

# Matplotlib - basic

Juyong Lee

# Matplotlib 란?

- Matplotlib란 Matlab에서 그려주는 그래프를 python에서 그릴 수 있도록 해주는 library

- Matlab 프로그램은 여러가지 과학 및 공학 계산을 수행해주는 유료 프로그램

- 강력한 성능과 예쁜 그래프를 지원함
- 하지만 가격이 비싸서 open-source인 대체품을 만든 것이 matplotlib임.
- Python와 결합되어 매우 편리함.

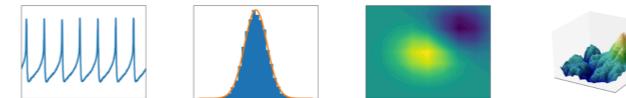


Version 3.1.1

[home](#) | [examples](#) | [tutorials](#) | [API](#) | [contents](#) »

fork me on GitHub

Matplotlib is a Python 2D plotting library which produces publication quality figures in a variety of hardcopy formats and interactive environments across platforms. Matplotlib can be used in Python scripts, the Python and IPython shells, the Jupyter notebook, web application servers, and four graphical user interface toolkits.



Matplotlib tries to make easy things easy and hard things possible. You can generate plots, histograms, power spectra, bar charts, errorcharts, scatterplots, etc., with just a few lines of code. For examples, see the [sample plots](#) and [thumbnail gallery](#).

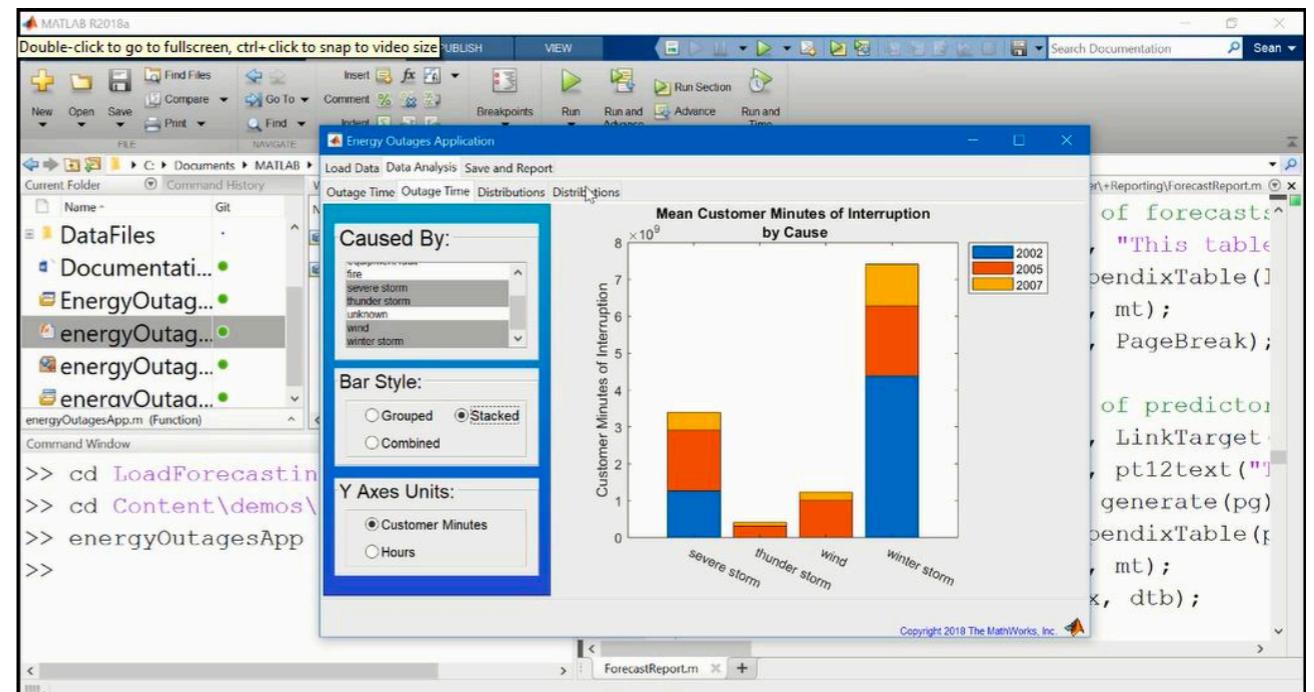
For simple plotting the `pyplot` module provides a MATLAB-like interface, particularly when combined with IPython. For the power user, you have full control of line styles, font properties, axes properties, etc, via an object oriented interface or via a set of functions familiar to MATLAB users.

## Installation

Visit the [Matplotlib installation instructions](#).

## Documentation

**<https://matplotlib.org>**



# How to install

- Ubuntu에서
  - pip을 이용
    - python -m pip install -U pip
    - python -m pip install -U matplotlib
- MacOS에서
  - brew를 이용
    - brew install libpng freetype pkg-config
- Anaconda를 사용한다면 default로 깔려있음.

# Import matplotlib

- 그림을 그리기 위해서는 matplotlib와 그 sub-module인 pyplot을 import 해야한다.
- 일반적으로 matplotlib는 mpl, pyplot은 plt로 줄여서 import한다.
- 이번 실습에서는 Jupyter-notebook을 사용할 예정입니다.

The screenshot shows a Jupyter Notebook interface with the following details:

- Header:** jupyter Matplotlib\_example Last Checkpoint: 2분 전 (autosaved)
- Toolbar:** Includes File, Edit, View, Insert, Cell, Kernel, Widgets, Help, Trusted, Logout, and Python 3.
- Cells:**
  - In [1]:

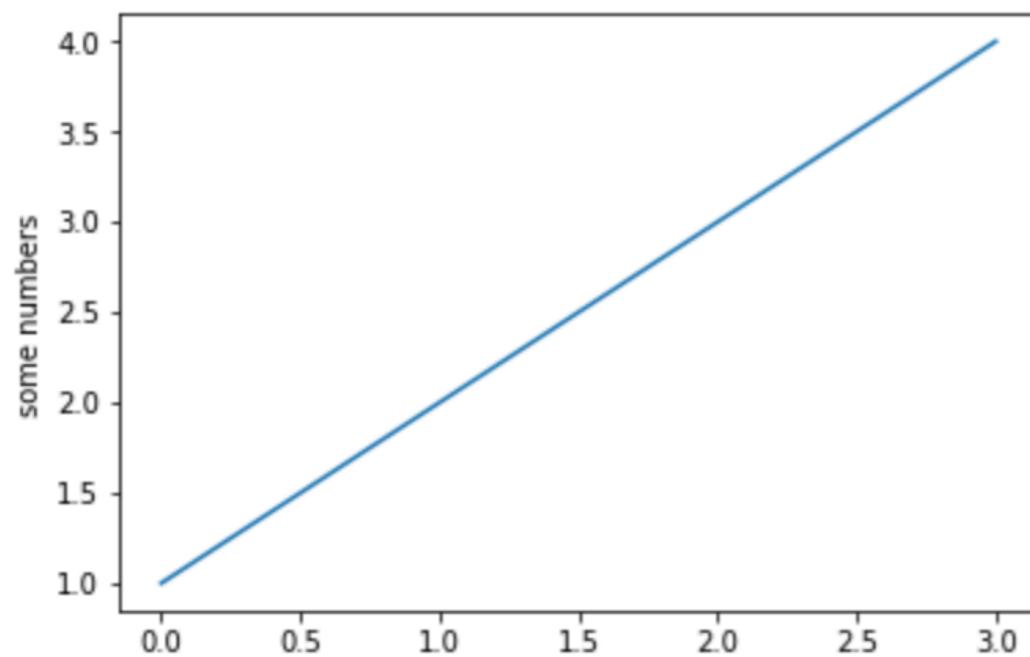
```
import matplotlib as mpl
import matplotlib.pyplot as plt
```
  - In [2]:

```
import numpy as np
```
  - In [ ]: (This cell is currently empty and highlighted with a green border.)

