ENGL238 English Phonetics

영어영문학과 2018130887 정세은

1)English Consonants, Vowels

-영어의 알파벳은 총 26개로, 자음 21개과 모음 5개로 이루어져 있다. 그 중, 모음은 monophthongs와 diphthongs로 나누어진다.

-모든 소리는 발음할 때 목이 떨리는지 안 떨리는지에 따라 voiced와 voiceless로 나누어진다. 모든 모음은 voiced이다.

-j(y)는 자음이다. year은 자음으로 시작하는 단어, ear은 모음으로 시작하는 단어이다. 헷갈리지 말 것!

2)Phonetics

-Phonology : 인지적, 추상적, 머릿속에서 일어나는 것 vs. Phonetics : 물리적, 실제로 소리 나는 것

-Articulatory phonetics (from mouth) : How to produce speech

-Acoustic phonetics (through air) : How to transmit speech

-Auditory phonetics (to ear) : How to hear speech

3)Articulation

-Vocal track(upper) : lip, teeth, alveolar ridge, hard palate, soft palate(velum), uvula

-Vocal track(lower) : lip, tongue(tip, frond, back, blade, center, root)

-5 speech organs : oro-nasal process(velum), articulatory process(lips, toungue tip, toungue body), phonation process(larynx)

4)Phonation process

-Larynx = voicebox

-voiced : can feel vibration (v,z,l,m,a,i..) -voiceless : can't feel vibration (f,s,k,p,h..)

5)Oro-nasal process in velum

-velum is lowered → nasal sound, when breathing -velum is raised → oral sound

6)Articulatory process: in lips, toungue tip, toungue body

7)Control of constrictios(articulators)

: Each constrictor(lips, tounge tip, tounge body) needs to be more specific in geometry

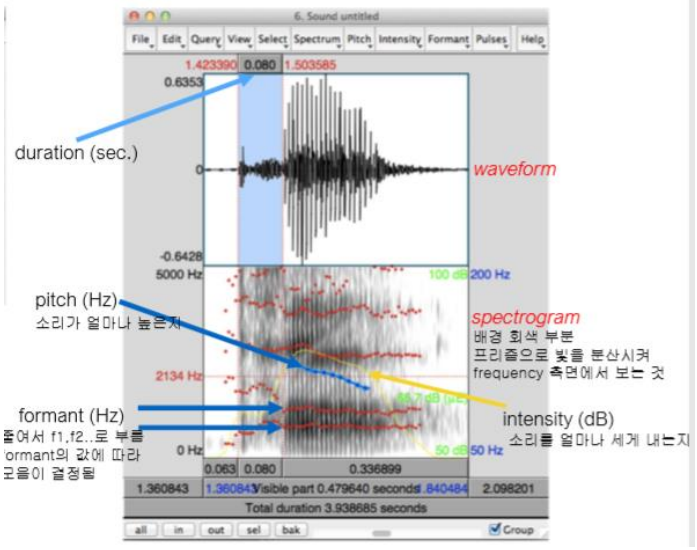
-Constriction location(CL) : lips, tounge body, tip

-Constiction degree(CD) : stops, fricatives, approximants, vowels

8)Phonemes : Individual sounds that form words, a combination of speech organ's actions

<Praat>

1)Acoustics in Praat



-duration : 소리가 지속되는 기간 (초)

-pitch : 소리의 높낮이

(male : 65-200Hz, female : 145-275Hz)

-formant(Hz) : formant의 값에 따라 모음이 결정됨

-intensity(dB) : 소리의 세기

-spectrogram : 배경 회색 부분. 프리즘으로 빛을 분산시켜 frequency 측면에서 보는 것

2)Vowel acoustics

-Measure pitch using Praat : praat 상의 큰 파도 = larynx가 닫혔다가 열리는 떨림. 큰 파도~큰 파도까지 몇 초 걸리는지 계산하고, 1/해당값 하면 Hz를 알 수 있음.

