English Phonetics

2016130743 영어영문학과 신지환

alveolar 치경음/ hard palate 경구개 / soft palate(velum) 연구개 / larynx 목구멍

Q. velum 이 raised 되면 nasal tract 이 막힐까? 막힘. 이때 나는 소리는 모든 모음. 비음을 뺀 모든 자음.

Q. 코로 숨을 쉴때 velum 이 raised or lower? 열림(lower). 소리내는 순간 올라감(closed).

oro-nasal process: nasal 과 아닌것을 구분. 내려가면(열리면) 비음

phonation process: 유성음 voiced 과 무성음 voiceless 으로 나눠지는 기준. 진동의 유무

lip tongue tip tongue body 모두 constrictor

cl 은 앞뒤 cd 는 상하(얼마나 막을 것인지)

tongue tip 은 constriction location 4 개 정도가 있음: dental alveolar palate-alveolar retroflex

은 응 p t k stops / 스 즈 th sh fricatives / approximants 는 l r wo yu

모든 모음은 constrictor 로써 tongue body 만 쓴다

Q velum raised(open) constrictor cl=alvelolar cd=stop : t

Q. 모음과 같은 constrictor 를 쓰는 자음을 쓰시오 k. 여기서 velum 이 lower 되면 응소리(larynx 가 closed 된 상태)

Acoustics: formant 값에 따라 무슨 모음인지 구분할 수 있다.

Praat: 분석하는 목소리의 성별에 따라서 레인지 다르게 설정해줘야함.

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

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 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.

피치는 일초에 몇번 움직였느냐에 대한 계산, 단위: Hz

sin wave 모양을 결정하는 두가지

1. frequency:1 초동안 몇번 반복되는지

2. magnitude: 크기

두번째 소리는 성대에서 녹음하는 소리. 성대에서 나오는 소리는 높낮이의 정도만 다르고 입에서 결정된다.

(중요)모든 소리(complex tone)는 다르게 생긴 여러 사인웨이브(simplex tone)의 조합이다.

우리가 듣는 모든 소리는 complex tone

frequency 가 낮으면 저음

magnitude 는 첫번째, 세번째, 두번째 순.

sin wave form 의 x 축은 시간 y 는 value 숫자값

오른쪽에 있는 spectrum 처럼 단순화될 수 있다(spectral analysis)

맨아래 spectrum 을 합성 synthesis

complex tone 의 반복되는 패턴(frequency)은 frequency 가 가장 낮은 simplex tone 의 frequency 와 같다

sum wave 의 frequency 가 첫번째 wave 와 같음. 이때 첫번째 wave 는 fundamental frequency(F0)이라고 함

pure tone=simplex tone

spectro-gram 은 spectrum 은 시간으로 visualize 한 것

spectrum 은 시간 개념이 없음.

y 축은 프리퀀시

source 에 입모양(filter)을 더해야 아에이오우 소리가 만들어진다.

첫번째 f0 가 pitch. 주파수가 점점 줄어드는게 모든 사람의 패턴

우측 하단 그래프에서도 필터된 후에도 등간격으로 source 가 유지되는 걸 볼 수 있음

사람목소리는 하모닉스로 이루어져있다. 사인웨이브들의 배음으로 이루어져 있다.

콤플렉스 톤은 심플렉스톤의 반대.

F0 는 래링스에서 보컬폴드가 몇번 떨리는지와 일치.

하이프리퀀시로 갈수록 앰플리튜드가 주는게 우리 목소리의 특징이다.

egg 는 보이스 소스에서 직접 녹음한거(필터거치기 전)

보컬트랙(입모양)에 의해 목소리가 만들어지는걸 필터라고 한다

밸리를 만들어주는게 필터. 산맥을 fromants

하모닉스에서 첫번째 산맥을 f0, 그게피치

일반 소리는 하모닉스가 아님. 사람목소리만 하모닉스. 기타소리도 하모닉스.

mono sound 로 합쳤을 때 나오는 것을 perse train 이라고 함.

spectral analysis 하면 천천히 감소하는게 보인다

output spectru 에서 첫번째 뾰족한 것을 formants

f1 f2 가 뭐냐고 했을 때, 첫번째, 두번째 formants 의 프리퀀시를 읽어주면 된다.

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!

숫자화되지 않은 데이터들이 어떻게 숫자화되는가. 숫자열로 표현되는가.
이미지: 모든데이터는 벡터의 상태일 때 다루기 쉽고 벡터화된다.
소리: 소리도 벡터화 된다.
텍스트: 5000번째 단어라고 했을 때 0000....1(5000개) 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)
```

array는 list를 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 np해주면 이렇게 바꿔쓸 수 있다.

numpy라는 패키지 안에는 여러 패키지가 포함될 수 있다. import numpy하면 numpy안에 있는 모든 것들을 쓸 수 있다. A안에 D안에 f가 있다고 하자. numpy.A.D.f라고 할 수 있고, from numpy import A를 하면 그다음부터는 A.D.f로 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 plt하면 plt만 해도 불러올 수 있음.

```
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.]])

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

X

array([[1, 2, 3],

[4, 5, 6]])

array는 리스트를 계산할 수 있게 만들어줌.

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

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

dtype=뭐라고 하냐에 따라 무슨 숫자가 나올지 정할 수 있다.

float다음 숫자는 얼마나 정확하냐와 연관되어있음.

arrange(0,10,2)는 0부터 10개 2씩 뛰면서

np.arange(5)

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

인덱스 0부터 다섯 개를 만든 것.

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

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

0부터 10까지 6개로 나눠주는 것. 그 간격이 다 같음.

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

array([[1, 2],

[3, 4],

[5, 6]])

[]대괄호가 두 개 나오면 이차원 세 개 나오면 삼차원

X.astype(np.float64)

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

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

type바꿔주는 것.

np.zeros_like(X)

array([[0, 0, 0],

[0, 0, 0]])

다 0으로 바꿔줌. X*0해도 됨

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

print(data)

plt.hist(data, bins=10)

plt.show()

100개의 랜덤한 데이터를 정규분포로 만들어라.

여기서 data.ndim하면 1(1차원이라는 의미) data.shape하면 100 (100개의 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.]]])
x의 엘리먼트는 2*3*4=24개로 동일하다. 무슨값인지 모르면 -1적으면 되고 4적어도됨.

np.allclose(X.reshape(-1, 3, 2), Y)
True
X랑 Y랑 같냐

a = np.random.randint(0, 10, [2, 3])
b = np.random.random([2, 3])
np.savez("test", a, b)
0과 10사이에서 숫자를 픽해서 [2,3]모양을 만들어라.
파일로 저장해주는 것. test.npz 생겨났을 것.

ls -al test*
있는지 확인

del a, b
who
뭐가 지금 available한지. 근데 delete해서 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

csv는 컴마로 분리된 variable. csv로부터 data로 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 sum을 할건가. 첫 번째 차원에서 sum을 해라.