

# 데이터 시각화 (2024)

데이터과학부 정진명

([jmjung@suwon.ac.kr](mailto:jmjung@suwon.ac.kr), 글로벌경상관 918호)

## 2 주차

Anaconda  
(python interpreter + packages)

# Anaconda

❖ **Anaconda:** python interpreter + 다양한 패키지

Anaconda

Python interpreter

+

450 여개의 package 보유

(데이터과학에서 주로 사용하는 아래의 6개 package 포함)

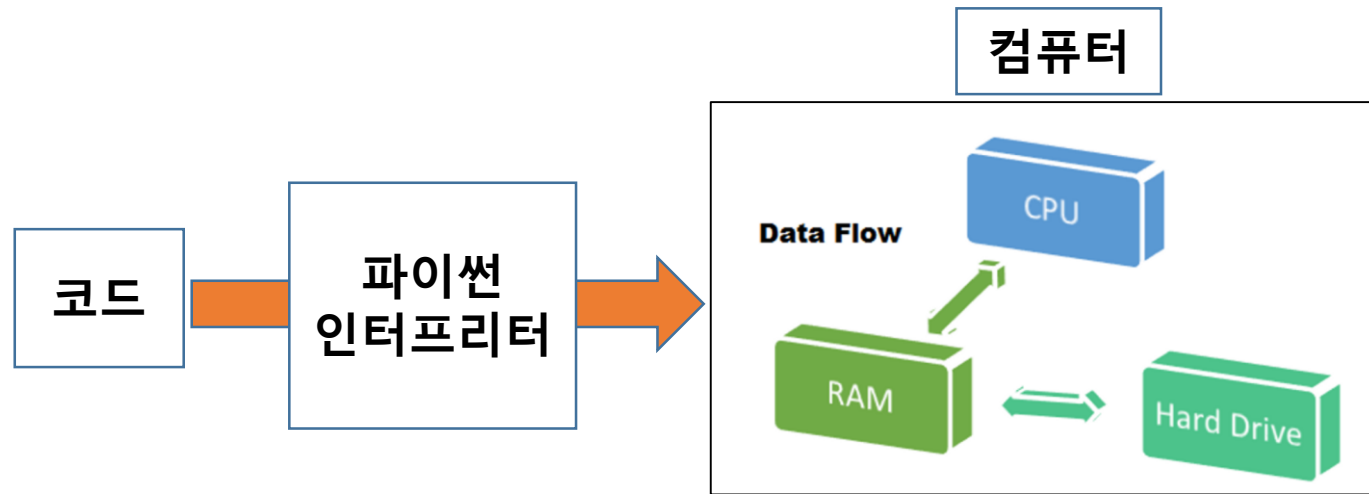
name	Summary / License
<a href="#">matplotlib</a>	Publication quality figures in Python / PSF-based
<a href="#">numpy</a>	Array processing for numbers, strings, records, and objects. / BSD 3-Clause
<a href="#">pandas</a>	High-performance, easy-to-use data structures and data analysis tools. / BSD 3-clause
<a href="#">scikit-learn</a>	A set of python modules for machine learning and data mining / BSD 3-Clause
<a href="#">scipy</a>	Scientific Library for Python / BSD 3-Clause
<a href="#">tensorflow</a>	TensorFlow is a machine learning library. / Apache 2.0

[jupyter notebook](#)

← 수업에 사용할 개발환경

# python interpreter

- **python interpreter:**
  - python 코드를 번역해서 실행시켜주는 **python 실행 프로그램**
  - 또는 python 각 명령어에 대한 **실행 방침**이 적혀있는 **문서**
- 작성한 코드는 **python interpreter**에 있는 **실행 방침**에 따라,  
컴퓨터의 **메모리(ram)**에 여러 **변수**들을 만들고 그들을 업데이트함
- python 코드의 문법이 업데이트 됨에 따라, **버전**이 증가



출처: <https://www.cloudsis.com/post/2018/09/12/cpu-ram-and-hard-drive-e2-80-93-what-e2-80-99s-the-difference>

# 패키지 (package)

- 특정 기능을 수행하는 **모듈(함수)**들의 묶음
- 분야 별로 나누어져 있음 (통계, 머신러닝, 시각화, ...)
- 패키지 함수 예제

```
import numpy  
numpy.var(l1)
```

3.44

**numpy Package에 정의된 함수 사용하는 경우**

```
l1=[2,3,4,6,7]  
m1=sum(l1)/len(l1)  
print(m1)  
  
l2=[(ii-m1)*(ii-m1) for ii in l1]  
var=sum(l2)/len(l1)  
print(var)
```

4.4

3.44

**직접 코딩하는 경우**

- 같은 결과 도출
- package 함수 사용이 훨씬 간단
- 패키지에 있는 함수들의 parameter와 return 값을 이해하고, 잘 사용하는 것도 중요한 프로그래밍 능력

# 패키지 (package) 예제: scipy ttest

- `scipy.stats` package에 구현되어 있는 `ttest_ind` 함수

## `scipy.stats.ttest_ind`

`scipy.stats.ttest_ind(a, b, axis=0, equal_var=True, nan_policy='propagate')`

[\[source\]](#)

Calculate the T-test for the means of *two independent* samples of scores.

This is a two-sided test for the null hypothesis that 2 independent samples have identical average (expected) values. This test assumes that the populations have identical variances by default.

**Parameters:** `a, b` : *array\_like*

The arrays must have the same shape, except in the dimension corresponding to *axis* (the first, by default).

`axis` : *int or None, optional*

Axis along which to compute test. If None, compute over the whole arrays, *a*, and *b*.

`equal_var` : *bool, optional*

If True (default), perform a standard independent 2 sample test that assumes equal population variances [1]. If False, perform Welch's t-test, which does not assume equal population variance [2].

*New in version 0.11.0.*

`nan_policy` : *{'propagate', 'raise', 'omit'}, optional*

Defines how to handle when input contains nan. The following options are available (default is 'propagate'):

- 'propagate': returns nan
- 'raise': throws an error
- 'omit': performs the calculations ignoring nan values

**Returns:** `statistic` : *float or array*

The calculated t-statistic.

`pvalue` : *float or array*

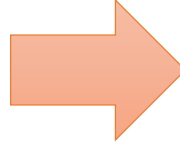
The two-tailed p-value.

# 패키지 (package) 버전

- package별로 구성 함수에 따라 **버전**이 다름

package A (v 1.1.1)

- func1
- func2
- func3



package A (v 1.3.2)

- func1
- func2 (수정)
- func3
- func4

**version a.b.c**

a 증가: 큰 수정  
b 증가: 중간 수정  
c 증가: 작은 수정



# 설치된 interpreter & package 버전 확인

anaconda prompt 실행 → conda 환경

```
(base) C:\Users\Wjmjun>python --version  
Python 3.11.7
```

```
(base) C:\Users\Wjmjun>conda list pandas  
# packages in environment at C:\Users\Wjmjun\anaconda3:  
#  
# Name            Version           Build    Channel  
pandas            2.1.4             py311hf62ec03_0
```

