

Praat

Amount of hz = the amount that 1 second is divided into

Intensity = 폭 (how strong the voice is)

Pitch (high and lowness of sound)

Praat shows sound's spectrum

빠르게 움직이면 고주파

느리게 움직이면 저주파 = pitch

Spectral analysis

Voiced – 유성음

Voiceless – 무성음

Phonetics – the study on speech

- Articulatory phonetics (from mouth) <- the most primitive

- How to produce speech

- ◆ Tone of sound = vibration/ open close of vocal folds
- ◆ Difference of sound (아,에,이,오,우) = shape of mouth (tongue etc.)
- ◆ *Velum is raised = nasal tract is closed = 모든 모음/ 비음을 뺀 자음
- ◆ *코로 숨을 쉴 때 nasal tract open and velum is lowered.

- Acoustic phonetics (*through air*)

- How to transmit speech

- ◆ Constriction location
 - Lips – 2 location
 - Tongue body – 2
 - Tongue lip – 4
- ◆ Constriction degree

- Auditory phonetics (to ear)

■ How to hear speech

◆ Praat

● Intensity

● Pitch

■ Hz = pitch = the amount of vibration per second

Phonology – study of sound system (different from phonetics)

Speech = sound of human

모든 영어의 소리는 유성음과 무성음으로 나뉜다.

Velum raised, glottis (larynx의 틈) open, tongue tip, alveolar stop =

모든 모음은 constrictor 로써 tongue body 만 쓴다

모음과 같은 tongue body 를 쓰는 자음 = k (velum lowered = 응 (nasal sound))

Computer language 다양함

모든 language들이 조금씩 다름 but 공통점 -> 단어, 어떻게 combine하는지 (문법)

단어는? -> 의미를 포함하고 있음 (정보를 담는 그릇)

단어 = 변수(variable)

변수에다가 어떤 정보 assign (variable assignment)

Conditioning에 대한 문법 필요 (if conditioning if...)

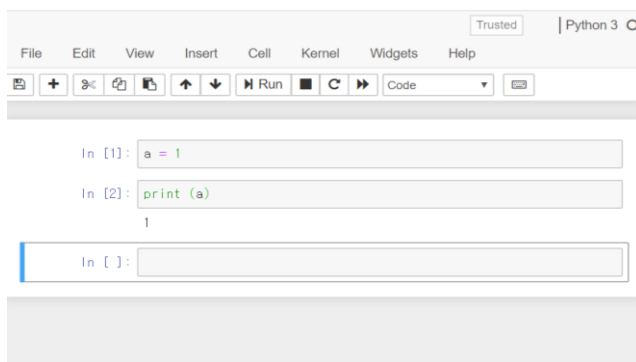
여러 번 반복(자동화) - for loop

함수 - 입력 넣으면 출력될 수 있게, 입력 - 출력의 관계 (packaging)

e.g. 두개의 자연수 -> 합 구함 = 함수

정보의 종류 - 2가지 -> 숫자, 글자

= sign은 오른쪽에 있는 정보를 왼쪽에 있는 sign으로 assign

A screenshot of a Jupyter Notebook interface. The top bar shows 'Trusted' and 'Python 3'. Below the menu bar (File, Edit, View, Insert, Cell, Kernel, Widgets, Help) is a toolbar with icons for undo, redo, run, and other actions. The main area contains three input cells. The first cell has the code 'a = 1'. The second cell has the code 'print (a)' and shows the output '1'. The third cell is empty and has a blue cursor at the start.

```
In [1]: a = 1

In [2]: print (a)
1

In [ ]:
```

Python 에서 사용하는 함수 = 누군가 만들어 놓은 유용한 함수

함수 이름 치고 () 안에 variable 넘

Print라는 함수의 입력 = a 출력 = 1

한 칸 = cell

파란색으로 바꿨을 때 'b' 누르면 below cell cell 밑에 cell 추가

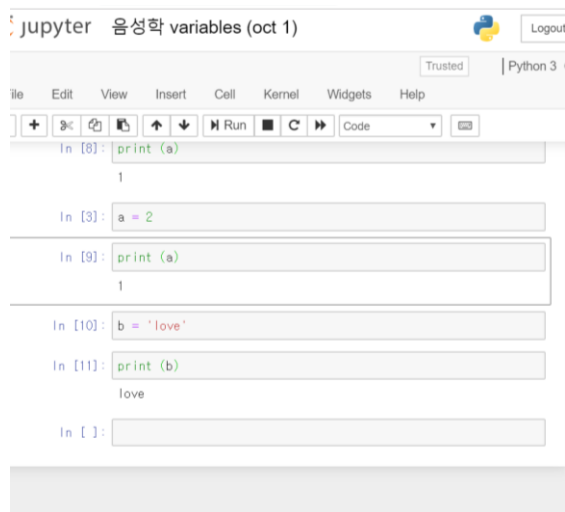
A -> above cell cell 위에 추가

X -> delete cell

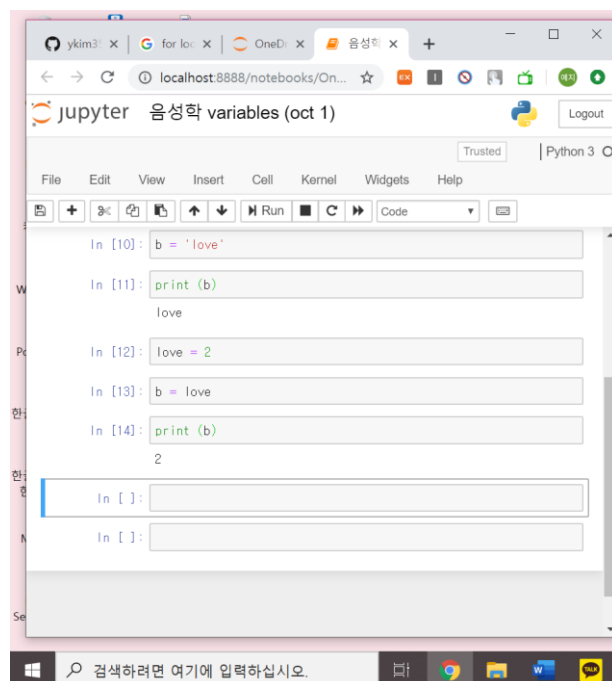
Play 단축키 -> shift enter

같은 변수 variable 에 다른 것을 노면 그냥 reset

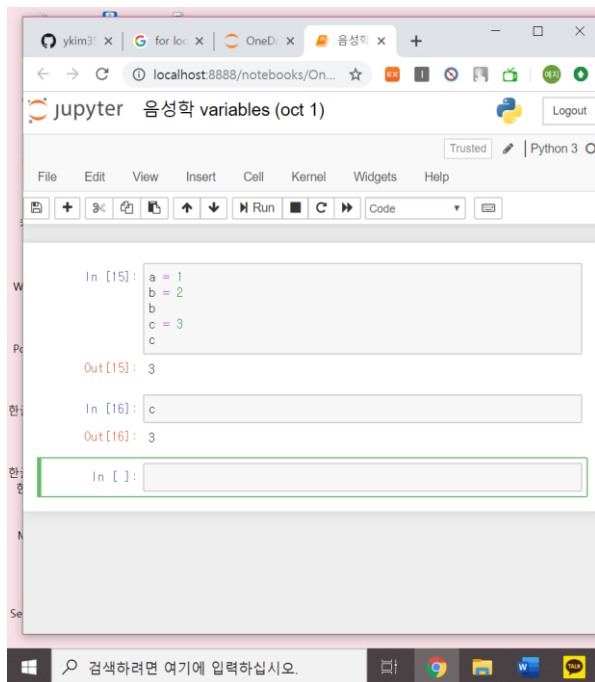
So even if the first line says a=1 if the next one is a=2 and it is "played" last then a = 2.
But if you click the cell that says a=1 at the top and click play then that is the newest and will be printed as 1



문자 입력 출력



제일 마지막에 변수명 하나만 치면 자동 'print' 됨 (아래)

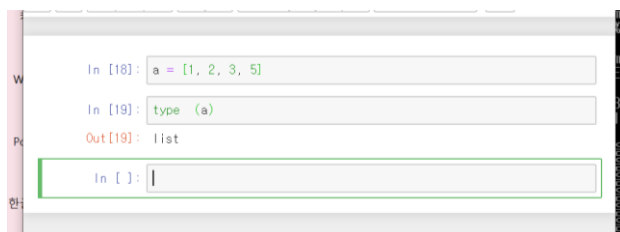


한줄에 길게 쓰고 싶으면 no enter but semicolon



한꺼번에 넣는거 = list (use [])

List 인지 그냥 숫자인지 어떻게 아나? -> type ()



```
In [20]: a = 1
In [21]: type (a)
Out [21]: int
In [ ]:
```

```
In [22]: a = 1.2
In [23]: type (a)
Out [23]: float
In [ ]:
```

```
In [24]: a = 'love'
In [25]: type (a)
Out [25]: str
In [ ]:
```

문자 = str

```
In [26]: a = [1, 2, 3, 5, 'love']
In [27]: type (a)
Out [27]: list
In [ ]:
```

```
In [26]: a = [1, 'love', [1, 'bye']]
In [27]: type (a)
Out [27]: list
In [ ]:
```

List 안에 세가지 항목 – int, str, list (list 안에 int, str)

```
In [28]: a = (1, 'love', (1, 'bye'))

In [29]: type(a)
Out[29]: tuple

In [ ]:
```

List and tuple is the same only difference is () and [] but () 이 더 보안에 강함

```
In [30]: a = {'a':'apple','b':'banana'}

In [31]: type(a)
Out[31]: dict

In [ ]:
```

{ } = dictionary ' ' 는 몇 개 표제어와 단어의 관계는 :

10/8/19

The screenshot shows a Jupyter Notebook window titled "음성학 variables (oct 1)". The top toolbar includes buttons for File, Edit, View, Insert, Cell, Kernel, Widgets, and Help. Below the toolbar, a code cell contains the following Python code:

```
a=(1,2)
b=(3,4)
c=a[0]+b[0]
c
```

The output of the code cell is 4. Above the code cell, a traceback for a `NameError` is visible, indicating that the variable `ab` is not defined. The error message is:

```
Traceback (most recent call last)
<ipython-input-7-605579547f61> in <module>
      1 a=1
      2 b=1
----> 3 c=ab
      4 c

NameError: name 'ab' is not defined
```

Why 0?

A에서 0번째 b에서 0번째 computer number starts from 0 so 0 here is 1 and 3

The screenshot shows a Jupyter Notebook interface with the title "음성학 variables (oct 1)". The top toolbar includes File, Edit, View, Insert, Cell, Kernel, Widgets, and Help. The code area contains a cell with the following code:

```
1 a=1
2 b=1
3 c=ab
4 c
```

The output shows a `NameError` message: `NameError: name 'ab' is not defined`. Below this, a new cell contains the following code:

```
In [8]: a=(1,2)
        b=(3,4)
        c=a[1]+b[1]
        c
```

The output for this cell is `Out[8]: 6`. The bottom of the notebook shows a search bar and a Windows taskbar.

1 = 두번째 숫자

대괄호 쓰고 그것의 index적으면??????

The screenshot shows a Jupyter Notebook interface with the title "음성학 variables (oct 1)". The top toolbar includes File, Edit, View, Insert, Cell, Kernel, Widgets, and Help. The code area contains a cell with the following code:

```
Out[10]: int

In [12]: a = 123; print(type(a)); print(a[1])
```

The output shows `<class 'int'>`. Below this, a new cell contains the following code:

```
In [ ]: 
```

The output shows a `TypeError` message: `TypeError: 'int' object is not subscriptable`. The bottom of the notebook shows a search bar and a Windows taskbar.

Why error? Because there is only 1 number in a so the print should be 0 not 1

A screenshot of a Jupyter Notebook interface. The browser address bar shows 'localhost:8888/notebooks/On...'. The notebook title is '음성학 variables (oct 1)'. The interface includes a menu bar (File, Edit, View, Insert, Cell, Kernel, Widgets, Help) and a toolbar with icons for file operations, running, and code execution. The code area contains two input cells. The first cell (In [13]) contains the following code:

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

The output for this cell is:

```
<class 'dict'>
apple
```

The second cell (In [16]) contains the following code:

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

The output for this cell is:

```
<class 'list'>
Out[16]: (1, 2, 3)
```

Below the code area is an empty input field labeled 'In []:'. The bottom of the notebook shows a Windows taskbar with a search bar and several application icons.

A의 0번째는 123 but the type is tuple so when you write what type it is it says tuple

A screenshot of a Jupyter Notebook interface, similar to the one above. The browser address bar shows 'localhost:8888/notebooks/On...'. The notebook title is '음성학 variables (oct 1)'. The interface includes a menu bar (File, Edit, View, Insert, Cell, Kernel, Widgets, Help) and a toolbar. The code area contains two input cells. The first cell (In [13]) contains the following code:

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

The output for this cell is:

```
<class 'dict'>
apple
```

The second cell (In [17]) contains the following code:

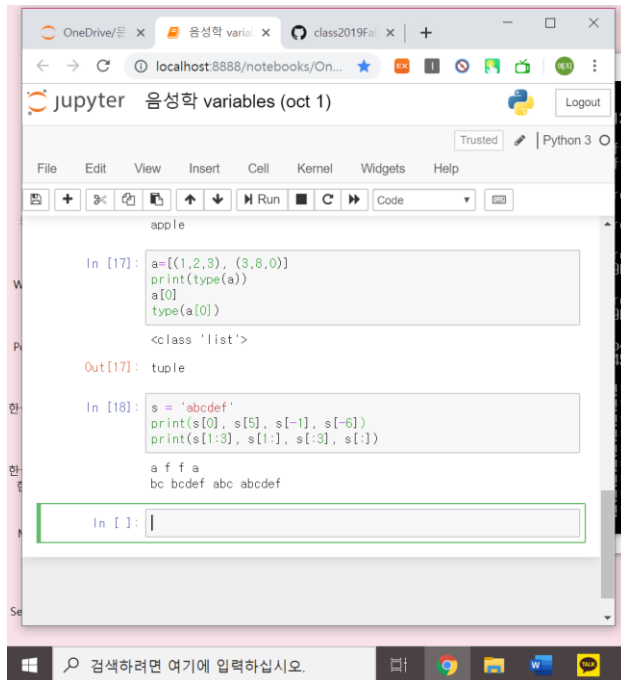
```
a = [(1,2,3), (3,8,0)]
print(type(a))
a[0]
type(a[0])
```

The output for this cell is:

```
<class 'list'>
Out[17]: tuple
```

Below the code area is an empty input field labeled 'In []:'. The bottom of the notebook shows a Windows taskbar with a search bar and several application icons.

