

JUPYTER NOTEBOOK 이해하기

Moon Yong Joon

IDE 사용하기

Moon Yong Joon



Kernel 실행

Jupyter notebook 커널 실행

jupyter notebook 커널 선택하여 실행

The image consists of two screenshots of the Jupyter Notebook web interface. The top screenshot shows the 'Files' tab with a file browser. A red dashed box highlights the 'New' button in the top right corner. A red arrow points from the 'New' button to a dropdown menu that lists 'Text File', 'Folder', 'Terminals Unavailable', 'Notebooks', and 'Python 2'. Below this, the text 'New를 눌러 Python2로 실행' (Click New to run with Python2) is written. The bottom screenshot shows the 'Untitled' notebook. A red dashed box highlights the 'Untitled' text in the top left corner. A red arrow points from the 'Untitled' text to a 'Rename Notebook' dialog box. The dialog box has a text input field containing 'TEST_J' and 'OK' and 'Cancel' buttons. Below this, the text 'Untitled 클릭하면 이름 변경 창이 나오고 이름 변경' (Clicking Untitled opens a name change window and you can change the name) is written. Another red dashed box highlights the 'Kernel' menu in the bottom screenshot, with a red arrow pointing from it to the text '파일이름 변경' (Change filename).

Untitled

File Edit View Insert

Rename Notebook

Enter a new notebook name:

TEST_J

OK Cancel

Untitled 클릭하면 이름 변경 창이 나오고 이름 변경

Upload New

Text File
Folder
Terminals Unavailable
Notebooks
Python 2

New를 눌러 Python2로 실행

jupyter TEST_J Last Checkpoint: 1 minutes ago (autosaved)

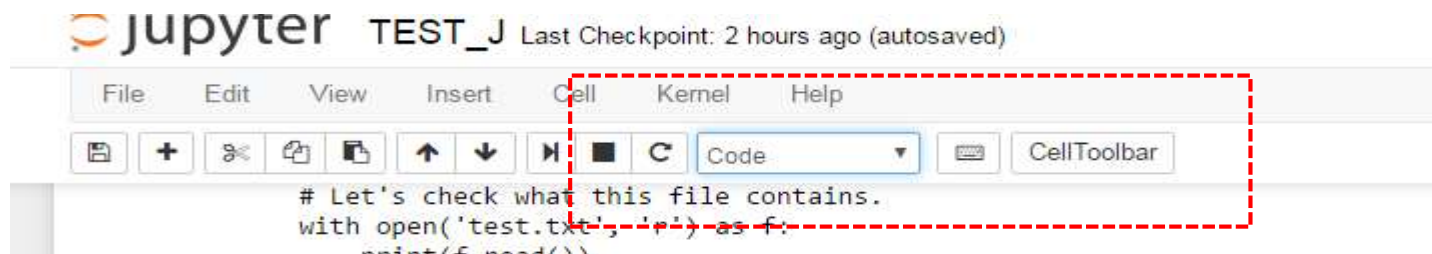
File Edit View Insert Cell Kernel Help

In []:

파일이름 변경

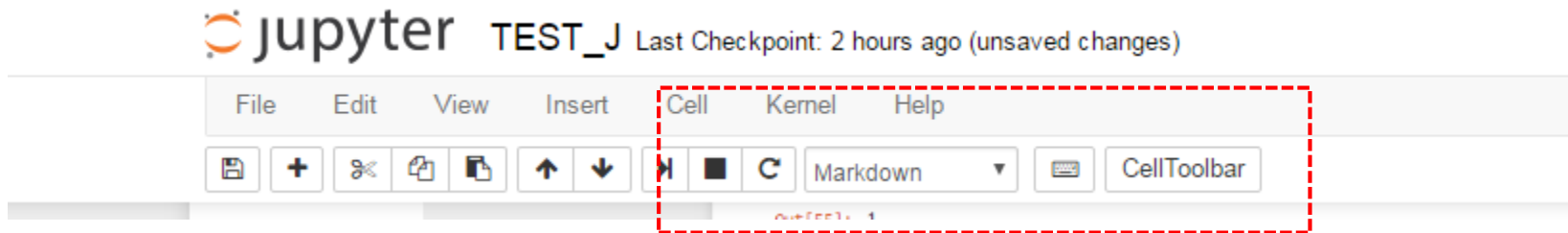
Cell type : code

Cell에 Python 코드가 입력되어 실행



Cell type : markdown

Cell에 markdown에 대한 표기법으로 수학적식이나 문서 등을 작성



Ipynb 파일 보기

! 또는 %(매직 commad)를 이용해서 notebook 파일 보기

In [39]: `% cd ..`

C:\Users\06411

In [40]: `!ls *.ipynb`

TEST_J.ipynb
Untitled.ipynb
dahlmoon.ipynb
test.ipynb

In [41]: `%ls *.ipynb`

C 드라이브의 볼륨: SYSTEM
볼륨 일련 번호: 4A5F-4E72

C:\Users\06411 디렉터리

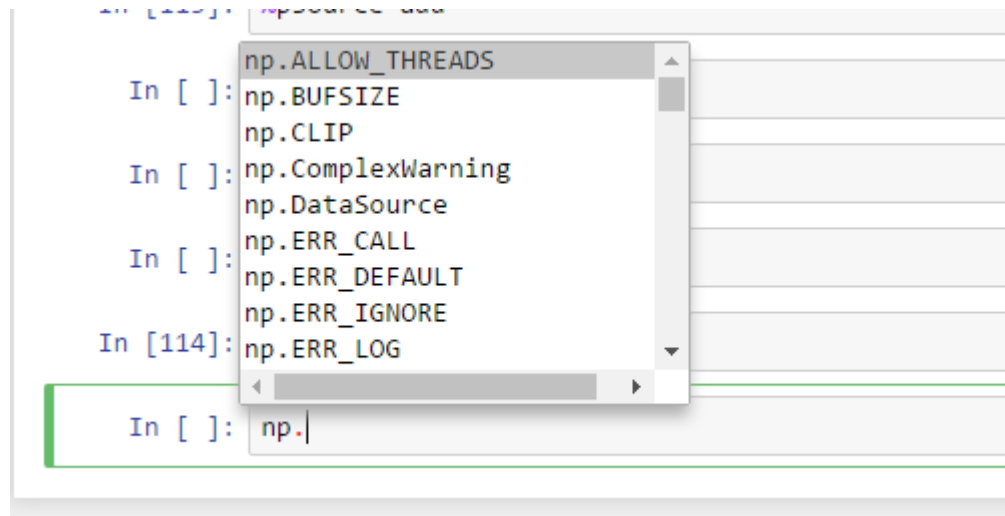
2016-05-02	오후 01:39	82,213	dahlmoon.ipynb
2016-06-27	오전 11:34	10,624	test.ipynb
2016-06-27	오후 01:16	34,883	TEST_J.ipynb
2016-06-27	오전 11:05	72	Untitled.ipynb
4개 파일		127,792	바이트
0개 디렉터리		41,370,497,024	바이트 남음



기본 기능 이해

자동완성: tab

ipython 처럼 입력하고 tab 키를 누르면 내부에 있는 요소들을 보여주므로 선택해서 사용 가능

A screenshot of the IPython interactive shell demonstrating tab completion. The prompt 'In []:' is followed by 'np.' and a vertical bar. A dropdown menu is displayed, listing various NumPy attributes: np.ALLOW_THREADS, np.BUFSIZE, np.CLIP, np.ComplexWarning, np.DataSource, np.ERR_CALL, np.ERR_DEFAULT, np.ERR_IGNORE, and np.ERR_LOG. The first item, np.ALLOW_THREADS, is highlighted. The background shows a list of previous inputs: 'In [113]: np.random.randn', 'In []: np.BUFSIZE', 'In []: np.ComplexWarning', 'In []: np.ERR_CALL', and 'In [114]: np.ERR_LOG'.

```
In [113]: np.random.randn
In [ ]: np.BUFSIZE
In [ ]: np.ComplexWarning
In [ ]: np.ERR_CALL
In [114]: np.ERR_LOG
In [ ]: np.
```

객체 정보 조회 : shift+TAB

변수를 키인하고 shift+TAB을 누르면 내부 특성이 조회 됨

```
import numpy as np
```

```
x = [1, 2, 3, 99, 99, 3, 2, 1]
```

```
x
```

Type: list

String form: [1, 2, 3, 99, 99, 3, 2, 1]

Length: 8

Docstring:

^ + x

함수 정보 조회 : shift+TAB

함수를 키인하고 shift+TAB을 누르면 내부 특성이 조회 됨

```
def add(x:int, y:int) ->int :  
    return x+y
```

add

Signature: add(x:int, y:int) -> int

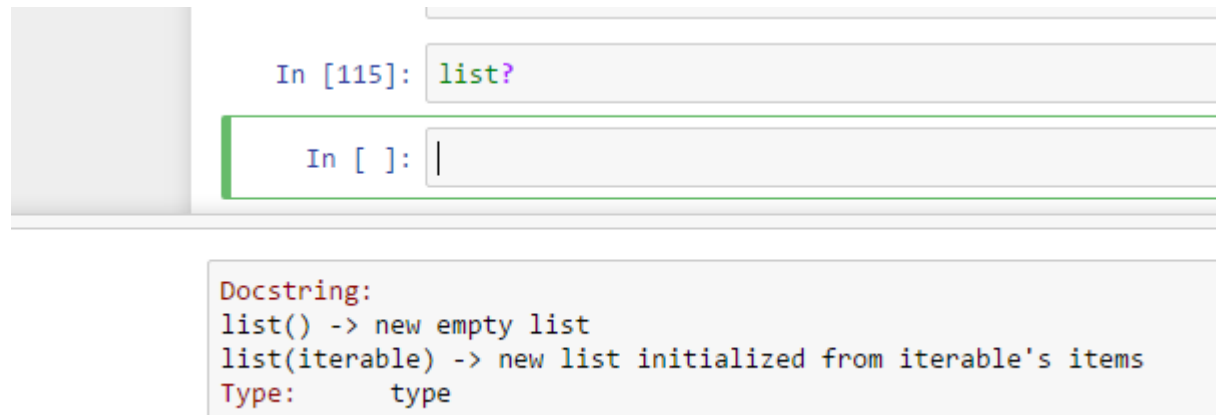
Docstring: <no docstring>

File: c:\users\06411\documents\<ipython-input-820-5bfdc7793909>

Type: function

객체 정보 조회 : ?

ipython 처럼 입력한 후 ?를 붙이고 실행시키면
내부 정보가 보임



```
In [115]: list?
```

```
In [ ]: |
```

```
Docstring:
list() -> new empty list
list(iterable) -> new list initialized from iterable's items
Type:      type
```

객체 정보 조회 : ??

정의된 객체에 다음에 ??를 이용하면 help 정보 (소스)가 나옴

```
In [117]: add??
```

```
In [ ]:
```


```
Signature: add(x, y)
```

```
Source:
```

```
def add(x,y) :  
    return x+y
```

```
File:      c:\users\06411\test\<ipython-input-111-b603eada704c>
```

```
Type:      function
```



파이썬 표현식 처리

Ide 기본 사용하기

표현식을 작성해서 실행을 시키면 실행 결과를 표시 `_`는 앞이 출력결과를 가져와서 다음을 실행

```
In [35]: "Hello World!!!"
```

```
Out[35]: 'Hello World!!!'
```

```
In [36]: 1+ 1
```

```
Out[36]: 2
```

```
In [37]: _+ 10
```

```
Out[37]: 12
```

함수 작성 후 처리

함수를 정의하고 사전에 함수를 매핑해서 실행

```
In [63]: def square(x):  
         """Square of x."""  
         return x*x  
  
         def cube(x):  
             """Cube of x."""  
             return x*x*x
```

```
In [64]: # create a dictionary of functions  
  
funcs = {  
    'square': square,  
    'cube': cube,  
}
```

```
In [65]: x = 2  
  
print square(x)  
print cube(x)  
  
for func in sorted(funcs):  
    print func, funcs[func](x)
```

```
4  
8  
cube 8  
square 4
```




파일 처리

File 저장 및 처리

텍스트 파일을 생성한 후에 file을 오픈한 후에 처리하기

```
In [46]: %cd TEST
```

```
C:\Users\06411\TEST
```

```
In [47]: %%writefile test.txt  
Hello world!
```

```
Writing test.txt
```

```
In [49]: # Let's check what this file contains.  
with open('test.txt', 'r') as f:  
    print(f.read())
```

```
Hello world!
```

파이썬 함수(.py) 실행하기

%run(매직 commad)을 이용해서 파이썬 모듈 실행

```
In [9]: %%writefile add.py
```

```
def add(x,y) :  
    return x+y  
  
print add(4,4)
```

Writing add.py

```
In [10]: %ls
```

C 드라이브의 볼륨: SYSTEM
볼륨 일련 번호: 4A5F-4E72

C:\Users\06411\TEST 디렉터리

2016-06-27	오후 12:32	<DIR>	.
2016-06-27	오후 12:32	<DIR>	..
2016-06-27	오후 12:32		50 add.py
		1개 파일	50 바이트
		2개 디렉터리	41,355,968,512 바이트 남음

```
In [11]: %run add.py
```



Markdown cell 처리

Markdown cell

마크다운 셀을 사용해서 로직이나 다양한 설명을 작성함

```
In [ ]: ### New paragraph
This is rich text with links(http://ipython.org),
equations:

$$\hat{f}(\xi) = \int_{-\infty}^{+\infty} f(x) \mathrm{e}^{-i \xi x} dx$$

code with syntax highlighting:
```python
print("Hello world!")
|
```

## New paragraph

This is *rich* **text** with [links](#), equations:

$$\hat{f}(\xi) = \int_{-\infty}^{+\infty} f(x) e^{-i\xi x} dx$$

code with syntax highlighting: ```python print("Hello world!")



# Ndarray 정의

# List 정의 후 ndarray로 변환

list 정의 후 ndarray로 변환되면 내부 속성으로는  
shape(배열의 모양), dtype(데이터타입),  
strides(size), ndim(차원)로 구성

```
In [220]: from random import random
list1 = [random() for _ in range(1000)]
list2 = [random() for _ in range(1000)]
```

```
In [221]: list1
```

```
Out[221]: [0.6613515102149676,
0.6596178750127825,
0.6631784970620159,
0.6787861801225666,
0.3139119330923984,
0.6341853132179451,
0.678287721363433,
0.5488762085798072,
0.5895861182148574,
0.7186348284471711,
0.17539981964497375,
0.5477082753241312,
0.29513879095323714,
0.6431061629632866,
0.4468474985971197,
0.13582680132246165]
```

```
In [224]: import numpy as np
arr1 = np.array(list1)
arr2 = np.array(list2)
```

```
In [225]: arr1.shape
```

```
Out[225]: (1000,)
```

```
In [226]: arr1.dtype
```

```
Out[226]: dtype('float64')
```

```
In [227]: arr1.strides
```

```
Out[227]: (8,)
```

```
In [228]: arr1.ndim
```

```
Out[228]: 1
```

# IPYTHON

## 모듈

## 사용하기





# 이미지 파일 처리

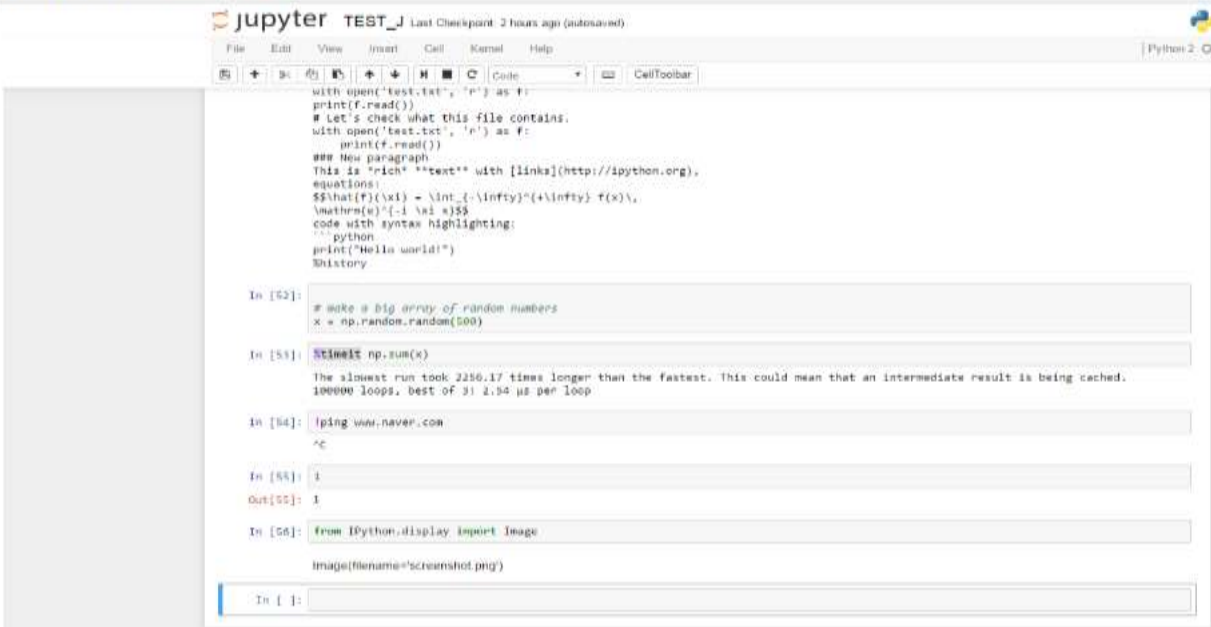
# IPython 이미지 처리

Jupyter notebook 이미지를 캡처해서 저장하고 이를 불러 출력