- interpreter 버전 확인  
→ python --version

- package 버전 확인 (package: pandas 일 경우)  
→ conda list pandas

# 새로운 package 설치

anaconda prompt 실행 → conda 환경

```
(base) C:\Users\Wjmjun>pip install folium
Collecting folium
  Downloading folium-0.16.0-py2.py3-none-any.whl.metadata (3.6 kB)
Collecting branca>=0.6.0 (from folium)
  Downloading branca-0.7.1-py3-none-any.whl.metadata (1.5 kB)
Requirement already satisfied: jinja2>=2.9 in c:\Users\Wjmjun\Wanaconda3\lib\site-packages (from folium)
Requirement already satisfied: numpy in c:\Users\Wjmjun\Wanaconda3\lib\site-packages (from folium)
Requirement already satisfied: requests in c:\Users\Wjmjun\Wanaconda3\lib\site-packages (from folium)
Requirement already satisfied: xyzservices in c:\Users\Wjmjun\Wanaconda3\lib\site-packages (from folium)
Requirement already satisfied: MarkupSafe>=2.0 in c:\Users\Wjmjun\Wanaconda3\lib\site-packages (from jinja2->folium)
Requirement already satisfied: charset-normalizer<4,>=2 in c:\Users\Wjmjun\Wanaconda3\lib\site-packages (from requests->folium)
Requirement already satisfied: idna<4,>=2.5 in c:\Users\Wjmjun\Wanaconda3\lib\site-packages (from requests->folium)
Requirement already satisfied: urllib3<3,>=1.21.1 in c:\Users\Wjmjun\Wanaconda3\lib\site-packages (from requests->folium)
Requirement already satisfied: certifi>=2017.4.17 in c:\Users\Wjmjun\Wanaconda3\lib\site-packages (from requests->folium)
Downloading folium-0.16.0-py2.py3-none-any.whl (100 kB)
----- 100.0/100.0 kB 823.6 kB/s
Downloading branca-0.7.1-py3-none-any.whl (25 kB)
Installing collected packages: branca, folium
Successfully installed branca-0.7.1 folium-0.16.0

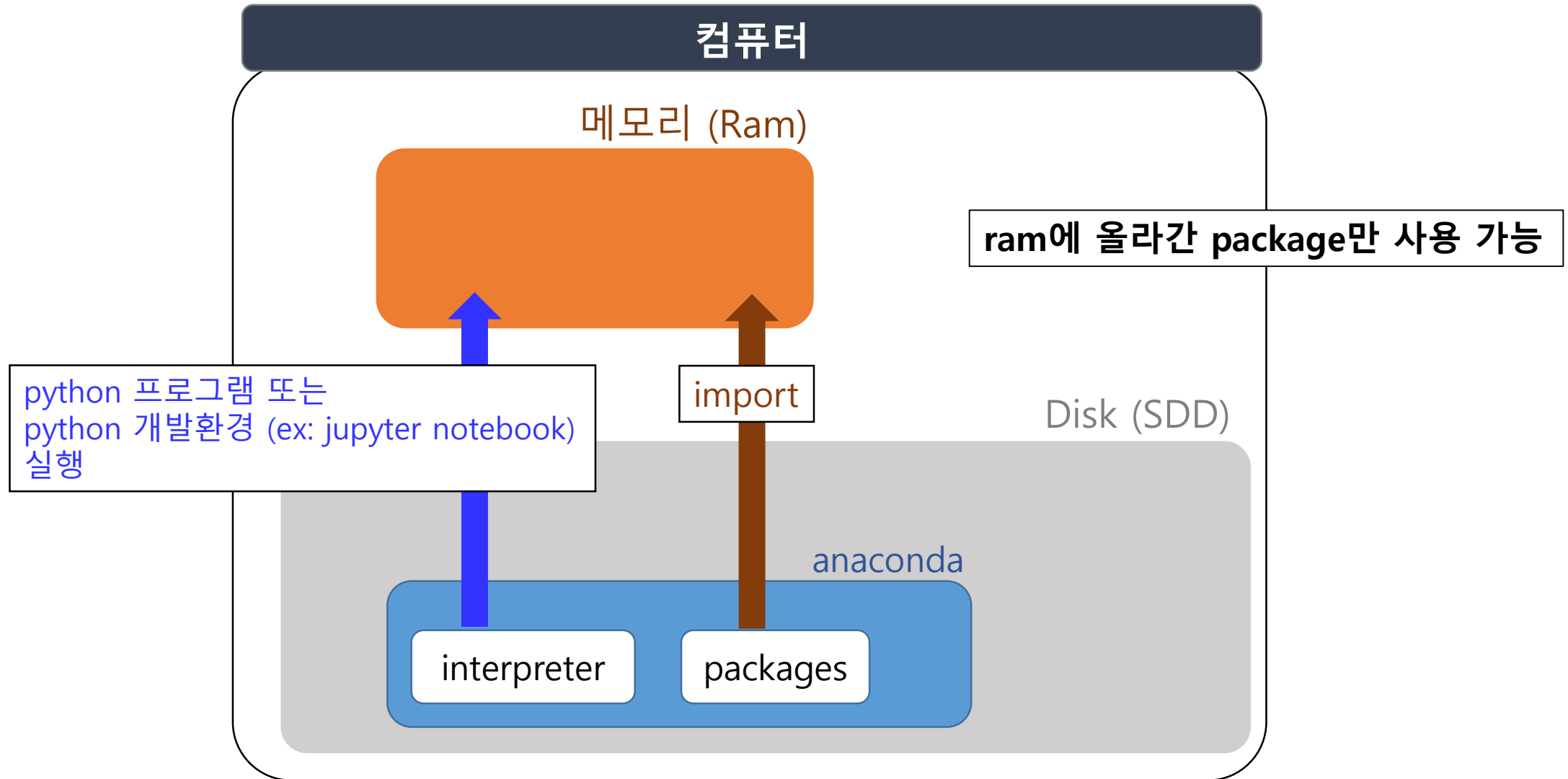
(base) C:\Users\Wjmjun>conda list folium
# packages in environment at C:\Users\Wjmjun\Wanaconda3:
#
# Name                    Version                   Build  Channel
folium                    0.16.0                   pypi_0    pypi

(base) C:\Users\Wjmjun>
```

package: folium 일 경우.

- pip install folium (최신 버전)
- pip install folium==0.15.1 (특정 버전)
- pip install folium --upgrade (최신버전으로 upgrade)

# import package



# Jupyter notebook

# Jupyter notebook

interaction  
mode

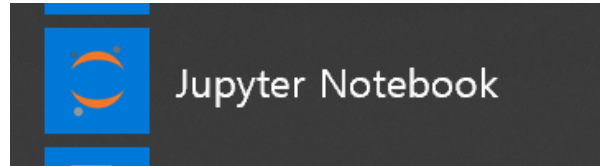
web  
mode



- IPython(<http://ipython.org>) 커널을 기반으로 한 대화형 파이썬 셸
- 일반적인 터미널 셸 + 웹 기반 데이터 분석 Notebook 제공
- 미디어, 텍스트, 코드, 수식 등을 하나의 문서로 표현 가능
- 사실상의 데이터 분석 Interactive Shell의 표준

# Jupyter notebook 실행방법

방법1: Jupyter Notebook 아이콘 클릭



방법2: anaconda prompt에서 jupyter notebook 입력 후 실행

◆ 이 경우 anaconda prompt에서 작업할 폴더(ipynb파일이 있는폴더)로 이동한 후에 jupyter notebook 실행하면 편리함.

1. 작업할 폴더로 이동: `cd` (change directory) 명령 사용 (window 탐색기에서 해당 폴더로 이동 후 주소복사)
2. jupyter notebook 입력

```
(base) C:\Users\Wjmjung>cd C:\Users\Wjmjung\Dropbox\Ongoing_lecture\2020_1
```

```
(base) C:\Users\Wjmjung\Dropbox\Ongoing_lecture\2020_1>jupyter notebook_
```

# Jupyter notebook ipynb 파일 열기

The screenshot shows the JupyterLab interface. At the top, there's a header with the Jupyter logo and 'Quit' and 'Logout' buttons. Below the header, there are tabs for 'Files', 'Running', 'Clusters', and 'Nbextensions'. The 'Files' tab is active, showing a file browser. The current path is '/ 강의자료\_backup / 4w'. The file list includes a folder 'data' and several files: '4w\_only\_전국아파트분양가분석.ipynb', '4w\_only\_전국아파트분양가분석\_html.html', '4w\_only\_전국아파트분양가분석\_결과삭제\_html.html', '4w\_p1.mp4', '4w\_p2.mp4', and '4w\_p3.mp4'. The file '4w\_only\_전국아파트분양가분석.ipynb' is highlighted with a blue box and labeled '클릭'. To the right, the 'New' button is highlighted with a red box and labeled '클릭'. A dropdown menu is open, showing options: 'Notebook: Python 3', 'Other: Text File', 'Folder', and 'Terminal'. The 'Python 3' option is highlighted with a red box and labeled '클릭'.

Quit Logout

Files Running Clusters Nbextensions

Select items to perform actions on them.

Upload New ↕

0 ▾ / 강의자료\_backup / 4w Name ▾

..

data

4w\_only\_전국아파트분양가분석.ipynb

4w\_only\_전국아파트분양가분석\_html.html

4w\_only\_전국아파트분양가분석\_결과삭제\_html.html

4w\_p1.mp4

4w\_p2.mp4

4w\_p3.mp4

Notebook: Python 3

Other: Text File Folder Terminal

## .ipynb 파일 여는 방법

1. 새로운 빈 파일 생성 후 코드 작성
2. 기존 ipynb 파일 클릭하여 사용

# Useful Tips for Jupyter notebook

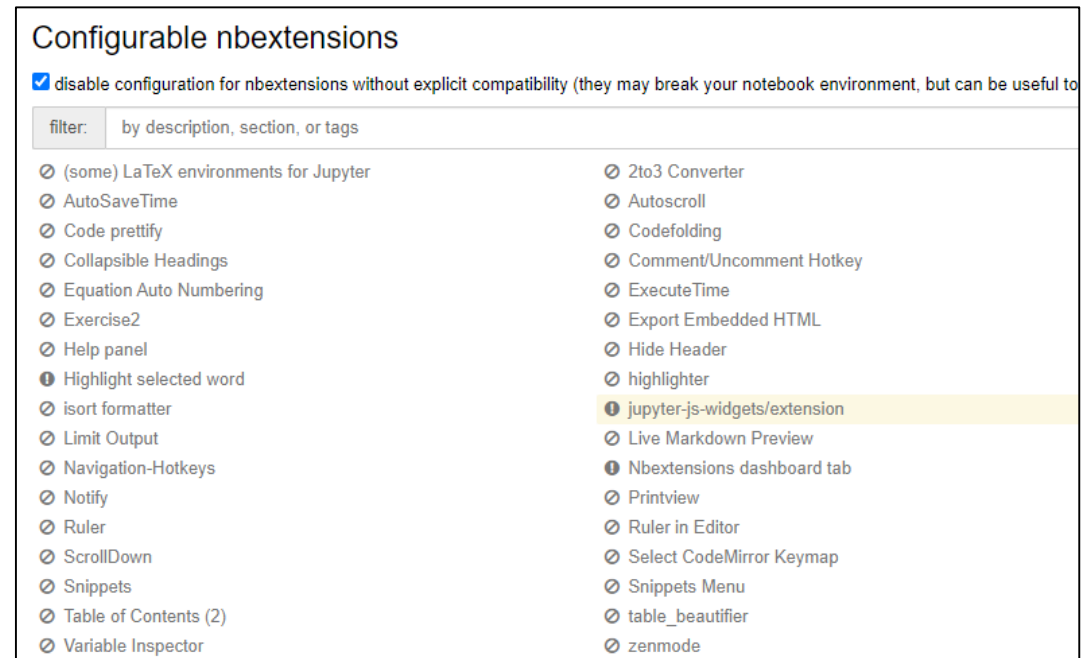
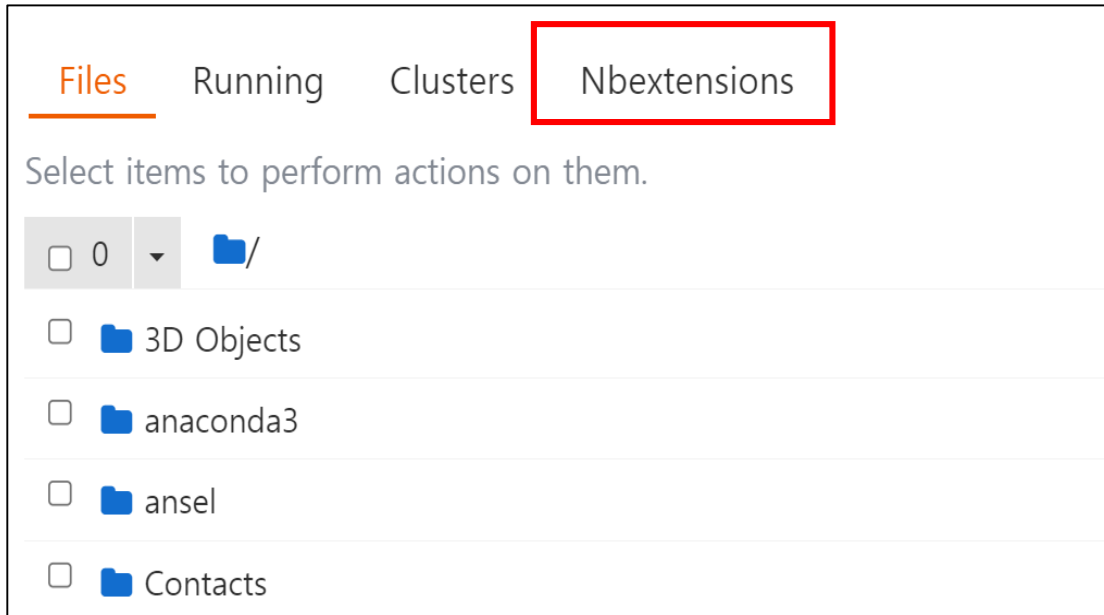