# Basic plot

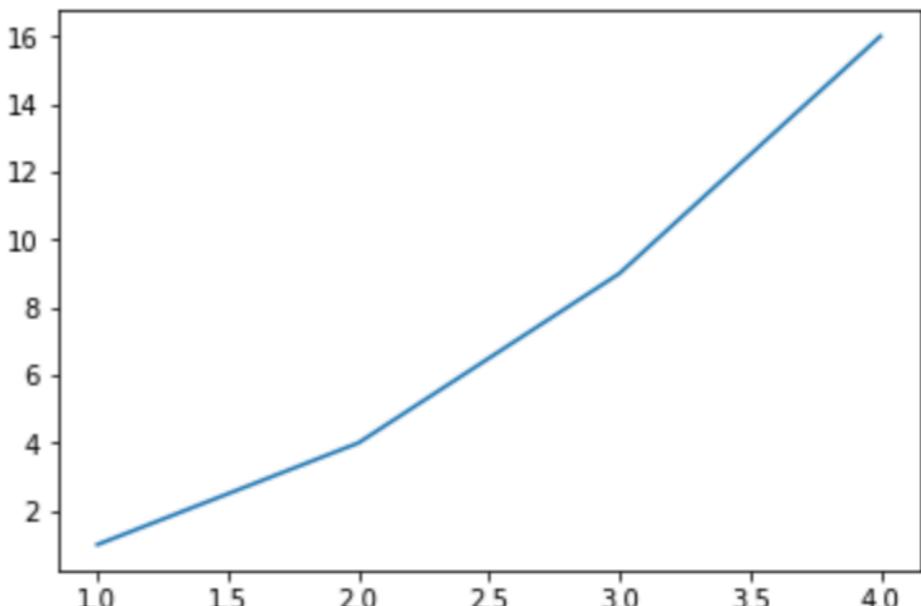
- plt.plot 함수
  - 가장 기본적인 plot함수
  - plt.plot(x\_data, y\_data)

In [3]: `plt.plot([1, 2, 3, 4])` x값은 자동으로 1, 2, 3, 4, ... 와 같은 식으로 매겨짐.  
`plt.ylabel('some numbers')`  
`plt.show()`



In [4]: `plt.plot([1, 2, 3, 4], [1, 4, 9, 16])` 리스트가 2개 주어지면 차례대로 x, y값으로 여겨짐.  
x값은 자동으로 1, 2, 3, 4, ... 와 같은 식으로 매겨짐.

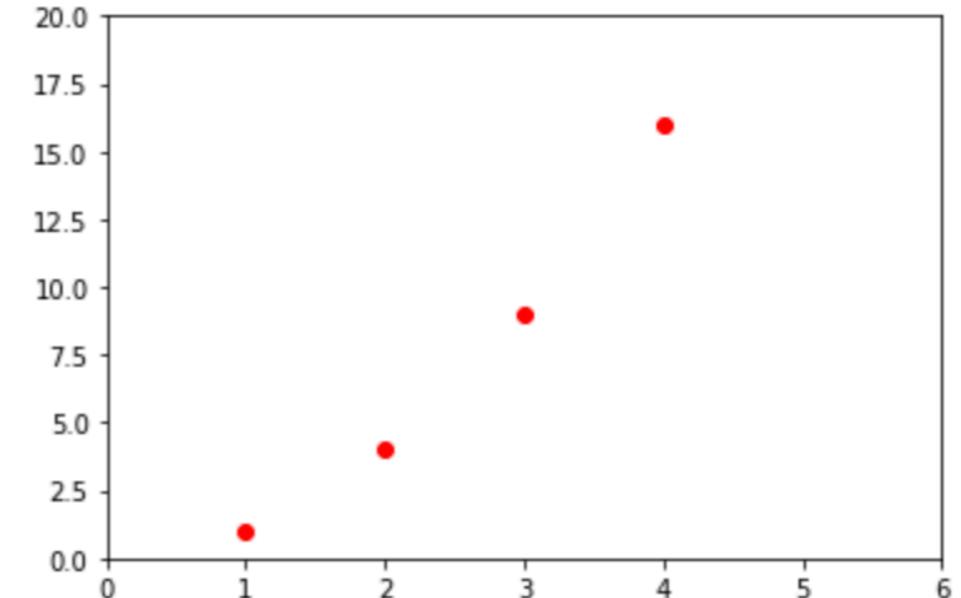
Out[4]: [<matplotlib.lines.Line2D at 0x119d7a9e8>]



# plot의 모양 바꾸기

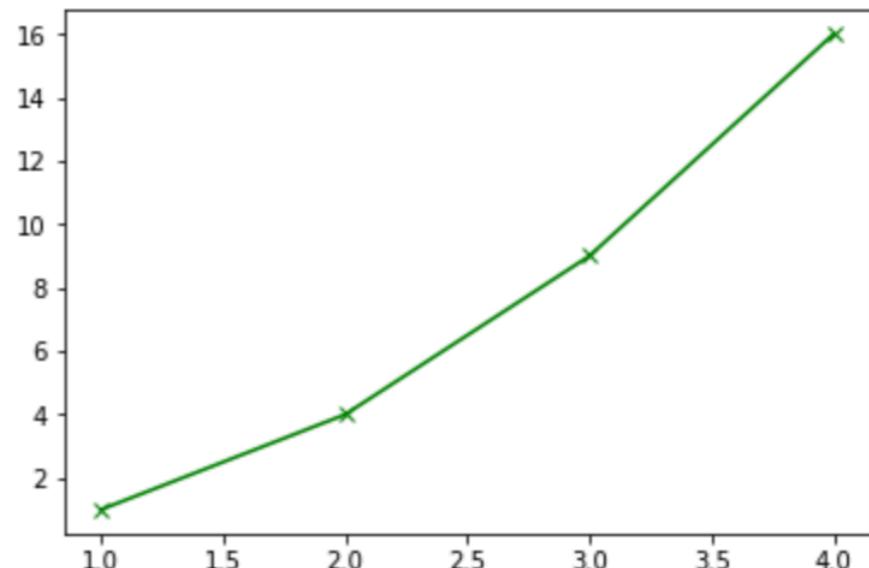
- plot method에 argument가 3개 주어지면 (x\_data, y\_data, format)으로 받아들임.
- Formatting option에서는 marker, line, color 세가지를 다음과 같이 지정한다.
  - fmt='[marker][line][color]'
  - 예시:
    - 'gx-': green, x-shape, solid line
    - 'bo-.': blue, o-dot, dotted line
- **plot.axis**에서는 x축과 y축의 min, max를 지정 한다.

```
In [5]: plt.plot([1, 2, 3, 4], [1, 4, 9, 16], 'ro')
plt.axis([0, 6, 0, 20])
plt.show()
```



```
In [6]: plt.plot([1, 2, 3, 4], [1, 4, 9, 16], 'gx-')
```

```
Out[6]: <matplotlib.lines.Line2D at 0x119effb00>
```



# Marker list

다양한 marker를 사용할 수 있다.

## 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

# Line style

```
In [7]: plt.plot([1, 2, 3, 4], [1, 4, 9, 16], 'b+:')
```

```
Out[7]: <matplotlib.lines.Line2D at 0x119fc5748>
```

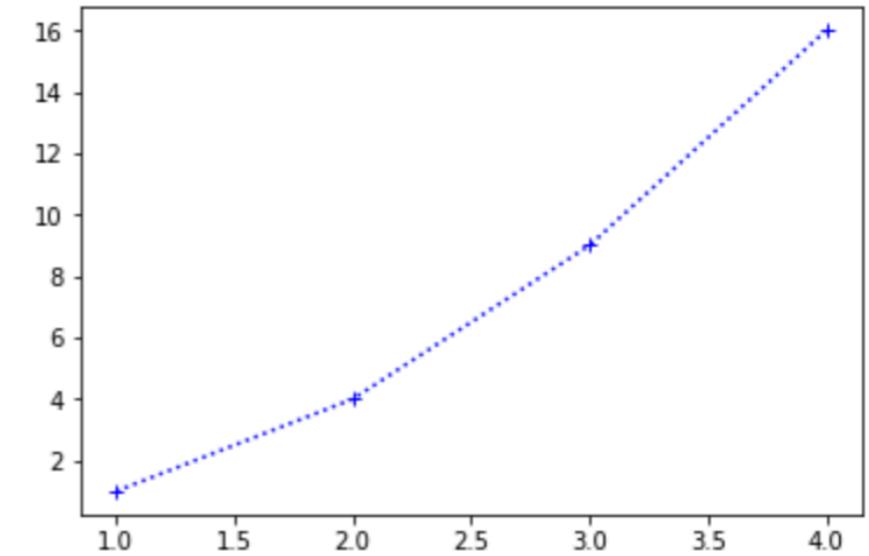
- 다음과 같은 line style을 지원한다.

## Line Styles

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

Example format strings:

```
'b'      # blue markers with default shape
'or'     # red circles
'g'      # green solid line
'-'      # dashed line with default color
'^k:'    # black triangle-up markers connected by a dotted line
```

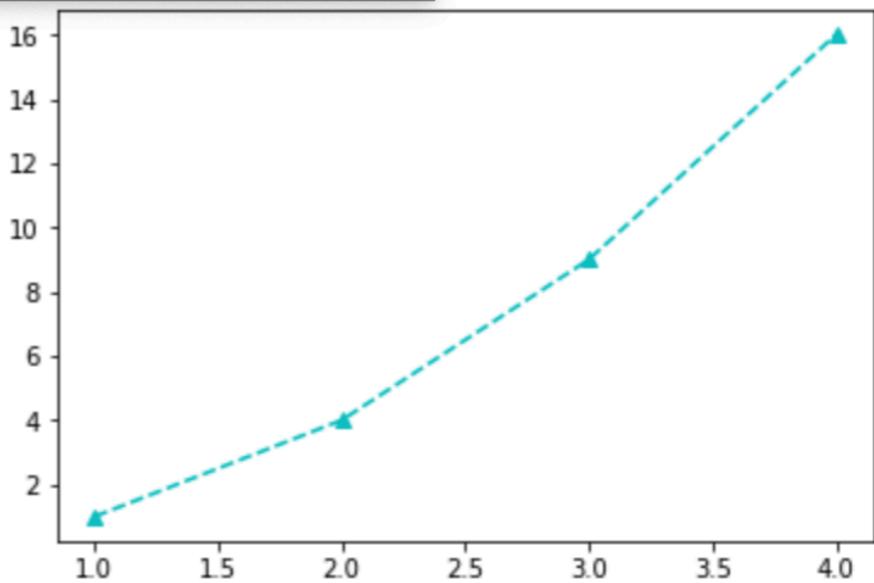


## Dotted line

```
In [8]: plt.plot([1, 2, 3, 4], [1, 4, 9, 16], 'c^--')
```

```
Out[8]: <matplotlib.lines.Line2D at 0x11a088358>
```

click to scroll output; double click to hide



## Dashed line

# Color list

- 다음에 나타나있는 기본 색은 알파 벳 한글자로 표현된다.
- 그 외의 임의의 색을 이용하려면 hex string ('#000FFF')를 이용할 수 있다.

## 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

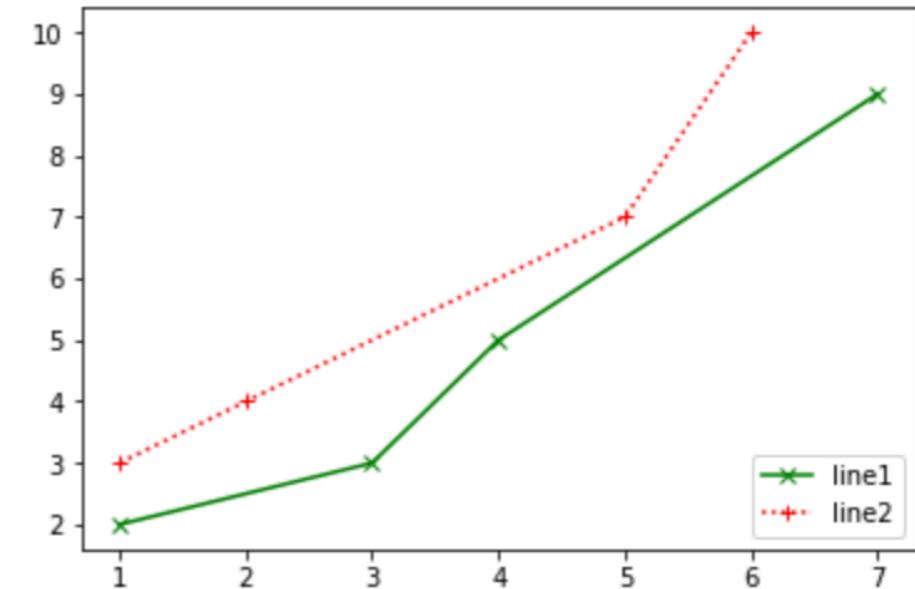
and the 'CN' colors that index into the default property cycle.

# 한 그래프에 여러개의 선 그리기

```
In [16]: plt.plot(x1, y1, 'gx-', label='line1')
plt.plot(x2,y2, 'r+:', label='line2')
plt.legend(loc='lower right')
```

```
Out[16]: <matplotlib.legend.Legend at 0x11a456630>
```

- Plot method를 여러번 사용하면 하나의 그래프에 여러개의 선이 나타난다.
- Label argument를 사용해주는 편이 좋다.
- **plt.legend**를 사용하면 label을 표시해준다.
- plt.legend는 legend의 위치를 loc을 이용해서 지정해 줄 수 있다.
- 가능한 location의 종류는 오른쪽과 같다



Location String	Location Code
'best'	0
'upper right'	1
'upper left'	2
'lower left'	3
'lower right'	4
'right'	5
'center left'	6
'center right'	7
'lower center'	8
'upper center'	9
'center'	10

# 축 label 붙이기

- 항상 그래프를 그릴 때는 x축, y축이 무엇인지 표시해주는 것이 중요하다.

- plt.xlabel**

- plt.ylabel**

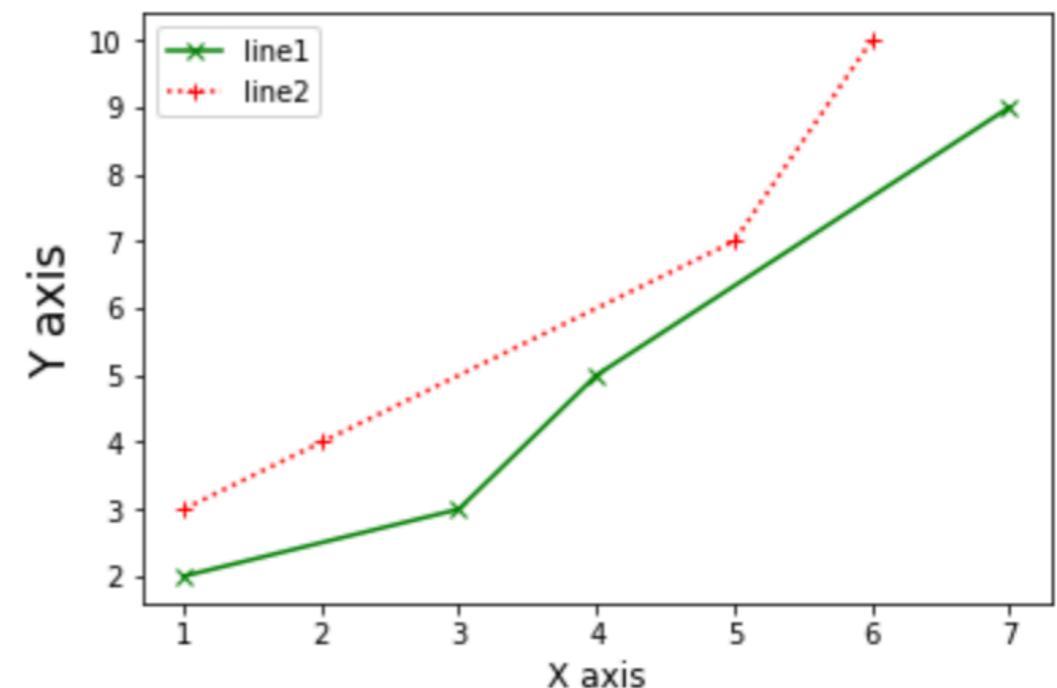
- 폰트의 사이즈를 조정하고 싶으면 **fontsize argument** 사용

- small, medium, large, xx-large, huge를 제공

- 더 미세 조정을 하고 싶으면 숫자 point를 줄 수도 있다

```
In [18]: plt.plot(x1, y1, 'gx-', label='line1')
plt.plot(x2,y2, 'r+:', label='line2')
plt.legend(loc='upper left')
plt.xlabel('X axis', fontsize='large')
plt.ylabel('Y axis', fontsize='xx-large')
```

Out[18]: Text(0, 0.5, 'Y axis')



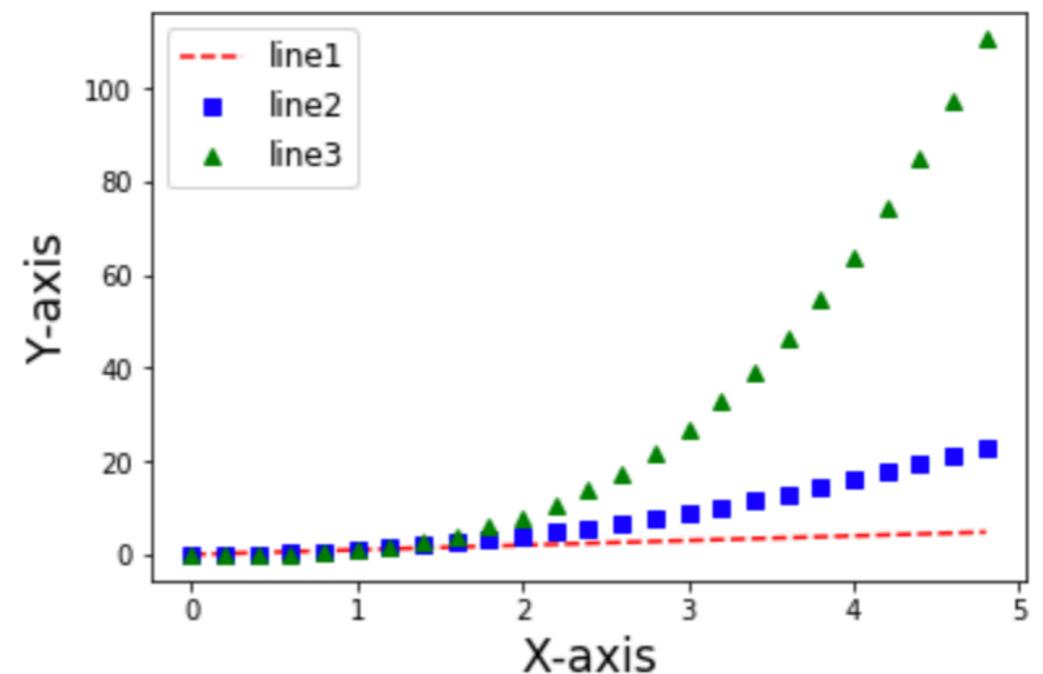
# Numpy array 사용하기

- Numpy array도 list와 동일하게 사용 가능하다.
- **format**에서 **line**을 지정해주지 않으면 선이 그려지지 않음.
  - Scatter plot이 된다.

In [24]:

```
# evenly sampled time at 200ms intervals
t = np.arange(0., 5., 0.2)

# red dashes, blue squares and green triangles
plt.plot(t, t, 'r--', label='line1')
plt.plot(t, t**2, 'bs', label='line2')
plt.plot(t, t**3, 'g^', label='line3')
plt.xlabel('X-axis', fontsize='xx-large')
plt.ylabel('Y-axis', fontsize='xx-large')
plt.legend(fontsize='large')
plt.show()
```



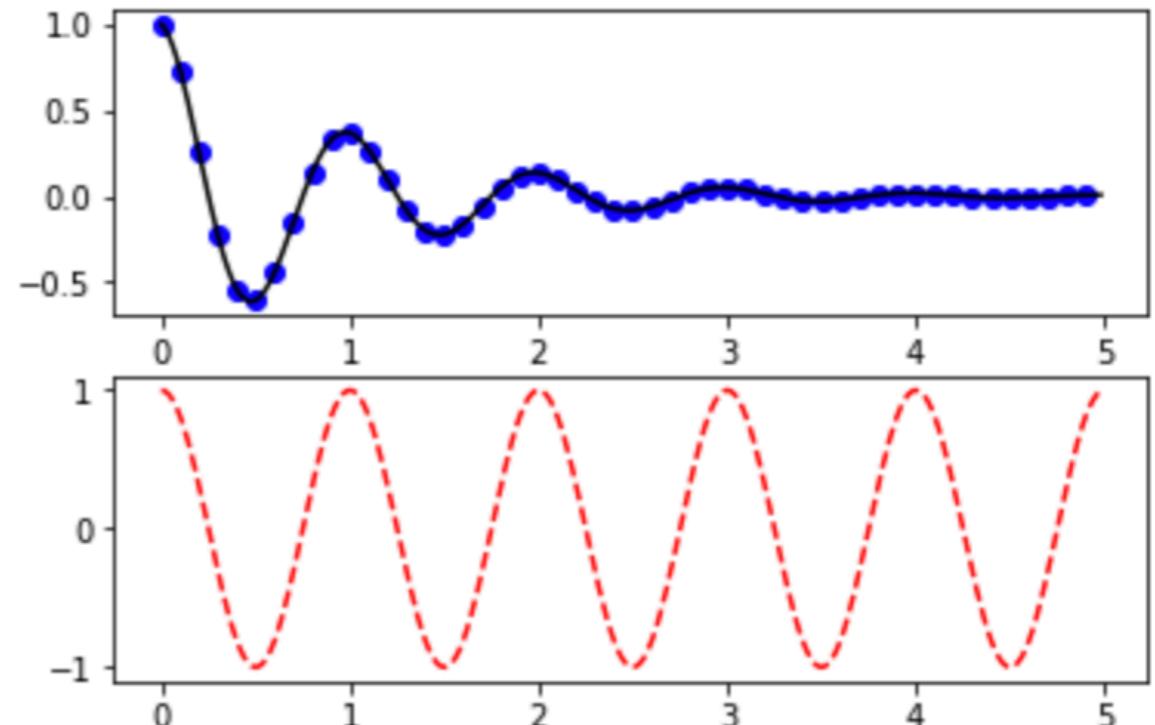
# 여러개의 plot을 한 그림에 넣기

- subplot 명령을 사용하면 하나의 그림에 여러개의 plot을 넣을 수 있다.
- subplot(211)의 의미
  - 괄호안의 3개의 숫자는 행, 열, 번호를 의미한다
  - 211은 2행, 1열짜리 plot을 만들겠다는 뜻
  - 마지막 1은 첫번째 plot
  - 마지막에 2가 오면 두번째 plot이라는 뜻임.

```
25]: def f(t):  
    return np.exp(-t) * np.cos(2*np.pi*t)  
  
t1 = np.arange(0.0, 5.0, 0.1)  
t2 = np.arange(0.0, 5.0, 0.02)
```

```
plt.figure()  
plt.subplot(211)  
plt.plot(t1, f(t1), 'bo', t2, f(t2), 'k')
```

```
plt.subplot(212)  
plt.plot(t2, np.cos(2*np.pi*t2), 'r--')  
plt.show()
```



# Subplot 다루기 (2)

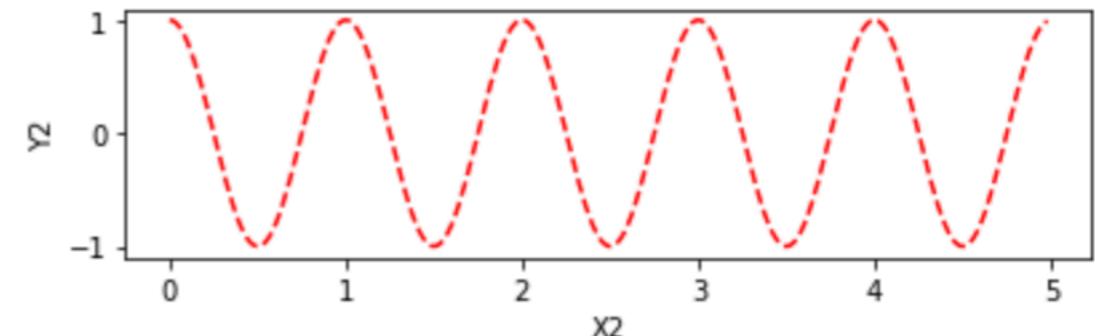
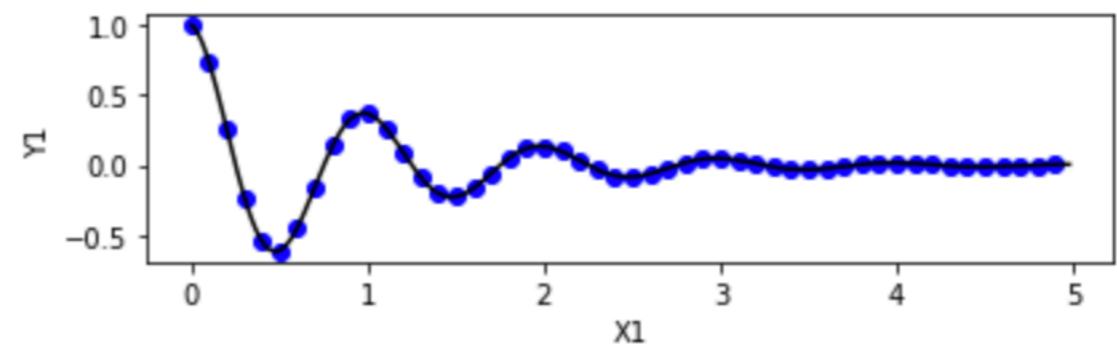
- subplot에도 xlabel을 넣을 수 있다.
- 오른쪽 예제에서 각 subplot의 xlabel과 ylabel을 넣어주었다.
- **matplotlib에서는 current plot을 암묵적으로 가정하고 있다는 점을 유념하자!**
- 마지막의 **plt.tight\_layout()** 은 그래프 모양을 그림 파일에 꽉 차게 만들어서 좀 더 보기 좋게 만들어준다.
  - 대부분의 경우 사용하는 것이 나음.

In [28]:

```
t1 = np.arange(0.0, 5.0, 0.1)
t2 = np.arange(0.0, 5.0, 0.02)
```

```
plt.figure()
plt.subplot(211)
plt.plot(t1, f(t1), 'bo', t2, f(t2), 'k')
plt.xlabel('X1')
plt.ylabel('Y1')
```

```
plt.subplot(212)
plt.plot(t2, np.cos(2*np.pi*t2), 'r--')
plt.xlabel('X2')
plt.ylabel('Y2')
plt.tight_layout()
```



# 그림 저장하기

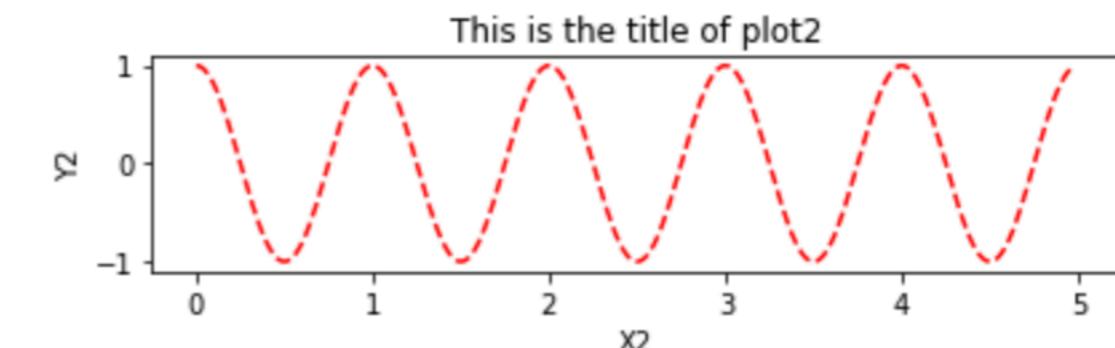
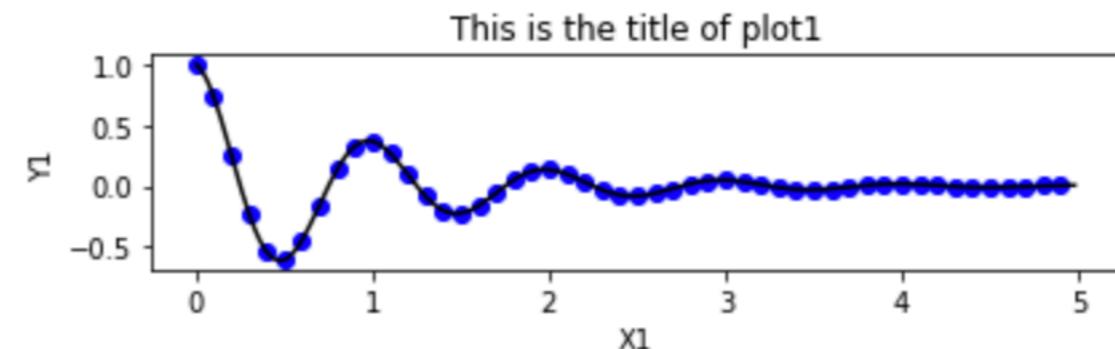
- **plt.title** 메소드를 사용하면 plot의 제목을 지정할 수 있다.
- **plt.savefig** 메소드를 사용하면 그림 파일로 저장 할 수 있다.
- 지원하는 format
  - eps, ps, tif, png, pdf, svg 등등 대부분의 그림 파일 형식을 지원한다.
  - 논문에 사용될 그래프라면 벡터 그래픽인 eps, pdf, tif를 주로 이용하자!
  - 주의점: plt.show() 다음에 사용하면 빈 그림을 만든다.

In [33]:

```
plt.figure()  
plt.subplot(211)  
plt.plot(t1, f(t1), 'bo', t2, f(t2), 'k')  
plt.xlabel('X1')  
plt.ylabel('Y1')  
plt.title('This is the title of plot1')
```

```
plt.subplot(212)  
plt.plot(t2, np.cos(2*np.pi*t2), 'r--')  
plt.xlabel('X2')  
plt.ylabel('Y2')
```

```
plt.title('This is the title of plot2')  
plt.tight_layout()  
plt.savefig('test.png', format='png')
```



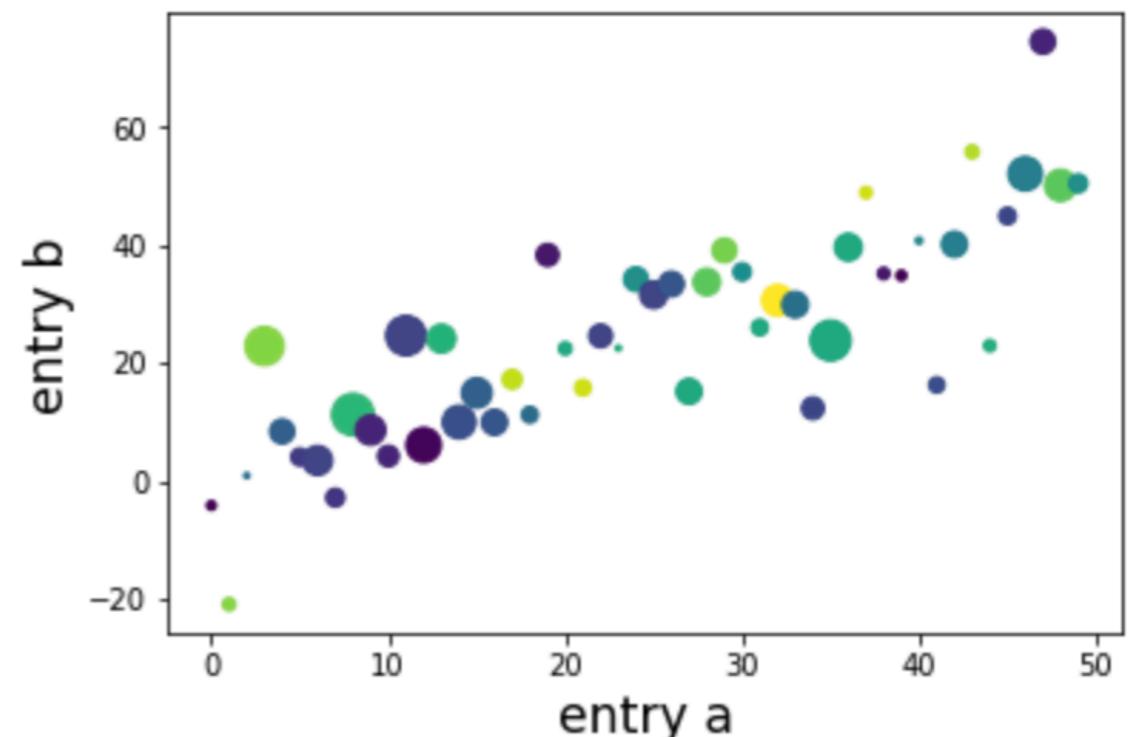
# Scatter plot

- Scatter plot: x와 y 데이터에 따라서 흩어진 점으로 그린 plot
- `plt.scatter(x, y, c=<color>, s=<area>)`
- 기본적으로 list 또는 array를 data로 가진다.
- 오른쪽 예제 처럼 dictionary를 data로 사용할 수 있다.
- c 또는 color argument를 통해서 색깔을 조정 할 수 있다.
- s 또는 area argument를 통해서 데이터 포인트의 크기를 조정할 수 있다.
- x, y의 위치, 색, 점의 크기를 통해서 4가지의 데이터를 동시에 표현할 수 있다.

[35]:

```
data = {'a': np.arange(50),
        'c': np.random.randint(0, 50, 50),
        'd': np.random.randn(50)}
data['b'] = data['a'] + 10 * np.random.randn(50)
data['d'] = np.abs(data['d']) * 100
```

```
plt.scatter('a', 'b', c='c', s='d', data=data)
plt.xlabel('entry a', fontsize='xx-large')
plt.ylabel('entry b', fontsize='xx-large')
plt.show()
```



# Scatter plot - 예시

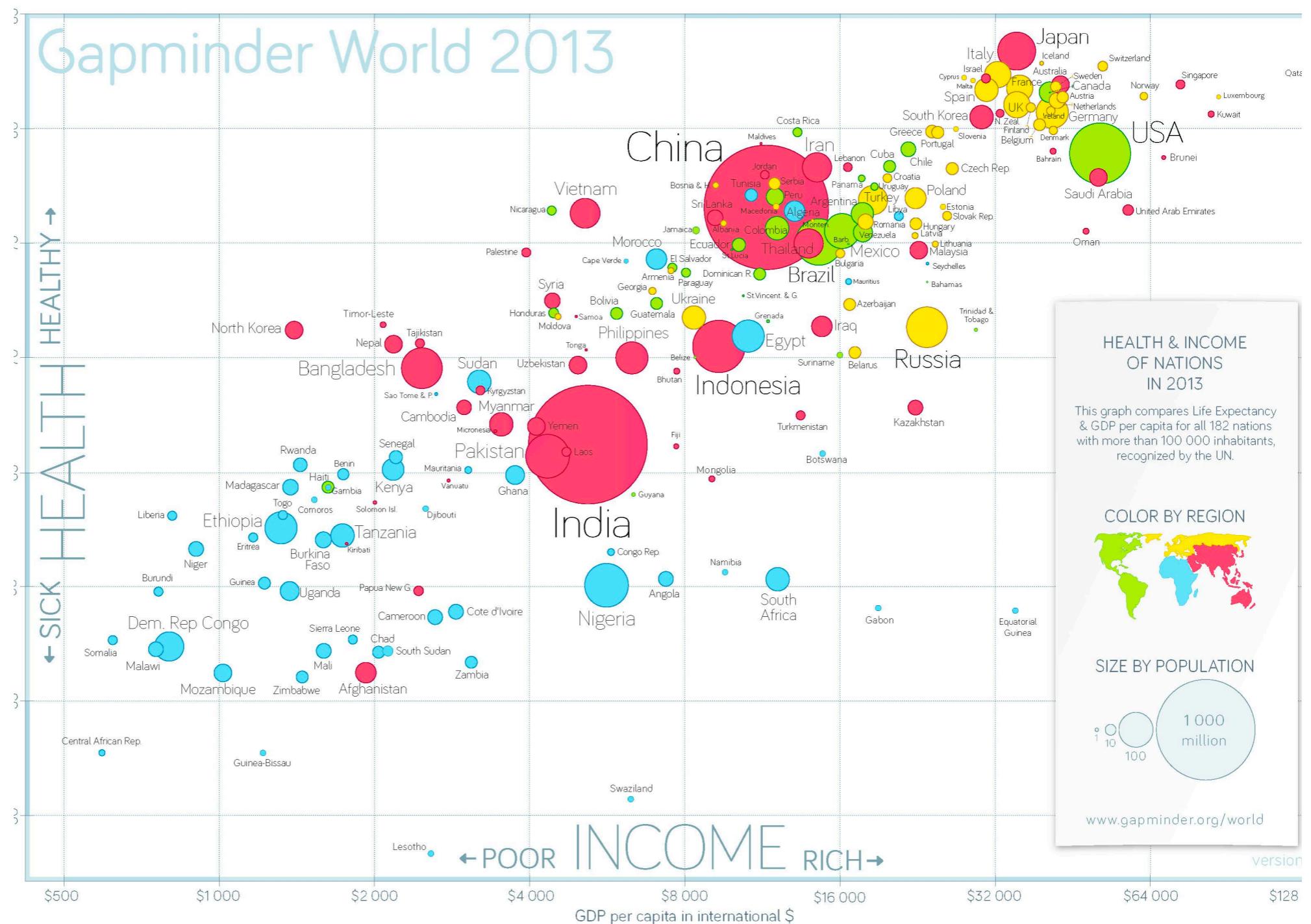
GDP와 Health 사이의 관계

x축: income

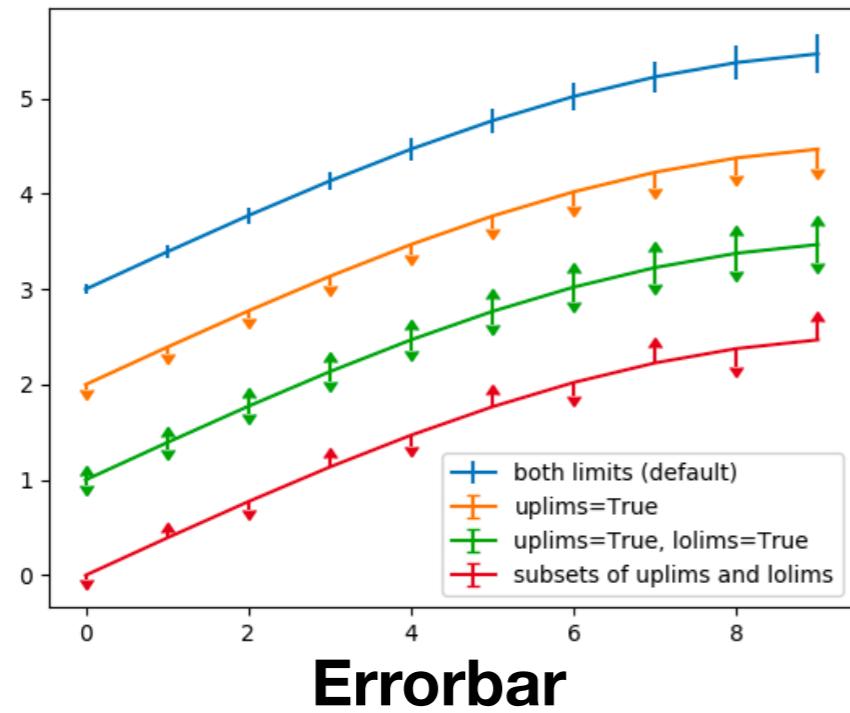
y축: health

원의 크기: 인구

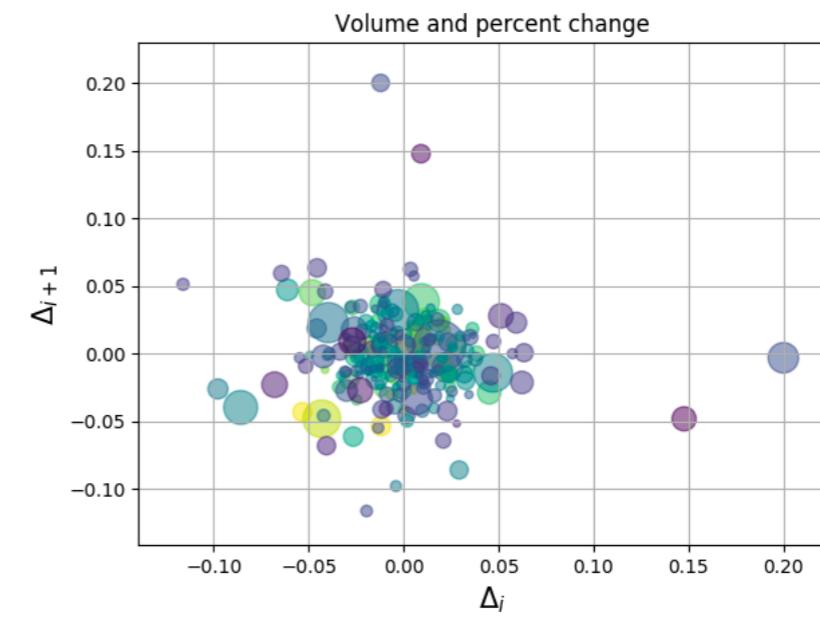
색: region



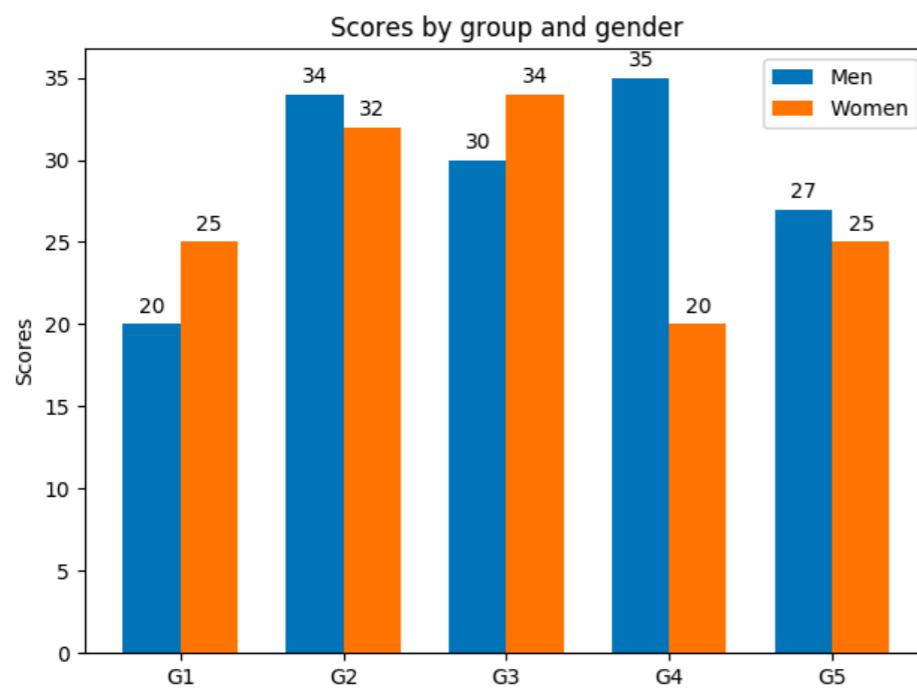
# matplotlib로 그릴 수 있는 그래프들



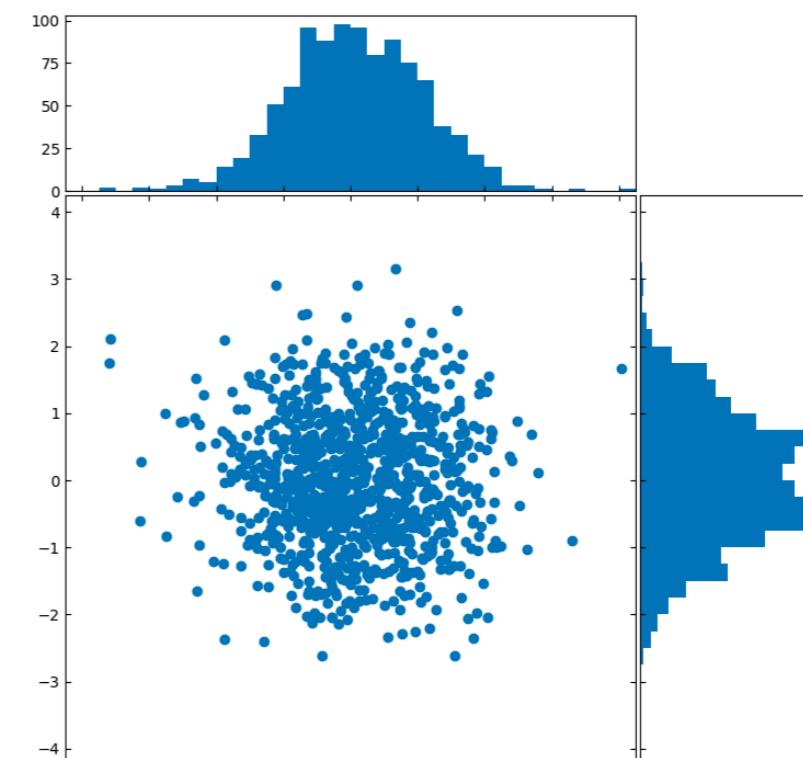
Errorbar



Scatter plot



Bar chart



Histogram