String



-1 is f because from a which is 0 it goes back again, so f is -1 e is -2 etc.

첫번째 는 0 마지막은 -1

Range 을 찾을 때 는 : so 1:3 는 1에서 3번째 전 까지

1: 은 1에서 끝까지

:3 은 첫번째에서 세번째 전까지

: 은 전부 다

The screenshot shows a Jupyter Notebook interface with the following code and output:

```
In [6]: s = 'abcdef'
print(s[0], s[5], s[-1], s[-6])
print(s[1:3], s[1:], s[:3], s[:])

<class 'str'>
a f f a
bc bcdef abc abcdef

In [3]: n = [100, 200, 300]
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]

In [4]: len(s)

Out[4]: 6

In [5]: s[1]+s[3]+s[4:] +10

Out[5]: 'bdefefefefefefefefef'

In [7]: s.upper()

Out[7]: 'ABCDEF'
```

String 과 list는 접근 방법이 같다

Len는 variable의 정보의 길이 len=length

The screenshot shows a Jupyter Notebook with the following code and output:

```
In [7]: s.upper()

Out[7]: 'ABCDEF'
```

Variable 에 . 찍고 upper 쓰면 대문자로 바뀜

NLP: natural language processing 자연 언어 처리

Variable 만들고 옆에 . 함수 쓰면

A Jupyter Notebook interface titled '음성학 variables (oct 1)'. The browser tabs show 'OneDrive/문', '음성학 vari...', and 'class2019Fa...'. The address bar is 'localhost:8888/notebooks/On...'. The notebook has a 'Trusted' status and 'Python 3' kernel. The code cells show:

```

In [21]: b="DFGE"
         b.lower()
Out[21]: 'dfge'

In [22]: s = ' this is a house built this year.\n'
         s
Out[22]: ' this is a house built this year.\n'

In [23]: result = s.find('house')
         result
Out[23]: 11

```

The input prompt 'In []:' is visible at the bottom of the code area.

Why 11? The word starts from the 11th including spaces

A Jupyter Notebook interface titled '음성학 variables (oct 1)'. The browser tabs show 'OneDrive/문', '음성학 vari...', and 'class2019Fa...'. The address bar is 'localhost:8888/notebooks/On...'. The notebook has a 'Notebook saved' status and 'Python 3' kernel. The code cells show:

```

In [22]: s = ' this is a house built this year.\n'
         s
Out[22]: ' this is a house built this year.\n'

In [23]: result = s.find('house')
         result
Out[23]: 11

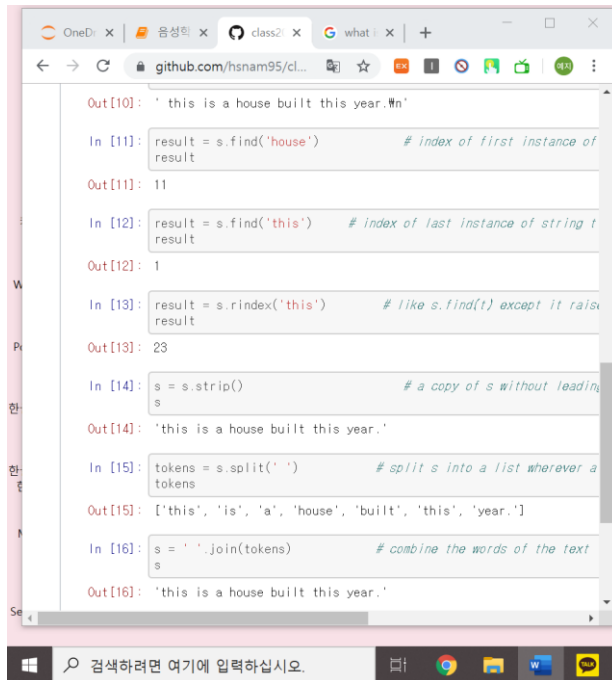
In [26]: result = s.find("this")
         result
Out[26]: 1

```

The input prompt 'In []:' is visible at the bottom of the code area.

There are 2 this but it finds the first one

Index : finding starting from the last



The screenshot shows a Jupyter Notebook with the following code and output:

```
Out[10]: ' this is a house built this year.\n'

In [11]: result = s.find('house')           # index of first instance of
result
Out[11]: 11

In [12]: result = s.find('this')           # index of last instance of string t
result
Out[12]: 1

In [13]: result = s.rindex('this')          # like s.find(t) except it raises
result
Out[13]: 23

In [14]: s = s.strip()                     # a copy of s without leading
s
Out[14]: 'this is a house built this year.'

In [15]: tokens = s.split(' ')             # split s into a list wherever a
tokens
Out[15]: ['this', 'is', 'a', 'house', 'built', 'this', 'year.']

In [16]: s = ' '.join(tokens)              # combine the words of the text
s
Out[16]: 'this is a house built this year.'
```

Rindex is 23 because it counts the last 'this'

Strip 는 나머지 다 없애고 순수한 text 만 남김

Split 는 자르는 것

Token is just a name

10/10

How to add comments to jupyter notebook

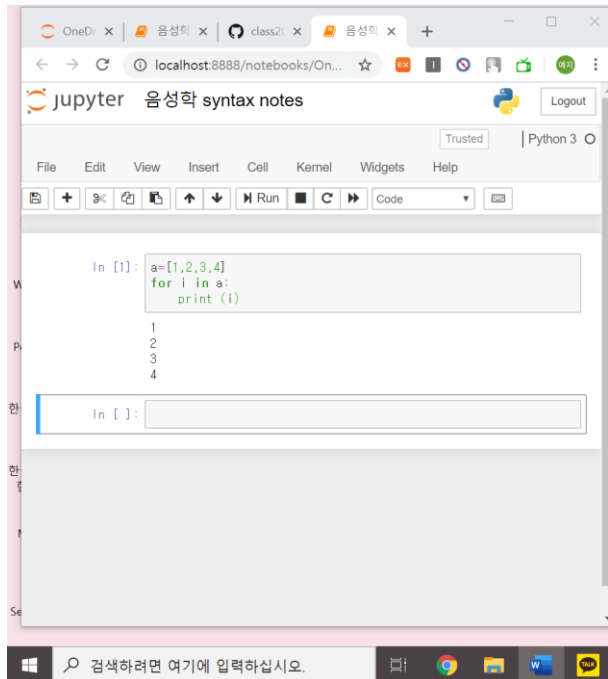
을 앞에 붙이면 comment 처럼 남길수 있음 (실행 안됨)

Code -> markdown으로 바꾸면 실행 안됨

'for' and 'if' used in all computer languages

문법 -> For ____ in ____ ;

In 뒤에 있는걸 하나씩 돌려서 I 가 받아서 무언가를 하라



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

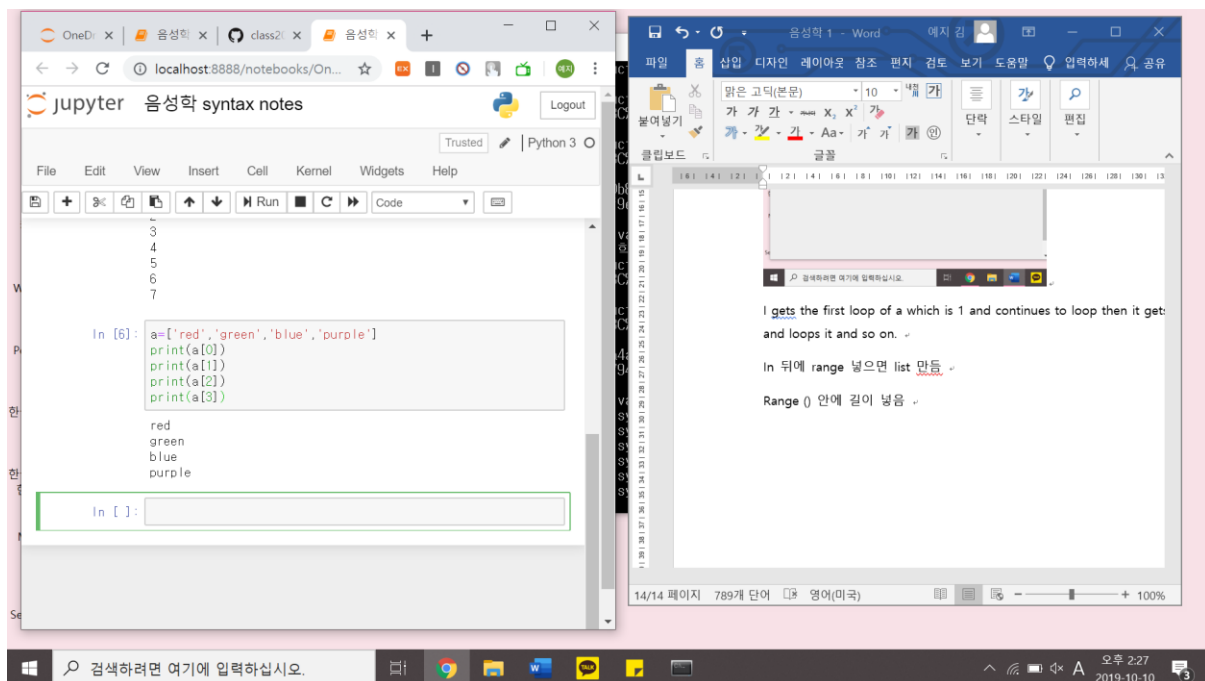
1
2
3
4

In [ ]:
```

I gets the first loop of a which is 1 and continues to loop then it gets the second loop which is 2 and loops it and so on.

In 뒤에 range 넣으면 list 만들

Range () 안에 길이 넣음



```
In [6]: a=['red','green','blue','purple']
        print(a[0])
        print(a[1])
        print(a[2])
        print(a[3])