# nbextension 설치

- jupyter notebook을 편리하게 사용할 수 있는 기능 제공 (변수 highlighting, table of contents등)
- 설치: [anaconda prompt](#)에서 다음 명령어 실행

```
1) pip install jupyter_contrib_nbextensions  
2) jupyter contrib nbextension install --user
```

\* 2) 실행시 *No module named 'notebook.base'* 에러 발생하면 아래 코드 수행후 다시 2) 수행,  
- pip install notebook==6.4.12



# 폰트 변경

<https://realblack0.github.io/2020/05/13/jupyter-notebook-themes.html>

- 1) jupyterthemes package 설치
- 2) 원하는 폰트 및 코딩 환경 지정

```
# jupytertheme 패키지 설치
```

```
pip install jupyterthemes
```

```
# 추천 테마 및 옵션 적용
```

```
jt -t onedork -fs 115 -nfs 125 -tfs 115 -dfs 115 -ofs 115 -cursc r -cellw 80% -lineh 115 -altmd -kl -T -N
```

```
# 끝
```

추천테마 옵션 예제:

- 1) jt -t onedork -fs 115 -nfs 125 -tfs 115 -dfs 115 -ofs 115 -cursc r -cellw 80% -lineh 115 -altmd -kl -T -N
- 2) jt -t grade3 -f dejavu -fs 14 -dfs 12 -ofs 12 -tfs 12 -nfs 13 -cellw 80% -lineh 150 -cursc r

# jupyter notebook 사용법

# Jupyter notebook 두 가지 모드

## 1. edit mode – cell 안 상태

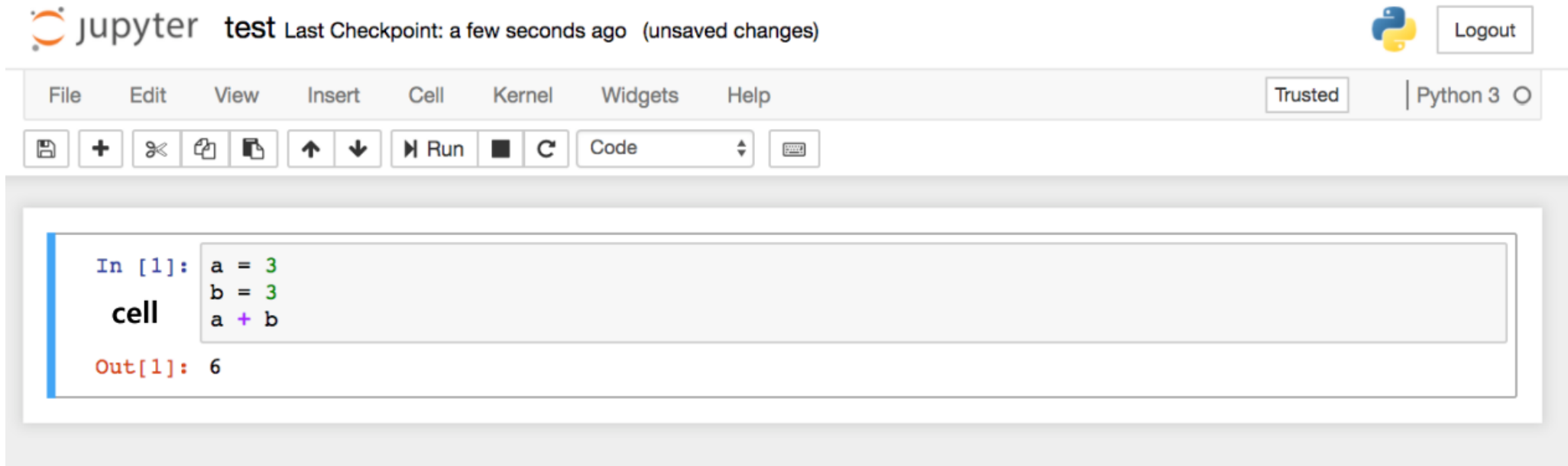
- enter를 치거나 cell 안을 클릭하면 edit mode로 바뀜
- 왼쪽 바의 색깔이 **초록색** (색깔은 개발환경 세팅에 따라 달라질 수 있음)
- 코드 작성시에 사용

## 2. commend mode – cell 밖 상태

- ESC를 치거나 cell 밖을 클릭하면 commend mode로 바뀜
- 왼쪽 바의 색깔이 **파란색** (색깔은 개발환경세팅값에 따라 달라질수 있음)
- commend mode에서는 타이핑을 해도 코드 작성이 되지않고, 미리 예약된 단축키가 작동함
- 코드 작성 외에 코드 관리에 편리한 작업을 수행

# 코드 실행 방법

## ■ 코드 실행 명령어



- cell 단위로 실행 → 실행 시점에 해당 코드가 memory에 올라감
- 실행 명령어 **ctrl + enter**, **shift + enter**, **alt + enter**
- cell의 위치가 아닌 실행 순서에 의하여 코드가 실행 됨
- 셀을 블록으로 지정후 실행하면 블록지정된 셀 이 순차적으로 모두 실행됨

# 코드 실행 과정

## 작성한 코드

```
A=3  
for pp in [1,2,3,4]:  
    print(pp)  
B=5  
C=A+B
```

## Python Interpreter:

각 명령어에 대한 실행 방침이 적혀있는 문서

**A=3:**

- **메모리**에 A 변수를 만들고 3을 넣어라

**for pp in list1:**

**print(pp)**

- **메모리**의 list1의 각 element를 순서대로 pp변수에 넣고 pp변수를 출력하라

**C=A+B**

- **메모리**에서 A변수 값과 B변수 값을 더하여 C라는 변수(없으면 만든 후)에 넣어라

....

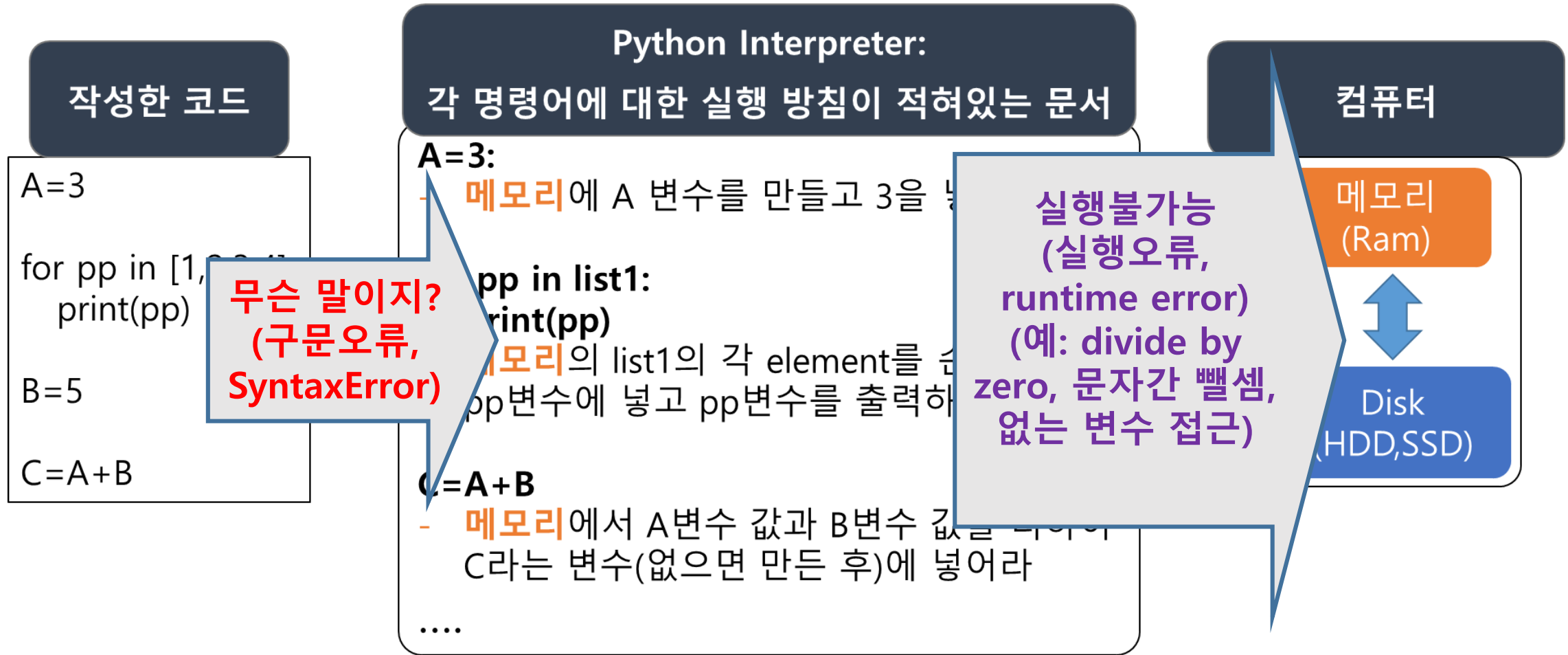
## 컴퓨터

메모리  
(Ram)