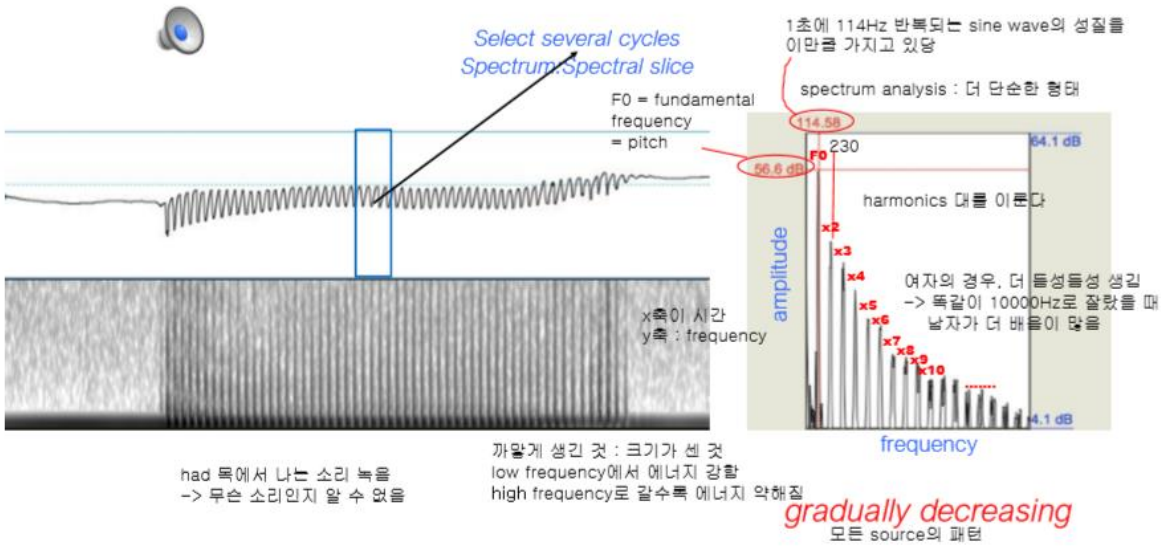
3)Human voice source

-larynx에서 나는 소리. complex tone(pure tone의 총합)임.

-ElectroGlottoGraph(EGG)로 측정됨.

-hamonics로 이루어져 있음.

-가장 낮은 pure tone을 Fundamental frequency(F0)이라고 부름. = 1초당 성대의 개폐가 반복되는 정도



4)Complex tone in spectrum

-모든 signal(sound 포함)은 여러 개의 서로 다른 sine wave의 결합으로 표현됨.

-sine wave: 리드미컬하게 반복되는 기본적인 형태. x축은 시간, y축은 value

-sine wave들을 합하여 새로운 wave를 만들 수 있음. 반복되는 형태는 첫번째 wave(가장 느린 wave)와 같음.

-sine wave를 spectrum 형태로 만들 수 있음.

5)Filter

-vocal tract에 의해 filtered된 소리

-peaks/mountain : vocal tract가 좋아하는 소리 = formants

-valleys : vocal tract가 싫어하는 소리

6)Synthesizing Source

-서로 다른 frequencies와 amplitude로 여러 개의 pure tone을 만든 다음 stereo로 결합하고, 그 다음 mono로 변환시킴.

