θi. e만 변하면 값이 나오는 것. 오일러 공식이 새로운 phasor.

θ

0,

π/2

π

3π/2

2π

cosθ

1

i

-1

-i

1

(a,b)

(1,0)

(0,1)

(-1,0)

(0,-1)

(1,0)

복소수를 plot 하는 방법을 배울 것.

복소 평면. x 축 a, y 축 b 이라고 하고 θ 값을 쭈욱 이어보면 원을 그리게 됨.

x 축과 원점-(a,b) 연결한 선이 이루는 각이 θ

```
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
```

Q. 첫줄은 import matplot.pyplot 와 같다.

1. 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
```

sr, freq 둘다 Hz 가 단위. 1 초에 몇 개이니까.

음질의 정도가 sr. 1 초동안 들어갈 수 있는 time tics. t=0.001 0.002 0.003....일 때 sr=1000

freq 는 1 초에 태극문양이 몇 개 들어가느냐.

나중에 변수값 바뀔 때 얘네만 바꿔주면 돼서 편리.

2. generate time

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

+1 은 제일 마지막꺼 더해주는 것. sr*dur 은 time tic 을 dur 까지 씀.

여기서는 1/10000 부터 5000/10000 까지인 것.

왜 시간을 만들어주어야 할까. phasor 함수들은 받아들이는 입력이 radian 만이기 때문.

이것만으로는 소리의 실체를 만들 수 없음.

3. generate phase

theta = t * 2*np.pi * freq

들어가는 각도값이 phase. 총 100 바퀴를 돌아라.

t 연동시키는 것

4. generate signal by cosine-phasor

s = np.sin(theta)

Q. time 과 theta 의 벡터사이즈는 같다.

5. 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')

실제 소리를 만들때는 t 가 필요해서 x 축으로. theta 는 그냥 한바퀴 반복, 뻔함.

Q. 점들의 개수는 몇 개인가 1000 개.

t 랑 s 랑 같아야 함. 다르면 왜 실행이 안되는지 알아야.

시간이 없다고 가정하면,

theta = np.arange(0, 2*np.pi,) 이때 theta 의 단위는 radian

s = np. sin(theta)

총 7 개의 값이 모두 corresponding 하다.

theta = np.arange(0, 2*np.pi, 0.1)하면 열배로 촘촘하게 값을 그리게 됨.

시간이 없음.

subplot 은 화면 분리해서 여러개 들어갈 수 있는 것.

111 은 1x1 의 첫 번째 221 는 2x2 의 첫 번째 222, 223, 224 도 가능.

non-linear 하다

41 분 유튜브 확인할것

6. generate signal by complex-phasor

c = np.exp(theta*1j)

c 의 표기법이 다 같은 이유는 형식(a+bi)을 통일시키는 것. -01 은 소수점 하나를 왼쪽으로 옮긴다는 의미

```
fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')
ax.plot(t[0:1000], c.real[0:1000], c.imag[0:1000], '.')
ax.set_xlabel('time (s)')
ax.set_ylabel('real')
ax.set_zlabel('imag')
```

c 라는 하나의 복소수 값에는 a,b 값도 포함되어있다.

7. ipd.Audio(s, rate=sr)

s 대신 c.imag(=s)을 넣어도 소리 나옴.