Disk  
(HDD,SSD)

# 두 가지 종류의 에러



- **구문오류:** 실행하기 전, interpreter 문법에 맞는지 체크해서 맞지 않으면 에러 발생, 코드 전체가 실행이 되지 않음
- **실행오류:** 실행하는 중, 실행이 불가능하여 생기는 에러, 실행오류 발생하기 전까지의 코드는 실행됨

# 사용법 (command mode)

## 주요 단축키 (command mode – cell 밖 상태)

- 아래 셀이랑 합치기 : shift + M
  - 셀 내려두기 : x    셀 copy : c    셀 붙여넣기 : v or shift + v
  - 셀 지우기 : d, d    셀 지우기 취소 : z
  - Markdown 변환 : m    Code로 변환 : y
- Code line: l (소문자 L)
  - Find and replace: f (해당 셀에만 적용)
  - Find: ctrl+f (web page 기능)



# 사용법 (edit mode – code 상태)

## 주요 단축키 (edit mode – cell 안 상태)

- 툴팁 표시하기 : Shift + Tab (사용 함수 설명)
- 셀 나누기 : ctrl + shift + -
- 들여쓰기: tab, shift+tab
- 변수나 함수 작성시 tab: 자동완성기능
- 입력 취소: ctrl + z
- 입력 취소 되돌리기 : ctrl + y
- 주석처리: ctrl + / (블록지정 후에도 가능)

### ■ 단축키 모음

<http://intellegibilisverum.tistory.com/entry/IPython-notebook-%EB%8B%A8%EC%B6%95%ED%82%A4-%EB%AA%A8%EC%9D%8C>

# 사용법 (edit mode – markdown 상태)

## 2. Header

마크다운의 가장 기본인 Header(제목)을 지정하는 방법이다.

"#"을 활용하여 header(제목) 기능을 사용할 수 있고, "#"의 개수로 header의 크기를 조절할 수 있다.(1~6개까지 지원)

# 한 개가 가장 크고, 개수가 늘 수록 작아진다.

```
# Header 1 : # 1개 사용
## Header 2 : # 2개 사용
### Header 3 : # 3개 사용
#### Header 4 : # 4개 사용
##### Header 5 : # 5개 사용
##### Header 6 : # 6개 사용
##### Header 7 : # 7개 사용; 6개까지 header로 활용 가능
```

결과는 다음과 같다. 6개까지만 지원하므로, 마지막 7개는 header로 변환되지 않았다.

**Header 1 : # 1개 사용**

**Header 2 : # 2개 사용**

**Header 3 : # 3개 사용**

**Header 4 : # 4개 사용**

**Header 5 : # 5개 사용**

**Header 6 : # 6개 사용**

##### Header 7 : # 7개 사용; 6개까지 header로 활용 가능

<https://leedakyeong.tistory.com/entry/Markdown-Jupyter-Notebook-%EC%A3%BC%ED%94%BC%ED%84%B0-%EB%85%B8%ED%8A%B8%EB%B6%81-%EB%A7%88%ED%81%AC%EB%8B%A4%EC%9A%B4-%EC%A0%95%EB%A6%AC>

# Magic command

해당 명령어들은 현재 메모리에 할당되어 있는 변수들의 정보를 보여줍니다.

<code>%who</code>	변수들을 출력합니다.
<code>%who_ls</code>	변수들을 리스트의 형태로 출력합니다.
<code>%whos</code>	변수들을 변수명과 유형, 그리고 데이터를 포함하여 출력합니다.

```
import numpy as np
```

```
a = 1234
b = "hello"
c = 1e-3
```

```
def foo(a, b):
    print(a * b)
```

```
%who
```

```
a      b      c      foo      np      plt
```

```
%who_ls
```

```
['a', 'b', 'c', 'foo', 'np', 'plt']
```

```
%whos
```

Variable	Type	Data/Info
----------	------	-----------

a	int	1234
b	str	hello
c	float	0.001
foo	function	<function foo at 0x00000296BED309D0>
np	module	<module 'numpy' from 'C:\>ges\numpy\__init__.py'>
plt	module	<module 'matplotlib.pyplot' from 'C:\>matplotlib\matplotlib.py'>

# Magic command

%%time	코드 블록(셀)의 실행 시간을 측정합니다.
%%timeit	코드 블록(셀)을 n회 반복 실행하여 평균 실행 시간을 측정합니다.

```
%%time  
sum(range(10000000))  
sum(range(1000000))
```

CPU times: total: 266 ms  
Wall time: 257 ms

499999500000

```
%%timeit  
sum(range(10000000))  
sum(range(1000000))
```

262 ms  $\pm$  1.77 ms per loop (mean  $\pm$  std. dev. of 7 runs, 1 loop each)

# Anaconda prompt 명령어 사용 (with %)

```
%pip install geopandas
```

```
Collecting geopandas
```

```
  Downloading geopandas-0.14.3-py3-none-any.whl.metadata
```

```
Collecting fiona>=1.8.21 (from geopandas)
```

```
  Downloading fiona-1.9.5-cp311-cp311-win_amd64.whl.metadata
```

```
----- 0.0/51.1
```

```
----- 20.5/51.1 |
```

```
----- 51.1/51.1 |
```

```
Requirement already satisfied: packaging in c:\users\j:
```

```
Requirement already satisfied: pandas>=1.4.0 in c:\use
```

```
1.4)
```

```
Collecting pyproj>=3.3.0 (from geopandas)
```

```
  Downloading pyproj-3.6.1-cp311-cp311-win_amd64.whl.me
```

```
Collecting shapely>=1.8.0 (from geopandas)
```

```
  Downloading shapely-2.0.3-cp311-cp311-win_amd64.whl.r
```

```
-----
```

# Anaconda prompt 명령어 사용 (with %)

In [6]:

```
pwd
```

```
'C:\\Users\\jjm\\Dropbox\\Ongoing_lecture\\2024_1\\공공데이터분석'
```

In [10]:

```
%cd C:\\Users\\jjm\\Dropbox\\Ongoing_lecture\\2024_1\\공공데이터분석\\da_5w
```

```
C:\\Users\\jjm\\Dropbox\\Ongoing_lecture\\2024_1\\공공데이터분석\\da_5w
```

In [11]:

```
pwd
```


```
'C:\\Users\\jjm\\Dropbox\\Ongoing_lecture\\2024_1\\공공데이터분석\\da_5w'
```

# Anaconda 설치 (참고)

# Anaconda 설치


- Anaconda 설치 방법
  - 아래의 site에 가서 본인 컴퓨터 OS bit 에 맞는 (보통 64bit) anaconda를 다운 받은 후 설치 한다.  
(<https://www.anaconda.com/products/individual>)


**Anaconda Distribution**

**Download** 

For Windows

Python 3.9 • 64-Bit Graphical Installer • 594 MB

Get Additional Installers    **클릭**

**Windows** 

Python 3.9

64-Bit Graphical Installer (594 MB)

32-Bit Graphical Installer (488 MB)

**MacOS** 

Python 3.9

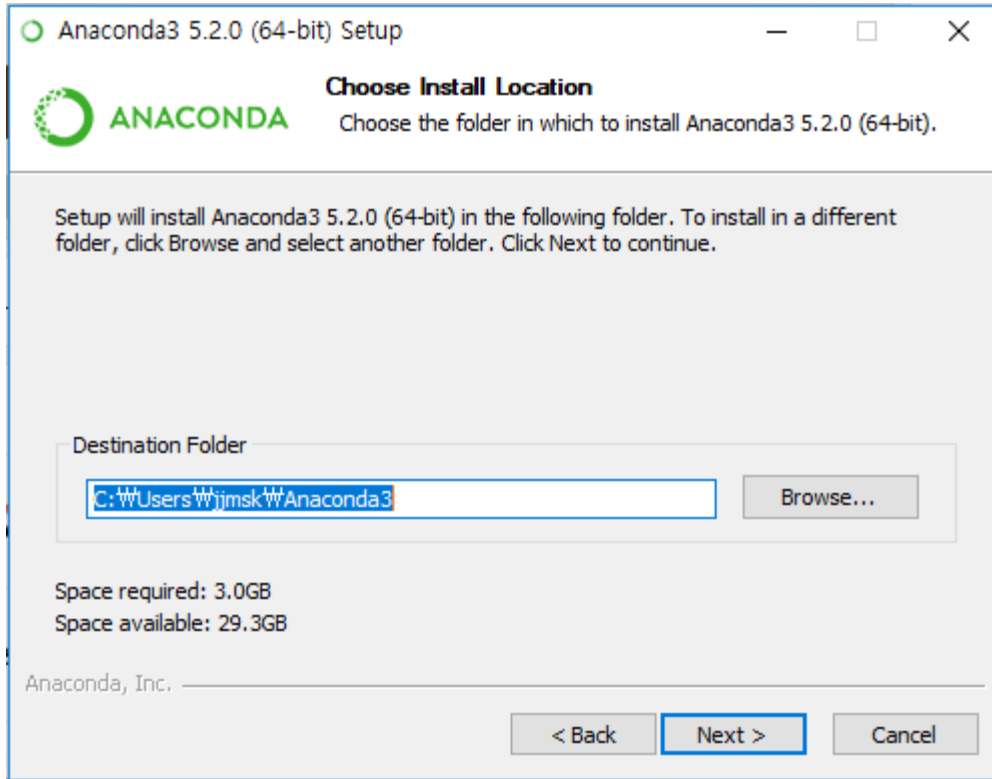
64-Bit Graphical Installer (591 MB)

64-Bit Command Line Installer (584 MB)