7)Vowel space

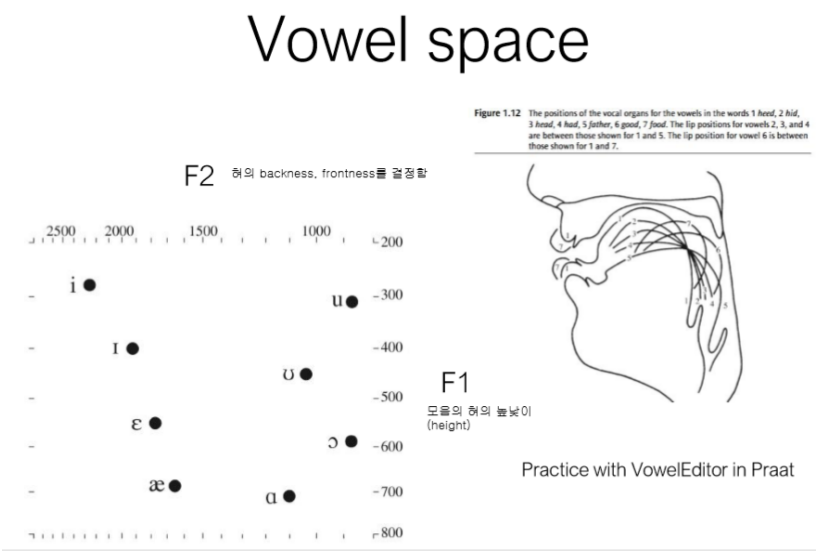


Figure 1.12 The positions of the vocal organs for the vowels in the words 1 *heed*, 2 *hid*, 3 *head*, 4 *had*, 5 *father*, 6 *good*, 7 *food*. The lip positions for vowels 2, 3, and 4 are between those shown for 1 and 5. The lip position for vowel 6 is between those shown for 1 and 7.

언어 = 단어 + 문법

단어: 정보를 담고 있는 그릇

정보를 담는 그릇(단어)이 변수(variable)로서 필요함

프로그래밍의 공통적인 문법

1)변수에 정보를 넣는 것(assign하는 것) variable assignment

2)컨디셔닝에 대한 문법이 필요함. (if문법) if conditioning

3)여러 번 반복 (for문법)

4)함수. 입력을 넣으면 출력이 나오게끔 packaging하는 것. 재사용 가능. 반복적으로 사용

두 개를 숫자를 주면, 처음부터 끝까지 자연수의 합을 구해라.

ex) 7, 10 입력 → 7+8+9+10=34

variable의 종류 : 숫자 or 글자

컴퓨터 프로그래밍에서 =는 equal의 의미가 아님. 오른쪽의 정보를 왼쪽의 variable로 assign한다고 생각하면 됨.

variable은 unique함. 처음에 1을 입력했다가, 다음에 2를 입력하면, 처음 값인 1은 사라짐.

문자를 입력할 때는 반드시 ' ' 안에!

shift + enter : 실행

b : below에 새로운 cell, a : above에 새로운 cell, x : 셀 삭제

```
In [25]: a = 1
         b = 2
         b
         c = 3
         c
Out[25]: 3
```

마지막에 친 variable의 값을 보여줌

```
In [ ]: a = [1, 2, 3, 5]
```

list로 여러 개 넣을 수 있음.

```
In [29]: type(a)
Out[29]: list
```

무슨 타입인지 알 수 있음

int(1) / list ([1, 2, 3]) / float (1.2) / str ('love')

```
In [41]: a = [1,'love',[1,'bye']]
```

Q. type은? list

속해있는 아이템은? int, str, list

```
In [44]: a = (1,'love',[1,'bye'])
In [45]: type(a)
Out[45]: tuple
```

list = tuple은 같은 이름.

list [ ] 사용, tuple은 ( ) 사용. tuple이 보안에 더 강함.

```
In [46]: a = {'a':'apple', 'b':'banana'}
In [47]: type(a)
Out[47]: dict
```

dictionary. { } 사용. **표제어:설명어** 처럼 쌍으로 항상 이루어져야 함.

```
In [6]: a=[1,2]
        b=[3,4]
        c=a[0]+b[0]
        c
Out[6]: 4
```

a와 b 각각의 list를 만들고, list에서 0번째 값을 가져와서 더할 수 있음.

```
In [10]: a=1; a=float(a); print(type(a))
<class 'float'>
```

a=1인 경우 원래 type은 int이지만, a=float(a)로 입력하여 type을 바꿀 수 있음.

```
In [13]: a=1.2; a=int(a); type(a)
Out[13]: int
```

어떤 variable의 내부 정보로 들어갈 때는 대괄호[ ]를 사용함. 대괄호 안에 들어가는 것은 index를 쓴다.

```
In [14]: a='123'; print(type(a)); print(a[1])
<class 'str'>
2
```

```
In [22]: a={"a":"apple", "b":"orange", "c":2014}
print(type(a))
print(a["a"])
<class 'dict'>
apple
```

dict에서는 pair 중 앞부분을 access의 index로 씀.