```
In [61]: from IPython.display import Image

In [62]: Image(filename='screen.png')
```

Out[62]:



The screenshot shows a Jupyter Notebook window titled 'jupyter TEST\_J'. The interface includes a menu bar (File, Edit, View, Insert, Cell, Kernel, Help) and a toolbar with icons for navigation and execution. The code cell contains the following Python code:

```
with open('test.txt', 'r') as f:
 print(f.read())
Let's check what this file contains.
with open('test.txt', 'r') as f:
 print(f.read())
New paragraph
This is "rich" text with [links](http://ipython.org),
equations:

$$\hat{f}(x) = \int_{-\infty}^{\infty} f(x) dx$$

code with syntax highlighting:
python
print("Hello world!")
%history
```

The output of the code cell shows the following results:

```
In [62]: # make a big array of random numbers
x = np.random.random(500)

In [53]: %timeit np.sum(x)
The slowest run took 2256.17 times longer than the fastest. This could mean that an intermediate result is being cached.
10000 loops, best of 3: 2.04 µs per loop

In [54]: !ping www.naver.com
^C

In [55]: 1
Out[55]: 1

In [56]: from IPython.display import Image
Image(filename='screenshot.png')
```

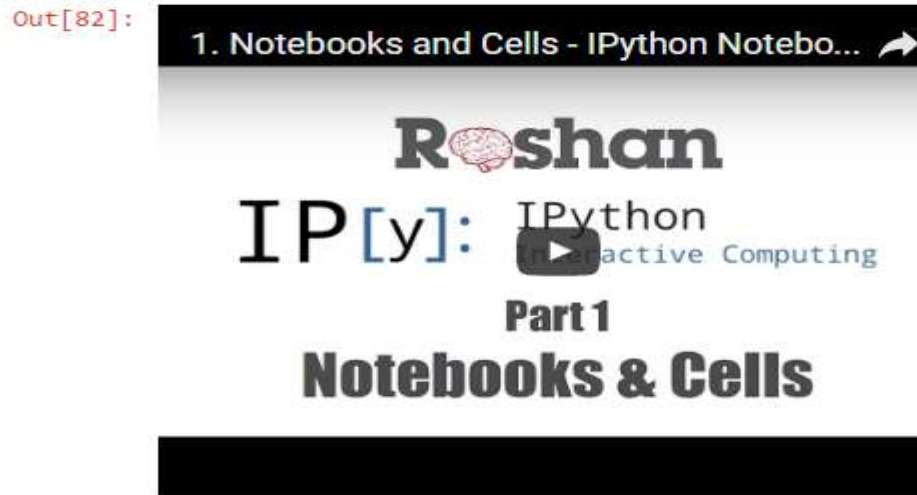
The final output shows the image 'screenshot.png' being displayed.

# IPython 유튜브 연계

Jupyter notebook에서 유튜브 영상을 처리할 수 있음, 유튜브의 파일명만 내부에 작성하면 호출이 됨

```
In [81]: from IPython.display import YouTubeVideo
```

```
In [82]: YouTubeVideo("lmoNmY-cmSI")
```





# ipython 단축키

# IPython 단축키

Jupyter notebook에서 ipython 단축키를 사용  
일부는 Windows에서는 실행되지 않을 수 있음.

Ctrl+P, 위 화살표 키	명령어 이력을 역순으로 검색
Ctrl+N, 아래 화살표 키	명령어 이력을 최근 순으로 검색
Ctrl+R	readline 명령어 형식의 이력 검색
Ctrl+Shift + V	클립보드에서 텍스트 붙여넣기
Ctrl+C	현재 실행중인 코드 중단하기
Ctrl+A	커서를 줄의 처음으로 이동하기
Ctrl+E	커서를 줄의 마지막으로 이동하기
Ctrl+K	커서가 놓인 곳부터 줄이 마지막까지 지우기
Ctrl+U	현재 입력된 모든 텍스트 지우기
Ctrl+F	커서의 앞으로 한글자씩 이동하기
Ctrl+B	커서를 뒤로 한글자씩 이동하기
Ctrl+L	화면 지우기



편집 / 명령모드

# 단축키:ctrl+enter

cell에 표현식을 입력하고 ctrl+ enter를치면 현재 cell이 실행되고 다음 cell이 활성화 되지 않음

```
In [160]: c = 10
```

```
In [161]: print c
```

```
10
```

# 단축키: shift+enter

cell에 표현식을 입력하고 shift+ enter(Alt-Enter )를치면 다음 셀이 활성화

```
In [155]: a=10
```

```
In [156]: print a
```

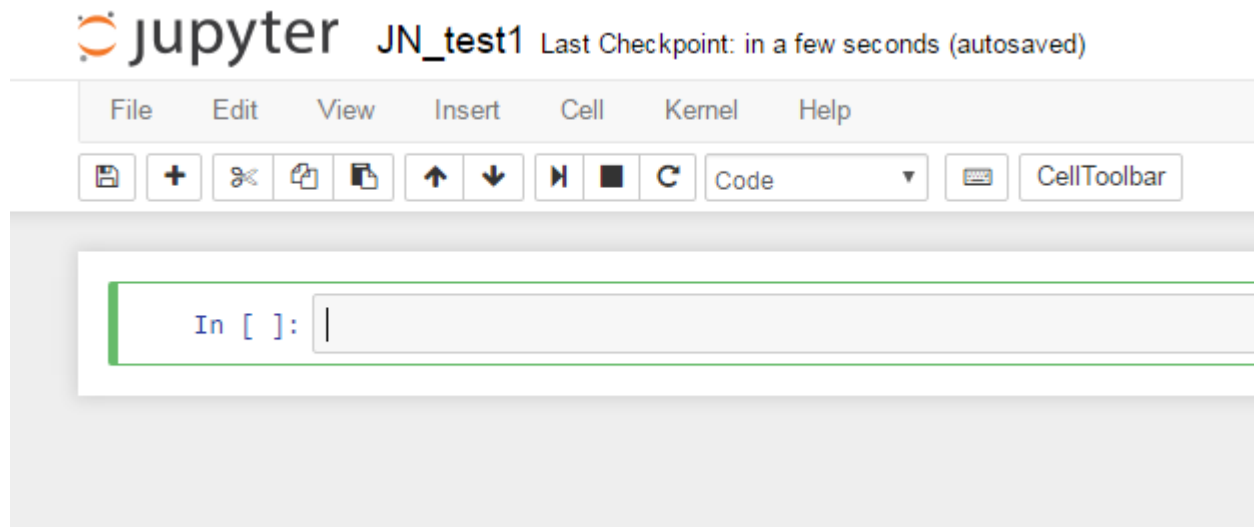
```
10
```

```
In []:
```



# 단축키:ctrl+S

현재 jupyter notebook에 저장되었다는 표시가  
남음

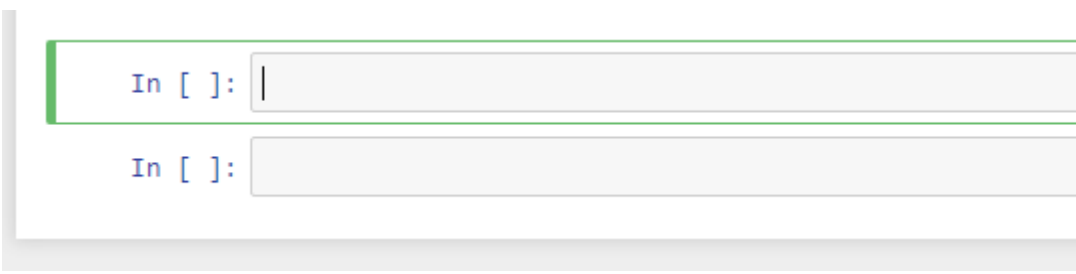




편집모드

# 단축키: ctrl + shift + “-”

편집모드에서 cell 분할



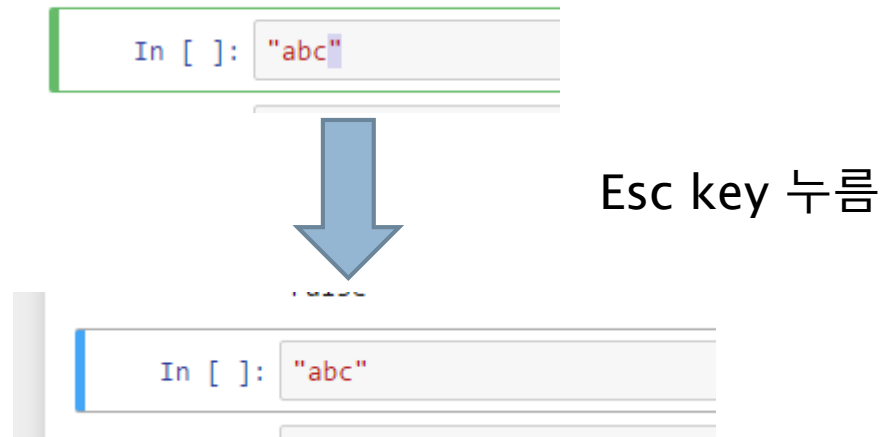
The image shows a Jupyter Notebook interface with two input cells. The top cell is highlighted with a green border and contains the text 'In [ ]: |'. The bottom cell contains the text 'In [ ]: '.

In [ ]: |

In [ ]:

# 단축키: Esc

편집모드에서 cell을 명령모드로 변경  
초록색에서 파란색으로 변경





os command 사용

# ! 키워드 : ls|grep 사용

디렉토리 내부에 특정 파일을 찾아서 표시하기

```
In [188]: files = !ls -l -S | grep .edges
```

```
In [189]: files
```

```
Out[189]: ['-rw-r--r-- 1 06411 Administ 600274 Dec 29 2012 1912.edges',
 '-rw-r--r-- 1 06411 Administ 523802 Dec 29 2012 107.edges',
 '-rw-r--r-- 1 06411 Administ 280354 Dec 29 2012 1684.edges',
 '-rw-r--r-- 1 06411 Administ 96188 Dec 29 2012 3437.edges',
 '-rw-r--r-- 1 06411 Administ 51066 Dec 29 2012 348.edges',
 '-rw-r--r-- 1 06411 Administ 37228 Dec 29 2012 0.edges',
 '-rw-r--r-- 1 06411 Administ 27082 Dec 29 2012 414.edges',
 '-rw-r--r-- 1 06411 Administ 26496 Dec 29 2012 686.edges',
 '-rw-r--r-- 1 06411 Administ 4320 Dec 29 2012 698.edges',
 '-rw-r--r-- 1 06411 Administ 2914 Dec 29 2012 3980.edges']
```

# Tab 사용해서 이름 검색

파일명 등을 세부 적으로 모를 경우 tab을 이용해서 조회

```
In [194]: !head -n5 1912.
 953 1323 1912.conjugate
 1789 1707 1912.denominator
 1175 1059 1912.edges
 1329 1559 1912.imag
 1804 1898 1912.numerator
 1912.real

In []: !head -n5 1912.
```

# MAGIC COMMAND

Moon Yong Joon





# Magic command

# Magic command

magic command에는 `line(%)`과 `cell(%%)`로 지정해서 처리할 수 있음

line

%magic command  
for example, %run foo.py 는 s  
file foo.py를 실행

Cell(전체)

%%magic command  
for example, %%latex 는 모든  
cell에 있는 latex를 번역

# Magic command 조회

%lsmagic을 이용해서 가지고 있는 command를  
전체 조회

```
In [42]: %lsmagic
```

```
Out[42]: Available line magics:
```

```
%alias %alias_magic %autocall %automagic %autosave %bookmark %cd %clear %cls %colors %config %connect_info %copy %ddir %debug %dhist %dirs %doctest_mode %echo %ed %edit %env %gui %hist %history %install_default_config %install_ext %install_profiles %killbgscripts %ldir %less %load %load_ext %loadpy %logoff %logon %logstart %logstate %logstop %ls %lsmagic %macro %magic %matplotlib %mkdir %more %notebook %page %pastebin %pdb %pdef %pdoc %pfile %pinfo %pinfo2 %popd %pprint %precision %profile %prun %psearch %psource %pushd %pwd %pycat %pylab %qtconsole %quickref %recall %rehashx %reload_ext %ren %rep %rerun %reset %reset_selective %rmdir %run %save %sc %set_env %store %sx %system %tb %time %timeit %unalias %unload_ext %who %who_ls %whos %xdel %xmode
```

```
Available cell magics:
```

```
%%! %%HTML %%SVG %%bash %%capture %%cmd %%debug %%file %%html %%javascript %%latex %%perl %%prun %%pypy %%python %%python2 %%python3 %%ruby %%script %%sh %%svg %%sx %%system %%time %%timeit %%writefile
```

```
Automagic is ON, % prefix IS NOT needed for line magics.
```

# 주요 Magic command 1

## %ismagic 내의 주요 명령어 설명

명령어	설명
%pwd, %cd	현재 위치 및 다른 디렉토리로 이동
%history	명령어 히스토리 출력
%reset	모든 정의된 변수 삭제
%%capture	실행되는 명령에 대한 정보의 결과를 저장
%whos	현재 정의된 변수 표시
%pdoc, %psource	Help 기능 실행
%timeit	평균 실행 시간을 출력
%bookmark	디렉토리에 대한 별칭을 저장하고 쉽게 이동할 수 있게 해줌
%%writefile	현재 디렉토리에 파일 생성
%load	디렉토리에 있는 파일을 셀에 로딩
%run	py 프로그램 파일을 실행
%matplotlib inline	matplotlib을 내부 셀에서 실행하기

# 주요 Magic command : 2

## %lsmagic 내의 주요 명령어 설명

명령어	설명
%ls	현재 디렉토리에 파일들을 보기
%magic	모든 매직 함수에 대한 상세 도움말 출력
%pdb	예외가 발생하면 자동적으로 디버거 진입.(한번 입력시 ON, 다시 입력시 OFF)
%debug	작성된 코드에 대한 debug 처리

# Magic command 내의 help

%magic comand 뒤에 ?를 입력하면 설명이 나온다

```
In [45]: %edit?
```

```
In []: |
```

**Docstring:**

Bring up an editor and execute the resulting code.

**Usage:**

%edit [options] [args]

%edit runs an external text editor. You will need to set the command for this editor via the ``TerminalInteractiveShell.editor`` option in your configuration file before it will work.

This command allows you to conveniently edit multi-line code right in your IPython session.

If called without arguments, %edit opens up an empty editor with a temporary file and will execute the contents of this file when you close it (don't forget to save it!).

# Magic command 확인하기

## magic command에 대한 설명 보기

```
In [118]: %magic
```

IPython's 'magic' functions

=====

The magic function system provides a series of functions which allow you to control the behavior of IPython itself, plus a lot of system-type features. There are two kinds of magics, line-oriented and cell-oriented.

Line magics are prefixed with the % character and work much like OS command-line calls: they get as an argument the rest of the line, where arguments are passed without parentheses or quotes. For example, this will time the given statement::

# %bookmark 처리

특정 디렉토리를 특정 이름으로 관리하고 싶을 때 지정해서 사용

```
%cd facebook
```

```
/notebooks/facebook
```

```
%ls
```

0.circles	1684.circles	3437.circles	3980.circles	686.circles
0.edges	1684.edges	3437.edges	3980.edges	686.edges
107.circles	1912.circles	348.circles	414.circles	698.circles
107.edges	1912.edges	348.edges	414.edges	698.edges