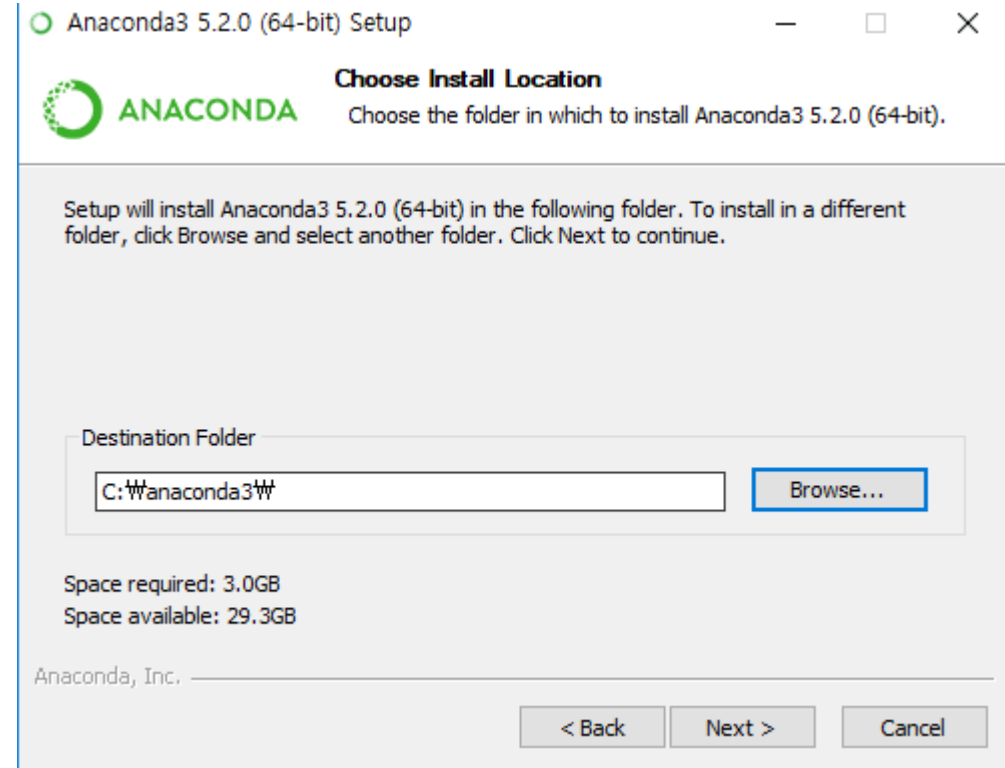
- 다운받은 후 실행하고, 변경사항 없이 다음 (next)를 여러 번 누르면 설치 됨  
(소요시간 약 15분 이내, 컴퓨터 사양에 따라 다를 수 있음)



# Anaconda install 시 자주 발생하는 에러 처리



8. 디폴트 Destination Folder 에서 Next해서 넘어가면 OK



8-1. **혹시 디폴트 Destination Folder 에 anaconda가 설치가 안되고 에러가 나면, C아래에 anaconda3이라는 폴더를 하나 만들고 그곳에 anaconda 설치하자**  
(- Destination Folder에 한글이 포함 되어있으면 에러 발생)

# Anaconda uninstall 방법

## ■ uninstall 방법

아래 경로의 Anaconda3 폴더에 가서 **Uninstall-Anaconda3** 실행



내 PC > 로컬 디스크 (C:) > 사용자 > **jmjung** > Anaconda3  
(또는 Users)

이 **jmjung** <user name>은 컴퓨터마다 다른  
컴퓨터 로그인 화면에 나오는 이름



qt.conf	CONF 파일	1KB
ucrtbase.dll	응용 프로그램 확장	993KB
<b>Uninstall-Anaconda3</b>	응용 프로그램	304KB
vccorlib140.dll	응용 프로그램 확장	358KB

# 기본파이썬 & pandas 복습

# 기초 파이썬 복습

## 실습1

주어진  $n$ 과  $k$ 에 대하여

1.  $[0, 1, 2, \dots, n-1]$  리스트를 만든다. (range)
2.  $k$ 의 배수의 값만을 갖는 리스트  $a$ 를 만든다 (slicing)
3.  $a$  리스트의 합을 출력한다. (sum)

```
n=15, k=3
```

```
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14]
```

```
[0, 3, 6, 9, 12]
```

```
sum: 30
```

## 실습2

1. 1~50 사이의 정수 중에 7의 배수 - 1 만 출력하기: (6,13,...,48)  
(for, if)

---

6

13

20

27

34

41

48

## 실습3

1. 주어진 문자열에서 각 문자가 등장한 횟수를 출력하시오  
(dictionary)

```
input a string: abbcccdde  
a 1  
b 2  
c 3  
d 4  
e 1
```

cf) pandas Series의 value\_counts함수를 사용해 보시오

## 실습4

1. 두 수와 연산자, 총 세 개의 변수를 parameter로하는 사칙연산 함수 `foo1()`을 만드시오 (연산자는 `+, -, /, *` 중 하나만 가능)

```
foo1(3,2,'+')  
foo1(1,5,'-')  
foo1(2,3,'*')  
foo1(4,8,'/')
```

5

-4

6

0.5



## 실습5

# 단어가 들어있는 list에서 숫자가 2개이상 포함되어있는 단어 출력하는 프로그램을 만드시오

```
lst1=['12ab','abc1','abc123','abc12','aaaaacc','11']
```

```
12ab  
abc123  
abc12  
11
```

pandas 복습

# 실습1

1. 아래와 같은 series sr1을 만드시오
  - Index: 0~9
  - Data: 0~9까지의 random value 10개
2. sr1에 3을 곱하고 1을 더한 sr2를 만들고,  
sr2 중에 data가 4의 배수인 index 합을 구하시오
3. sr1에서 두번째로 큰 data의 값과 index를 구하시오

## 실습2

- pandas\_data1.txt를 읽어서 다음을 출력하시오 ('이름'을 index로 선택)
  1. 최하나학생의 기말점수
  2. 김다섯학생의 모든 정보
  3. 처음부터 '김열' 학생까지의 데이터
  4. 마지막 5명 학생의 출석, 과제 점수

(한글파일 read\_table 시: encoding='cp949')

## 실습3

- pandas\_data3.txt를 읽어서 다음을 수행하시오
  1. final이 40점 이상인 학생의 수
  2. att이 5점 이하인 학생의 기말 점수들을 구하고, 그 점수들의 평균
  3. proj가 100점인 학생의 mid, final, att 점수
  4. "mid\_grade" column 만들어, mid 점수가 25점 이상 A, 20점 이상 B, 나머지는 C를 넣으시오

## 실습4

pandas\_data3.txt 파일을 읽어서,

- 1) gender 별 중간점수 최대값을 구하시오
- 2) 학년 별 기말점수 평균을 구하시오

	year	mid	final	att	proj
name					
kim2	1	12	36	5	75
kim3	2	17	20	5	96
kim6	2	28	20	4	83
kim7	2	20	44	6	82
kim8	1	20	24	8	88
kim9	3	23	45	10	92
kim10	3	16	47	5	83
lee5	3	21	35	5	92
lee9	3	20	36	5	87
lee10	4	28	41	9	87

matplotlib

# Figure & Axes



# Figure 구조

- Figure
  - 그림을 담는 가장 큰 틀
  - 보통 한 개만 설정
- Axes
  - Figure에 들어갈 그림의 개수 결정
  - 여러 개 가능
- Axis
  - 각 axes마다 존재