red
green
blue
purple

In [ ]:
```

읽은 고딕(본문) 10
가 가 간 - X₂ X² 가
글꼴

I gets the first loop of a which is 1 and continues to loop then it get and loops it and so on. ~

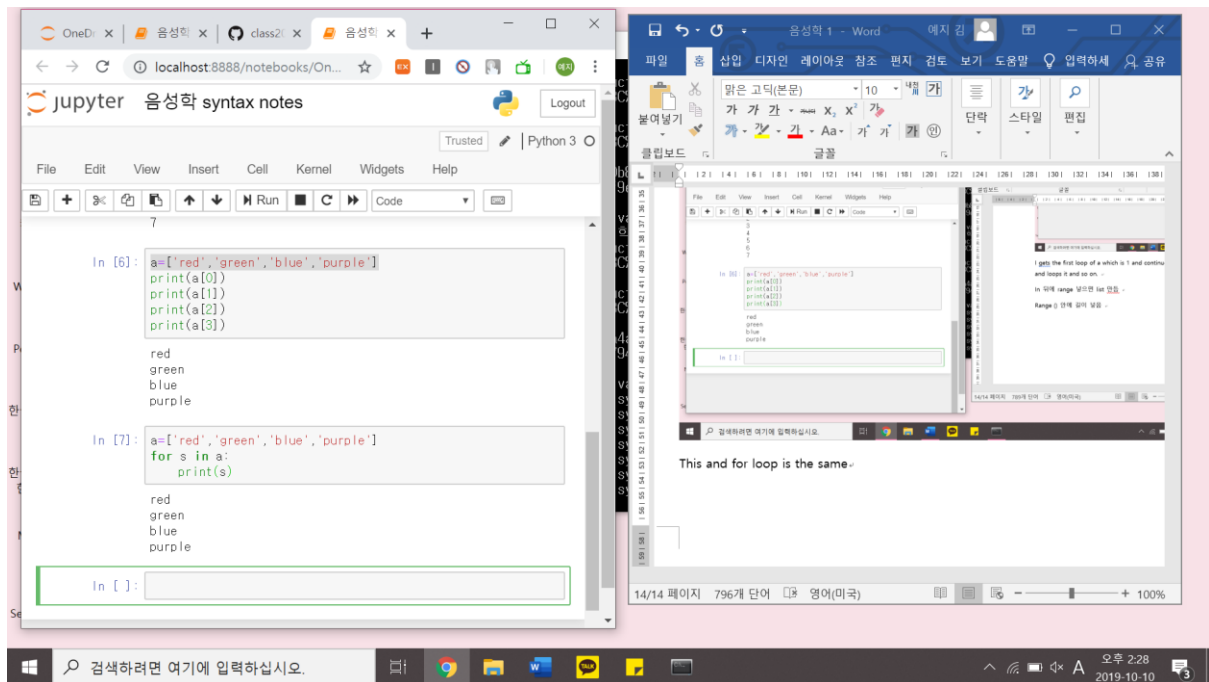
In 뒤에 range 넣으면 list 만들 ~

Range () 안에 길이 넣음 ~

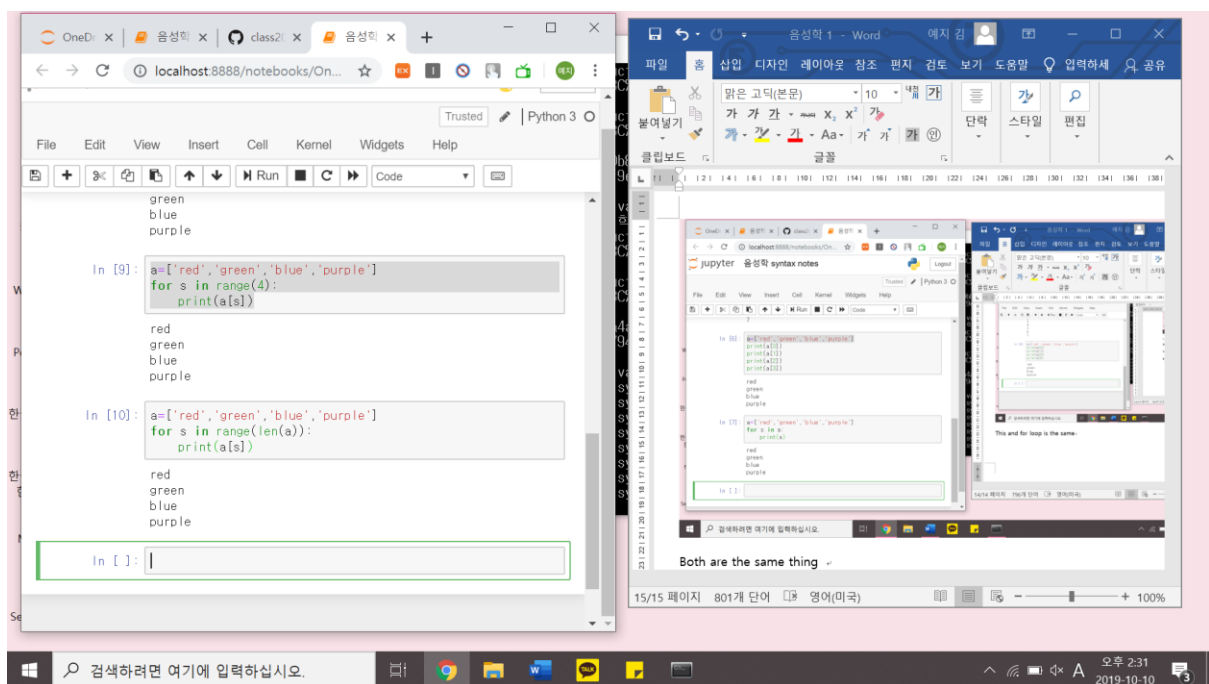
14/14 페이지 789개 단어 영어(미국) 100%

오후 2:27 2019-10-10

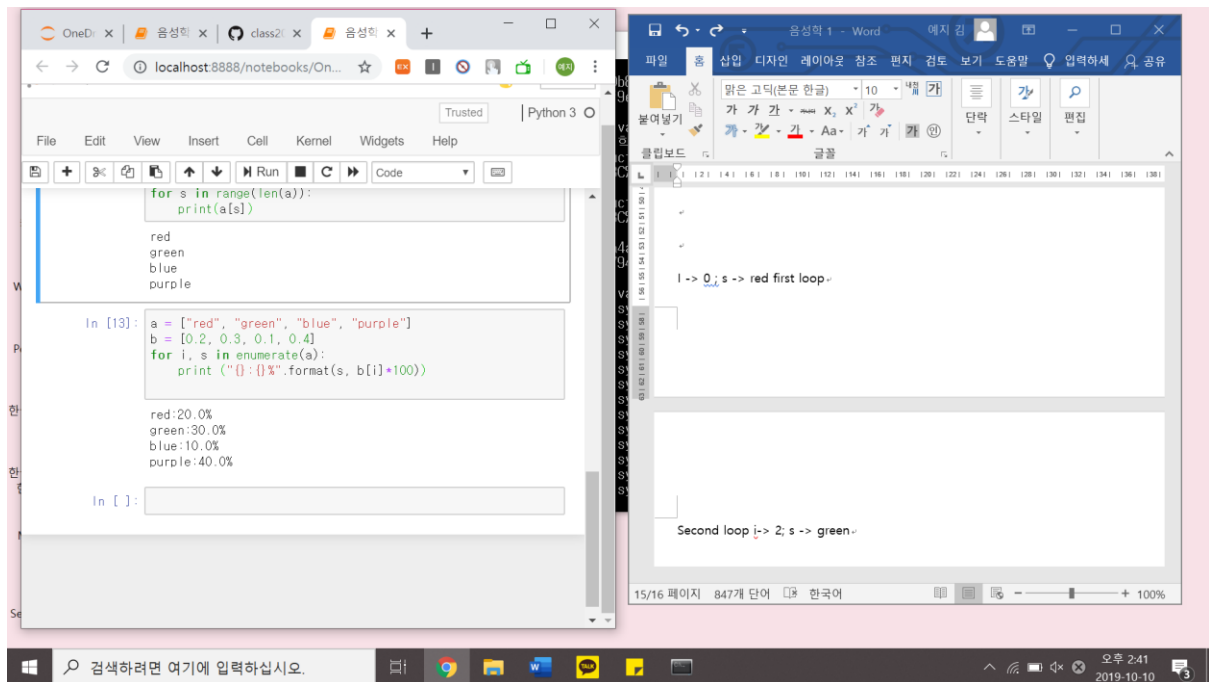
This and for loop is the same



Both are the same thing



Both are the same except that len is more productive in that you don't have to count how many there are but 'a' counts for how many number there are.

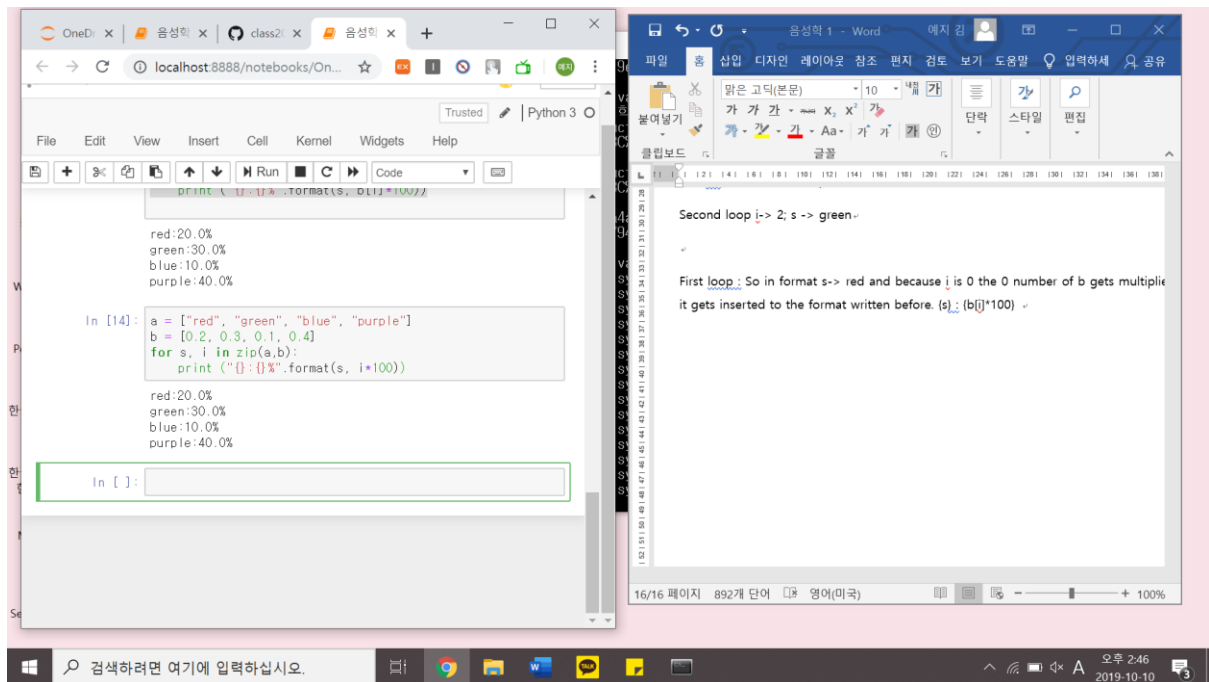


*a and b has to be the same length of list

I -> 0 ; s -> red first loop

Second loop i-> 2; s -> green

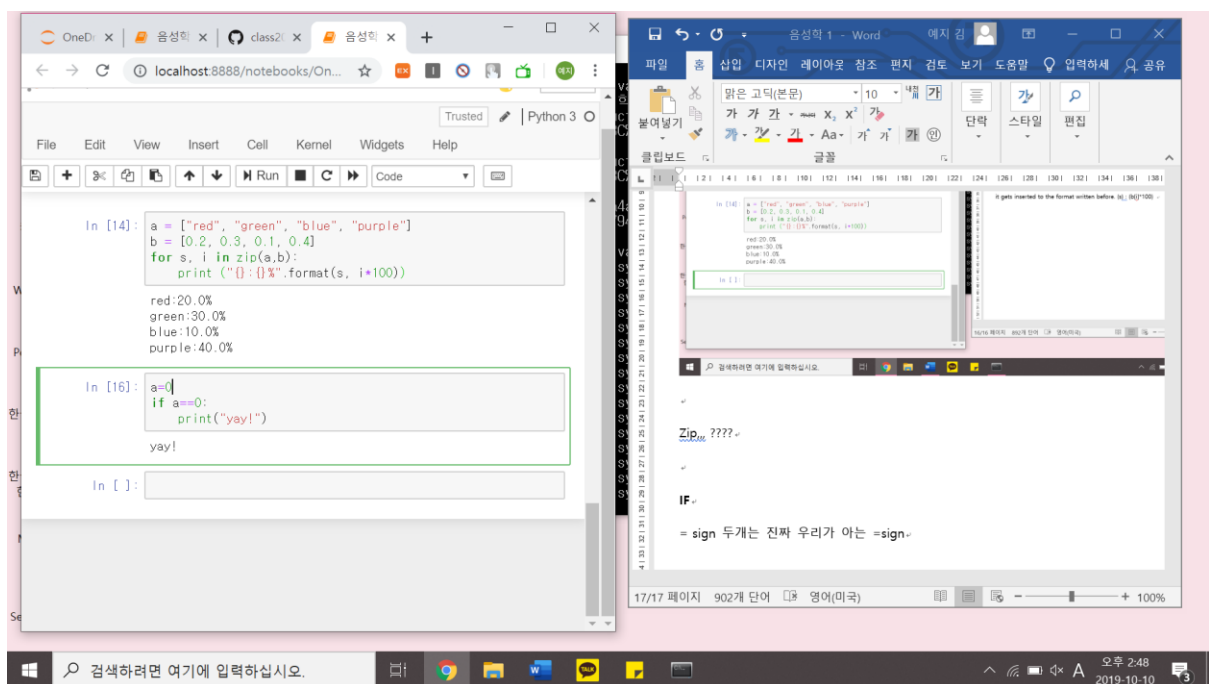
First loop : So in format s-> red and because i is 0 the 0 number of b gets multiplied by 100 and it gets inserted to the format written before. {s} : {b[i]*100}



Zip,,, ????

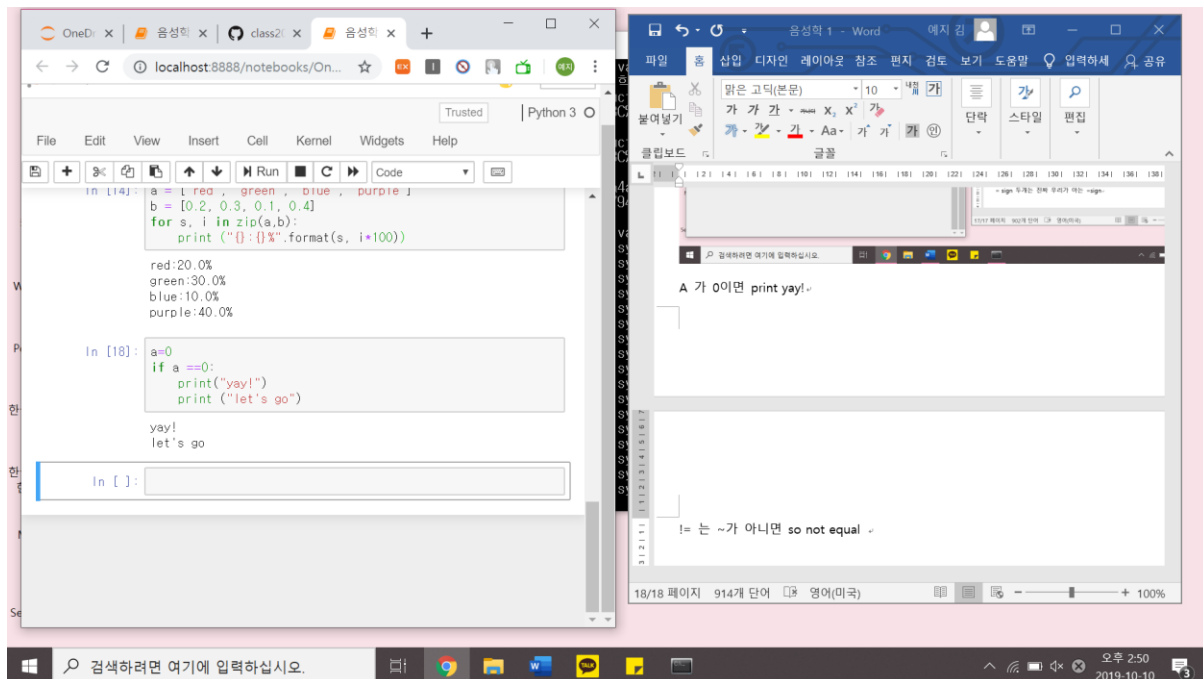
IF

= sign 두개는 진짜 우리가 아는 =sign

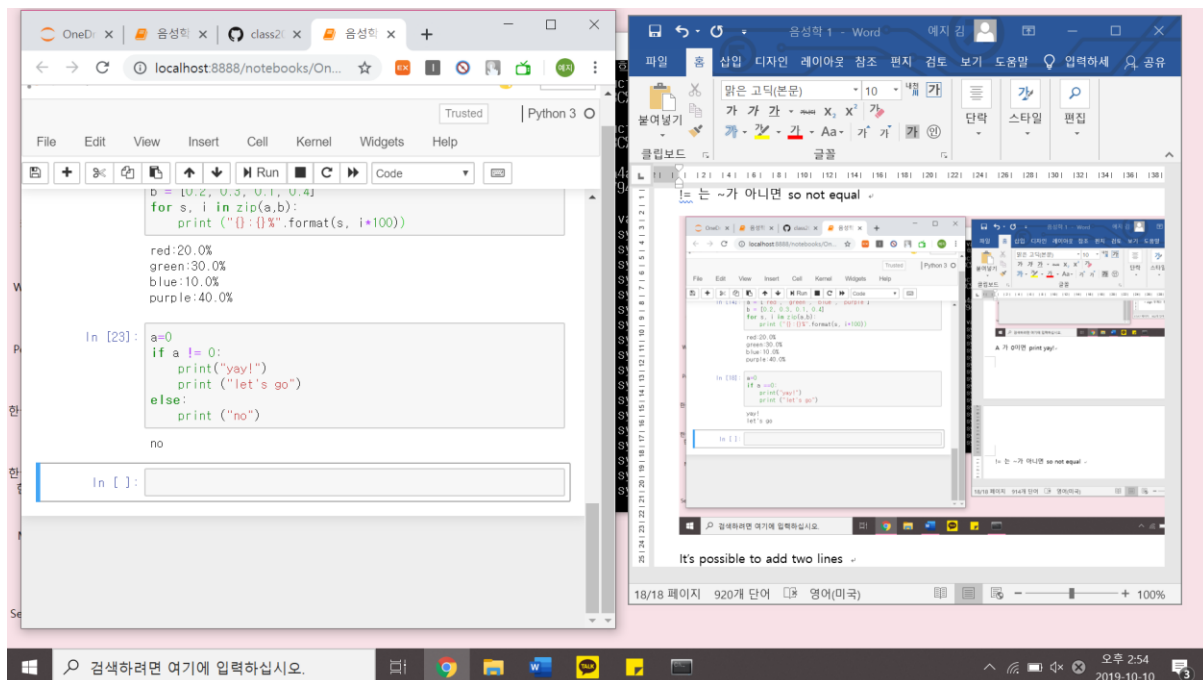


A 가 0이면 print yay!

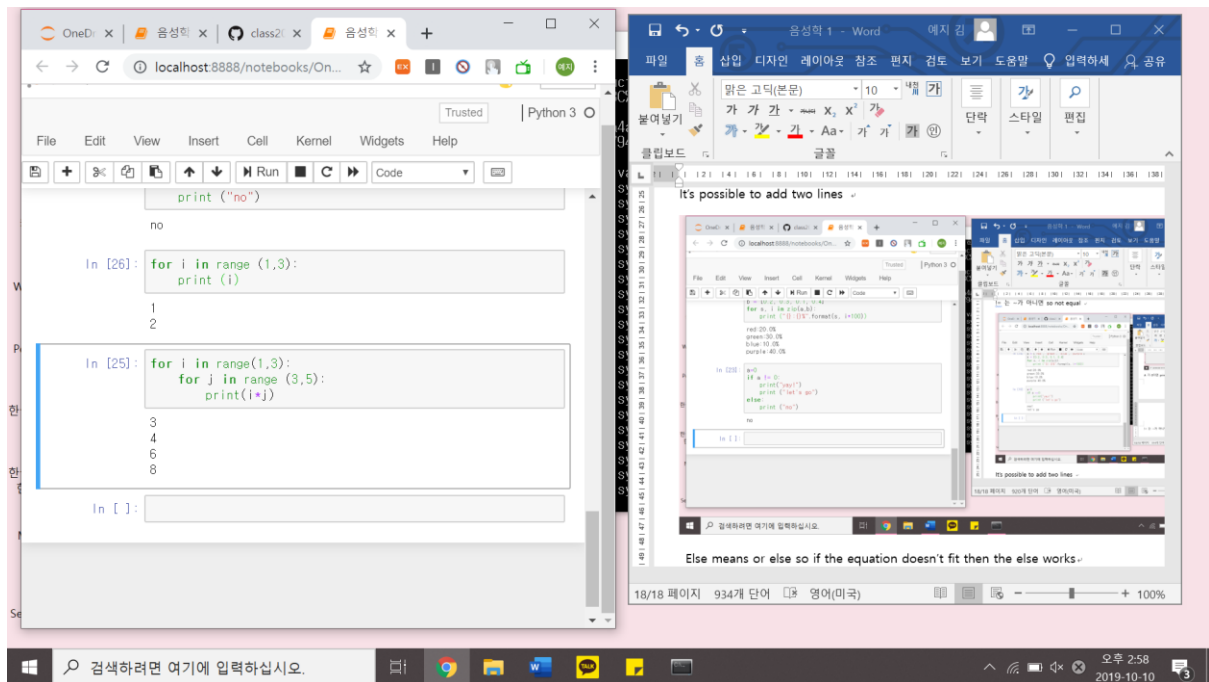