```
%bookmark fbdata
```

```
%pwd
```

```
u'/notebooks/facebook'
```

```
%cd ..
```

```
/notebooks
```

```
%cd fbdata
```

```
(bookmark:fbdata) -> /notebooks/facebook
/notebooks/facebook
```



# %capture 처리

## %%capture 파일명 후 실제 실행하는 매직명령어의 실행결과를 별도로 저장해서 처리

```
%%capture output
%ls
```

output

```
<IPython.utils.capture.CapturedIO at 0x7efd77fef50>
```

```
output.stdout
```

```
u' withfile.txt \x1b[0m\x1b[01;34mfacebook\x1b[0m/\r\n1_hello_tensorflow.ipynb fo
o.txt\r\n2_getting_started.ipynb foo1.txt\r\n3_mnist_from_scratch.ipynb image.jpg\r\nC:\\Users
411\\Downloads\\line_plot_plus.pdf line_plot_plus.pdf\r\nLICENSE mod.py\r\nMatplotlib_test1.
ipynb mod.pyc\r\nMover_ch2.pyde mod_f.py\r\nUntitled.ipynb
 mod_f.pyc\r\nUntitled1.ipynb newfile.txt\r\nUntitled2.ipynb test.txt\r
\r\nUntitled3.ipynb test2.ipynb\r\nnarraystore.nd test_1.ipynb\r\nndata.txt
 understanding Python_20160815.ipynb\r\nodoctest.py withfile.txt\r
\r\nodoctest.pyc yum-2.0.7.tar.gz\r\n'
```

```
output.show()
```

withfile.txt	facebook/
1_hello_tensorflow.ipynb	foo.txt
2_getting_started.ipynb	foo1.txt
3_mnist_from_scratch.ipynb	image.jpg
C:\Users\411\Downloads\line_plot_plus.pdf	line_plot_plus.pdf
LICENSE	mod.py
Matplotlib_test1.ipynb	mod.pyc
Mover_ch2.pyde	mod_f.py
Untitled.ipynb	mod_f.pyc
Untitled1.ipynb	newfile.txt
Untitled2.ipynb	test.txt
Untitled3.ipynb	test2.ipynb
arraystore.nd	test_1.ipynb
data.txt	understanding_Python_20160815.ipynb
doctest.py	withfile.txt



# 변수 관리

# Notebook 내의 변수 삭제

`%reset`(매직 commad)은 현재 실행되는 notebook 내의 모든 변수를 삭제함

```
In [77]: x
```

```
Out[77]: 10
```

```
In [78]: %reset
```

```
Once deleted, variables cannot be recovered. Proceed (y/[n])? y
```

```
In [79]: x
```

```

NameError Traceback (most recent call last)
<ipython-input-79-401b30e3b8b5> in <module>()
----> 1 x

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

변수가 삭제되어 오류 메시지

# Whos: 변수들 표시

현재 실행환경 내의 Variables을 표시, similar to Matlab's whos

```
In [100]: %whos
```

Variable	Type	Data/Info
YouTubeVideo	type	<class 'IPython.lib.display.YouTubeVideo'>
add	function	<function add at 0x0643EF30>

```
In [101]: %reset
```

Once deleted, variables cannot be recovered. Proceed (y/[n])? y

```
In [102]: %whos
```

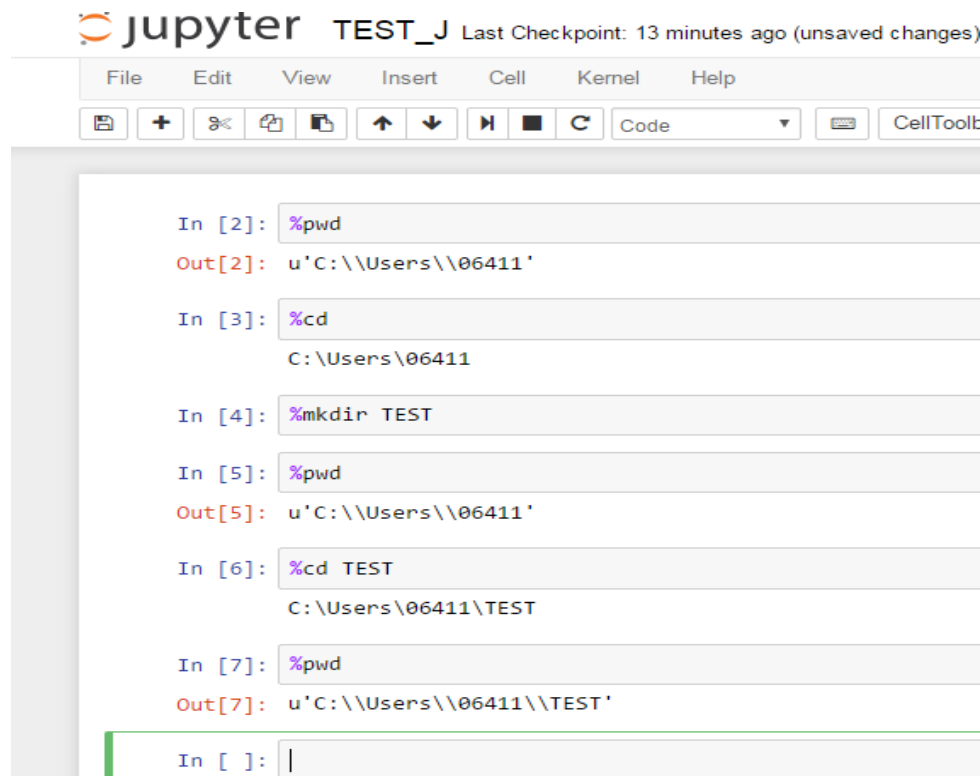
Interactive namespace is empty.



# 작업 위치 정하기

# Directory 만들고 이동

%(매직 commad)를 이용해서 현재 위치 및 디렉토리 생성 및 이동



The image shows a Jupyter Notebook interface with the title 'TEST\_J' and a status bar indicating 'Last Checkpoint: 13 minutes ago (unsaved changes)'. The interface includes a menu bar (File, Edit, View, Insert, Cell, Kernel, Help) and a toolbar with icons for file operations and execution. The notebook contains several code cells demonstrating directory creation and navigation using magic commands.

```
In [2]: %pwd
Out[2]: u'C:\\Users\\06411'
```

```
In [3]: %cd
C:\\Users\\06411
```

```
In [4]: %mkdir TEST
```

```
In [5]: %pwd
Out[5]: u'C:\\Users\\06411'
```

```
In [6]: %cd TEST
C:\\Users\\06411\\TEST
```

```
In [7]: %pwd
Out[7]: u'C:\\Users\\06411\\TEST'
```

```
In []: |
```



# 파일 생성 및 실행

# 파이썬 파일 만들고 확인하기

%%writefile(매직 commad)를 이용해서 현재 위치에 add.py 생성하고 조회

```
In [9]: %%writefile add.py
```

```
def add(x,y) :
 return x+y

print add(4,4)
```

Writing add.py

```
In [10]: %ls
```

C 드라이브의 볼륨: SYSTEM  
볼륨 일련 번호: 4A5F-4E72

C:\Users\06411\TEST 디렉터리

2016-06-27	오후 12:32	<DIR>	.
2016-06-27	오후 12:32	<DIR>	..
2016-06-27	오후 12:32		50 add.py
		1개 파일	50 바이트
		2개 디렉터리	41,355,968,512 바이트 남음



# 파이썬 파일 실행하기

%run(매직 commad)을 이용해서 파이썬 모듈 실행

```
In [9]: %%writefile add.py
```

```
def add(x,y) :
 return x+y

print add(4,4)
```

Writing add.py

```
In [10]: %ls
```

C 드라이브의 볼륨: SYSTEM  
볼륨 일련 번호: 4A5F-4E72

C:\Users\06411\TEST 디렉터리

2016-06-27	오후 12:32	<DIR>	.
2016-06-27	오후 12:32	<DIR>	..
2016-06-27	오후 12:32		50 add.py
		1개 파일	50 바이트
		2개 디렉터리	41,355,968,512 바이트 남음

```
In [11]: %run add.py
```

8



# 파일 로드하기

# 파이썬 파일 실행하기

%loadpy(매직 commad)를 이용해서 python 파일을 로드하고 실행하면 결과가 나옴

```
In []: %loadpy add.py
```

```
In []: # %Load add.py
```

```
def add(x,y) :
 return x+y
```

```
print add(4,4)
```

```
In [23]: # %Load add.py
```

```
def add(x,y) :
 return x+y
```

```
print add(4,4)
```



# 실행시간 점검

# %timeit 사용

%timeit 를 사용해서 실행하는 시간을 점검

```
%history
```

In [52]:

```
make a big array of random numbers
x = np.random.random(500)
```

In [53]: %timeit np.sum(x)

The slowest run took 2256.17 times longer than the fastest. This could mean that an ir  
100000 loops, best of 3: 2.54 µs per loop

In [54]:



help 기능

# 매직커맨드 + ?

매직커맨드에 ? 를 붙이면 매직커맨드에 대한 help 기능이 실행됨

```
In [424]: %history?
```

```
By default, all input history from the current session is displayed.
Ranges of history can be indicated using the syntax:
```

```
``4``
```

```
 Line 4, current session
```

```
``4-6``
```

```
 Lines 4-6, current session
```

```
``243/1-5``
```

# %pdoc: 객체에 대한 doc 조회

“%pdoc 객체”를 입력해서 docstring을 조회

```
In [103]: %pdoc list
```

```
In []:
```

```
Class docstring:
 list() -> new empty list
 list(iterable) -> new list initialized from iterable's items
Init docstring:
 x.__init__(...) initializes x; see help(type(x)) for signature
```



# 함수에 대한 소스와 헤드 조회

“%pdef 객체”, “%psource”를 입력해서 함수의  
헤드와 소스를 조회

```
In [111]: def add(x,y) :
 return x+y
```

```
In [112]: %pdef add
 add(x, y)
```

```
In [113]: %psource add
```

```
In []: |
```

```
def add(x,y) :
 return x+y
```



history

# Cell에 입력된 history 확인하기

## Cell에 입력된 이력을 출력

In [51]: `%history`

```
%pwd
%pwd
%cd
%mkdir TEST
%pwd
%cd TEST
%pwd
%writefile add.py

def add(x,y) :
 return x+y

print add(4,4)
%%writefile add.py

def add(x,y) :
 return x+y

print add(4,4)
```

# Cell에 입력된 일부 history 확인

현재 명령된 이전 명령 5개만 읽어오기

```
In [192]: %history -l 5
%ls
files = !ls -l -S | grep .edges
files
%history
%history?
```

# 방문한 모든 디렉토리 history 확인

%dhist로 현재까지 방문한 모든 디렉토리 이력을 읽어오기

```
In [193]: %dhist
```

```
Directory history (kept in _dh)
0: C:\Users\06411\TEST
1: C:\Users\06411\TEST\facebook
```

---



debug

# Cell 입력한 로직 오류 점검

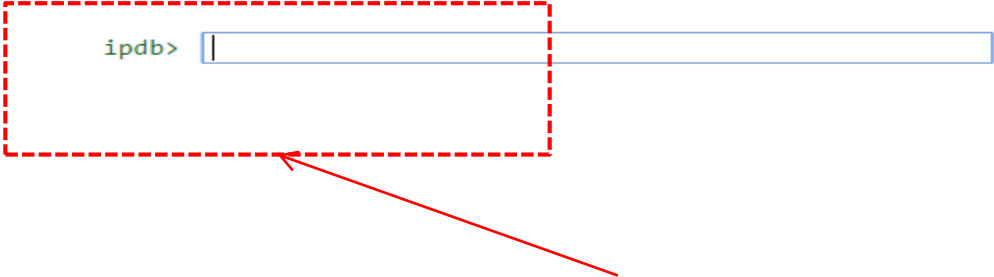
Cell에 입력된 로직에 대한 debug.

```
In [*]: def sub(x,y) :
 aaaaa
 return x + y

In [*]: %debug

> <ipython-input-120-ee2d5fc03e64>(2)sub()
1 def sub(x,y) :
----> 2 a
 3 return x + y

ipdb> |
```



**s(tep)**  
function).

**n(ext)**

**unt(il)**

frame returns.

**r(eturn)**

**c(ontinue)**

-- Execute the current line, stop at the first possible occasion (either in a function that is called or in the current

-- Continue execution until the next line in the current function is reached or it returns.

-- Continue execution until the line with a number greater than the current one is reached or until the current

-- Continue execution until the current function returns.

-- Continue execution, only stop when a breakpoint is encountered.

# SHELL COMMAND

Moon Yong Joon





Shell command

# !shell command

Cell에 !shell command 실행하면 작동되고 이를 파이썬 변수에 할당할 수 있음

```
!ls
```

```
BeautifulSoup_test_20170120.ipynb
GitHub
My Music
My Pictures
My Videos
Processing
__pycache__
add_a_workbook.xlsx
arraystore.nd
arraystore.npy
arraystore.txt
bdsc4j.csv.xlsx
bs4_google_20170130.ipynb
country_data.xml
data.bin
. . .
```

```
contents = !ls
print(contents[:2])
```

```
!cd
path = !cd
print(path)
```

```
!echo "printing from the shell"
message = "hello from Python"
!echo {message}
```

```
['BeautifulSoup_test_20170120.ipynb', 'GitHub']
C:\\Users\\06411\\Documents
['C:\\\\Users\\\\06411\\\\Documents']
"printing from the shell"
hello from Python
```

# Shell과 관련된 magic command

Cell에서 shell과 magic 명령이 같은 부분이 존재 os별로 상이함

shell	magic	설명
!pwd, !cd	%pwd, %cd	현재 위치 및 다른 디렉토리로 이동
!env	%env	컴퓨터 환경정보 보기
!echo	%echo	메시지 출력하기
!cp	%cp	카피하기
!ls	%ls	현재 디렉토리의 리스트
!mkdir	%mkdir	디렉토리 생성
!rmdir	%rmdir	디렉토리 삭제
!mv	%mv	파일 이동
!rm	%rm	파일 삭제

# 데이터 가져오기

Moon Yong Joon



# Wget 설치

# wget

## 인터넷상에서 데이터를 가져오기

```
!wget https://raw.githubusercontent.com/ipython-books/minibook-2nd-data/master/nyc_taxi.zip
--2016-09-06 01:08:26-- https://raw.githubusercontent.com/ipython-books/minibook-2nd-data/master/nyc_taxi.zip
Resolving raw.githubusercontent.com (raw.githubusercontent.com)... 151.101.100.133
Connecting to raw.githubusercontent.com (raw.githubusercontent.com)|151.101.100.133|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 91078102 (87M) [application/octet-stream]
Saving to: 'nyc_taxi.zip'

100%[=====>] 91,078,102 10.6MB/s in 9.2s

2016-09-06 01:08:45 (9.48 MB/s) - 'nyc_taxi.zip' saved [91078102/91078102]
```

# Sudo apt-get install

sudo apt-get install wget 명령으로 설치  
기 설치되어 있으면 아래의 메시지가 나옴

```
!sudo apt-get install wget
```

```
Reading package lists... Done
Building dependency tree
Reading state information... Done
wget is already the newest version.
0 upgraded, 0 newly installed, 0 to remove and 105 not upgraded.
```

---

# Wget 실행



# wget

## 인터넷상에서 데이터를 가져오기

```
!wget https://raw.githubusercontent.com/ipython-books/minibook-2nd-data/master/nyc_taxi.zip
--2016-09-06 01:08:26-- https://raw.githubusercontent.com/ipython-books/minibook-2nd-data/master/nyc_taxi.zip
Resolving raw.githubusercontent.com (raw.githubusercontent.com)... 151.101.100.133
Connecting to raw.githubusercontent.com (raw.githubusercontent.com)|151.101.100.133|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 91078102 (87M) [application/octet-stream]
Saving to: 'nyc_taxi.zip'

100%[=====>] 91,078,102 10.6MB/s in 9.2s

2016-09-06 01:08:45 (9.48 MB/s) - 'nyc_taxi.zip' saved [91078102/91078102]
```



unzip 실행

# unzip

## 압축파일을 받을 경우 unzip 처리

```
%ls
```

```
JN_test1.ipynb add.pyc file_append.txt foo.txt
Untitled.ipynb class_func.ipynb file_read.txt nyc_taxi.zip
add.py file-write.txt file_write.txt test1_0831.ipynb
```

```
!unzip nyc_taxi.zip
```

```
Archive: nyc_taxi.zip
 creating: data/
 inflating: data/nyc_data.csv
 inflating: data/nyc_fare.csv
```

```
%ls
```

```
JN_test1.ipynb class_func.ipynb file_read.txt test1_0831.ipynb
Untitled.ipynb data/ file_write.txt
add.py file-write.txt foo.txt
add.pyc file_append.txt nyc_taxi.zip
```