<-> 보통은 0번째, 1번째 .. 이런식으로 사용

```
In [23]: a={1:"apple", "b":"orange", "c":2014}
print(type(a))
print(a[1])
<class 'dict'>
apple
```

dict에서 표제어를 str이 아니라 int로 해도 가능.

```
In [30]: s = 'abcdef'
n = [100, 200, 300]
print(s[0], s[5], s[-1], s[-6])
a f f a
```

맨 앞부터 0번째, 1번째 ...

반대로 맨 앞에서부터 반대 방향으로 -1번째, -2번째, ...

제일 첫번째 것과 제일 마지막 거는 일일이 세지 않아도 0번째, -1번째로 찾으면 됨

```
In [31]: s = 'abcdef'
n = [100, 200, 300]
print(s[0], s[5], s[-1], s[-6])
print(s[1:3], s[1:], s[:3], s[:])
a f f a
bc bcdef abc abcdef
```

print(s[1:3]) 은 첫번째에서 3번째의 직전 것(=두번째꺼)까지

print(s[1:] 은 첫번째부터 끝까지

print(s[:3]) 은 0번째부터 3번째의 직전 것(=두번째꺼)까지

```
In [32]: print(n[0], n[2], n[-1], n[-3])
print(n[1:2], n[1:], n[:2], n[:])
100 300 300 100
[200] [200, 300] [100, 200] [100, 200, 300]
```

list도 위의 string과 똑 같은 접근방식 사용함.

```
In [33]: len(s)
Out[33]: 6
```

variable의 정보의 길이(length)를 알 수 있는 함수

```
In [36]: s[1]+s[3]+s[4:]*10
Out[36]: 'bdefefefefefefefefefef'
```

값들을 계산하듯이 할 수 있음

```
In [37]: s.upper()
Out[37]: 'ABCDEF'
```

대문자로 바꾸는 함수

variable을 만들고 그 옆에 .을 쓰면 함수가 실행됨

```
In [42]: s=' this is a house built this year.₩n'
         s
Out[42]: ' this is a house built this year.₩n'
```

```
In [43]: result=s.find('house')
         result
Out[43]: 11
```

11번째에서 'house'가 시작됨

```
In [49]: result=s.rindex('this')
         result
Out[49]: 28
```

rindex는 마지막 단어의 위치를 찾아줌

```
In [51]: s = s.strip()
         s
Out[51]: 'this is a house built this year.'
```

불필요한 것들을 제거하고 순수한 텍스트만 남겨주는 함수

```
In [52]: tokens = s.split(' ')
         tokens
Out[52]: ['this', 'is', 'a', 'house', 'built', 'this', 'year.']
```

긴 string을 스페이스를 기준으로 잘라서 단어들로 모아서 list 만들 수 있는 함수

```
In [54]: tokens = s.split(',')
         tokens
Out[54]: [' this is a house', ' built this year.₩n']
```

,을 기준으로 자를 수도 있음.

```
In [60]: s = ' '.join(tokens)
         s
Out[60]: ' this is a house  built this year.₩n'
```

token에 있는 잘라진 단어들을 붙일 수 있음

```
In [65]: s=','.join(tokens)
         s
Out[65]: 'this,is,a,house,,built,this,year.'
```

```
In [64]: s=s.replace('this', 'that')
         s
Out[64]: 'that is a house  built that year.'
```

모든 this를 that으로 바꿔라.

#을 앞에 붙이면 실행이 되지 않고 note로 남게 됨.

또는 Markdown ▼

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

for i in a

print(i)

in 뒤에 있는 것(a)을 하나씩 불러서 i가 그것을 행하고 무언가(print)를 해라.

```
In [5]: a=[1, 2, 3, 4, 5, 6, 7]
        for i in range(len(a)):
            print(a[i])
1
2
3
4
5
6
7
```

range 뒤에 숫자가 나오면 list를 만들어 줌.

ex) range(4)는 4개의 index를 만들어 줌. (0부터 3까지)

In [16]:
```
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%
```

enumerate(a)는 a의 variable에 번호를 매김.

(i,s)=(0,red) (1,green) …

print("{}:{}%".format(s,b[i]*100)) 은

{}:{}%를 출력하는데, (s,b[i]*100)의 형태로 출력함

In [17]:
```
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))
```

```
red:20.0%
green:30.0%
blue:10.0%
purple:40.0%
```

len(a)와 len(b)는 같아야 함.

zip→a와 b가 각각 pair로 연결됨

In [24]:
```
a = 0
if a == 0:
    print("yay!")
    print("let's go")
```

```
yay!
let's go
```

a=0이라면 "yay!"를 출력하라.

a=0이라면 "let's go"를 출력하라.

In [35]:
```
a = 0
if a < 0:
    print("yay!")
    print("let's go")
else:
    print("no")
```

```
no
```

<= 또는 >=처럼 부등호가 먼저 나와야 함.

In [36]:
```
for i in range(1,3):
    for j in range(3,5):
        print(i*j)
```

```
3
4
6
8
```

range(1,3)은 1,2,3이 아니라 1,2

첫번째 루프는 1,2이니까 두 번 돌고
두번째 루프는 3,4이니까 두 번 돎.

→ 총 4번 돌게 됨.

행렬: 행과 열을 직사각형 안에 넣어 놓은 것. image

1장이면 흑백, 3장이면 색상 (RGB) → 사진

시간별로 계속 있는 것 → 동영상

image는 2차원(행과 열), color image는 3차원, 시간까지 있으면 4차원

→(이미지, 소리, 텍스트는 벡터화될 수 있음.) 벡터: 숫자가 한 줄로 쭉 나열되어 있는 것. 모든 data는 벡터의 형태로 되어 있어야 다루기 쉬움.

dictionary. 첫번째 단어는 10000...000(5만개), 두번째 단어는 0100....00000

벡터는 수학적 과정을 거쳐야 함. + - x 등

```
In [4]: A = numpy.array(a)
        B = numpy.array(b)

In [5]: A+B
Out[5]: array([ 3,  7, 11])

In [7]: type(A)
Out[7]: numpy.ndarray
```

array → list를 계산이 가능하게 바꿔줌.

```
In [9]: import numpy as np

In [10]: A = np.array(a)
         B = np.array(b)
```

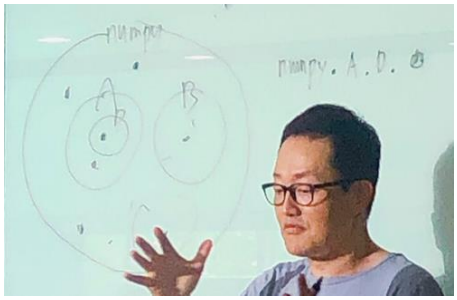
np로 줄여서 쓰기도 함

```
In [11]: X=np.array([[1,2,3],[4,5,6]])

In [12]: X

Out[12]: array([[1, 2, 3],
                [4, 5, 6]])
```

2개의 행과 3개의 열이 생김

```
In [13]: X.shape

Out[13]: (2, 3)
```

차원을 알려줌. 2열 3행의 행렬이다.



import numpy

numpy.A.D.f 이렇게 불러와도 되고

from numpy import A.D (numpy에 있는 A를 불러오자)

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

두번째꺼랑 같은 의미로 쓰이는 거 = from matploylib import pyplot as plt

numpy를 쓰는 이유? 수학적으로 계산을 가능하게 함. 모든 data 처리는 list로 하지 말고 numpy로 처리해야 함.

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

Out[3]: array([[8, 0, 0],
               [0, 0, 0]])
```

입력이 [2,3]로 list로 들어감. 2행 3열.

```
In [4]: np.zeros([2,3])
Out[4]: array([[0., 0., 0.],
               [0., 0., 0.]])
```

np라는 라이브러리 안에 있는 zeros 함수.

```
In [6]: np.array([[0, 0, 0], [0, 0, 0]])
Out[6]: array([[0, 0, 0],
               [0, 0, 0]])
```

list를 array로 convert 해줌. 위의 것과 결과가 똑같음.

```
In [7]: np.ones([2,3])
Out[7]: array([[1., 1., 1.],
               [1., 1., 1.]])
```

.이 있는 것은 int가 아니라 float이라는 의미.

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

.이 사라짐

```
In [10]: np.ones([2,3], dtype='float64')
Out[10]: array([[1., 1., 1.],
                [1., 1., 1.]])
```

정확도는 높아지지만, 데이터 메모리 차지는 많아짐. 반비례함

```
In [12]: np.arange(5)
Out[12]: array([0, 1, 2, 3, 4])
```

```
In [13]: np.arange(0, 10)
Out[13]: array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])
```

계산 가능한 array를 만들어준다. 0부터 5개 → 0,1,2,3,4

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

2만큼 건너뛰면서 만듦.

```
In [11]: np.arange(0,10,2, dtype='float64')
Out[11]: array([0., 2., 4., 6., 8.])
```

float 뒤의 숫자는 precious와 관련이 있음

```
In [15]: np.linspace(0,10,6)
Out[15]: array([ 0.,  2.,  4.,  6.,  8., 10.])
```

0부터 10까지 6개로 똑같이 나눈다.

```
In [16]: np.linspace(0,10,7)
Out[16]: array([ 0.        ,  1.66666667,  3.33333333,  5.        ,  6.66666667,
                8.33333333, 10.        ])
```

0부터 10까지 7개로 똑같이 나눈다.

```
In [17]: X = np.array([[1,2,3],[4,5,6]])
         X
Out[17]: array([[1, 2, 3],
                [4, 5, 6]])
```

2 by 3 만듦

```
In [19]: X = np.array([[1,2],[4,5],[8,9]])
         X
Out[19]: array([[1, 2],
                [4, 5],
                [8, 9]])
```

```
In [20]: X = np.array([[[1,2],[4,5],[8,9]],[[1,2],[4,5],[8,9]]])
         X
Out[20]: array([[[1, 2],
                 [4, 5],
                 [8, 9]],

                [[1, 2],
                 [4, 5],
                 [8, 9]]])
```

대괄호가 두개 나오면 → 2차원 행렬(직사각형), 대괄호가 세개 나오면 → 3차원 (직육면체)

```
In [23]: X.ndim
Out[23]: 3
```

3차원임

```
In [24]: X.shape
Out[24]: (2, 3, 2)
```

2x3x2의 3차원임

```
In [25]: X.dtype
Out[25]: dtype('int32')
```

type 알 수 있음

```
In [26]: X.astype(np.float64)
Out[26]: array([[[1., 2.],
                 [4., 5.],
                 [8., 9.]],

                [[1., 2.],
                 [4., 5.],
                 [8., 9.]]])
```

바꿀 수 있음

```
In [27]: np.zeros_like(X)