!= 는 ~가 아니면 so not equal



It's possible to add two lines

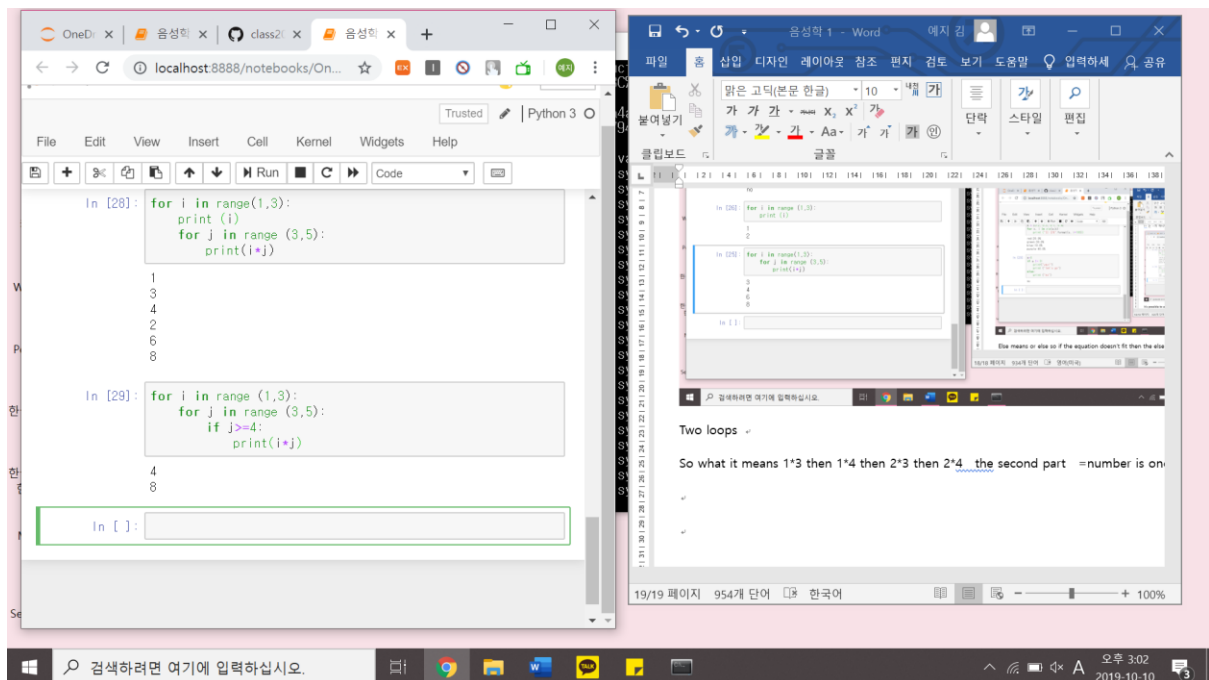


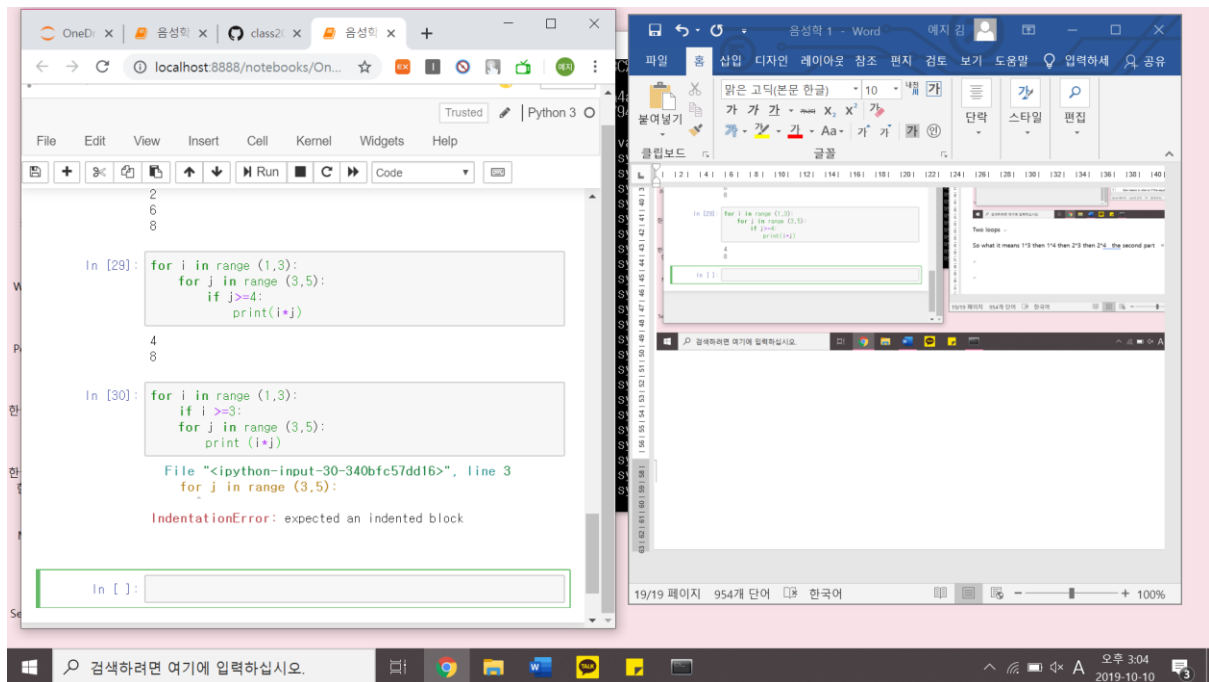
Else means or else so if the equation doesn't fit then the else works



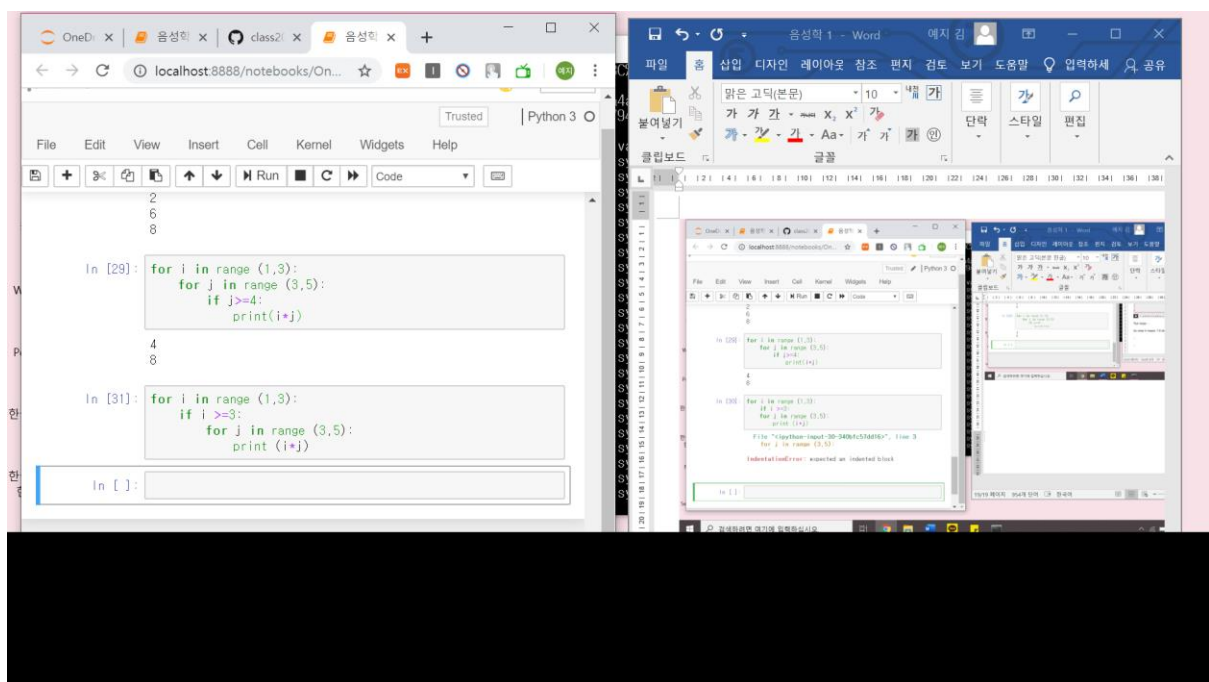
Two loops

So what it means 1*3 then 1*4 then 2*3 then 2*4 the second part =number is one below





This is error because it is not indented but if it is indented



This is intended so there is no error but just no answer.

List and string are similar , how? -> 정보에 어떻게 access 하는가 string 속에 있는 data를 access 하는 방법과 list 속에있는 data를 access하는 방법이 동일

String을 만들고 나서 split이라고 하면 string 을 split 해서 list으로 만들어 줌.

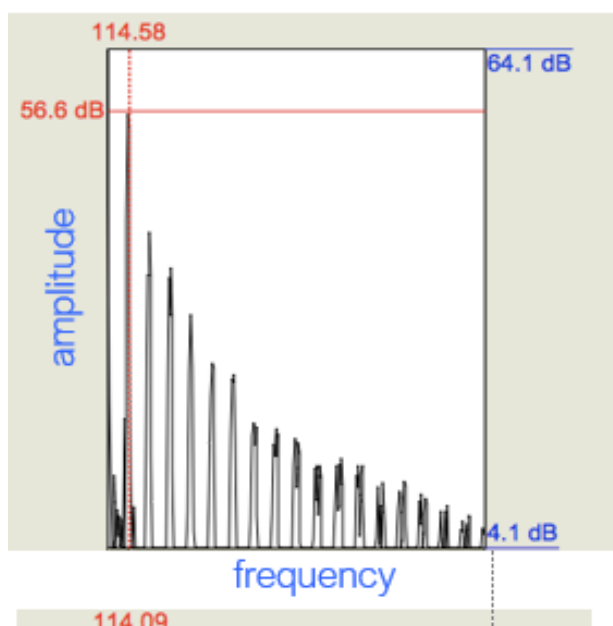
Syntax for if

함수 잘 알아야 함

모음 CL 정의 안함

Amplitude pressure 로 결정됨

여자가 f0가 (frequency) 더 높음 -> 음성음성있는게 여자꺼



주어진 frequency range 속에 pure tone이 여자가 더 적게 가지고 있다 TF?

남자가 pure tone 이 더 많음

그 숫자 filtered되도 변하지 않음 (그 숫자?????)

10/29/19

모든 데이터는 vector의 형태

이미지는 행열이다

numpy

리스트 안에 숫자가 들어갈 때

10/31

. = 뭔 안에 뭐

Import numpy

Numpy.A.B.D

Or

From numpy import A.D

In [4]:

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

Out[4]:

```
[1, 3, 5, 2, 4, 6]
```

리스트 안에서는 수학적 계산 x

In [6]:

```
import numpy
```

In [7]:

```
A = numpy.array (a)
B= numpy.array (b)
```

array makes calculation possible but numpy 먼저 import 해야함

In [8]:

```
A+B
```

Out[8]:

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

In [9]:

```
type (A)
```

Out[9]:

```
numpy.ndarray
```

In [10]:

```
import numpy as np
```

In [11]:

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

Out[11]:

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

In [12]:

```
X.shape
```

Out[12]:

(2, 3)

shape = 2 by 3 matrix

In []:

In []:

In [1]:

```
import numpy as np
import matplotlib.pyplot as plt

from matplotlib import pyplot as plt is the same this as above
```

In [1]:

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

```
-----
-
NameError                                Traceback (most recent call last)
<ipython-input-1-3d3be3b5e906> in <module>
----> 1 np.empty([2,3], dtype='int')
```

NameError: name 'np' is not defined

dtype = data type

In [7]:

```
np.zeros([2,3])
```

Out[7]:

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

2 by 3의 matrix을 0으로 채움

In []:

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

계산을 못하니까 쓸모가 없음

In [11]:

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

Out[11]:

```
array([[0, 0, 0],
       [0, 0, 0]])
```

array 사용

In [12]:

```
np.ones([2,3], dtype = 'float64')
```

Out[12]:

```
array([[1., 1., 1.],
       [1., 1., 1.]])
```

ones 라는 함수 float 64는 64 소수자리까지 (정확하지만 데이터 많이 차지)

In [13]:

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

Out[13]:

```
array([[1, 1, 1],
       [1, 1, 1]])
```

In [14]:

```
np.arange(5)
```

Out[14]:

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

float loop 때 range랑 비슷함

In [15]:

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

Out[15]:

```
array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])
```

In [17]:

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

Out[17]:

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

0부터 10까지 2개 차이로

In [16]:

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

Out[16]:

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

In [18]:

```
np.linspace (0,10,6)
```

Out[18]:

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

0부터 10까지 6개로 나눔

In [20]:

```
x= np.array ([[1,2],[4,5],[8,9]])  
x
```

Out[20]:

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

대괄호 2개 = 2차원 대괄호 3개 = 3차원

In [25]:

```
x= np.array ([[ [1,2],[4,5],[8,9]],[[1,2],[4,5],[8,9]]])  
x
```

Out[25]:

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

In [26]:

```
x. ndim
```

Out[26]:

```
3
```

몇차원인지 알려줌

In [27]:

```
x.shape
```

Out[27]:

```
(2, 3, 2)
```

In [28]:

```
x.dtype
```

Out[28]:

```
dtype('int32')
```

In [29]:

```
x.astype(np.float64)
```

Out[29]:

```
array([[[1., 2.],
        [4., 5.],
        [8., 9.]],

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

In [30]:

```
np.zeros_like(x)
```

Out[30]:

```
array([[[0, 0],
        [0, 0],
        [0, 0]],

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

생긴건 똑같이 숫자를 0으로 바꿈

In [31]:

```
x*0
```

Out[31]:

```
array([[[0, 0],
        [0, 0],
        [0, 0]],

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

In [32]:

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

```
[ 0.5948894  0.62901742 -2.43481232 -0.64520163 -2.25552614  0.19489435
-0.78682143 -1.25504291 -0.27158939 -0.58122381 -0.3975076  0.19090601
-0.7051918  0.85886849 -0.1332941  1.01240101 -0.37015608 -0.3629753
 0.62532226 -0.55471334  0.51391598 -0.46922885 -0.13815178 -0.95075819
 0.28241838  0.76452934  1.1830317  -1.68797608 -0.27279998  1.34480936
 0.38829424  1.41437356  0.21335677  1.51224885  0.24974161 -0.25661151
 0.16083075 -0.21285222 -0.06453529  0.28284785  0.19434011 -0.58968835
-1.84297084  1.38741169 -0.0083715  -0.50905944  1.47782995 -1.34516351
 0.77349661  1.09383414  1.09127973 -1.11432754  0.94883338 -0.23355717
-0.39135684 -1.81958121  0.23487778  0.82448781 -0.87749371 -0.73695733
-0.12413642  2.71239267  0.89010278  1.174496  -0.66606796  0.00658702
-0.50367648 -1.21744002 -0.59010186 -0.58975187 -2.62121707  1.21332786
 0.25443795 -0.64268828  2.12788179  0.94568845  1.24823598 -1.05314019
 1.36540336  0.22325166 -0.4619426  0.78764955 -1.13887395 -0.57905073
 0.46988864  0.44221884 -0.80193795  1.03495679  2.23371987 -0.58567336
-0.06988392  0.24613168 -1.43981342 -2.03215962  0.95582361 -0.27585455
 0.34924642 -0.40700961  0.14194833  0.0361661 ]
```

normal = normal distribution 만들어줌 0 = mean 1= standard deviation 100은 100개의 data 만들어라

In [34]:

```
data.ndim
```

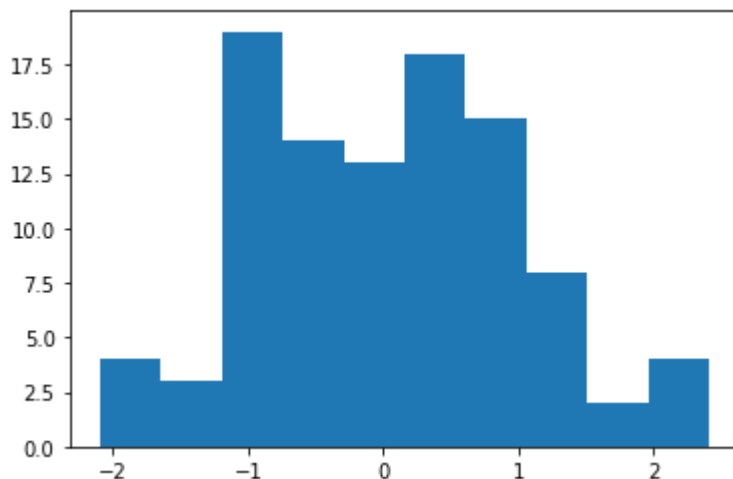
Out [34]:

1

In [35]:

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

```
[-0.29052981  0.3142496 -0.30092923  0.23294338 -0.0096937 -1.75264288
 -1.09379186  1.22418643  0.09211239  0.65077238 -0.18112944  0.59968089
  1.02008837  2.41507312  0.24426123 -0.61227634  2.210503 -0.42432102
  0.41848103  0.17453703 -0.86170595  1.58677576  0.8008617 -2.09665586
  1.36270504 -0.7862358  1.06420937 -0.05100987 -0.14991648 -0.75866774
 -1.12325 -0.86547804 -0.66849136 -0.75090565  0.71276998  0.17085606
  1.82536286 -1.32038878  0.75468933  0.27739382  1.15321934 -0.72314527
  1.08439659  1.48315776  2.13391077 -0.49218686  0.43490225 -0.09114682
 -0.80881791  0.80532131  0.73983968 -0.45784457 -1.62985832  0.38751317
 -0.53320662 -0.80517678  0.26518977 -1.11002782 -0.67592721  0.15333034
 -0.95616868  0.51447594  1.12573406  0.45444624 -0.85809141  0.84831613
  2.33190843  0.03833018 -1.08656145  0.62538228 -0.50315794  0.6696119
  0.54005942 -1.01313403  0.82195499 -0.77879309 -0.3755143 -0.97419932
  0.05934476  0.54322135  0.29534863 -0.2355785  0.11673703 -0.93300181
  1.06537317 -0.64259085 -0.63545322 -0.98052767 -1.55429745  0.81891179
  0.67071758 -2.04890039  0.70647498  0.68675609 -0.87092778 -0.65607294
  0.51223153 -0.04768571 -1.71196897  0.57827161]
```



bins =10 은 graph 에서 10개로 나눔 위에 graph의 값을 다 더하면 100개

In [37]:

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

Out[37]:

```
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.]])
```

2,3,4 = 3차원 2,3,4,5 = 4차원

In [38]:

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

Out[38]:

```
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.]]])
```

reshape = shape바꾸는 것 -1은 뭔지 모르겠다 너가 알아서 해라 but 4써도 똑같이 나옴

In [42]:

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

Out[42]:

True

X reshape 한거랑 y가 똑같은가

In [47]:

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

randint = 0~10에서 숫자 골라서 2 3 matrix 만듦 np.savez 파일로 저장

In [43]:

```
ls -al test*
```

C 드라이브의 볼륨에는 이름이 없습니다.
볼륨 일련 번호: AE98-5A30

C:\Users\Wykim3\OneDrive\문서\영어음성학 디렉터리

C:\Users\Wykim3\OneDrive\문서\영어음성학 디렉터리

파일을 찾을 수 없습니다.

In [48]:

```
who
```

X Y a b data np plt x

who 는 지금 어떤 variable 이 available 한지

In [54]:

```
del a,b
```

```
-----  
-  
NameError                                Traceback (most recent call last)  
<ipython-input-54-a9b92be7739b> in <module>  
----> 1 del a,b
```

NameError: name 'a' is not defined

In [55]:

```
who
```

X Y data np plt x

In [56]:

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

Out[56]:

['arr_0', 'arr_1']

In [57]:

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

Out[57]:

```
array([[1, 6, 6],
       [2, 2, 8]])
```

npzfiles은 저장된 파일불러오기

skiprows 첫번째 row skip because it is title

In [58]:

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

In [59]:

```
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
```

In [62]:

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

In [63]:

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

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


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]])
```

비교할때 dimention이랑 shape 같아야함

In [67]:

```
a
```

Out[67]:

```
array([[1, 2, 3],
       [4, 5, 6],
       [7, 8, 9]])
```

In [69]:

```
a.sum()
```

Out[69]:

45

In [70]:

```
np.sum(a)
```

Out[70]:

45

In [71]:

```
a.sum(axis=0)
```

Out[71]:

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

axis = 몇번째 차원에서 실행

In [72]:

```
a.sum(axis=1)
```

Out[72]:

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

In [73]:

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

Out[73]:

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

In [74]:

```
a = np.arange(1, 25).reshape(4, 6)  
a
```

Out[74]:

```
array([[ 1,  2,  3,  4,  5,  6],  
       [ 7,  8,  9, 10, 11, 12],  
       [13, 14, 15, 16, 17, 18],  
       [19, 20, 21, 22, 23, 24]])
```

In [75]:

```
a+100
```

Out[75]:

```
array([[101, 102, 103, 104, 105, 106],  
       [107, 108, 109, 110, 111, 112],  
       [113, 114, 115, 116, 117, 118],  
       [119, 120, 121, 122, 123, 124]])
```

In [76]:

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

Out[76]:

```
array([0, 1, 2, 3, 4, 5])
```

In [77]:

```
a+b
```

Out[77]:

```
array([[ 1,  3,  5,  7,  9, 11],
       [ 7,  9, 11, 13, 15, 17],
       [13, 15, 17, 19, 21, 23],
       [19, 21, 23, 25, 27, 29]])
```

In []:

In [1]:

```
import numpy as np
import matplotlib.pyplot as plt

from matplotlib import pyplot as plt is the same this as above
```

In [6]:

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

Out[6]:

```
array([[ -1662389008,      367,         0],
       [         0,    131074,   538970682]])
```

dtype = data type

In [7]:

```
np.zeros([2,3])
```

Out[7]:

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

2 by 3의 matrix을 0으로 채움

In []:

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

계산을 못하니까 쓸모가 없음

In [11]:

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

Out[11]:

```
array([[0, 0, 0],  
       [0, 0, 0]])
```

array 사용

In [12]:

```
np.ones([2,3], dtype = 'float64')
```

Out[12]:

```
array([[1., 1., 1.],  
       [1., 1., 1.]])
```

ones 라는 함수 float 64는 64 소수자리까지 (정확하지만 데이터 많이 차지)

In [13]:

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

Out[13]:

```
array([[1, 1, 1],  
       [1, 1, 1]])
```

In [14]:

```
np.arange(5)
```

Out[14]:

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

float loop 때 range랑 비슷함

In [15]:

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

Out[15]:

```
array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])
```

In [17]:

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

Out[17]:

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

0부터 10까지 2개 차이로

In [16]:

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

Out[16]:

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

In [18]:

```
np.linspace (0,10,6)
```

Out[18]:

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

0부터 10까지 6개로 나눔

In [20]:

```
x= np.array ([[1,2],[4,5],[8,9]])  
x
```

Out[20]:

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

대괄호 2개 = 2차원 대괄호 3개 = 3차원

In [25]:

```
x= np.array ([[[1,2],[4,5],[8,9]],[[1,2],[4,5],[8,9]]])  
x
```

Out[25]:

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

In [26]:

```
x. ndim
```

Out[26]:

```
3
```

몇차원인지 알려줌

In [27]:

```
x.shape
```

Out[27]:

```
(2, 3, 2)
```

In [28]:

```
x.dtype
```

Out[28]:

```
dtype('int32')
```

In [29]:

```
x.astype(np.float64)
```

Out[29]:

```
array([[[1., 2.],
        [4., 5.],
        [8., 9.]],

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

In [30]:

```
np.zeros_like(x)
```

Out[30]:

```
array([[[0, 0],
        [0, 0],
        [0, 0]],

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

생긴건 똑같이 숫자를 0으로 바꿈

In [31]:

```
x*0
```

Out[31]:

```
array([[0, 0],
       [0, 0],
       [0, 0]],

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

In [32]:

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

```
[ 0.5948894  0.62901742 -2.43481232 -0.64520163 -2.25552614  0.19489435
 -0.78682143 -1.25504291 -0.27158939 -0.58122381 -0.3975076  0.19090601
 -0.7051918  0.85886849 -0.1332941  1.01240101 -0.37015608 -0.3629753
 0.62532226 -0.55471334  0.51391598 -0.46922885 -0.13815178 -0.95075819
 0.28241838  0.76452934  1.1830317  -1.68797608 -0.27279998  1.34480936
 0.38829424  1.41437356  0.21335677  1.51224885  0.24974161 -0.25661151
 0.16083075 -0.21285222 -0.06453529  0.28284785  0.19434011 -0.58968835
 -1.84297084  1.38741169 -0.0083715  -0.50905944  1.47782995 -1.34516351
 0.77349661  1.09383414  1.09127973 -1.11432754  0.94883338 -0.23355717
 -0.39135684 -1.81958121  0.23487778  0.82448781 -0.87749371 -0.73695733
 -0.12413642  2.71239267  0.89010278  1.174496  -0.66606796  0.00658702
 -0.50367648 -1.21744002 -0.59010186 -0.58975187 -2.62121707  1.21332786
 0.25443795 -0.64268828  2.12788179  0.94568845  1.24823598 -1.05314019
 1.36540336  0.22325166 -0.4619426  0.78764955 -1.13887395 -0.57905073
 0.46988864  0.44221884 -0.80193795  1.03495679  2.23371987 -0.58567336
 -0.06988392  0.24613168 -1.43981342 -2.03215962  0.95582361 -0.27585455
 0.34924642 -0.40700961  0.14194833  0.0361661 ]
```

normal = normal distribution 만들어줌 0 = mean 1= standard deviation 100은 100개의 data 만들어라

In [34]:

```
data.ndim
```

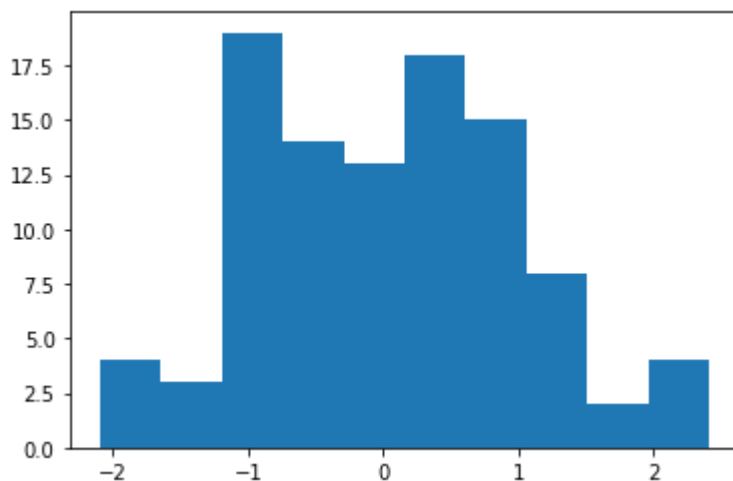
Out[34]:

1

In [35]:

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

```
[-0.29052981  0.3142496 -0.30092923  0.23294338 -0.0096937 -1.75264288
 -1.09379186  1.22418643  0.09211239  0.65077238 -0.18112944  0.59968089
  1.02008837  2.41507312  0.24426123 -0.61227634  2.210503 -0.42432102
  0.41848103  0.17453703 -0.86170595  1.58677576  0.8008617 -2.09665586
  1.36270504 -0.7862358  1.06420937 -0.05100987 -0.14991648 -0.75866774
 -1.12325 -0.86547804 -0.66849136 -0.75090565  0.71276998  0.17085606
  1.82536286 -1.32038878  0.75468933  0.27739382  1.15321934 -0.72314527
  1.08439659  1.48315776  2.13391077 -0.49218686  0.43490225 -0.09114682
 -0.80881791  0.80532131  0.73983968 -0.45784457 -1.62985832  0.38751317
 -0.53320662 -0.80517678  0.26518977 -1.11002782 -0.67592721  0.15333034
 -0.95616868  0.51447594  1.12573406  0.45444624 -0.85809141  0.84831613
  2.33190843  0.03833018 -1.08656145  0.62538228 -0.50315794  0.6696119
  0.54005942 -1.01313403  0.82195499 -0.77879309 -0.3755143 -0.97419932
  0.05934476  0.54322135  0.29534863 -0.2355785  0.11673703 -0.93300181
  1.06537317 -0.64259085 -0.63545322 -0.98052767 -1.55429745  0.81891179
  0.67071758 -2.04890039  0.70647498  0.68675609 -0.87092778 -0.65607294
  0.51223153 -0.04768571 -1.71196897  0.57827161]
```



bins =10 은 graph 에서 10개로 나눔 위에 graph의 값을 다 더하면 100개

In [37]:

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

Out[37]:

```
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.]])
```

2,3,4 = 3차원 2,3,4,5 = 4차원

In [38]:

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

Out[38]:

```
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.]])
```

reshape = shape바꾸는 것 -1은 뭔지 모르겠다 너가 알아서 해라 but 4썬도 똑같이 나옴

In [42]:

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

Out[42]:

True

X reshape 한거랑 y가 똑같은가

In [47]:

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

randint = 0~10에서 숫자 골라서 2 3 matrix 만듦 np.savez 파일로 저장

In [43]:

```
ls -al test*
```

C 드라이브의 볼륨에는 이름이 없습니다.
볼륨 일련 번호: AE98-5A30

C:\Users\Wykim3\OneDrive\문서\영어음성학 디렉터리

C:\Users\Wykim3\OneDrive\문서\영어음성학 디렉터리

파일을 찾을 수 없습니다.

In [48]:

```
who
```

X Y a b data np plt x

who 는 지금 어떤 variable 이 available 한지

In [54]:

```
del a,b
```

```
-----  
-  
NameError                                Traceback (most recent call last)  
<ipython-input-54-a9b92be7739b> in <module>  
----> 1 del a,b
```

NameError: name 'a' is not defined

In [55]:

```
who
```

X Y data np plt x

In [56]:

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

Out[56]:

['arr_0', 'arr_1']

In [57]:

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

Out[57]:

```
array([[1, 6, 6],
       [2, 2, 8]])
```

npzfiles은 저장된 파일불러오기

skiprows 첫번째 row skip because it is title

In [58]:

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

In [59]:

```
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
```

In [62]:

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

In [63]:

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

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

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]])
```

비교할때 dimention이랑 shape 같아야함

In [67]:

```
a
```

Out[67]:

```
array([[1, 2, 3],
       [4, 5, 6],
       [7, 8, 9]])
```

In [69]:

```
a.sum()
```

Out[69]:

45

In [70]:

```
np.sum(a)
```

Out[70]:

```
45
```

In [71]:

```
a.sum(axis=0)
```

Out[71]:

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

axis = 몇번째 차원에서 실행

In [72]:

```
a.sum(axis=1)
```

Out[72]:

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

In [73]:

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

Out[73]:

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

In [74]:

```
a = np.arange(1, 25).reshape(4, 6)
a
```

Out[74]:

```
array([[ 1,  2,  3,  4,  5,  6],
       [ 7,  8,  9, 10, 11, 12],
       [13, 14, 15, 16, 17, 18],
       [19, 20, 21, 22, 23, 24]])
```

In [75]:

```
a+100
```

Out[75]:

```
array([[101, 102, 103, 104, 105, 106],
       [107, 108, 109, 110, 111, 112],
       [113, 114, 115, 116, 117, 118],
       [119, 120, 121, 122, 123, 124]])
```

In [76]:

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

Out[76]:

```
array([0, 1, 2, 3, 4, 5])
```

In [77]:

```
a+b
```

Out[77]:

```
array([[ 1,  3,  5,  7,  9, 11],
       [ 7,  9, 11, 13, 15, 17],
       [13, 15, 17, 19, 21, 23],
       [19, 21, 23, 25, 27, 29]])
```

In []:

In [1]:

```
import numpy as np
import matplotlib.pyplot as plt

from matplotlib import pyplot as plt is the same this as above
```

In [6]:

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

Out[6]:

```
array([[ -1662389008,      367,         0],
       [         0,    131074,   538970682]])
```

dtype = data type

In [7]:

```
np.zeros([2,3])
```

Out[7]:

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

2 by 3의 matrix을 0으로 채움

In []:

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

계산을 못하니까 쓸모가 없음

In [11]:

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

Out[11]:

```
array([[0, 0, 0],
       [0, 0, 0]])
```

array 사용

In [12]:

```
np.ones([2,3], dtype = 'float64')
```

Out[12]:

```
array([[1., 1., 1.],
       [1., 1., 1.]])
```

ones 라는 함수 float 64는 64 소수자리까지 (정확하지만 데이터 많이 차지)

In [13]:

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

Out[13]:

```
array([[1, 1, 1],
       [1, 1, 1]])
```

In [14]:

```
np.arange(5)
```

Out[14]:

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

float loop 때 range랑 비슷함

In [15]:

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

Out[15]:

```
array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])
```

In [17]:

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

Out[17]:

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

0부터 10까지 2개 차이로

In [16]:

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

Out[16]:

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

In [18]:

```
np.linspace (0,10,6)
```

Out[18]:

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

0부터 10까지 6개로 나눔

In [20]:

```
x= np.array ([[1,2],[4,5],[8,9]])  
x
```

Out[20]:

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

대괄호 2개 = 2차원 대괄호 3개 = 3차원

In [25]:

```
x= np.array ([[[1,2],[4,5],[8,9]],[[1,2],[4,5],[8,9]]])  
x
```

Out[25]:

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

In [26]:

```
x. ndim
```

Out[26]:

```
3
```

몇차원인지 알려줌

In [27]:

```
x.shape
```

Out[27]:

```
(2, 3, 2)
```

In [28]:

```
x.dtype
```

Out[28]:

```
dtype('int32')
```

In [29]:

```
x.astype(np.float64)
```

Out[29]:

```
array([[[1., 2.],
        [4., 5.],
        [8., 9.]],

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

In [30]:

```
np.zeros_like(x)
```

Out[30]:

```
array([[[0, 0],
        [0, 0],
        [0, 0]],

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

생긴건 똑같이 숫자를 0으로 바꿈

In [31]:

```
x*0
```

Out[31]:

```
array([[0, 0],
       [0, 0],
       [0, 0]],

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

In [32]:

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

```
[ 0.5948894  0.62901742 -2.43481232 -0.64520163 -2.25552614  0.19489435
 -0.78682143 -1.25504291 -0.27158939 -0.58122381 -0.3975076  0.19090601
 -0.7051918  0.85886849 -0.1332941  1.01240101 -0.37015608 -0.3629753
  0.62532226 -0.55471334  0.51391598 -0.46922885 -0.13815178 -0.95075819
  0.28241838  0.76452934  1.1830317  -1.68797608 -0.27279998  1.34480936
  0.38829424  1.41437356  0.21335677  1.51224885  0.24974161 -0.25661151
  0.16083075 -0.21285222 -0.06453529  0.28284785  0.19434011 -0.58968835
 -1.84297084  1.38741169 -0.0083715  -0.50905944  1.47782995 -1.34516351
  0.77349661  1.09383414  1.09127973 -1.11432754  0.94883338 -0.23355717
 -0.39135684 -1.81958121  0.23487778  0.82448781 -0.87749371 -0.73695733
 -0.12413642  2.71239267  0.89010278  1.174496  -0.66606796  0.00658702
 -0.50367648 -1.21744002 -0.59010186 -0.58975187 -2.62121707  1.21332786
  0.25443795 -0.64268828  2.12788179  0.94568845  1.24823598 -1.05314019
  1.36540336  0.22325166 -0.4619426  0.78764955 -1.13887395 -0.57905073
  0.46988864  0.44221884 -0.80193795  1.03495679  2.23371987 -0.58567336
 -0.06988392  0.24613168 -1.43981342 -2.03215962  0.95582361 -0.27585455
  0.34924642 -0.40700961  0.14194833  0.0361661 ]
```

normal = normal distribution 만들어줌 0 = mean 1= standard deviation 100은 100개의 data 만들어라