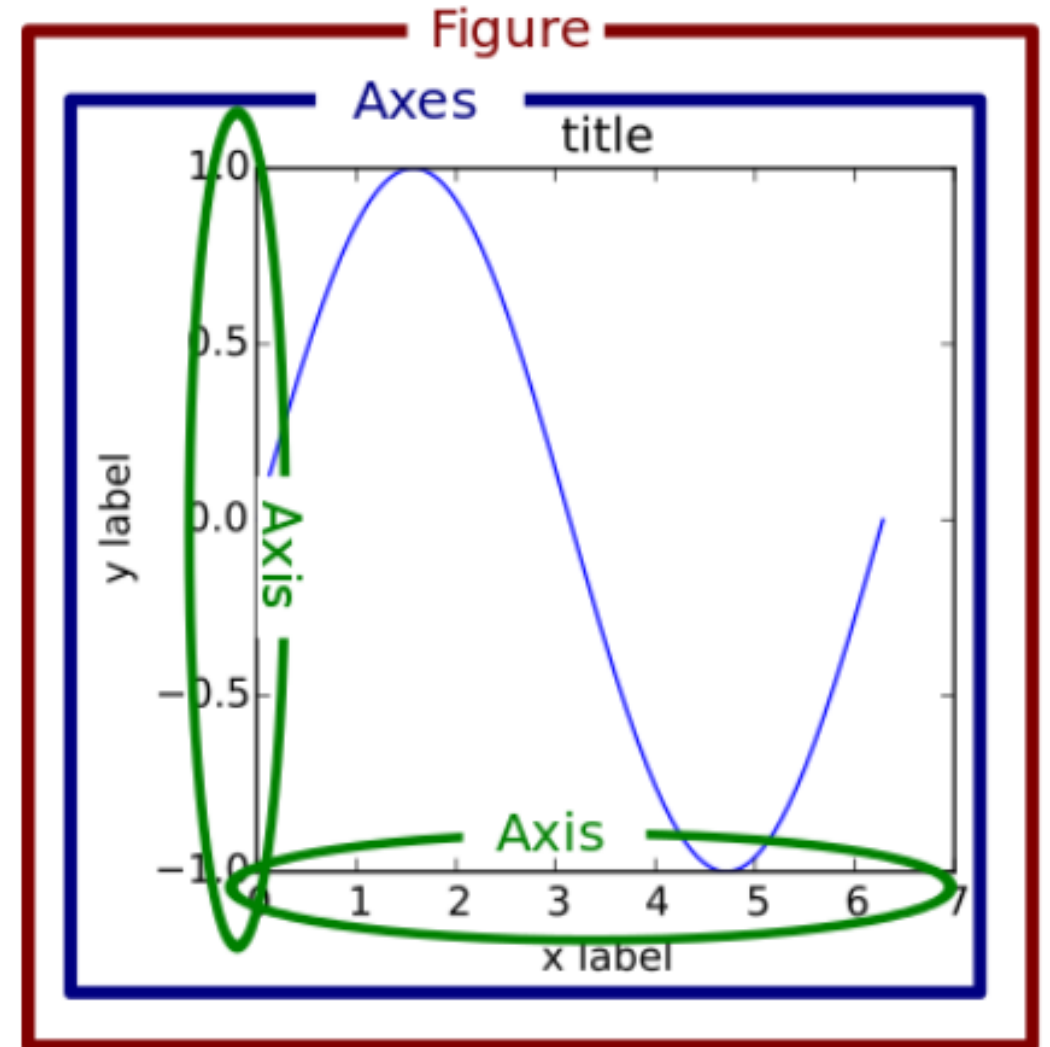


Figure 1, axes 1

## Figure 구조 예제

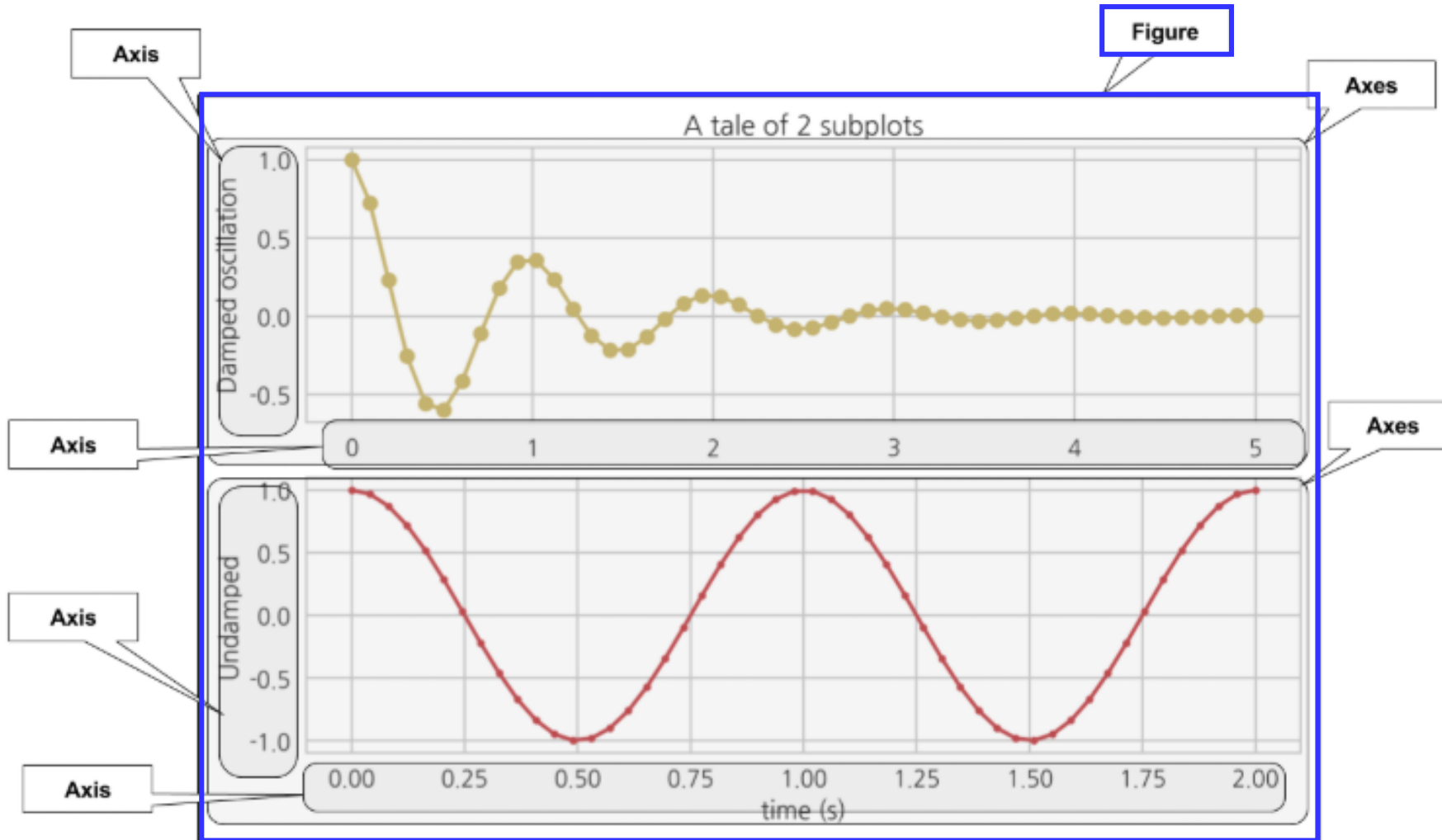


Figure 1, axes 2

# Figure 생성

- `plt.figure()`: figure 생성 함수
  - `parameter`
    - `figsize`: figure size (가로, 세로)
    - `dpi`: dots per inch, 1인치당 들어가는 도트의 개수, 해상도



- ...
- `return`
  - 생성된 figure (보통 **fig**라는 변수를 사용)

# Axes 생성

- Axes 를 생성하는 세 가지 함수
  - `fig.subplots` (가장 간단한 함수로 우선 이것부터 사용)
  - `fig.add_subplot`
  - `fig.subplot2grid`
- **`fig.subplots()`**: 생성한 figure에 axes를 생성하는 함수, 같은 크기의 axes가 생성됨
  - **`fig`**: `plt.figure()`의 return 객체
  - **`parameter`**
    - `nrows`: axes의 행의 수
    - `ncols`: axes의 열의 수
    - ...
  - **`return`**
    - 생성된 axes들의 모음

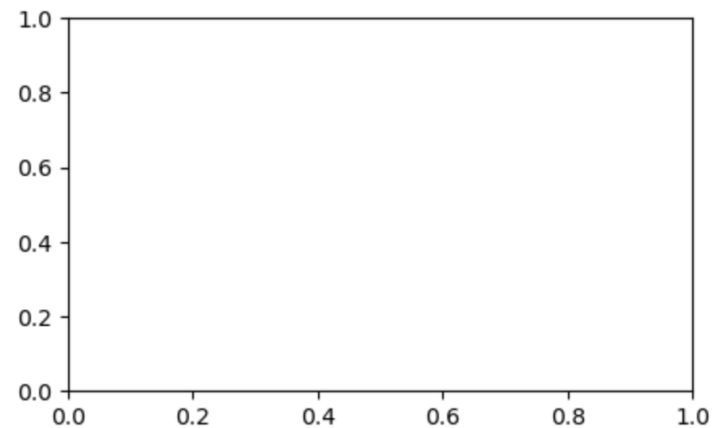
# Axes 생성 (1개)

```
fig=plt.figure(figsize=(5,3), dpi=100)
ax=fig.subplots(1,1) # positional argument <- default paramter
# 또는 fig.subplots() (nrows=1, ncols=1)
type(fig)
ax
type(ax)
#fig.show() <- jupyter notebook에서는 show 함수 호출 할 필요 없음
```

matplotlib.figure.Figure

<Axes: >

matplotlib.axes.\_axes.Axes



# Axes 생성 (2개)

```
fig=plt.figure(figsize=(6,3), dpi=100)
axs=fig.subplots(1,2)
type(fig)
axs
type(axs)
axs[0]
type(axs[0])
```

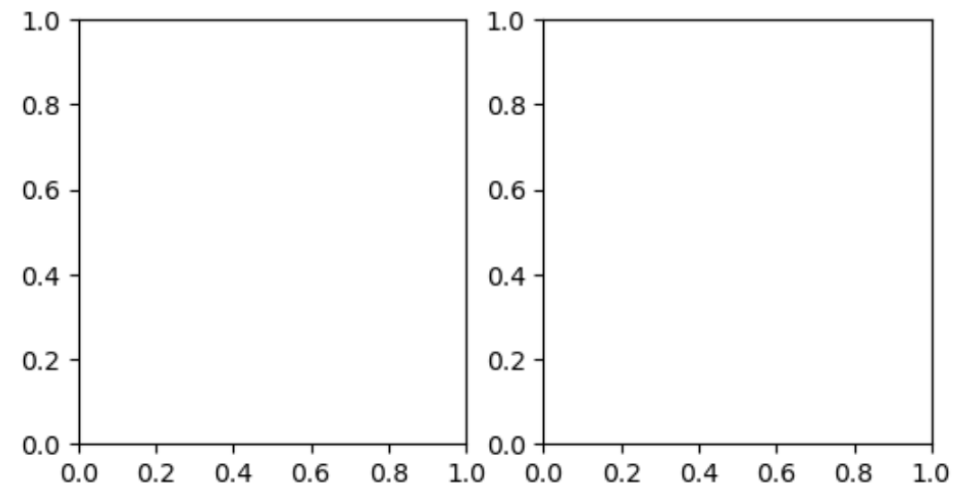
matplotlib.figure.Figure

array([<Axes: >, <Axes: >], dtype=object)

numpy.ndarray

<Axes: >

matplotlib.axes.\_axes.Axes



# plot

## matplotlib.axes.Axes.plot

```
Axes.plot(self, *args, scalex=True, scaley=True, data=None, **kwargs)
```

Plot y versus x as lines and/or markers.

← 주어진 X, Y 데이터를 line 또는 marker로 그리는 함수

### Parameters:

**x, y** : array-like or scalar ← not optional

The horizontal / vertical coordinates of the data points. x values are optional and default to `range(len(y))`.

Commonly, these parameters are 1D arrays.

They can also be scalars, or two-dimensional (in that case, the columns represent separate data sets).

These arguments cannot be passed as keywords.

**fmt** : str, optional

A format string, e.g. 'ro' for red circles. See the *Notes* section for a full description of the format strings.

Format strings are just an abbreviation for quickly setting basic line properties. All of these and more can also be controlled by keyword arguments.

This argument cannot be passed as keyword.

**data** : indexable object, optional

An object with labelled data. If given, provide the label names to plot in x and y.

### Note

Technically there's a slight ambiguity in calls where the second label is a valid *fmt*. `plot('n', 'o', data=obj)` could be `plt(x, y)` or `plt(y, fmt)`. In such cases, the former interpretation is chosen, but a warning is issued. You may suppress the warning by adding an empty format string `plot('n', 'o', '', data=obj)`.

### Returns:

list of [Line2D](#)

A list of lines representing the plotted data.

# plot 설명 및 예제

plot(X,Y)

X=[x1, x2, x3]

Y=[y1, y2, y3]

1. 화면에 (x1, y1), (x2,y2), (x3,y3)을 마커를 찍는다. (default 마커: 점)
2. 각 점을 차례로 선으로 연결한다. (default 선: 실선)

```
In [44]: fig=plt.figure(figsize=(4,4), dpi=100)
         ax=fig.subplots()

         X=range(100)
         Y=[value**2 for value in X]

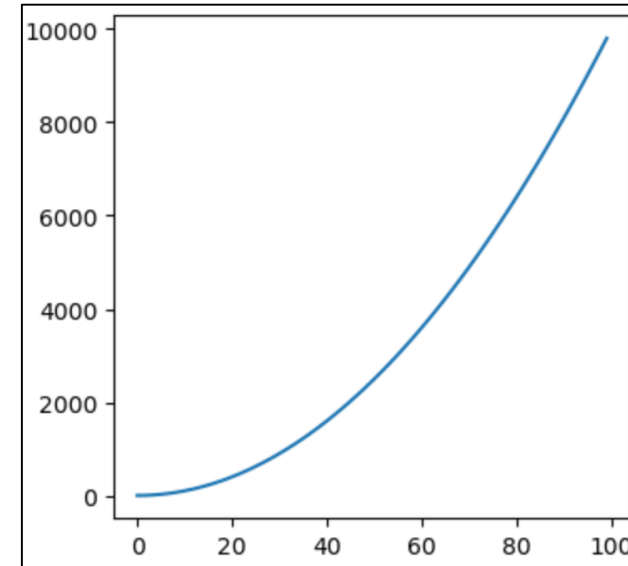
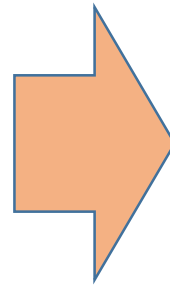
         list(X[:5])
         list(Y[:5])

         ax.plot(X,Y)
```

Out[44]: [0, 1, 2, 3, 4]

Out[44]: [0, 1, 4, 9, 16]

Out[44]: [<matplotlib.lines.Line2D at 0x1f83fc00130>]



← ax.plot(X,Y) return 객체 출력  
\_=ax.plot(X,Y) 처럼 dummy 변수(\_)로 받으면 출력 안됨



# ax.plot (stateless) vs plt.plot (stateful)

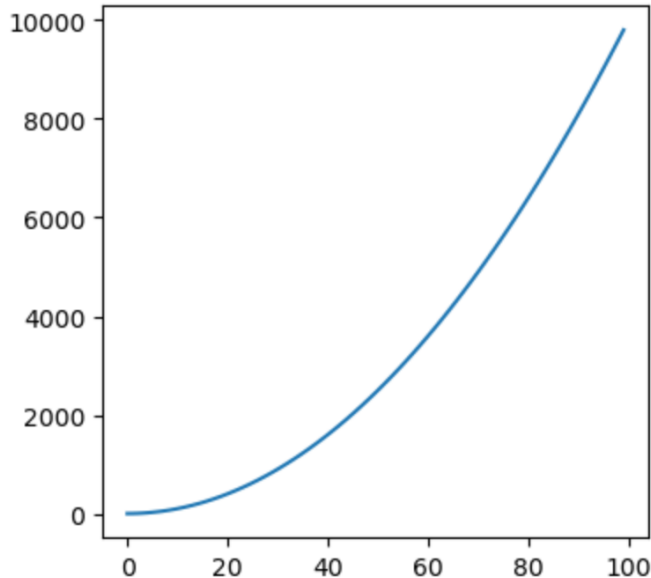
- `ax.plot()` 이 아닌 `plt.plot()` 을 사용하면 figure, axes 설정하지 않고도 같은 그림을 그릴 수 있다

```
fig=plt.figure(figsize=(4,4), dpi=100)  
ax=fig.subplots()
```

```
X=range(100)  
Y=[value**2 for value in X]
```

```
ax.plot(X,Y)
```

```
[<matplotlib.lines.Line2D at 0x25b403e3070>]
```

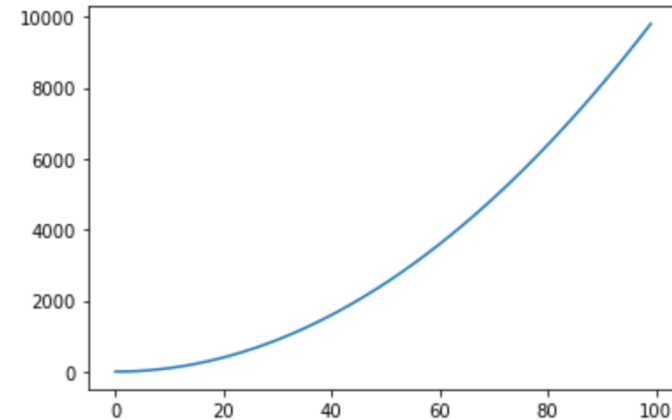


- stateless 방법
- 단점: figure, axes를 명시해야 하는 불편함
- 장점: axes를 원하는 대로 작업할 수 있음

```
[5]: X=range(100)  
Y=[value**2 for value in X]
```

```
plt.plot(X,Y)
```

```
t[5]: [<matplotlib.lines.Line2D at 0x25b3f6e3bb0>]
```



- stateful 방법
- 장점: figure와 axes를 알아서 생성함
- 단점: axes를 명시할 수 없어서, axes가 두 개 이상이면 혼란스러움

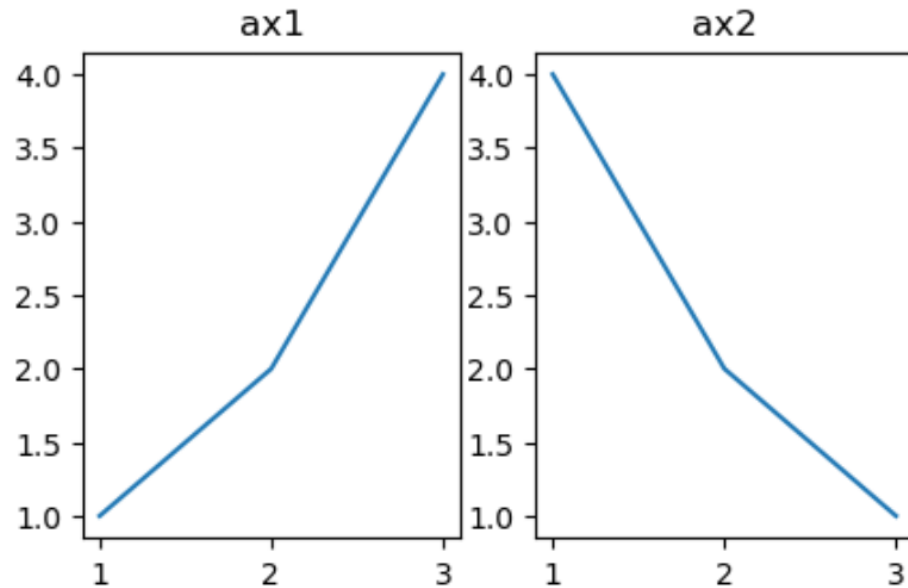
- 본 수업에서는 stateless 방법을 주로 사용

# plot on two axes

```
[15]: fig=plt.figure(figsize=(5,3), dpi=100)
      (ax1, ax2) = fig.subplots(1,2) # axs = fig.subplots(1,2) <- axs[0], axs[1]

      _=ax1.set_title('ax1')
      _=ax1.plot([1,2,3],[1,2,4])
      _=ax2.set_title('ax2')
      _=ax2.plot([1,2,3],[4,2,1])
```

- 각 axes 별로 작업 가능
- stateful 방법으로는 어려움



# plot with format

`plot(X,Y, fmt)`

`X=[x1, x2, x3]`

`Y=[y1, y2, y3]`

`fmt='[marker][line][color]'`

fmt 에서 정의한 대로

1. 화면에  $(x1, y1)$ ,  $(x2, y2)$ ,  $(x3, y3)$ 을 찍는다.
2. 각 점을 차례로 연결한다.

구체적인 컬러&스타일 정의는 나중에 다시 나옴

# plot with format

## Line Styles

character	description
' - '	solid line style
' - - '	dashed line style
' - . '	dash-dot line style
' : '	dotted line style

## Colors

The supported color abbreviations are the single letter codes

character	color
' b '	blue
' g '	green
' r '	red
' c '	cyan
' m '	magenta
' y '	yellow
' k '	black
' w '	white

[https://matplotlib.org/3.1.1/api/\\_as\\_gen/matplotlib.axes.Axes.plot.html](https://matplotlib.org/3.1.1/api/_as_gen/matplotlib.axes.Axes.plot.html)

## Markers

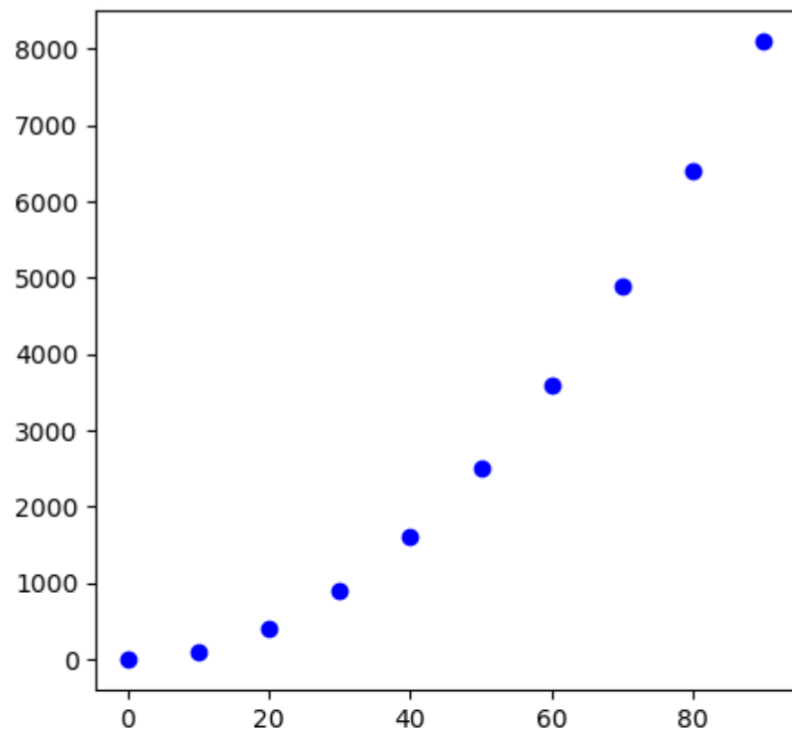
character	description
' . '	point marker
' , '	pixel marker
' o '	circle marker
' v '	triangle_down marker
' ^ '	triangle_up marker
' < '	triangle_left marker
' > '	triangle_right marker
' 1 '	tri_down marker
' 2 '	tri_up marker
' 3 '	tri_left marker
' 4 '	tri_right marker
' s '	square marker
' p '	pentagon marker
' * '	star marker
' h '	hexagon1 marker
' H '	hexagon2 marker
' + '	plus marker
' x '	x marker
' D '	diamond marker
' d '	thin_diamond marker
'   '	vline marker
' _ '	hline marker

# plot with format 예제

```
[17]: fig=plt.figure(figsize=(5,5), dpi=100)
      ax=fig.subplots()
      |
      X=np.arange(0,100,10)
      Y=X**2

      ax.plot(X,Y, 'ob')
```

t[17]: [<matplotlib.lines.Line2D at 0x25b41717d60>]

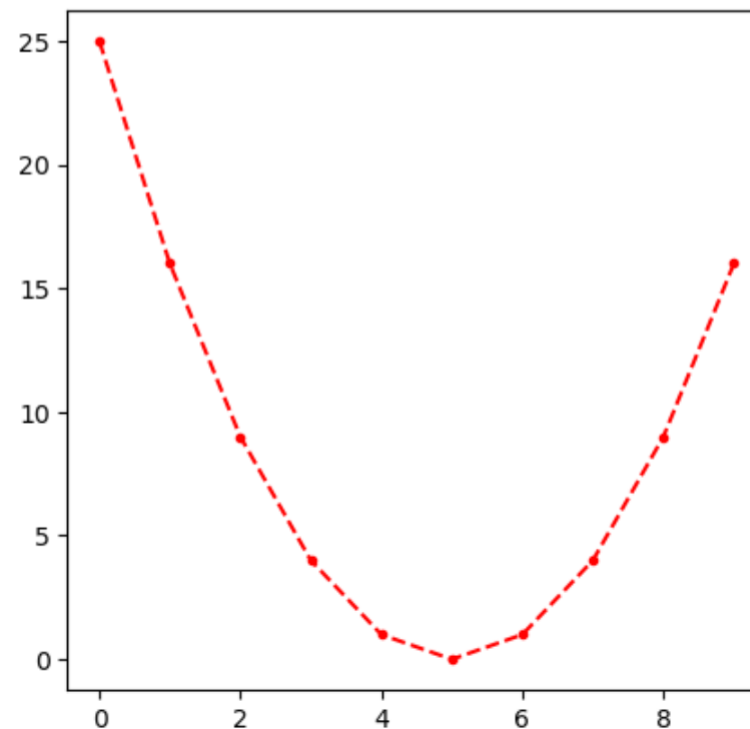


```
9]: fig=plt.figure(figsize=(5,5), dpi=100)
   ax=fig.subplots()

   X=np.arange(10)
   Y=(X-5)**2

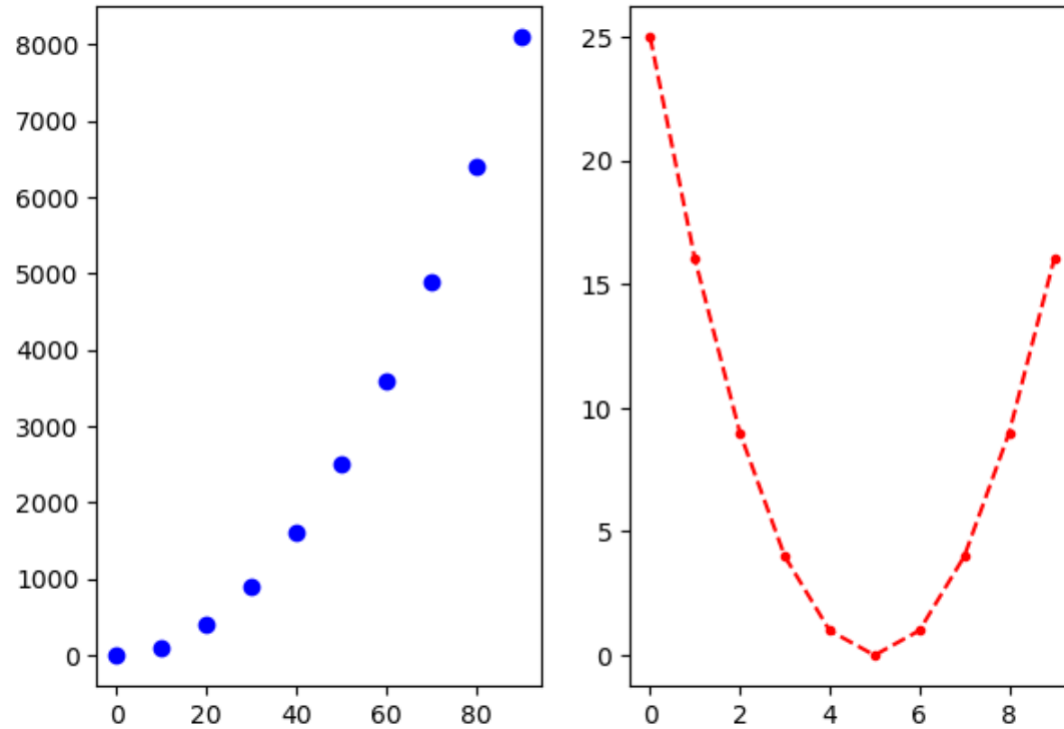
   ax.plot(X,Y, '.-r')
```

9]: [<matplotlib.lines.Line2D at 0x25b417cd6d0>]



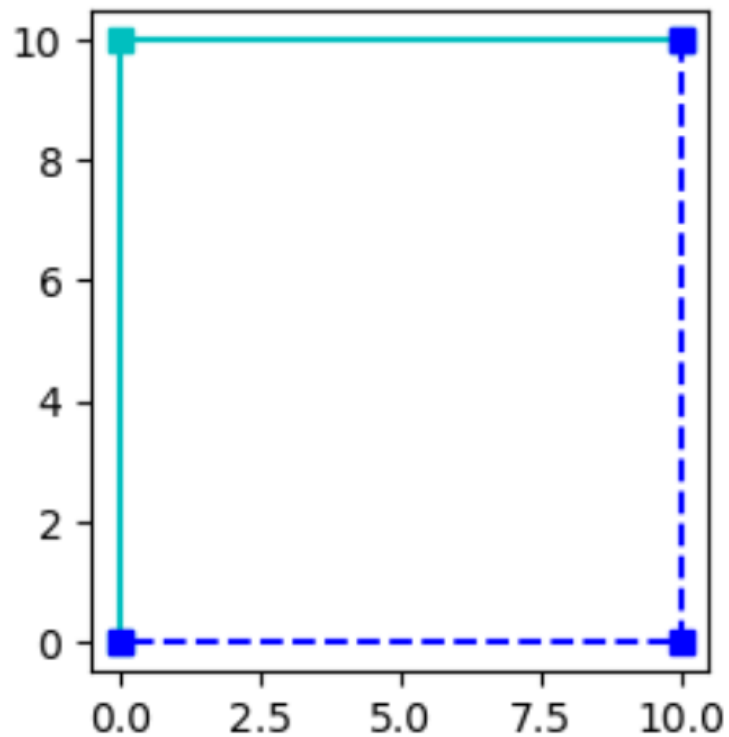
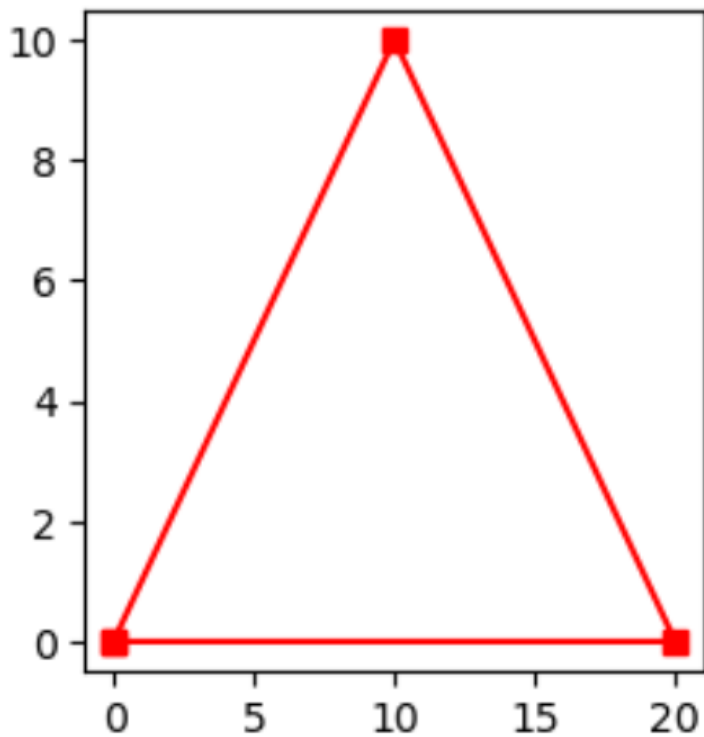
# 실습 1

- 전 슬라이드 두 개의 그림을 하나의 figure에 나란히 그리시오



## 실습 2

- 다음을 그려보시오



두번째 axes

- 두 개의 plot 함수를 호출
- `axe`에 여러 개의 그림을 그리면, 위에 각 그림들이 나중에 그린 그림이 그 전 그림을 가림

## Q & A

Thank you