# NUMPY

# 처리

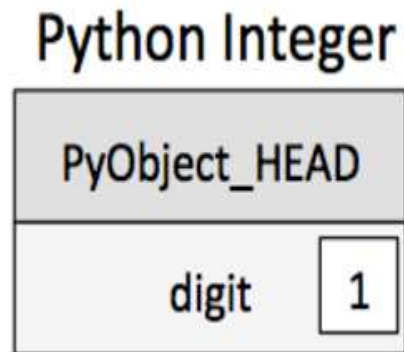
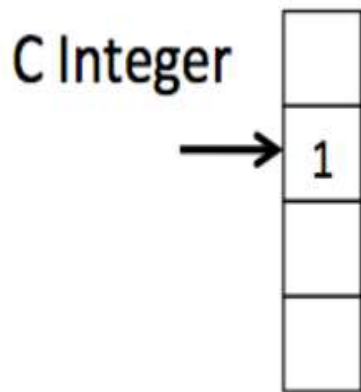
Moon Yong Joon



# 배열의 구조

# Python 객체 구조 : int

int 객체의 인스턴스에 대한 보관 방법



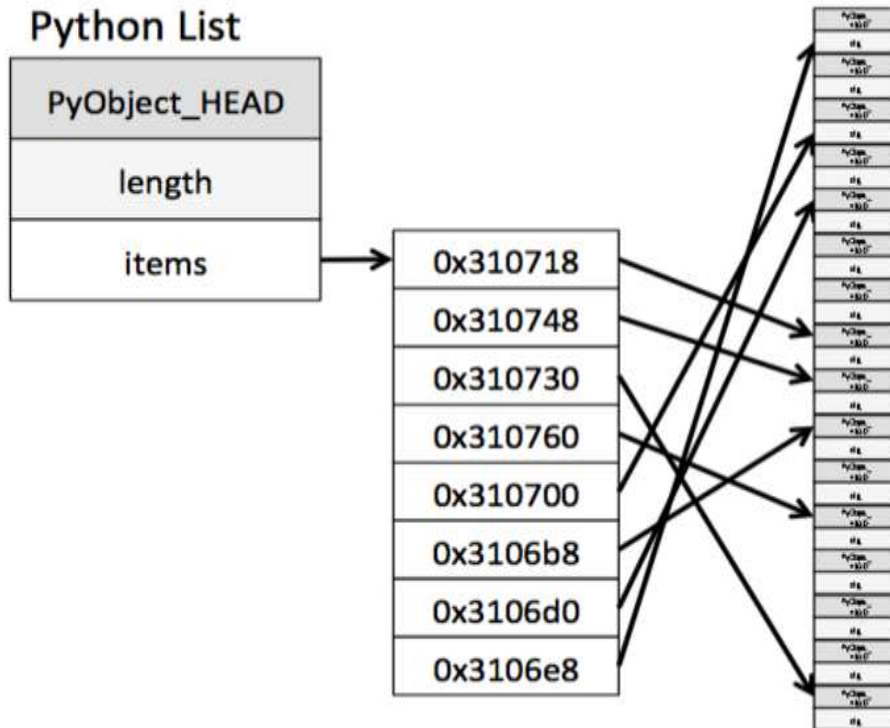
```
a = int(1)

print(a)
print(type(a))
print(isinstance(a, int))
```

1  
<class 'int'>  
True

# Python 객체 구조 : list

list 객체 인스턴스는 다양한 타입의 객체를 보관



```
lt = [1, 's', ['b', 'c']]
print(id(lt), type(lt))
print(id(lt[0]), type(lt[0]))
print(id(lt[1]), type(lt[1]))
print(id(lt[2]), type(lt[2]))
```

```
81466440 <class 'list'>
1562313200 <class 'int'>
6119296 <class 'str'>
95722760 <class 'list'>
```

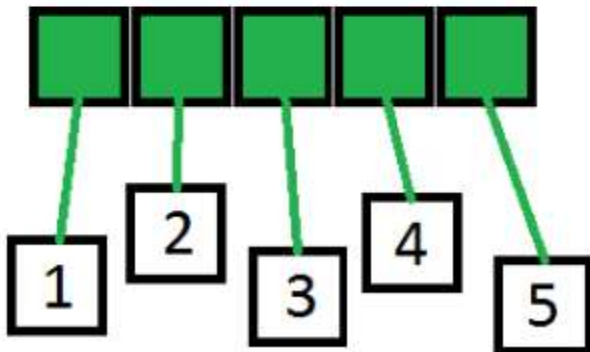
# Python 객체 구조 : array

array.array 객체 인스턴스는 고정 타입의 객체를 보관

Array of values



Array of references



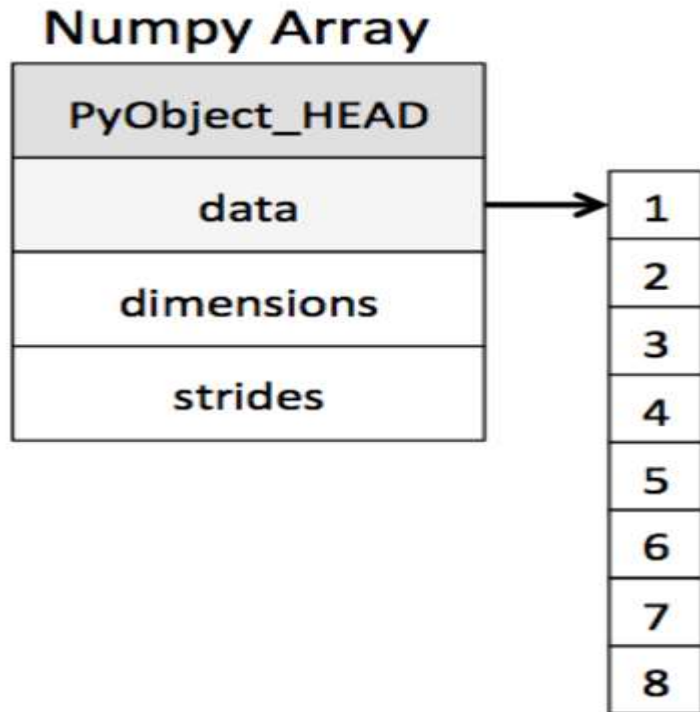
```
import array
L = list(range(10))
A = array.array('i', L)
print(A)
print(A.typecode)
print(A.tolist())
print(A.buffer_info())
```

```
array('i', [0, 1, 2, 3, 4, 5, 6, 7, 8, 9])
i
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
(69207456, 10)
```



# Python 객체 구조 : ndarray

ndarray 객체 인스턴스는 타입의 객체를 보관



```
import numpy as np

nda = np.array([1,2,3,4,5])
print(nda.shape)
print(nda.dtype)
print(nda.ndim)
print(nda.strides)
print(nda.data.tolist())
```

(5,)  
int32  
1  
(4,)  
[1, 2, 3, 4, 5]



# 배열의 생성

# List로 생성

list, list comprehension으로 ndarray 객체 인스턴스 생성

```
import numpy as np

integer array:
i = np.array([1, 4, 2, 5, 3])
print(i)
print(i.dtype)

float array:
f = np.array([3.14, 4, 2, 3])
print(f)
print(f.dtype)

nested lists result in multi-dimensional arrays
mi = np.array([range(i, i + 3) for i in [2, 4, 6]])
print(mi)
```

```
[1 4 2 5 3]
int32
[3.14 4. 2. 3.]
float64
[[2 3 4]
 [4 5 6]
 [6 7 8]]
```

# zeros / ones / full함수로 생성

## zeros / ones / full함수로 ndarray 객체 인스턴스 생성

```
Create a length-10 integer array filled with zeros
ze = np.zeros(10, dtype=int)
print(ze)
Create a 3x5 floating-point array filled with ones
one = np.ones((3, 5), dtype=float)
print(one)
Create a 3x5 array filled with 3.14
fl = np.full((3, 5), 3.14)
print(fl)
```

```
[0 0 0 0 0 0 0 0 0 0]
[[1. 1. 1. 1. 1.]
 [1. 1. 1. 1. 1.]
 [1. 1. 1. 1. 1.]]
[[3.14 3.14 3.14 3.14 3.14]
 [3.14 3.14 3.14 3.14 3.14]
 [3.14 3.14 3.14 3.14 3.14]]
```

# eye/empty 함수로 생성

eye/empty 함수로 ndarray 객체 인스턴스 생성

```
import numpy as np

Create a 3x3 identity matrix
ey = np.eye(3)
print(ey)

Create an uninitialized array of three integers
The values will be whatever happens to already exist at that memory location
em = np.empty(3)
print(em)
```

```
[[1. 0. 0.]
 [0. 1. 0.]
 [0. 0. 1.]]
[1. 1. 1.]
```

# arange/linspace함수로 생성

arange/linspace함수로 ndarray 객체 인스턴스 생성

```
import numpy as np
|
Create an array filled with a linear sequence
Starting at 0, ending at 20, stepping by 2
(this is similar to the built-in range() function)
ar = np.arange(0, 20, 2)
print(ar)

Create an array of five values evenly spaced between 0 and 1
ls = np.linspace(0, 1, 5)
print(ls)
```

```
[0 2 4 6 8 10 12 14 16 18]
[0. 0.25 0.5 0.75 1.]
```

# random 모듈의 함수로 생성

## random 모듈의 함수로 ndarray 객체 인스턴스 생성

```
import numpy as np

Create a 3x3 array of uniformly distributed
random values between 0 and 1
rr = np.random.random((3, 3))
print(rr)

Create a 3x3 array of normally distributed random values
with mean 0 and standard deviation 1
rn = np.random.normal(0, 1, (3, 3))
print(rn)

Create a 3x3 array of random integers in the interval [0, 10)
ri = np.random.randint(0, 10, (3, 3))
print(ri)
```

```
[[0.88050071 0.79042114 0.81195593]
 [0.10219646 0.477742 0.71939974]
 [0.29968495 0.31610957 0.42161167]]
[[0.27567609 -0.00974696 0.35981263]
 [1.55050238 0.26103931 -0.06907681]
 [-0.24800385 -1.48016813 -0.45919365]]
[[3 2 4]
 [8 1 1]
 [7 2 6]]
```

# 배열의 data type 지정하기



# ndarray 생성시 data type 지정

ndarray 객체 인스턴스 생성시 데이터 타입을 문자열이나 np.int16 처럼 지정 가능

```
import numpy as np

z1 = np.zeros(10, dtype='int16')
print(z1)
print(z1.dtype)

z2 = np.zeros(10, dtype=np.int16)
print(z2)
print(z2.dtype)
```

[0 0 0 0 0 0 0 0 0 0]  
int16  
[0 0 0 0 0 0 0 0 0 0]  
int16

# ndarray 생성시 data type 1

## ndarray 객체 인스턴스 생성시 데이터 타입

Data type	Description
bool_	Boolean (True or False) stored as a byte
int_	Default integer type (same as C long; normally either int64 or int32)
intc	Identical to C int (normally int32 or int64)
intp	Integer used for indexing (same as C ssize_t; normally either int32 or int64)
int8	Byte (-128 to 127)
int16	Integer (-32768 to 32767)
int32	Integer (-2147483648 to 2147483647)
int64	Integer (-9223372036854775808 to 9223372036854775807)
uint8	Unsigned integer (0 to 255)
uint16	Unsigned integer (0 to 65535)

# ndarray 생성시 data type 2

## ndarray 객체 인스턴스 생성시 데이터 타입

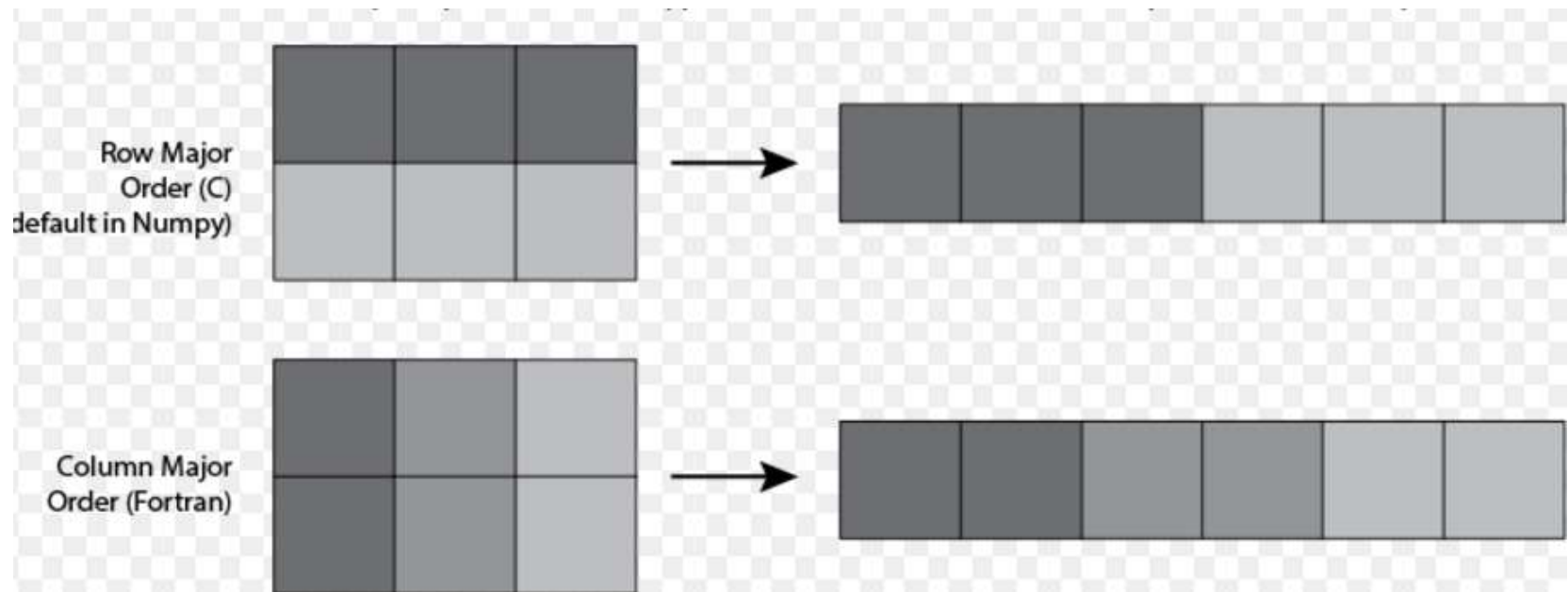
Data type	Description
uint32	Unsigned integer (0 to 4294967295)
uint64	Unsigned integer (0 to 18446744073709551615)
float_	Shorthand for float64.
float16	Half precision float: sign bit, 5 bits exponent, 10 bits mantissa
float32	Single precision float: sign bit, 8 bits exponent, 23 bits mantissa
float64	Double precision float: sign bit, 11 bits exponent, 52 bits mantissa
complex_	Shorthand for complex128.
complex64	Complex number, represented by two 32-bit floats
complex128	Complex number, represented by two 64-bit floats



# 배열의 주요 속성

# ndarray 배열의 구성

ndarray 객체의 배열의 구성은 평면 배열로 구성하면 기본이 행단위로 분리되고 Fortran 언어 스타일일 경우 열단위로 구성



# ndarray : 1차원

## ndarray 객체 인스턴스 생성된 1차원 속성

```
import numpy as np
np.random.seed(0) # seed for reproducibility

x1 = np.random.randint(10, size=6) # One-dimensional array

print(x1.flatten())
print("x1 차원 : ", x1.ndim)
print("x1 형태 :", x1.shape)
print("x1 size: :", x1.size)
print("x1 dtype :", x1.dtype)
print("itemsize:", x1.itemsize, "bytes")
print("nbytes:", x1.nbytes, "bytes")
```

```
[5 0 3 3 7 9]
x1 차원 : 1
x1 형태 : (6,)
x1 size: : 6
x1 dtype : int32
itemsize: 4 bytes
nbytes: 24 bytes
```