In [34]:

```
data.ndim
```

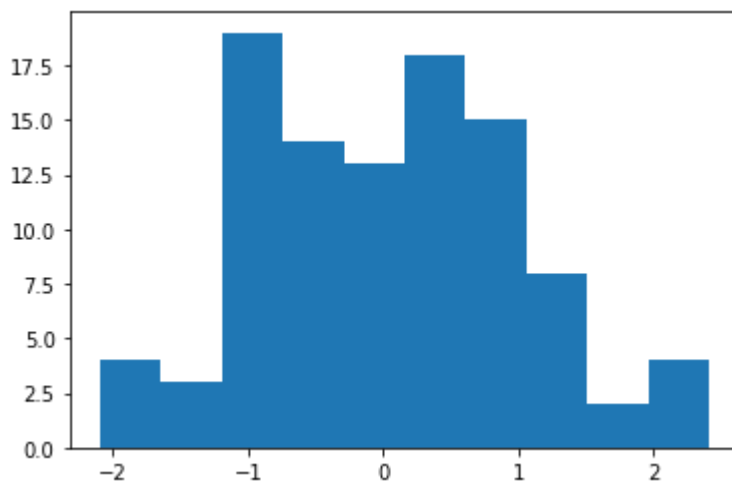
Out[34]:

1

In [35]:

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

```
[-0.29052981  0.3142496 -0.30092923  0.23294338 -0.0096937 -1.75264288
 -1.09379186  1.22418643  0.09211239  0.65077238 -0.18112944  0.59968089
  1.02008837  2.41507312  0.24426123 -0.61227634  2.210503 -0.42432102
  0.41848103  0.17453703 -0.86170595  1.58677576  0.8008617 -2.09665586
  1.36270504 -0.7862358  1.06420937 -0.05100987 -0.14991648 -0.75866774
 -1.12325 -0.86547804 -0.66849136 -0.75090565  0.71276998  0.17085606
  1.82536286 -1.32038878  0.75468933  0.27739382  1.15321934 -0.72314527
  1.08439659  1.48315776  2.13391077 -0.49218686  0.43490225 -0.09114682
 -0.80881791  0.80532131  0.73983968 -0.45784457 -1.62985832  0.38751317
 -0.53320662 -0.80517678  0.26518977 -1.11002782 -0.67592721  0.15333034
 -0.95616868  0.51447594  1.12573406  0.45444624 -0.85809141  0.84831613
  2.33190843  0.03833018 -1.08656145  0.62538228 -0.50315794  0.6696119
  0.54005942 -1.01313403  0.82195499 -0.77879309 -0.3755143 -0.97419932
  0.05934476  0.54322135  0.29534863 -0.2355785  0.11673703 -0.93300181
  1.06537317 -0.64259085 -0.63545322 -0.98052767 -1.55429745  0.81891179
  0.67071758 -2.04890039  0.70647498  0.68675609 -0.87092778 -0.65607294
  0.51223153 -0.04768571 -1.71196897  0.57827161]
```



bins =10 은 graph 에서 10개로 나눔 위에 graph의 값을 다 더하면 100개

In [37]:

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

Out[37]:

```
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.]])
```

2,3,4 = 3차원 2,3,4,5 = 4차원

In [38]:

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

Out[38]:

```
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.]])
```

reshape = shape바꾸는 것 -1은 뭔지 모르겠다 너가 알아서 해라 but 4썬도 똑같이 나옴

In [42]:

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

Out[42]:

True

X reshape 한거랑 y가 똑같은가

In [47]:

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

randint = 0~10에서 숫자 골라서 2 3 matrix 만듦 np.savez 파일로 저장

In [43]:

```
ls -al test*
```

C 드라이브의 볼륨에는 이름이 없습니다.
볼륨 일련 번호: AE98-5A30

C:\Users\Wykim3\OneDrive\문서\영어음성학 디렉터리

C:\Users\Wykim3\OneDrive\문서\영어음성학 디렉터리

파일을 찾을 수 없습니다.

In [48]:

```
who
```

X Y a b data np plt x

who 는 지금 어떤 variable 이 available 한지

In [54]:

```
del a,b
```

```
-----  
-  
NameError                                Traceback (most recent call last)  
<ipython-input-54-a9b92be7739b> in <module>  
----> 1 del a,b
```

NameError: name 'a' is not defined

In [55]:

```
who
```

X Y data np plt x

In [56]:

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

Out[56]:

['arr_0', 'arr_1']

In [57]:

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

Out[57]:

```
array([[1, 6, 6],  
       [2, 2, 8]])
```

npzfiles은 저장된 파일불러오기

skiprows 첫번째 row skip because it is title

In [58]:

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

In [59]:

```
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
```

In [62]:

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

In [63]:

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

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

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]])
```

비교할때 dimention이랑 shape 같아야함

In [67]:

```
a
```

Out[67]:

```
array([[1, 2, 3],
       [4, 5, 6],
       [7, 8, 9]])
```

In [69]:

```
a.sum()
```

Out[69]:

45

In [70]:

```
np.sum(a)
```

Out[70]:

45

In [71]:

```
a.sum(axis=0)
```

Out[71]:

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

axis = 몇번째 차원에서 실행

In [72]:

```
a.sum(axis=1)
```

Out[72]:

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

In [73]:

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

Out[73]:

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

In [74]:

```
a = np.arange(1, 25).reshape(4, 6)  
a
```

Out[74]:

```
array([[ 1,  2,  3,  4,  5,  6],  
       [ 7,  8,  9, 10, 11, 12],  
       [13, 14, 15, 16, 17, 18],  
       [19, 20, 21, 22, 23, 24]])
```

In [75]:

```
a+100
```

Out[75]:

```
array([[101, 102, 103, 104, 105, 106],  
       [107, 108, 109, 110, 111, 112],  
       [113, 114, 115, 116, 117, 118],  
       [119, 120, 121, 122, 123, 124]])
```


In [76]:

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

Out[76]:

```
array([0, 1, 2, 3, 4, 5])
```

In [77]:

```
a+b
```

Out[77]:

```
array([[ 1,  3,  5,  7,  9, 11],
       [ 7,  9, 11, 13, 15, 17],
       [13, 15, 17, 19, 21, 23],
       [19, 21, 23, 25, 27, 29]])
```

In []:

In [1]:

```
import numpy as np
import matplotlib.pyplot as plt

from matplotlib import pyplot as plt is the same this as above
```

In [6]:

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

Out[6]:

```
array([[ -1662389008,      367,         0],
       [         0,    131074,   538970682]])
```

dtype = data type

In [7]:

```
np.zeros([2,3])
```

Out[7]:

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

2 by 3의 matrix을 0으로 채움

In []:

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

계산을 못하니까 쓸모가 없음

In [11]:

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

Out[11]:

```
array([[0, 0, 0],  
       [0, 0, 0]])
```

array 사용

In [12]:

```
np.ones([2,3], dtype = 'float64')
```

Out[12]:

```
array([[1., 1., 1.],  
       [1., 1., 1.]])
```

ones 라는 함수 float 64는 64 소수자리까지 (정확하지만 데이터 많이 차지)

In [13]:

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

Out[13]:

```
array([[1, 1, 1],  
       [1, 1, 1]])
```

In [14]:

```
np.arange(5)
```

Out[14]:

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

float loop 때 range랑 비슷함

In [15]:

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

Out[15]:

```
array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])
```

In [17]:

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

Out[17]:

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

0부터 10까지 2개 차이로

In [16]:

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

Out[16]:

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

In [18]:

```
np.linspace (0,10,6)
```

Out[18]:

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

0부터 10까지 6개로 나눔

In [20]:

```
x= np.array ([[1,2],[4,5],[8,9]])  
x
```

Out[20]:

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

대괄호 2개 = 2차원 대괄호 3개 = 3차원

In [25]:

```
x= np.array ([[[1,2],[4,5],[8,9]],[[1,2],[4,5],[8,9]]])  
x
```

Out[25]:

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

In [26]:

```
x. ndim
```

Out[26]:

```
3
```

몇차원인지 알려줌

In [27]:

```
x.shape
```

Out[27]:

```
(2, 3, 2)
```

In [28]:

```
x.dtype
```

Out[28]:

```
dtype('int32')
```

In [29]:

```
x.astype(np.float64)
```

Out[29]:

```
array([[[1., 2.],
        [4., 5.],
        [8., 9.]],

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

In [30]:

```
np.zeros_like(x)
```

Out[30]:

```
array([[[0, 0],
        [0, 0],
        [0, 0]],

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

생긴건 똑같이 숫자를 0으로 바꿈

In [31]:

```
x*0
```

Out[31]:

```
array([[0, 0],
       [0, 0],
       [0, 0]],

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

In [32]:

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

```
[ 0.5948894  0.62901742 -2.43481232 -0.64520163 -2.25552614  0.19489435
 -0.78682143 -1.25504291 -0.27158939 -0.58122381 -0.3975076  0.19090601
 -0.7051918  0.85886849 -0.1332941  1.01240101 -0.37015608 -0.3629753
 0.62532226 -0.55471334  0.51391598 -0.46922885 -0.13815178 -0.95075819
 0.28241838  0.76452934  1.1830317  -1.68797608 -0.27279998  1.34480936
 0.38829424  1.41437356  0.21335677  1.51224885  0.24974161 -0.25661151
 0.16083075 -0.21285222 -0.06453529  0.28284785  0.19434011 -0.58968835
 -1.84297084  1.38741169 -0.0083715  -0.50905944  1.47782995 -1.34516351
 0.77349661  1.09383414  1.09127973 -1.11432754  0.94883338 -0.23355717
 -0.39135684 -1.81958121  0.23487778  0.82448781 -0.87749371 -0.73695733
 -0.12413642  2.71239267  0.89010278  1.174496  -0.66606796  0.00658702
 -0.50367648 -1.21744002 -0.59010186 -0.58975187 -2.62121707  1.21332786
 0.25443795 -0.64268828  2.12788179  0.94568845  1.24823598 -1.05314019
 1.36540336  0.22325166 -0.4619426  0.78764955 -1.13887395 -0.57905073
 0.46988864  0.44221884 -0.80193795  1.03495679  2.23371987 -0.58567336
 -0.06988392  0.24613168 -1.43981342 -2.03215962  0.95582361 -0.27585455
 0.34924642 -0.40700961  0.14194833  0.0361661 ]
```

normal = normal distribution 만들어줌 0 = mean 1= standard deviation 100은 100개의 data 만들어라

In [34]:

```
data.ndim
```

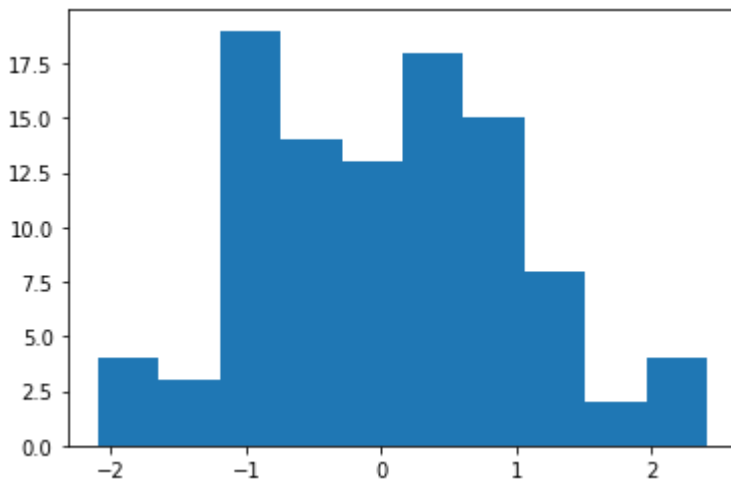
Out[34]:

1

In [35]:

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

```
[-0.29052981  0.3142496 -0.30092923  0.23294338 -0.0096937 -1.75264288
-1.09379186  1.22418643  0.09211239  0.65077238 -0.18112944  0.59968089
 1.02008837  2.41507312  0.24426123 -0.61227634  2.210503 -0.42432102
 0.41848103  0.17453703 -0.86170595  1.58677576  0.8008617 -2.09665586
 1.36270504 -0.7862358  1.06420937 -0.05100987 -0.14991648 -0.75866774
-1.12325 -0.86547804 -0.66849136 -0.75090565  0.71276998  0.17085606
 1.82536286 -1.32038878  0.75468933  0.27739382  1.15321934 -0.72314527
 1.08439659  1.48315776  2.13391077 -0.49218686  0.43490225 -0.09114682
-0.80881791  0.80532131  0.73983968 -0.45784457 -1.62985832  0.38751317
-0.53320662 -0.80517678  0.26518977 -1.11002782 -0.67592721  0.15333034
-0.95616868  0.51447594  1.12573406  0.45444624 -0.85809141  0.84831613
 2.33190843  0.03833018 -1.08656145  0.62538228 -0.50315794  0.6696119
 0.54005942 -1.01313403  0.82195499 -0.77879309 -0.3755143 -0.97419932
 0.05934476  0.54322135  0.29534863 -0.2355785  0.11673703 -0.93300181
 1.06537317 -0.64259085 -0.63545322 -0.98052767 -1.55429745  0.81891179
 0.67071758 -2.04890039  0.70647498  0.68675609 -0.87092778 -0.65607294
 0.51223153 -0.04768571 -1.71196897  0.57827161]
```



bins =10 은 graph 에서 10개로 나눔 위에 graph의 값을 다 더하면 100개

In [37]:

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

Out[37]:

```
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.]])
```

2,3,4 = 3차원 2,3,4,5 = 4차원

In [38]:

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

Out[38]:

```
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.]])
```

reshape = shape바꾸는 것 -1은 뭔지 모르겠다 너가 알아서 해라 but 4썬도 똑같이 나옴

In [42]:

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

Out[42]:

True

X reshape 한거랑 y가 똑같은가

In [47]:

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

randint = 0~10에서 숫자 골라서 2 3 matrix 만듦 np.savez 파일로 저장

In [43]:

```
ls -al test*
```

C 드라이브의 볼륨에는 이름이 없습니다.
볼륨 일련 번호: AE98-5A30

C:\Users\Wykim3\OneDrive\문서\영어음성학 디렉터리

C:\Users\Wykim3\OneDrive\문서\영어음성학 디렉터리

파일을 찾을 수 없습니다.