Out[27]: array([[[0, 0],
                 [0, 0],
                 [0, 0]],

                [[0, 0],
                 [0, 0],
                 [0, 0]]])
```

```
In [28]: X*0

Out[28]: array([[[0, 0],
                 [0, 0],
                 [0, 0]],

                [[0, 0],
                 [0, 0],
                 [0, 0]]])
```

다 0으로 바꿔줌

```
In [31]: data = np.random.normal(0,1, 100)
         print(data)
```

```
[-0.51048031  0.30354927  0.69515769 -1.09809421  0.18932242  0.65589217
  0.47284639 -1.66429725 -0.45553925  0.52562938 -0.44065576  0.91759086
  0.80663343 -0.34077449  1.45868986  1.55439178  0.6568207  -0.58500626
 -1.4122915  -0.26511916  0.63584113  1.80348869 -2.30447781 -1.06615858
 -1.09751041 -2.3248913   0.60997386  0.5927123  -0.36080345 -1.1213405
 -0.48170778  0.30965389  0.28696136 -1.3260088  -0.62593426  1.14230951
  0.38694749 -1.10817032  1.23217793 -1.54371235 -0.6664032  -0.45395769
  1.69052055  0.34691211  1.62053399 -1.60217715  1.43704219 -0.34642402
 -1.36520726  0.26612245  1.24734809  2.31211207  1.58042259 -0.58236402
 -1.14055756 -0.47944035  0.3222183  -1.0594763   1.1238066   1.28752646
 -2.26826773 -2.26593815 -1.39305724 -2.11124727  0.6974243  -1.96505694
 -1.10355344 -1.57793283 -1.87813503  0.13170728  1.75389073 -0.17110248
 -0.25942723 -0.38751004  0.01083037 -1.80757692 -0.47937368 -0.38300193
 -1.13250278  0.12217807 -0.15444545  0.15666062  0.89813448 -0.93263712
 -0.55231385  0.4114982   0.17981612  0.2104676   1.78040981  0.69366526
  2.0961827   1.03983991 -1.08066019 -0.36485845  0.3100939   1.17665745
 -2.16278897 -0.10670459  0.20451387  0.065758  ]
```

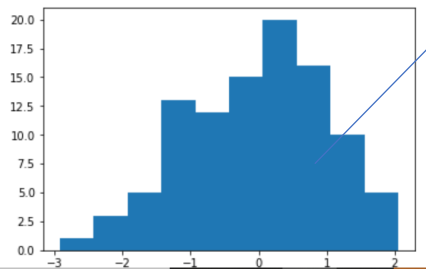
```
In [32]: data.ndim

Out[32]: 1
```

```
In [33]: data.shape

Out[33]: (100,)
```

random.normal → normal distribution의 data를 만들어줌. 0=mean, 1=standard deviation, 100=data의 개수

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

```
[-0.8398779  -0.66062864  1.30115222  0.50450206  0.95667305  0.60700552
 -1.01855037  0.95351948  0.08821886  0.99512932  0.41601486  0.60358232
 -1.38789209  0.29844135  0.57459243  0.22407801  0.88837517  1.17633349
  0.56068165  2.05564786  1.08076566  0.31183736 -0.64344222  0.9709739
 -0.47544633 -1.70769647  1.00774162 -1.34928956 -1.6282407  -0.19706846
 -0.02835946 -1.5466965   0.26731914 -0.00338448 -0.54668531 -0.64565518
 -2.41783182 -1.09289534 -1.13593424 -1.35908579 -0.04977131  0.07063047
  0.13563216 -0.57646697 -0.01080657 -0.16509508 -1.4860257   0.32366803
  0.17495132  0.54037639 -0.45249398 -0.94513949  0.09107664 -0.09690239
  0.04488749 -1.06009831 -0.3998957   1.33848238  0.66665498 -0.98073074
  1.11223601 -0.1388641   0.38457044  0.59639466 -0.11875297  0.10574101
  1.61805566  1.25025494 -0.87618395  0.83266569  1.59814316 -1.71705042
 -0.54227006 -1.11169427 -0.55534861  0.83079971 -1.13066581  0.39353686
  1.48271714 -2.92470339  1.19285915 -0.03397218  1.7172698   1.67298858
 -2.23940405 -1.17391172 -0.97346826 -0.70007253  0.8866518   0.51946561
 -0.02202387 -0.37008492  0.66716361  0.87427593  0.18242455 -2.01162127
  1.07647129  0.00683464  1.10721925  0.10419802]
```



bins=10 → 10개의 바구니를 만들어서 값들을 각각 넣어줌

y값에 해당하는 것은 소수점이 나올 수 없음. 무조건 정수값.

모두 몇 개? 100개

```
In [11]: X = np.ones([2, 3, 4])
         X
```

```
Out[11]: 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.]]])
```

총 2x3x4=24개 들어있음.

1x6x4(=24) 이런 식으로만 바꿀 수 있음.

```
In [22]: Y = X.reshape(-1, 3, 2)
         Y
```

```
Out[22]: 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.]]])
```

-1은 알아서 하라는 뜻

-1 대신 4 써도 됨

```
In [23]: np.allclose(X.reshape(-1, 3, 2), Y)
```

```
Out[23]: True
```

x랑 y가 close하냐? true

```
In [40]: a = np.random.randint(0, 10, [2, 3])
         b = np.random.random([2, 3])
         np.savez("test", a, b)
```

0부터 10까지 숫자를 pick해서 2x3으로 만들어라

그냥 random하게 만들어라.

savez → a,b라는 variable을 실제 file로 저장해줌.

```
In [41]: !ls -al test*
-rw-r--r--  1 jookai  staff     562 Apr  2 00:35 test.npz
-rw-r--r--  1 jookai  staff  123438 Mar 14 23:19 test.wav
```

실제 만들어졌는지 확인

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

del 하면 뒤의 a,b라는 variable이 사라짐.

%who는 몰라도 됨

```
In [41]: %who
X       Y       a       b       data    np      plt
```

지금까지 assign한 variable이 뭐뭐 있는지 다 알려줌.

```
In [46]: npzfiles = np.load("test.npz")
         npzfiles.files
Out[46]: ['arr_0', 'arr_1']

In [48]: npzfiles['arr_0']
Out[48]: array([[1, 5, 2],
                [1, 7, 0]])
```

저장한 것을 불러오고 [ ]안에 이름을 넣으면 불러져옴.

```
In [49]: data = np.loadtxt("regression.csv", delimiter=",", skiprows=1, dtype={'names':("X", "Y"), 'formats':('f',
         'f')})
         data

Out[49]: array([( 3.3  , 1.7  ), ( 4.4  , 2.76 ), ( 5.5  , 2.09 ), ( 6.71 , 3.19 ),
                ( 6.93 , 1.694), ( 4.168, 1.573), ( 9.779, 3.366), ( 6.182, 2.596),
                ( 7.59 , 2.53 ), ( 2.167, 1.221), ( 7.042, 2.827), (10.791, 3.465),
                ( 5.313, 1.65 ), ( 7.997, 2.904), ( 5.654, 2.42 ), ( 9.27 , 2.94 ),
                ( 3.1  , 1.3  )], dtype=[('X', '<f4'), ('Y', '<f4')])

In [50]: np.savetxt("regression_saved.csv", data, delimiter=",")
         !ls -al regression_saved.csv

         -rw-r--r--@ 1 jookai  staff  850 Apr  2 00:38 regression_saved.csv
```

찾아서 직접 해보고 올리기

```
In [51]: arr = np.random.random([5,2,3])
```

random으로 5x2x3으로 만들어줘라.

```
In [54]: 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
```

len=5 맨 처음것

shape은 우리가 정한 거 그대로

ndim은 3차원

size = 5x2x3 = 30

dtype은 float64

```
In [55]: a = np.arange(1, 5)
         b = np.arange(9, 5, -1)

In [56]: print(a - b)
         print(a * b)

         [-8 -6 -4 -2]
         [ 9 16 21 24]
```

5.2 Comparison

```
In [64]: 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 [65]: a == b

Out[65]: array([[False, False, False],
                [False,  True, False],
                [False, False, False]])

In [66]: a > b

Out[66]: array([[False, False, False],
                [False, False,  True],
                [ True,  True,  True]])
```

두개의 dimension과 size과 완전히 똑같아야 함

5.3 Aggregate (sum, min, max, mean, median, std)

```
In [67]: a.sum(), np.sum(a)

Out[67]: (45, 45)

In [68]: a.sum(axis=0), np.sum(a, axis=0)

Out[68]: (array([12, 15, 18]), array([12, 15, 18]))

In [69]: a.sum(axis=1), np.sum(a, axis=1)

Out[69]: (array([ 6, 15, 24]), array([ 6, 15, 24]))
```

sum은 다 더하는 것

axis → 몇번째 차원에서 sum을 할 것인가

axis=0 → 첫번째 차원에서 더함

1초를 표현하는 숫자의 개수가 몇 개가 있는가?

44100 → 일반인들이 구별할 수 있는 최대 수치