# ndarray : 2차원

## ndarray 객체 인스턴스 생성된 2차원 속성

```
import numpy as np
np.random.seed(0) # seed for reproducibility

x2 = np.random.randint(10, size=(3, 4)) # Two-dimensional array

print(x2.flatten())
print("x2 차원 : ", x2.ndim)
print("x2 형태 : ", x2.shape)
print("x2 size: ", x2.size)
print("x2 dtype : ", x2.dtype)
print("itemsize:", x2.itemsize, "bytes")
print("nbytes:", x2.nbytes, "bytes")
```

```
[5 0 3 3 7 9 3 5 2 4 7 6]
```

```
x2 차원 : 2
x2 형태 : (3, 4)
x2 size: : 12
x2 dtype : int32
itemsize: 4 bytes
nbytes: 48 bytes
```

# ndarray : 3차원

## ndarray 객체 인스턴스 생성된 3차원 속성

```
import numpy as np
np.random.seed(0) # seed for reproducibility

x3 = np.random.randint(10, size=(3, 4, 5)) # Three-dimensional array
print(x3.flatten())
print("x3 차원 : ", x3.ndim)
print("x3 형태 : ", x3.shape)
print("x3 size: : ", x3.size)
print("x3 dtype : ", x3.dtype)
print("itemsize:", x3.itemsize, "bytes")
print("nbytes:", x3.nbytes, "bytes")
```

[5 0 3 3 7 9 3 5 2 4 7 6 8 8 1 6 7 7 8 1 5 9 8 9 4 3 0 3 5 0 2 3 8 1 3 3 3  
7 0 1 9 9 0 4 7 3 2 7 2 0 0 4 5 5 6 8 4 1 4 9]

x3 차원 : 3  
x3 형태 : (3, 4, 5)  
x3 size: : 60  
x3 dtype : int32  
itemsize: 4 bytes  
nbytes: 240 bytes





# 배열 내부 조회

# 원소 읽기(indexing)

ndarray 객체 인스턴스 생성된 차원에 따라 원소 읽기

```
import numpy as np
np.random.seed(0) # seed for reproducibility

x1 = np.random.randint(6, size=6) # One-dimensional array
x2 = np.random.randint(8, size=(3, 2)) # Two-dimensional array
x3 = np.random.randint(12, size=(3, 2, 2)) # Three-dimensional array

print(x1[0])
print(x2[0])
print(x3[0])

print(x1[-1])
print(x2[-1])
print(x3[-1])
```

```
4
[7 1]
[[7 6]
 [8 8]]
3
[2 4]
[[7 8]
 [1 5]]
```

# subarrays 읽기(slicing)

ndarray 객체 인스턴스 생성된 차원에 따라 동일 타입의 subarray를 읽기

```
import numpy as np
np.random.seed(0) # seed for reproducibility

x1 = np.random.randint(6, size=6) # One-dimensional array
x2 = np.random.randint(8, size=(3, 2)) # Two-dimensional array
x3 = np.random.randint(12, size=(3, 2, 2)) # Three-dimensional array

print(x1[:2])
print(x2[:2])
print(x3[:2])

print(x1[-1:])
print(x2[-1:])
print(x3[-1:])
```

```
[4 5]
[[7 1]
 [3 5]]
[[[7 6]
 [8 8]]

 [[10 1]
 [6 7]]]
[3]
[[2 4]]
[[[7 8]
 [1 5]]]
```

# slices separated by commas

ndarray 객체 인스턴스 생성된 차원에 따라  
commas로 분리해서 읽기

```
import numpy as np
np.random.seed(0) # seed for reproducibility

x2 = np.random.randint(8, size=(3, 2)) # Two-dimensional array

print(x2)
print(x2[1, 1]) # 두번째 행, 두번째 열 원소 검색
print(x2[1, :]) # 두번째 행
print(x2[:, -1, ::-1]) # 역으로 출력
```

```
[[4 7]
 [5 0]
 [3 3]]
0
[5 0]
[[3 3]
 [0 5]
 [7 4]]
```



# 배열 원소 변경

# ndarray는 기본 view 제공

ndarray는 기본 view만 제공하므로 조회 후 변경시 원본 배열도 갱신됨

```
import numpy as np

x2 = np.random.randint(8, size=(3, 2)) # Two-dimensional array
print(x2)
print(" slice 배열 조회")
x2_sub = x2[:2, :2]
print(x2_sub)
print(" slice 배열 변경")
x2_sub[0, 0] = 99
print(x2_sub)
print("원본 배열 ")
print(x2)
```

```
[[4 7]
 [6 0]
 [0 4]]
 slice 배열 조회
[[4 7]
 [6 0]]
 slice 배열 변경
[[99 7]
 [6 0]]
원본 배열
[[99 7]
 [6 0]
 [0 4]]
```

# ndarray 조회 후 copy 사용

ndarray는 기본 view만 제공하므로 원본을 유지하려면 copy해서 사용해야 함

```
import numpy as np

x2 = np.random.randint(8, size=(3, 2)) # Two-dimensional array
print(x2)
print(" slice 배열 조회 및 복사 ")
x2_sub_copy = x2[:2, :2].copy()
print(x2_sub_copy)
print(" slice 배열 변경")
x2_sub_copy[0, 0] = 42
print(x2_sub_copy)
print("원본 배열 ")
print(x2)
```

```
[[0 1]
 [5 1]
 [5 0]]
slice 배열 조회 및 복사
[[0 1]
 [5 1]]
slice 배열 변경
[[42 1]
 [5 1]]
원본 배열
[[0 1]
 [5 1]
 [5 0]]
```



# 배열 형태 변경



# reshape

배열을 바꾸면 새로운 배열이 만들어 짐

```
import numpy as np

#reshape 메소드 처리
grid = np.arange(1, 10).reshape((3, 3))
print(grid)

#reshape 함수 처리

x = np.array([1, 2, 3, 4, 5, 6])
xx = np.reshape(x, (2, 3))
print(xx)
```

```
[[1 2 3]
 [4 5 6]
 [7 8 9]]
[[1 2 3]
 [4 5 6]]
```

# newaxis

기존 배열에 축을 추가해서 새로운 배열이 만들기

```
import numpy as np

#reshape 메소드 처리
x = np.arange(0, 3)
row vector via newaxis
rx = x[np.newaxis, :]
print(rx)

column vector via reshape
xs = x.reshape((3, 1))
print(xs)
xc = x[:, np.newaxis]
print(xc)
```

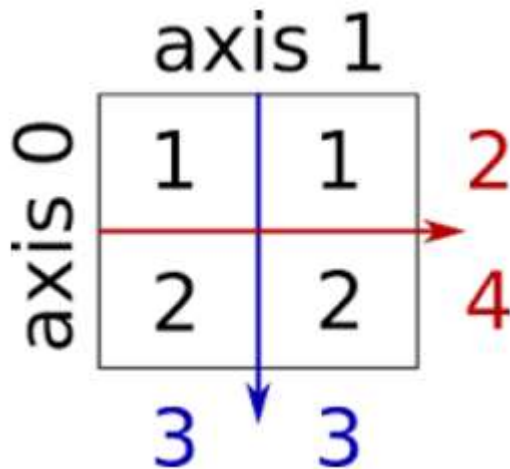
```
[[0 1 2]]
[[0]
 [1]
 [2]]
[[0]
 [1]
 [2]]
```



배열 행 / 열 축으로 통합

# concatenate

배열을 행 또는 열 축으로 통합 axis=0이면 행이 추가, axis=1이면 열로 추가



```
import numpy as np

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

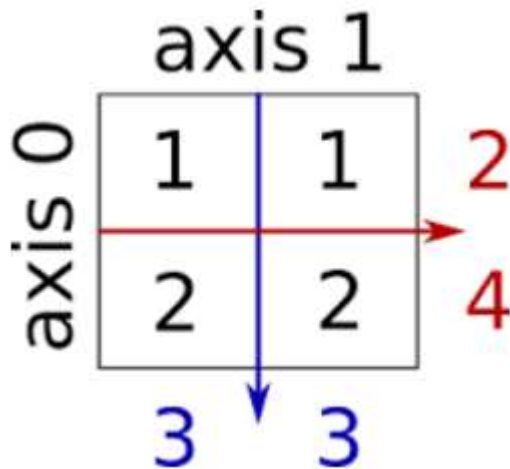
concatenate along the first axis
ax0 = np.concatenate([grid, grid], axis=0)
print(ax0)

concatenate along the second axis (zero-indexed)
ax1 = np.concatenate([grid, grid], axis=1)
print(ax1)
```

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

# vstack/hstack

배열을 행 또는 열 축으로 통합 vstack이면 행이 추가, hstack이면 열로 추가



```
import numpy as np
grid = np.array([[1, 2, 3],[4, 5, 6]])
```

```
vertically stack the arrays
av = np.vstack([grid, grid])
print(av)
```

```
horizontally stack the arrays
```

```
ah = np.hstack([grid, grid])
print(ah)
```

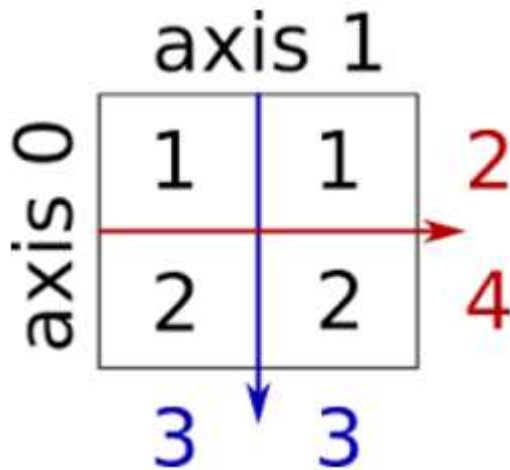
```
[[1 2 3]
 [4 5 6]
 [1 2 3]
 [4 5 6]]
[[1 2 3 1 2 3]
 [4 5 6 4 5 6]]
```



배열 행 / 열 축으로 분리

# split

배열을 행 또는 열 축으로 통합  $\text{axis}=0$ 이면 행 분리,  $\text{axis}=1$ 이면 열 분리



```
import numpy as np
```

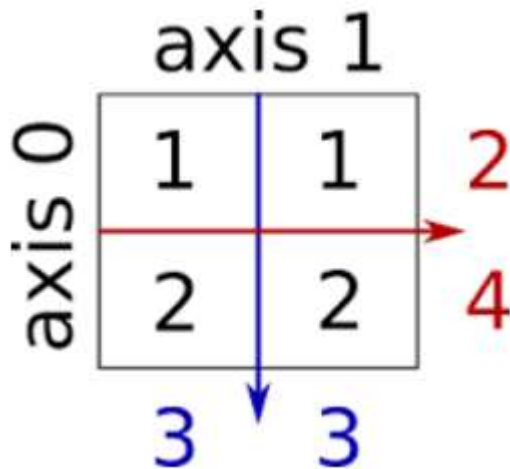
```
x = [1, 2, 3, 99, 99, 3, 2, 1]
x1, x2, x3 = np.split(x, [3,5])
print(x1, x2, x3)
```

```
x = [[1, 2, 3, 99, 100, 88, 99], [99, 3, 2, 1, 100, 88, 99]]
x1, x2, x3, x4 = np.split(x, [2,5,7], axis=1)
print(x1)
print(x2)
print(x3)
print(x4)
```

```
[1 2 3] [99 99] [3 2 1]
[[1 2]
 [99 3]]
[[3 99 100]
 [2 1 100]]
[[88 99]
 [88 99]]
[]
```

# vsplit/hsplit

배열을 행 또는 열 축으로 통합 vsplit이면 행 분리, hsplit이면 열로 분리



```
import numpy as np

x = [1, 2, 3, 99, 99, 3, 2, 1]

xa = np.array(x).reshape(4,2)
upper, lower = np.vsplit(xa, [2])
print(upper)
print(lower)

left, right = np.hsplit(xa , [1])
print(left)
print(right)
```

```
[[1 2]
 [3 99]
 [99 3]
 [2 1]]

[[1]
 [3]
 [99]
 [2]]

[[2]
 [99]
 [3]
 [1]]
```



# MATPLOTLIB

## 처리



`%matplotlib inline`

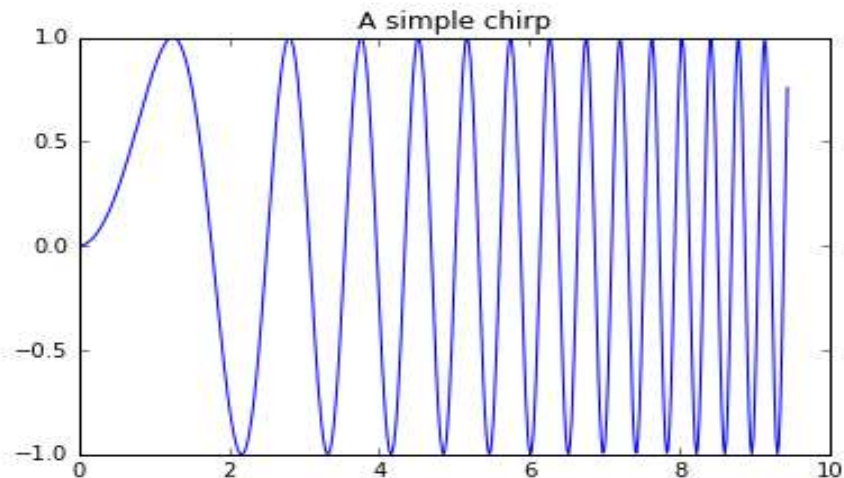
# Matplotlib 사용

%matplotlib inline 을 실행한 후에 코딩해서 실행하면 matplotlib이 처리된 결과가 출력 됨

```
In [18]: %matplotlib inline
```

```
In [19]: import matplotlib.pyplot as plt
import numpy as np

x = np.linspace(0, 3*np.pi, 500)
plt.plot(x, np.sin(x**2))
plt.title('A simple chirp');
```





%matplotlib notebook

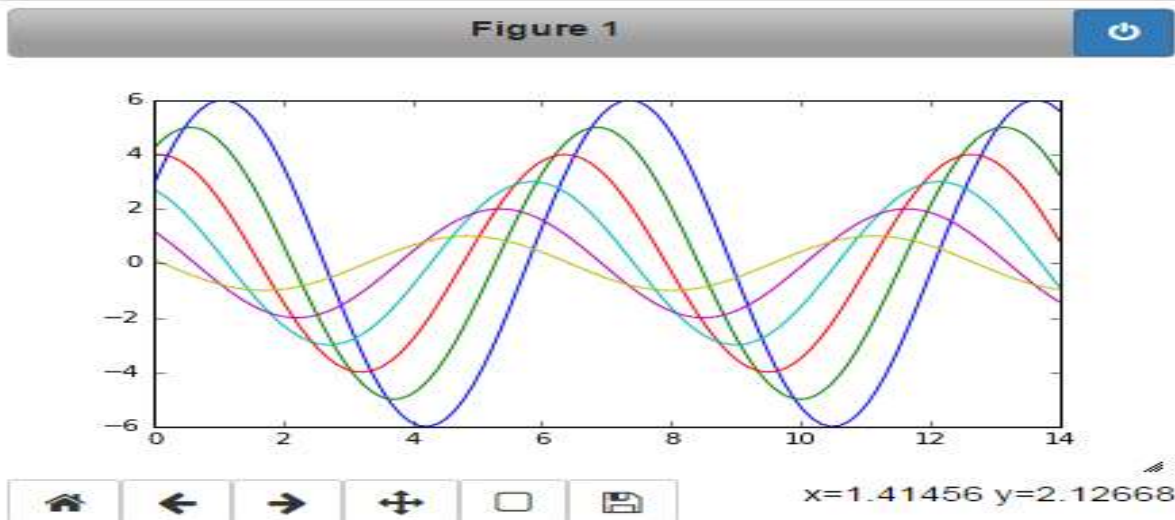
# %matplotlib notebook 사용

%matplotlib notebook 은 가상환경에서 qt창을 보기 위해 지정하면 inline에 표시

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

```
: def sinplot(flip=1):
 x = np.linspace(0, 14, 100)
 for i in range(1, 7):
 plt.plot(x, np.sin(x + i * .5) * (7 - i) * flip)
```

```
: %matplotlib notebook
sinplot()
```



# PANDAS

# 처리

Moon Yong Joon

# pandas.read\_csv 파일 읽기

pandas.read\_csv("xxx.csv") 로 읽고 확인하기

```
data_filename = 'nyc_data_01.csv'
print data_filename
fare_filename = 'nyc_fare_01.csv'
print fare_filename
```

```
nyc_data_01.csv
nyc_fare_01.csv
```

```
import pandas as pd

data = pd.read_csv(data_filename)
fare = pd.read_csv(fare_filename)
```

```
data.head(5)
```

	medallion	hack_license	vendor
0	76942C3205E17D7E7FE5A9F709D16434	25BA06A87905667AA1FE5990E33F0E2E	VT
1	517C6B330DBB3F055D007B07512628B3	2C19FBEE1A6E05612EFE4C958C14BC7F	VT
2	ED15611F168E41B33619C83D900FE266	754AEBD7C80DA17BA1D81D89FB6F4D1D	CMT
3	B33E704CC189E80C9671230C16527BBC	6789C77E1C8DC850C450D72204702976	VT
4	BD5CC6A22D05EB2D5C8235526A2A4276	5E8F2C93B5220A922699FEB AFC2F7A54	VT

```
fare.tail(5)
```

	medallion	hack_license	vendor
72326	6C8EA4103E23404EA0E0A0808FB5FEC0	390D55D08BCD554802CD22FC3D34FEC1B	VT
72327	2DBC1E95106344420C9818DA00FEA9AC	E467C8BCF6F93D3CFB1848C124BFA43C	VT
72328	5BAA40F73352E3A1C8648ACDC5445D05	9D21F436A113D106FC8169F6AC67F126	VT
72329	4150704BFA77421C4FC9C41D3A9E2A52	D019D1DF479317FE0918D35D6D9169EC	CN
72330	7B42640E2575D1C97848B1D1CCFA7307	74809FB2CFCFCE1D5A0FE52AAFD84F46	CN

# DataFrame.describe()

data.describe()를 이용해서 통계 기본 가져오기

```
data.describe()
```

	rate_code	passenger_count	trip_time_in_secs	trip_distance	pickup_longitude	pickup_latitude	dropoff_longitude	dropoff_latitude
count	72331.000000	72331.000000	72331.000000	72331.000000	72331.000000	72331.000000	72331.000000	72331.000000
mean	1.026987	1.702023	681.408151	2.742523	-73.975341	40.750951	-73.974562	40.751526
std	0.233373	1.364674	489.732353	3.236069	0.034110	0.026638	0.031892	0.030094
min	0.000000	0.000000	0.000000	0.000000	-74.089478	40.524044	-74.099770	40.523472
25%	1.000000	1.000000	360.000000	1.000000	-73.992176	40.736893	-73.991318	40.736305
50%	1.000000	1.000000	553.000000	1.700000	-73.982010	40.753441	-73.980499	40.754227
75%	1.000000	2.000000	880.000000	3.030000	-73.968018	40.767635	-73.965538	40.768353
max	6.000000	6.000000	8401.000000	96.300000	-73.370308	40.899021	-73.370308	40.941383



# DataFrame 속성

data.column(열정보), data.index(행정보) 가져오기

```
data.columns
```

```
Index([u'medallion', u'hack_license', u'vendor_id', u'rate_code',
 u'store_and_fwd_flag', u'pickup_datetime', u'dropoff_datetime',
 u'passenger_count', u'trip_time_in_secs', u'trip_distance',
 u'pickup_longitude', u'pickup_latitude', u'dropoff_longitude',
 u'dropoff_latitude'],
 dtype='object')
```

```
data.index
```

```
RangeIndex(start=0, stop=72331, step=1)
```

# DataFrame 특정 칼럼 추출

data[ [칼럼명,칼럼명]]을 넣어서 인덱싱 처리

```
: data.head(5)
```

	medallion	hack_license	vendor_id	rate_code
0	76942C3205E17D7E7FE5A9F709D16434	25BA06A87905667AA1FE5990E33F0E2E	VTS	1
1	517C6B330DBB3F055D007B07512628B3	2C19FBEE1A6E05612EFE4C958C14BC7F	VTS	1
2	ED15611F168E41B33619C83D900FE266	754AEBD7C80DA17BA1D81D89FB6F4D1D	CMT	1
3	B33E704CC189E80C9671230C16527BBC	6789C77E1CBDC850C450D72204702976	VTS	1
4	BD5CC6A22D05EB2D5C8235526A2A4276	5E8F2C93B5220A922699FEB AFC2F7A54	VTS	1

```
: data[['pickup_datetime', 'dropoff_datetime']].head(5)
```

	pickup_datetime	dropoff_datetime
0	2013-01-01 00:00:00	2013-01-01 00:05:00
1	2013-01-01 00:05:00	2013-01-01 00:21:00
2	2013-01-01 00:05:00	2013-01-01 00:12:00
3	2013-01-01 00:06:00	2013-01-01 00:06:00
4	2013-01-01 00:06:00	2013-01-01 00:12:00

# DataFrame 특정 행 추출

data.loc[ 행번호]을 넣어서 인덱싱 처리

```
fare.tail(2)
```

	medallion	hack_license	vendor_id	pickup_datetime
<b>72329</b>	4150704BFA77421C4FC9C41D3A9E2A52	D019D1DF479317FE0918D35D6D9169EC	CMT	2013-01-31 23:59:00
<b>72330</b>	7B42640E2575D1C97848B1D1CCFA7307	74809FB2CFCFCE1D5A0FE52AAFD84F46	CMT	2013-01-31 23:59:00

```
fare.loc[0].head(3)
```

```
medallion 76942C3205E17D7E7FE5A9F709D16434
hack_license 25BA06A87905667AA1FE5990E33F0E2E
vendor_id VTS
Name: 0, dtype: object
```

```
fare.loc[0].tail(5)
```

```
surcharge 0.5
mta_tax 0.5
tip_amount 0
tolls_amount 0
total_amount 6
Name: 0, dtype: object
```



# Seaborn 사용하기

# 히스토그램 그리기

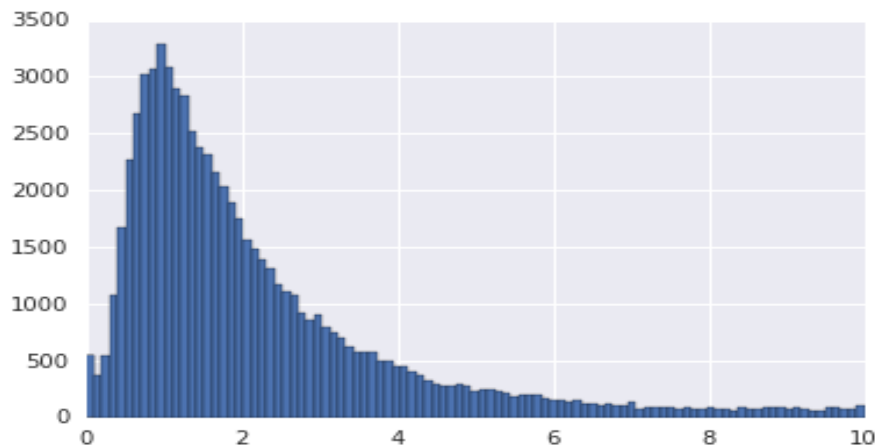
%matplotlib inline 을 실행한 후에 코딩해서 실행하면 seaborn이 처리됨

```
import seaborn as sns
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
sns.__version__
```

'0.7.1'

```
data.trip_distance.hist(bins=np.linspace(0.,10.,100))
```

<matplotlib.axes.\_subplots.AxesSubplot at 0x7fefc8c01a10>



# SEABORN

# 처리

Moon Yong Joon



Import seaborn

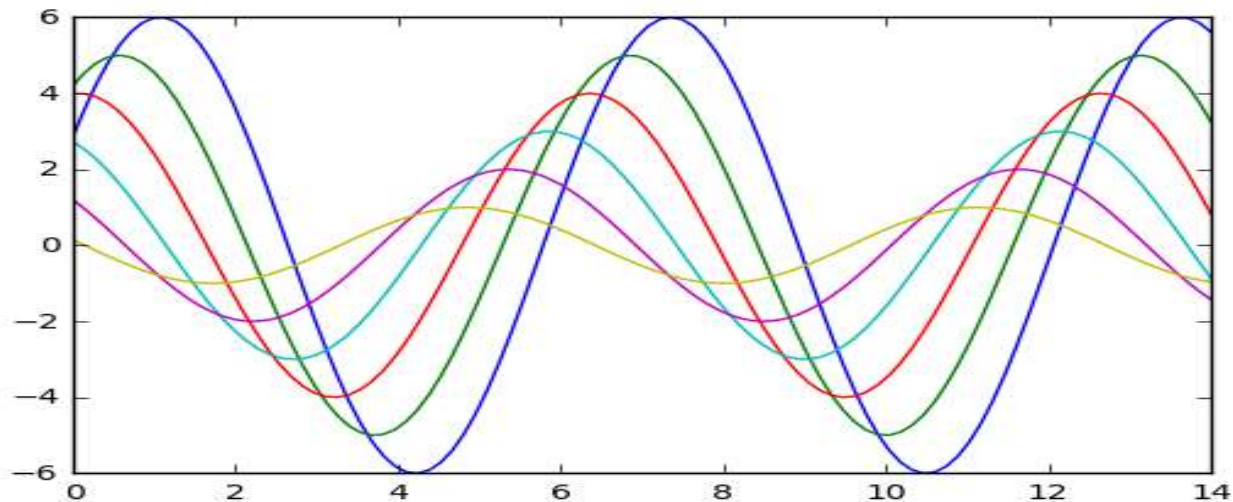
# Matplotlib 사용

## matplotlib 사용하기

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

```
def sinplot(flip=1):
 x = np.linspace(0, 14, 100)
 for i in range(1, 7):
 plt.plot(x, np.sin(x + i * .5) * (7 - i) * flip)
```

```
sinplot()
```



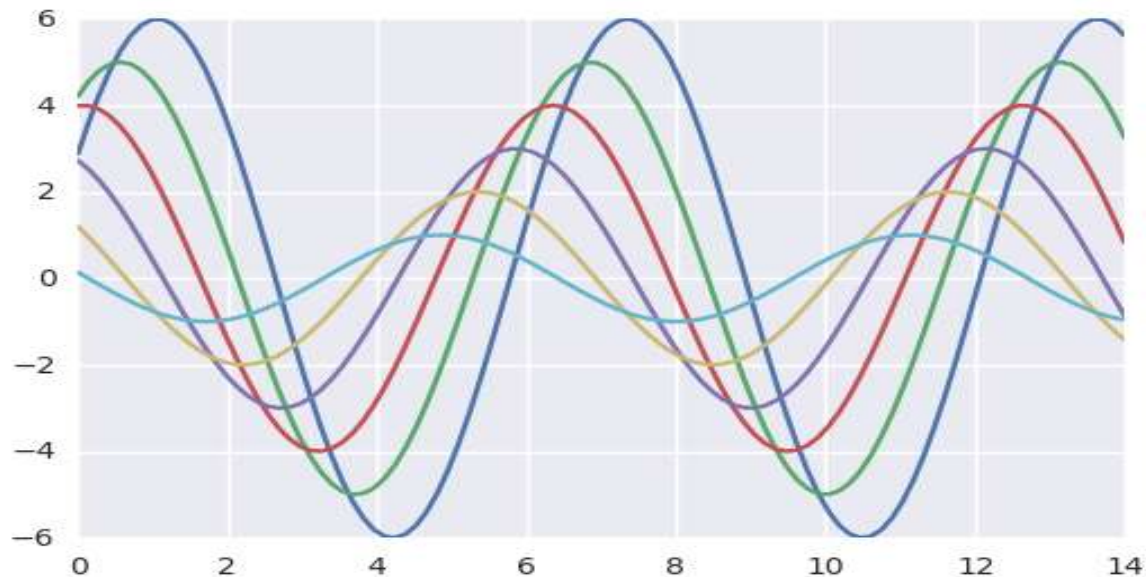


# Import seaborn

matplotlib으로 그린 것을 import seaborn만 해도 그래프가 변경

## Seaborn 그래프로 변경하기

```
import seaborn as sns
sinplot()
```





# 배경스타일 변경

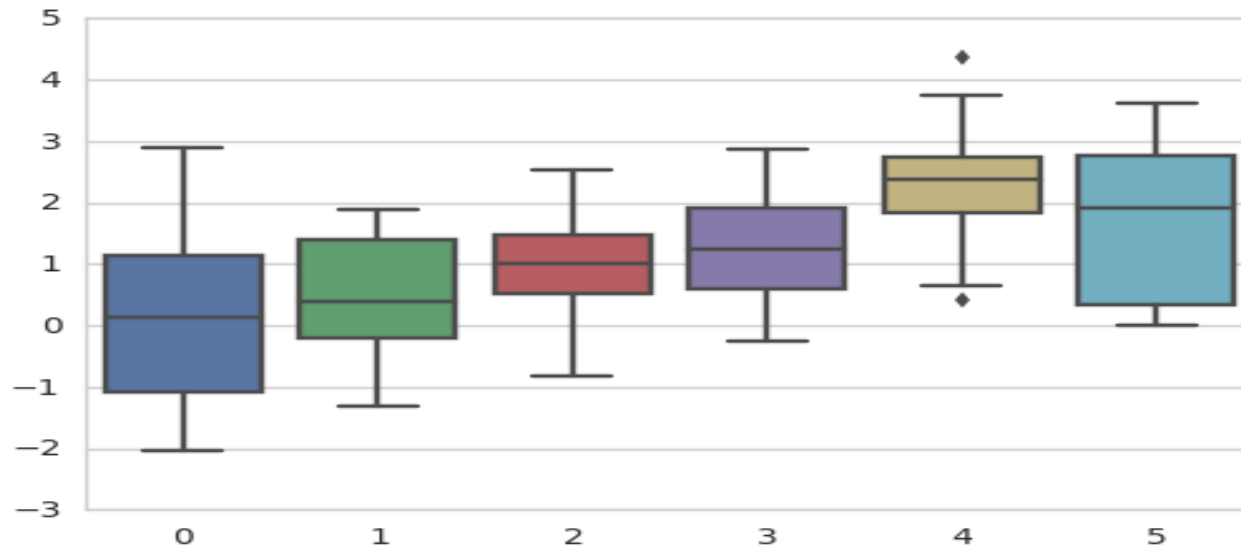
# 배경스타일: whitegrid

배경 그리드를 white와 grid로 처리

배경 스타일을 whitegrid로 세팅

```
sns.set_style("whitegrid")
data = np.random.normal(size=(20, 6)) + np.arange(6) / 2
sns.boxplot(data=data)
```

<matplotlib.axes.\_subplots.AxesSubplot at 0x7fc1f101e450>



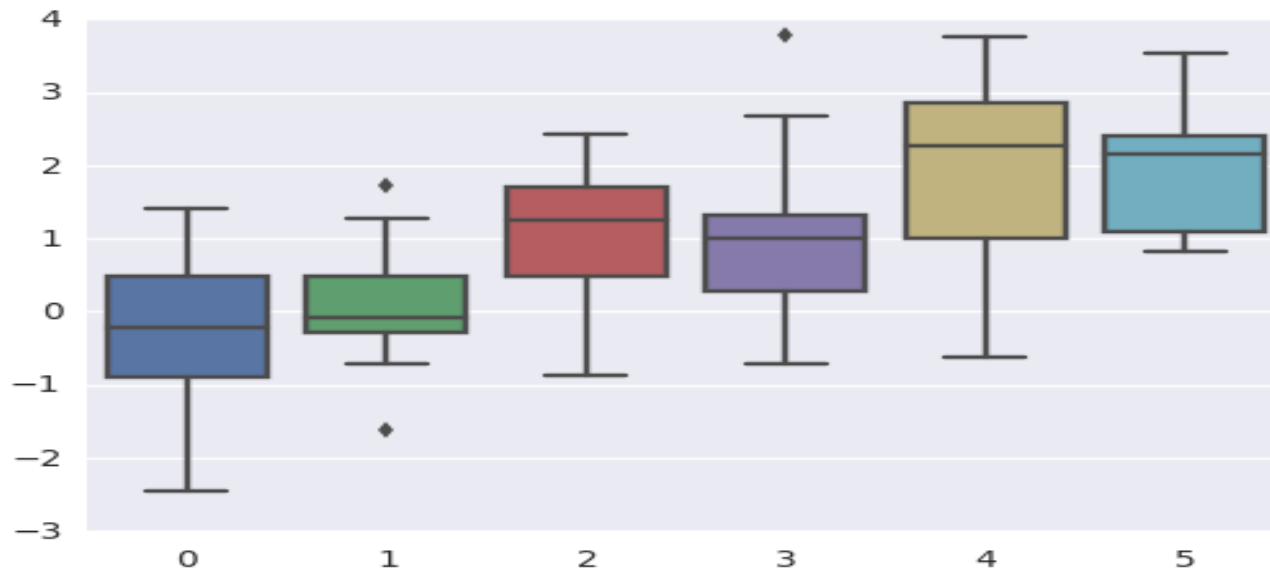
# 배경스타일: darkgrid

배경 그리드를 dark와 grid로 처리(default)

배경 스타일을 기본값인 darkgrid로 세팅 ¶

```
sns.set_style("darkgrid")
data = np.random.normal(size=(20, 6)) + np.arange(6) / 2
sns.boxplot(data=data)
```

<matplotlib.axes.\_subplots.AxesSubplot at 0x7fc1f10ff6d0>



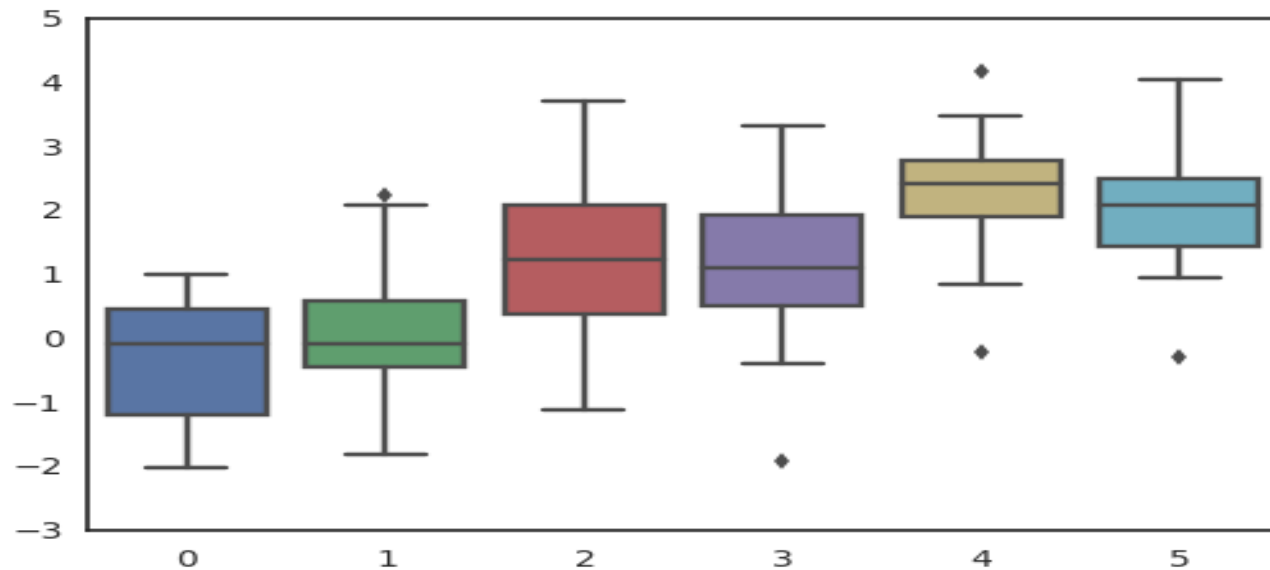
# 배경스타일: white

## 배경 그리드를 white로 처리

### 배경스타일을 white로 세팅

```
sns.set_style("white")
data = np.random.normal(size=(20, 6)) + np.arange(6) / 2
sns.boxplot(data=data)
```

<matplotlib.axes.\_subplots.AxesSubplot at 0x7fc1f0ba8d10>



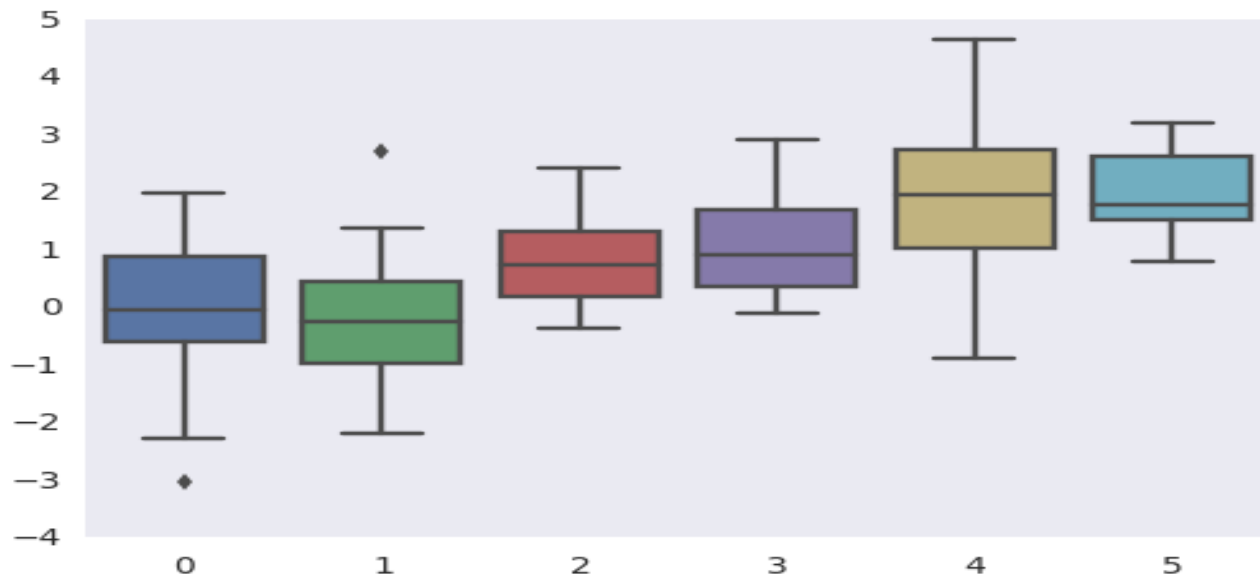
# 배경스타일: dark

## 배경 그리드를 dark로 처리

배경스타일을 dark => grid선이 빠짐 📌

```
sns.set_style("dark")
data = np.random.normal(size=(20, 6)) + np.arange(6) / 2
sns.boxplot(data=data)
```

<matplotlib.axes.\_subplots.AxesSubplot at 0x7fc1f09b44d0>





# 그래프 그리기

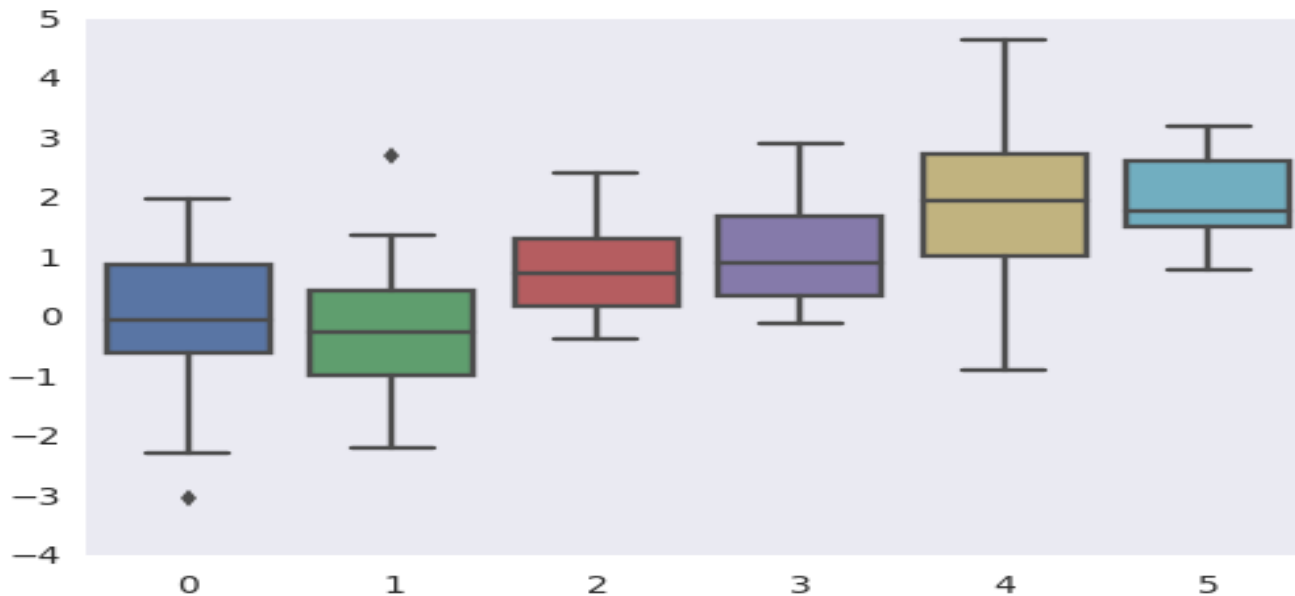
# boxplot

## box 모양으로 그리기

배경스타일을 dark => grid선이 빠짐 📏

```
sns.set_style("dark")
data = np.random.normal(size=(20, 6)) + np.arange(6) / 2
sns.boxplot(data=data)
```

<matplotlib.axes.\_subplots.AxesSubplot at 0x7fc1f09b44d0>





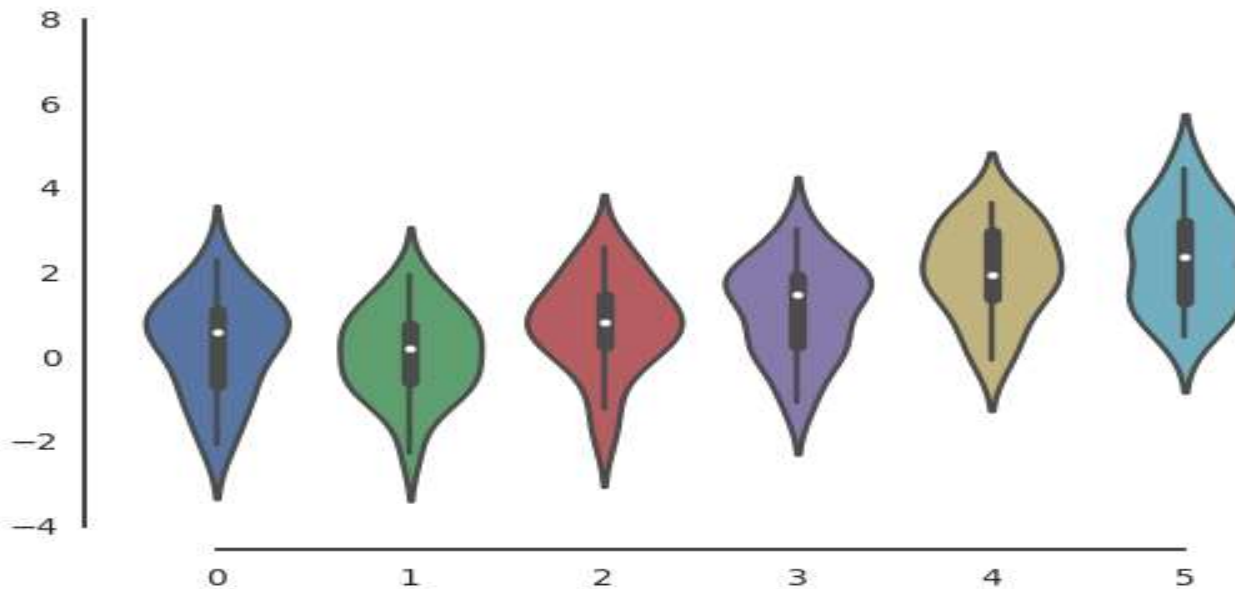
# violinplot:

## 바이올린 모양으로 도표 그리기

### violinplot 과 despine 사용하기

박스그래프를 violinplot 으로 변경하고 despine 함수를 이용해서 축선의 offset을 조정

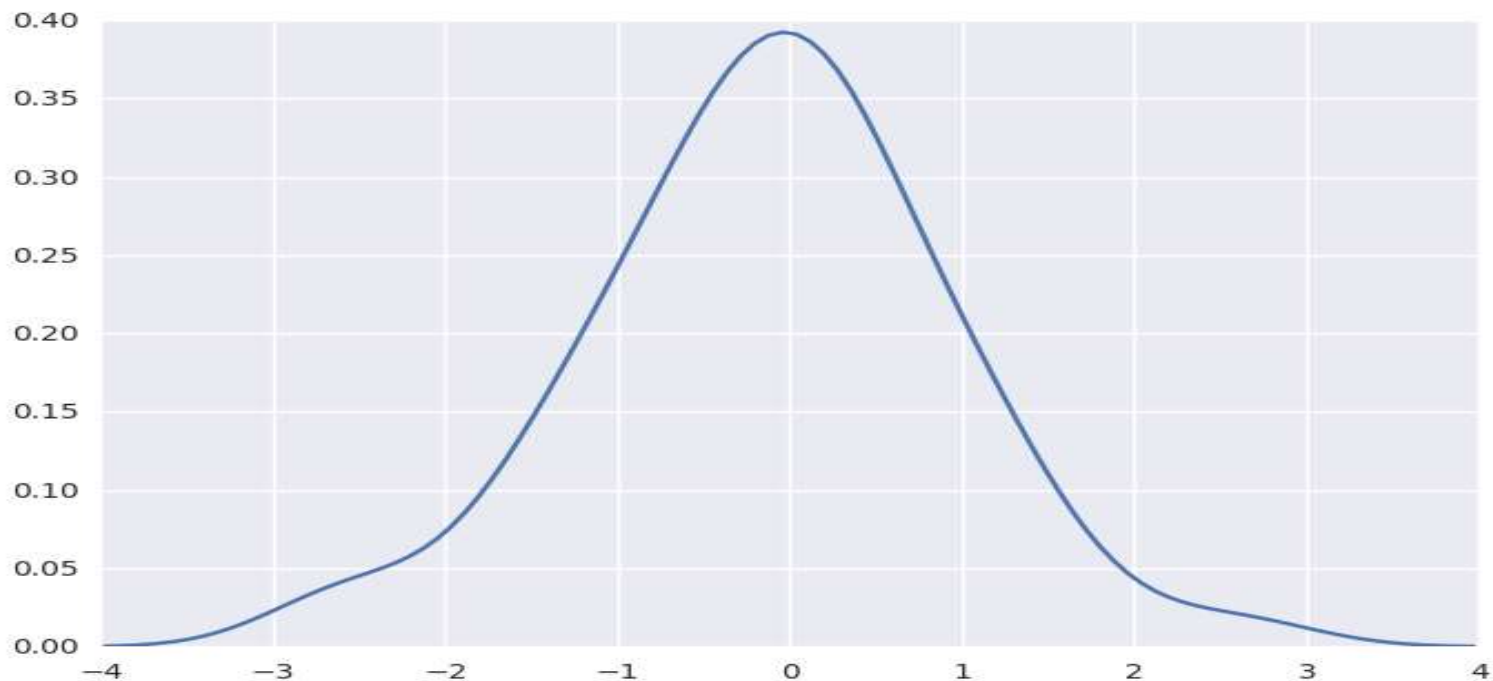
```
f, ax = plt.subplots()
sns.violinplot(data=data)
sns.despine(offset=10, trim=True);
```



# kdeplot:

## 도표 그리기

```
import numpy as np;
np.random.seed(10)
import seaborn as sns
sns.set(color_codes=True)
mean, cov = [0, 2], [(1, .5), (.5, 1)]
x, y = np.random.multivariate_normal(mean, cov, size=50).T
+ ax = sns.kdeplot(x)
```



# OS COMMAND

Moon Yong Joon

# OS 명령어로 디렉토리 만들기

!mkdir를 이용해서 디렉토리 생성

```
!mkdir facebook
```

```
%ls
```

withfile.txt	facebook/
1_hello_tensorflow.ipynb	foo.txt
2_getting_started.ipynb	foo1.txt
3_mnist_from_scratch.ipynb	image.jpg
C:\Users411\Downloads\line_plot_plus.pdf	line_plot_plus.pdf
LICENSE	mod.py
Matplotlib_test1.ipynb	mod.pyc
Mover_ch2.pyde	mod_f.py
Untitled.ipynb	mod_f.pyc
Untitled1.ipynb	newfile.txt
Untitled2.ipynb	test.txt
Untitled3.ipynb	test2.ipynb
arraystore.nd	test_1.ipynb
data.txt	understanding Python_20160815.ipynb
doctest.py	withfile.txt
doctest.pyc	yum-2.0.7.tar.gz

```
%cd facebook
```

```
/notebooks/facebook
```

# OS 명령어로 파일명 가져오기

! 입력한 후 실제 os 명령어 사용

```
files = !ls -l -s | grep .edges
```

```
files
```

```
[' 40 -rw-r--r-- 1 root root 37228 Aug 10 05:15 0.edges',
'512 -rw-r--r-- 1 root root 523802 Aug 10 05:15 107.edges',
'276 -rw-r--r-- 1 root root 280354 Aug 10 05:15 1684.edges',
'588 -rw-r--r-- 1 root root 600274 Aug 10 05:15 1912.edges',
' 96 -rw-r--r-- 1 root root 96188 Aug 10 05:15 3437.edges',
' 52 -rw-r--r-- 1 root root 51066 Aug 10 05:15 348.edges',
' 4 -rw-r--r-- 1 root root 2914 Aug 10 05:15 3980.edges',
' 28 -rw-r--r-- 1 root root 27082 Aug 10 05:15 414.edges',
' 28 -rw-r--r-- 1 root root 26496 Aug 10 05:15 686.edges',
' 8 -rw-r--r-- 1 root root 4320 Aug 10 05:15 698.edges']
```

# OS 명령어로 파일 내용 읽기

!head -line=2k를 입력하고 실행하면 파일 내의 데이터를 가져옴

```
!head --lines=2K foo.txt
```

```
Life is too short, you need python
```

```
!head --lines=2K mod.py
```

```
Tangerine = "Living reflection of a dream"
```

```
def apple():
 print "I AM APPLES!"
```

```
!head -n3 mod.py
```

```
Tangerine = "Living reflection of a dream"
```

```
def apple():
```

---

# LATEX

# 사용하기

Moon Yong Joon



$$(TeX shorthand)$



# 글자 처리하기 : 그리스 알파벳

markdown으로 지정하고 \$\$와 \$\$(\$와 \$) 사이에 문자를 입력하고 실행하면 문자가 표현

```
In []: $\alpha, \Alpha, \beta, \Beta, \gamma, \Gamma, \pi, \Pi, \phi, \varphi, \mu, \Phi$
```

$\alpha, \Alpha, \beta, \Beta, \gamma, \Gamma, \pi, \Pi, \phi, \varphi, \mu, \Phi$

# 산식 작성 : 제곱과 인덱스

산식을 표현할 때, 제곱(^)과 인덱스(\_{ }) 표시:

```
In []: $k_{n+1} = n^2 + k_n^2 - k_{n-1}$
```

$$k_{n+1} = n^2 + k_n^2 - k_{n-1}$$

# 산식 작성 : 문장 표현

산식을 표현할 때, 문장과 산식을 표현하고  
inline으로 표시

```
In [179]: %%latex
The mass-energy equivalence is described by the famous equation

$$E=mc^2$$

discovered in 1905 by Albert Einstein.
In natural units ($c = 1$), the formula expresses the identity

\begin{equation}
E=m
\end{equation}
```

The mass-energy equivalence is described by the famous equation

$$E = mc^2$$

discovered in 1905 by Albert Einstein. In natural units ( $c = 1$ ), the formula expresses the identity

$$E = m$$

# 산식 처리하기 : sqrt

markdown으로 지정하고 \$\$와 \$\$(\$와 \$) 사이에 수학산식을 입력하고 실행하면 실제 수학산식이 표현됨

```
In []: $$c = \sqrt{a^2 + b^2}$$
```

$$c = \sqrt{a^2 + b^2}$$



# Latex, Math import

# 산식 처리하기 : Latex, Math

code로 지정하고 IPython.display 모듈을 이용해서 수학산식을 입력하고 실행하면 실제 수학산식이 표현됨

```
In [150]: from IPython.display import display, Math, Latex
display(Math(r'F(k) = \int_{-\infty}^{\infty} f(x) e^{2\pi i k} dx'))
```

$$F(k) = \int_{-\infty}^{\infty} f(x) e^{2\pi i k} dx$$

# 산식 처리하기 : Latex

Latex를 import하고 latex 정의를 하면 실제 산식으로 표현됨

```
In [152]: from IPython.display import Latex
Latex(r"""\begin{eqnarray}
\nabla \times \vec{\mathbf{B}} - \frac{1}{c} \frac{\partial \vec{\mathbf{E}}}{\partial t} &= \frac{4\pi}{c} \vec{\mathbf{j}} \\
\nabla \cdot \vec{\mathbf{E}} &= 4\pi \rho \\
\nabla \times \vec{\mathbf{E}} + \frac{1}{c} \frac{\partial \vec{\mathbf{B}}}{\partial t} &= \vec{\mathbf{0}} \\
\nabla \cdot \vec{\mathbf{B}} &= 0
\end{eqnarray}""")
```

```
Out[152]:
```

$$\left. \begin{aligned} \nabla \times \vec{\mathbf{B}} - \frac{1}{c} \frac{\partial \vec{\mathbf{E}}}{\partial t} &= \frac{4\pi}{c} \vec{\mathbf{j}} \\ \nabla \cdot \vec{\mathbf{E}} &= 4\pi \rho \\ \nabla \times \vec{\mathbf{E}} + \frac{1}{c} \frac{\partial \vec{\mathbf{B}}}{\partial t} &= \vec{\mathbf{0}} \\ \nabla \cdot \vec{\mathbf{B}} &= 0 \end{aligned} \right|$$



%%latex 이용



# 산식 예시 1

## 산식들을 표현하는 예시

```
In [205]: %%latex
\[\int\limits_0^1 x^2 + y^2 dx \]
```

$$\int_0^1 x^2 + y^2 dx$$

```
In [207]: %%latex
\[a_{n_i} \]
```

$$a_{n_i}$$

```
In [209]: %%latex
\[\int_{i=1}^n \]
```

$$\int_{i=1}^n$$

```
In [211]: %%latex
\[\sum_{i=1}^{\infty} \]
```

$$\sum_{i=1}^{\infty}$$

# 산식 예시 2

## 산식들을 표현하는 예시

```
In [212]: %%latex
 \[\prod_{i=1}^n \]
```

$$\prod_{i=1}^n$$

```
In [213]: %%latex
 \[\cup_{i=1}^n \]
```

$$\cup_{i=1}^n$$

```
In [214]: %%latex
 \[\cap_{i=1}^n \]
```

$$\cap_{i=1}^n$$

# 산식 예시 3

## 산식들을 표현하는 예시

```
In [215]: %%latex
 \[\oint_{i=1}^n\]
```

$$\oint_{i=1}^n$$

```
In [216]: %%latex
 \[\coprod_{i=1}^n\]
```

$$\coprod_{i=1}^n$$

# 산식 작성 : 여러 라인 처리

여러 라인 처리는 `\begin{name}`, `\end{name}` 안에 여러 라인의 산식을 표현

```
In [176]: %%latex
\begin{equation}
x = a_0 + \cfrac{1}{a_1 + \cfrac{1}{a_2 + \cfrac{1}{a_3 + \cfrac{1}{a_4}}}}
\end{equation}
```

$$x = a_0 + \frac{1}{a_1 + \frac{1}{a_2 + \frac{1}{a_3 + \frac{1}{a_4}}}}$$

# 여러 라인 산식 작성 흐름

%%latex로 지정하고 latex 정의를 하면 실제 산식으로 표현됨

```
In [163]: %%latex
 \begin{align}
 \end{align}
```

Latex 시작과 끝

```
In [164]: %%latex
 \begin{align}
 \mathbf{\vec A}
 \end{align}
```

$\vec{A}$

```
In [165]: %%latex
 \begin{align}
 \mathbf{\vec A \cdot \vec B}
 \end{align}
```

$\vec{A} \cdot \vec{B}$

```
In [167]: %%latex
 \begin{align}
 \mathbf{\vec A \cdot \vec B = \sum_{i=1}^n A_i B_i}
 \end{align}
```

$\vec{A} \cdot \vec{B} = \sum_{i=1}^n A_i B_i$

수학 산식 추가

# 산식 처리 예시 : 단일 라인

%%latex로 지정하고 latex 정의(\[ \])를 하면 실제 산식을 한 라인으로 표현됨

```
In [177]: %%latex
 \[x^n + y^n = z^n \]
```

$$x^n + y^n = z^n$$

# 산식 처리 예시 : 여러 라인 1

%%latex로 지정하고 latex 정의(\begin \end)를 하면 실제 여러줄 산식으로 표현됨

```
In [151]: %%latex
\begin{align}
\nabla \times \vec{\mathbf{B}} - \frac{1}{c} \frac{\partial \vec{\mathbf{E}}}{\partial t} &= \frac{4\pi}{c} \vec{\mathbf{j}} \\
\nabla \cdot \vec{\mathbf{E}} &= 4\pi \rho \\
\nabla \times \vec{\mathbf{E}} + \frac{1}{c} \frac{\partial \vec{\mathbf{B}}}{\partial t} &= \vec{\mathbf{0}} \\
\nabla \cdot \vec{\mathbf{B}} &= 0
\end{align}
```

$$\left. \begin{aligned} \nabla \times \vec{\mathbf{B}} - \frac{1}{c} \frac{\partial \vec{\mathbf{E}}}{\partial t} &= \frac{4\pi}{c} \vec{\mathbf{j}} \\ \nabla \cdot \vec{\mathbf{E}} &= 4\pi \rho \\ \nabla \times \vec{\mathbf{E}} + \frac{1}{c} \frac{\partial \vec{\mathbf{B}}}{\partial t} &= \vec{\mathbf{0}} \\ \nabla \cdot \vec{\mathbf{B}} &= 0 \end{aligned} \right|$$

# 산식 처리 예시 : 여러 라인 2

%%latex로 지정하고 latex 정의를 하면 실제 산식으로 표현됨

```
In [170]: %%latex
\begin{align}
\dot{x} &= \sigma(y-x) \\\
\dot{z} &= -\beta z + xy \\\
\end{align}
```

$$\begin{aligned} \dot{x} &= \sigma(y-x) \\ \dot{z} &= -\beta z + xy \end{aligned}$$



# 산식 처리 예시 : 여러 라인 3

%%latex로 지정하고 latex 정의를 하면 실제 산식으로 표현됨

```
In [172]: %%latex
\begin{equation}
\int f(x) \, dx = F(x)
\label{eq1}
\tag{eq1text}
\end{equation}
```

$$\int f(x) \, dx = F(x)$$

(eq1text)



# Latex 기호

# 그리스 알파벳

## 문자를 표시할 때 사용

### Greek letters

$\alpha A$	<code>\alpha A</code>	$\nu N$	<code>\nu N</code>
$\beta B$	<code>\beta B</code>	$\xi \Xi$	<code>\xi \Xi</code>
$\gamma \Gamma$	<code>\gamma \Gamma</code>	$o O$	<code>o O</code>
$\delta \Delta$	<code>\delta \Delta</code>	$\pi \Pi$	<code>\pi \Pi</code>
$\epsilon \epsilon E$	<code>\epsilon \epsilon E</code>	$\rho \varrho P$	<code>\rho \varrho P</code>
$\zeta Z$	<code>\zeta Z</code>	$\sigma \Sigma$	<code>\sigma \Sigma</code>
$\eta H$	<code>\eta H</code>	$\tau T$	<code>\tau T</code>
$\theta \vartheta \Theta$	<code>\theta \vartheta \Theta</code>	$\upsilon \Upsilon$	<code>\upsilon \Upsilon</code>
$\iota I$	<code>\iota I</code>	$\phi \varphi \Phi$	<code>\phi \varphi \Phi</code>
$\kappa K$	<code>\kappa K</code>	$\chi X$	<code>\chi X</code>
$\lambda \Lambda$	<code>\lambda \Lambda</code>	$\psi \Psi$	<code>\psi \Psi</code>
$\mu M$	<code>\mu M</code>	$\omega \Omega$	<code>\omega \Omega</code>

# Miscellaneous symbols

기호를 표시할 때 사용

## Miscellaneous symbols

$\infty$	<code>\infty</code>	$\forall$	<code>\forall</code>
$\Re$	<code>\Re</code>	$\Im$	<code>\Im</code>
$\nabla$	<code>\nabla</code>	$\exists$	<code>\exists</code>
$\partial$	<code>\partial</code>	$\nexists$	<code>\nexists</code>
$\emptyset$	<code>\emptyset</code>	$\varnothing$	<code>\varnothing</code>
$\wp$	<code>\wp</code>	$\complement$	<code>\complement</code>
$\neg$	<code>\neg</code>	$\cdots$	<code>\cdots</code>
$\square$	<code>\square</code>	$\sqrt{\phantom{x}}$	<code>\sqrt{\phantom{x}}</code>
$\blacksquare$	<code>\blacksquare</code>	$\triangle$	<code>\triangle</code>

# Binary Op/Relation Symbols

기호를 표시할 때 사용

## Binary Operation/Relation Symbols

$\times$	<code>\times</code>	$\otimes$	<code>\otimes</code>
$\div$	<code>\div</code>	$\cap$	<code>\cap</code>
$\cup$	<code>\cup</code>	$\neq$	<code>\neq</code>
$\leq$	<code>\leq</code>	$\geq$	<code>\geq</code>
$\in$	<code>\in</code>	$\perp$	<code>\perp</code>
$\notin$	<code>\notin</code>	$\subset$	<code>\subset</code>
$\simeq$	<code>\simeq</code>	$\approx$	<code>\approx</code>
$\wedge$	<code>\wedge</code>	$\vee$	<code>\vee</code>
$\oplus$	<code>\oplus</code>	$\otimes$	<code>\otimes</code>
$\Box$	<code>\Box</code>	$\boxtimes$	<code>\boxtimes</code>
$\equiv$	<code>\equiv</code>	$\cong$	<code>\cong</code>

# Brackets and Parentheses

## 기호를 표시할 때 사용

L <sup>A</sup> T <sub>E</sub> X markup	Renders as
<code>\big( \Big( \bigg( \Bigg(</code>	$(((($
<code>\big) \Big) \bigg) \Bigg)</code>	$))]]$
<code>\big\{ \Big\{ \bigg\{ \Bigg\{</code>	${{{{$
<code>\big \langle \Big \langle \bigg \langle \Bigg \langle</code>	$\langle\langle\langle\langle$
<code>\big \rangle \Big \rangle \bigg \rangle \Bigg \rangle</code>	$\rangle\rangle\rangle\rangle$

# spacing commands

## 기호를 표시할 때 사용

L <sup>A</sup> T <sub>E</sub> X code	Description
<code>\quad</code>	space equal to the current font size (= 18 <b>mu</b> )
<code>\,</code>	3/18 of <code>\quad</code> (= 3 mu)
<code>\:</code>	4/18 of <code>\quad</code> (= 4 mu)
<code>\;</code>	5/18 of <code>\quad</code> (= 5 mu)
<code>\!</code>	-3/18 of <code>\quad</code> (= -3 mu)
<code>\</code> (space after backslash!)	equivalent of space in normal text
<code>\qquad</code>	twice of <code>\quad</code> (= 36 mu)

# text alignment

기호를 표시할 때 사용

Alignment	Environment	Switch command	ragged2e environment	ragged2e switch command
Left	<code>flushleft</code>	<code>\raggedright</code>	<code>FlushLeft</code>	<code>\RaggedRight</code>
Right	<code>flushright</code>	<code>\raggedleft</code>	<code>FlushRight</code>	<code>\RaggedLeft</code>
Centre	<code>center</code>	<code>\centering</code>	<code>Center</code>	<code>\Centering</code>
Fully justified			<code>justify</code>	<code>\justify</code>



# 화살표

## 기호를 표시할 때 사용

### Arrows

$\leftarrow$	<code>\leftarrow</code>	$\Lleftarrow$	<code>\Leftarrow</code>
$\rightarrow$	<code>\rightarrow</code>	$\Rrightarrow$	<code>\Rightarrow</code>
$\leftrightarrow$	<code>\leftrightarrow</code>	$\rightleftharpoons$	<code>\rightleftharpoons</code>
$\uparrow$	<code>\uparrow</code>	$\downarrow$	<code>\downarrow</code>
$\Uparrow$	<code>\Uparrow</code>	$\Downarrow$	<code>\Downarrow</code>
$\Leftrightarrow$	<code>\Leftrightarrow</code>	$\Updownarrow$	<code>\Updownarrow</code>
$\mapsto$	<code>\mapsto</code>	$\longmapsto$	<code>\longmapsto</code>
$\nearrow$	<code>\nearrow</code>	$\searrow$	<code>\searrow</code>
$\swarrow$	<code>\swarrow</code>	$\nwarrow$	<code>\nwarrow</code>
$\leftharpoonup$	<code>\leftharpoonup</code>	$\rightharpoonup$	<code>\rightharpoonup</code>
$\leftharpoondown$	<code>\leftharpoondown</code>	$\rightharpoondown$	<code>\rightharpoondown</code>
$\rightleftharpoons$	<code>\rightleftharpoons</code>		