In [48]:

```
who
```

X Y a b data np plt x

who 는 지금 어떤 variable 이 available 한지

In [54]:

```
del a,b
```

```
-----  
-  
NameError                                Traceback (most recent call last)  
<ipython-input-54-a9b92be7739b> in <module>  
----> 1 del a,b
```

NameError: name 'a' is not defined

In [55]:

```
who
```

X Y data np plt x

In [56]:

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

Out[56]:

['arr_0', 'arr_1']

In [57]:

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

Out[57]:

```
array([[1, 6, 6],  
       [2, 2, 8]])
```

npzfiles은 저장된 파일불러오기

skiprows 첫번째 row skip because it is title

In [58]:

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

In [59]:

```
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
```

In [62]:

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

In [63]:

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

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

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]])
```

비교할때 dimention이랑 shape 같아야함

In [67]:

```
a
```

Out[67]:

```
array([[1, 2, 3],
       [4, 5, 6],
       [7, 8, 9]])
```

In [69]:

```
a.sum()
```

Out[69]:

45

In [70]:

```
np.sum(a)
```

Out[70]:

45

In [71]:

```
a.sum(axis=0)
```

Out[71]:

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

axis = 몇번째 차원에서 실행

In [72]:

```
a.sum(axis=1)
```

Out[72]:

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

In [73]:

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

Out[73]:

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

In [74]:

```
a = np.arange(1, 25).reshape(4, 6)  
a
```

Out[74]:

```
array([[ 1,  2,  3,  4,  5,  6],  
       [ 7,  8,  9, 10, 11, 12],  
       [13, 14, 15, 16, 17, 18],  
       [19, 20, 21, 22, 23, 24]])
```

In [75]:

```
a+100
```

Out[75]:

```
array([[101, 102, 103, 104, 105, 106],  
       [107, 108, 109, 110, 111, 112],  
       [113, 114, 115, 116, 117, 118],  
       [119, 120, 121, 122, 123, 124]])
```

In [76]:

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

Out[76]:

```
array([0, 1, 2, 3, 4, 5])
```

In [77]:

```
a+b
```

Out[77]:

```
array([[ 1,  3,  5,  7,  9, 11],
       [ 7,  9, 11, 13, 15, 17],
       [13, 15, 17, 19, 21, 23],
       [19, 21, 23, 25, 27, 29]])
```

In []:

In [1]:

```
import numpy as np
import matplotlib.pyplot as plt

from matplotlib import pyplot as plt is the same this as above
```

In [6]:

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

Out[6]:

```
array([[ -1662389008,      367,         0],
       [         0,    131074,   538970682]])
```

dtype = data type

In [7]:

```
np.zeros([2,3])
```

Out[7]:

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

2 by 3의 matrix을 0으로 채움

In []:

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

계산을 못하니까 쓸모가 없음

In [11]:

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

Out[11]:

```
array([[0, 0, 0],
       [0, 0, 0]])
```

array 사용

In [12]:

```
np.ones([2,3], dtype = 'float64')
```

Out[12]:

```
array([[1., 1., 1.],
       [1., 1., 1.]])
```

ones 라는 함수 float 64는 64 소수자리까지 (정확하지만 데이터 많이 차지)

In [13]:

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

Out[13]:

```
array([[1, 1, 1],
       [1, 1, 1]])
```

In [14]:

```
np.arange(5)
```

Out[14]:

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

float loop 때 range랑 비슷함

In [15]:

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

Out[15]:

```
array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])
```

In [17]:

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

Out[17]:

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

0부터 10까지 2개 차이로

In [16]:

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

Out[16]:

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

In [18]:

```
np.linspace (0,10,6)
```

Out[18]:

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

0부터 10까지 6개로 나눔

In [20]:

```
x= np.array ([[1,2],[4,5],[8,9]])  
x
```

Out[20]:

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

대괄호 2개 = 2차원 대괄호 3개 = 3차원

In [25]:

```
x= np.array ([[[1,2],[4,5],[8,9]],[[1,2],[4,5],[8,9]]])  
x
```

Out[25]:

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

In [26]:

```
x. ndim
```

Out[26]:

```
3
```

몇차원인지 알려줌

In [27]:

```
x.shape
```

Out[27]:

```
(2, 3, 2)
```

In [28]:

```
x.dtype
```

Out[28]:

```
dtype('int32')
```

In [29]:

```
x.astype(np.float64)
```

Out[29]:

```
array([[[1., 2.],
        [4., 5.],
        [8., 9.]],

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

In [30]:

```
np.zeros_like(x)
```

Out[30]:

```
array([[[0, 0],
        [0, 0],
        [0, 0]],

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

생긴건 똑같이 숫자를 0으로 바꿈

In [31]:

```
x*0
```

Out[31]:

```
array([[0, 0],
       [0, 0],
       [0, 0]],

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

In [32]:

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

```
[ 0.5948894  0.62901742 -2.43481232 -0.64520163 -2.25552614  0.19489435
 -0.78682143 -1.25504291 -0.27158939 -0.58122381 -0.3975076  0.19090601
 -0.7051918  0.85886849 -0.1332941  1.01240101 -0.37015608 -0.3629753
  0.62532226 -0.55471334  0.51391598 -0.46922885 -0.13815178 -0.95075819
  0.28241838  0.76452934  1.1830317  -1.68797608 -0.27279998  1.34480936
  0.38829424  1.41437356  0.21335677  1.51224885  0.24974161 -0.25661151
  0.16083075 -0.21285222 -0.06453529  0.28284785  0.19434011 -0.58968835
 -1.84297084  1.38741169 -0.0083715  -0.50905944  1.47782995 -1.34516351
  0.77349661  1.09383414  1.09127973 -1.11432754  0.94883338 -0.23355717
 -0.39135684 -1.81958121  0.23487778  0.82448781 -0.87749371 -0.73695733
 -0.12413642  2.71239267  0.89010278  1.174496  -0.66606796  0.00658702
 -0.50367648 -1.21744002 -0.59010186 -0.58975187 -2.62121707  1.21332786
  0.25443795 -0.64268828  2.12788179  0.94568845  1.24823598 -1.05314019
  1.36540336  0.22325166 -0.4619426  0.78764955 -1.13887395 -0.57905073
  0.46988864  0.44221884 -0.80193795  1.03495679  2.23371987 -0.58567336
 -0.06988392  0.24613168 -1.43981342 -2.03215962  0.95582361 -0.27585455
  0.34924642 -0.40700961  0.14194833  0.0361661 ]
```

normal = normal distribution 만들어줌 0 = mean 1= standard deviation 100은 100개의 data 만들어라

In [34]:

```
data.ndim
```

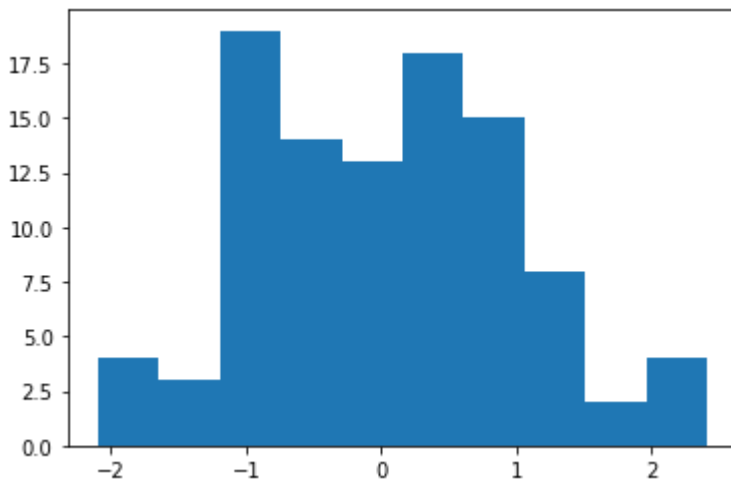
Out[34]:

1

In [35]:

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

```
[-0.29052981  0.3142496 -0.30092923  0.23294338 -0.0096937 -1.75264288
-1.09379186  1.22418643  0.09211239  0.65077238 -0.18112944  0.59968089
 1.02008837  2.41507312  0.24426123 -0.61227634  2.210503 -0.42432102
 0.41848103  0.17453703 -0.86170595  1.58677576  0.8008617 -2.09665586
 1.36270504 -0.7862358  1.06420937 -0.05100987 -0.14991648 -0.75866774
-1.12325 -0.86547804 -0.66849136 -0.75090565  0.71276998  0.17085606
 1.82536286 -1.32038878  0.75468933  0.27739382  1.15321934 -0.72314527
 1.08439659  1.48315776  2.13391077 -0.49218686  0.43490225 -0.09114682
-0.80881791  0.80532131  0.73983968 -0.45784457 -1.62985832  0.38751317
-0.53320662 -0.80517678  0.26518977 -1.11002782 -0.67592721  0.15333034
-0.95616868  0.51447594  1.12573406  0.45444624 -0.85809141  0.84831613
 2.33190843  0.03833018 -1.08656145  0.62538228 -0.50315794  0.6696119
 0.54005942 -1.01313403  0.82195499 -0.77879309 -0.3755143 -0.97419932
 0.05934476  0.54322135  0.29534863 -0.2355785  0.11673703 -0.93300181
 1.06537317 -0.64259085 -0.63545322 -0.98052767 -1.55429745  0.81891179
 0.67071758 -2.04890039  0.70647498  0.68675609 -0.87092778 -0.65607294
 0.51223153 -0.04768571 -1.71196897  0.57827161]
```



bins =10 은 graph 에서 10개로 나눔 위에 graph의 값을 다 더하면 100개

In [37]:

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

Out[37]:

```
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.]])
```

2,3,4 = 3차원 2,3,4,5 = 4차원

In [38]:

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

Out[38]:

```
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.]])
```

reshape = shape바꾸는 것 -1은 뭔지 모르겠다 너가 알아서 해라 but 4썬도 똑같이 나옴

In [42]:

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

Out[42]:

True

X reshape 한거랑 y가 똑같은가

In [47]:

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

randint = 0~10에서 숫자 골라서 2 3 matrix 만듦 np.savez 파일로 저장

In [43]:

```
ls -al test*
```

C 드라이브의 볼륨에는 이름이 없습니다.
볼륨 일련 번호: AE98-5A30

C:\Users\Wykim3\OneDrive\문서\영어음성학 디렉터리

C:\Users\Wykim3\OneDrive\문서\영어음성학 디렉터리

파일을 찾을 수 없습니다.

In [48]:

```
who
```

X Y a b data np plt x

who 는 지금 어떤 variable 이 available 한지

In [54]:

```
del a,b
```

```
-----  
-  
NameError                                Traceback (most recent call last)  
<ipython-input-54-a9b92be7739b> in <module>  
----> 1 del a,b
```

NameError: name 'a' is not defined

In [55]:

```
who
```

X Y data np plt x

In [56]:

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

Out[56]:

['arr_0', 'arr_1']

In [57]:

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

Out[57]:

```
array([[1, 6, 6],
       [2, 2, 8]])
```

npzfiles은 저장된 파일불러오기

skiprows 첫번째 row skip because it is title

In [58]:

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

In [59]:

```
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
```

In [62]:

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

In [63]:

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

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

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]])
```

비교할때 dimention이랑 shape 같아야함

In [67]:

```
a
```

Out[67]:

```
array([[1, 2, 3],
       [4, 5, 6],
       [7, 8, 9]])
```

In [69]:

```
a.sum()
```

Out[69]:

45

In [70]:

```
np.sum(a)
```

Out[70]:

45

In [71]:

```
a.sum(axis=0)
```

Out[71]:

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

axis = 몇번째 차원에서 실행

In [72]:

```
a.sum(axis=1)
```

Out[72]:

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

In [73]:

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

Out[73]:

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

In [74]:

```
a = np.arange(1, 25).reshape(4, 6)  
a
```

Out[74]:

```
array([[ 1,  2,  3,  4,  5,  6],  
       [ 7,  8,  9, 10, 11, 12],  
       [13, 14, 15, 16, 17, 18],  
       [19, 20, 21, 22, 23, 24]])
```

In [75]:

```
a+100
```

Out[75]:

```
array([[101, 102, 103, 104, 105, 106],  
       [107, 108, 109, 110, 111, 112],  
       [113, 114, 115, 116, 117, 118],  
       [119, 120, 121, 122, 123, 124]])
```

In [76]:

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

Out[76]:

```
array([0, 1, 2, 3, 4, 5])
```

In [77]:

```
a+b
```

Out[77]:

```
array([[ 1,  3,  5,  7,  9, 11],
       [ 7,  9, 11, 13, 15, 17],
       [13, 15, 17, 19, 21, 23],
       [19, 21, 23, 25, 27, 29]])
```

In [2]:

```
./nbconvert.py --format=pdf yourfile.ipynb
```

File "<ipython-input-2-2849e3d77fc9>", line 1

```
./nbconvert.py --format=pdf yourfile.ipynb
^
```

SyntaxError: invalid syntax

In []: