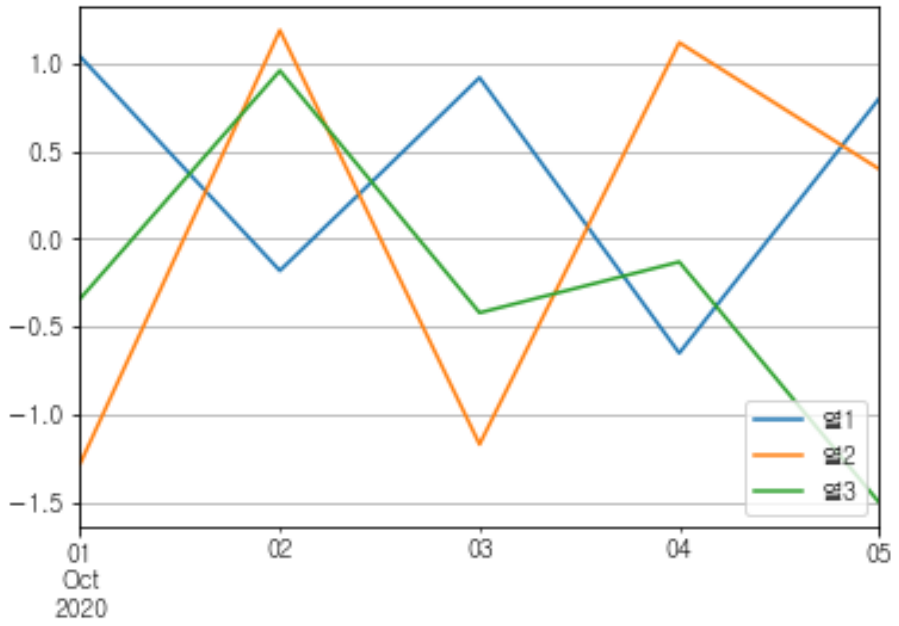
Matplotlib

Matplotlib

- 그래프를 그리거나, 분포를 보여주는 등 시각화를 위한 파이썬 패키지
- 연구용으로 많이 쓰인 MATLAB의 코드 스타일을 모방 (Matlab-style Plotting Library)
- 스위스 맥가이버 칼과 같이 다재다능하나, 사용하기는 약간 불편함



matplotlib

Matplotlib – 기본 꺾은선 차트 그리기

- Matplotlib 라이브러리 임포트

```
from matplotlib import pyplot as plt
%matplotlib inline

# 한글 폰트 설정 (윈도우 Malgun Gothic,  리눅스&Mac AppltGothic)
plt.rcParams['font.family'] = ['Malgun Gothic', 'AppleGothic']
```

- X축, Y축 데이터 준비

```
# 현대자동차 20일간 주가 변화
# X축 : 날짜
날짜_리스트 = [
            '2018-06-01',
            '2018-06-04',
            '2018-06-05',
            '2018-06-07',
            '2018-06-08',
            '2018-06-11',
            '2018-06-12',
            '2018-06-14',
            '2018-06-15',
            '2018-06-18',
           ]
```

```
# Y축 : 날짜별 주가
날짜별_주가_리스트 = [
                  140000,
                  143000,
                  142500,
                  142500,
                  141000,
                  139500,
                  140500,
                  135000,
                  134000,
                  135000,
                 ]
```

Matplotlib – 기본 꺾은선 차트 그리기

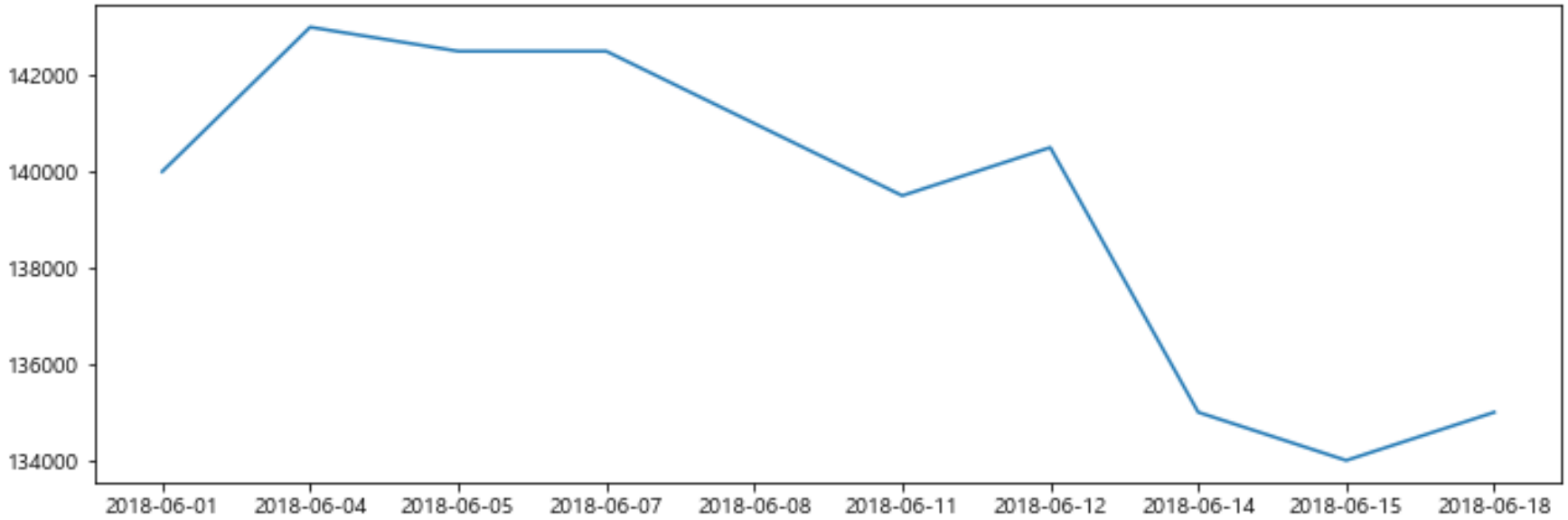
- plt.plot()으로 꺾은선 차트 그리기

```
plt.plot( [X축 데이터], [Y축 데이터] )
```

X축 데이터와 Y축 데이터를 가지고 꺾은선 차트를 그리는(plot) 코드

데이터 X와 Y의 길이는 같아야 합니다.

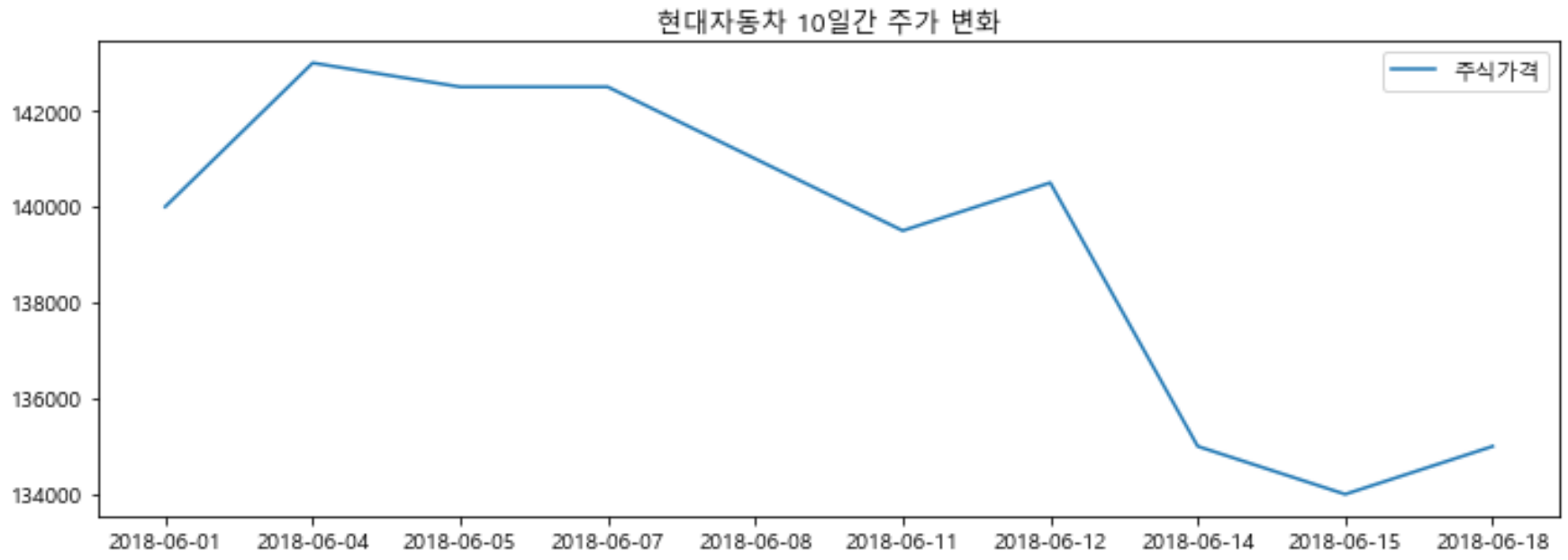
```
plt.figure(figsize=(12, 4))
plt.plot(날짜_리스트, 날짜별_주가_리스트)
plt.show() # 그래프 출력
```



Matplotlib – 기본 꺾은선 차트 그리기

- plt.title()로 차트 제목 추가하기
- label 및 plt.legend()로 범례 추가하기

```
plt.figure(figsize=(12, 4))
plt.plot(날짜_리스트, 날짜별_주가_리스트, label='주식가격')
plt.title('현대자동차 10일간 주가 변화')
plt.legend() # 범례 설정
plt.show() # 그래프 출력
```



Matplotlib – 기본 Bar 차트 그리기

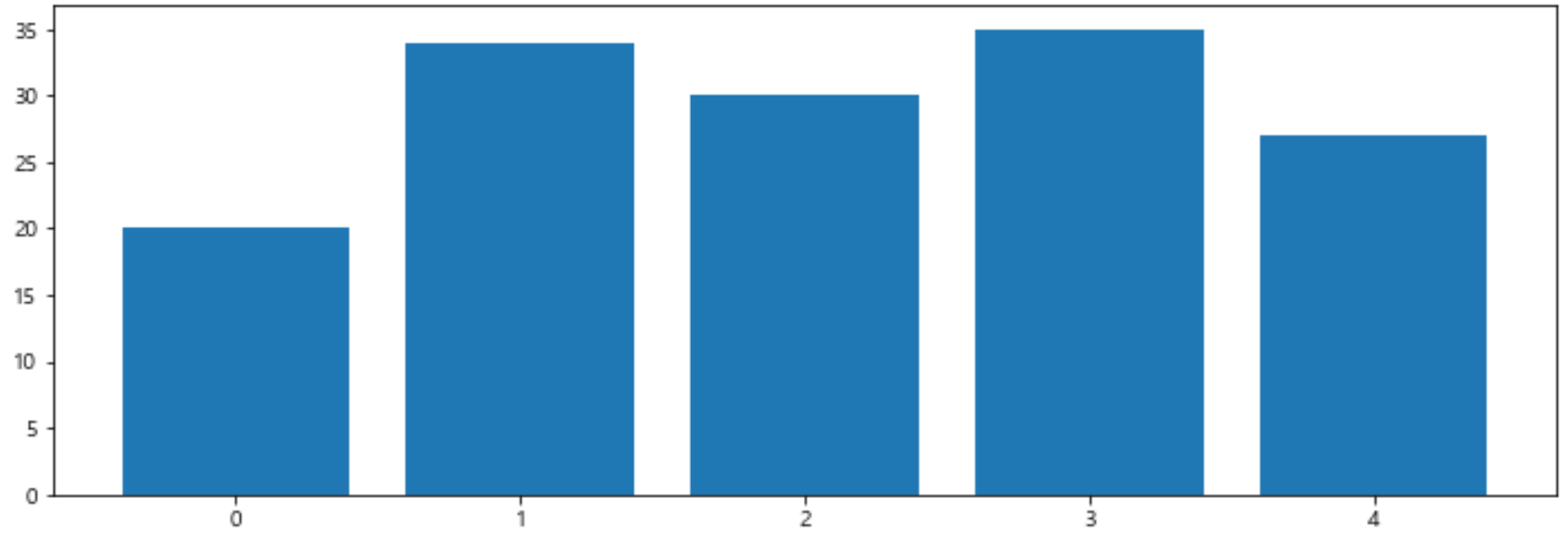
- plt.bar()로 Bar 차트 그리기

```
plt.bar( [X축 데이터], [Y축 데이터], width=바너비 )
```

X축 데이터와 Y축 데이터를 가지고 바 차트를 그리는(plot) 코드

데이터 X와 Y의 길이는 같아야 합니다.

```
바너비 = 0.8

plt.figure(figsize=(12,4))
plt.bar(X, 광고별_남성_클릭횟수, width=바너비, label='남성')
plt.show()
```



wanted

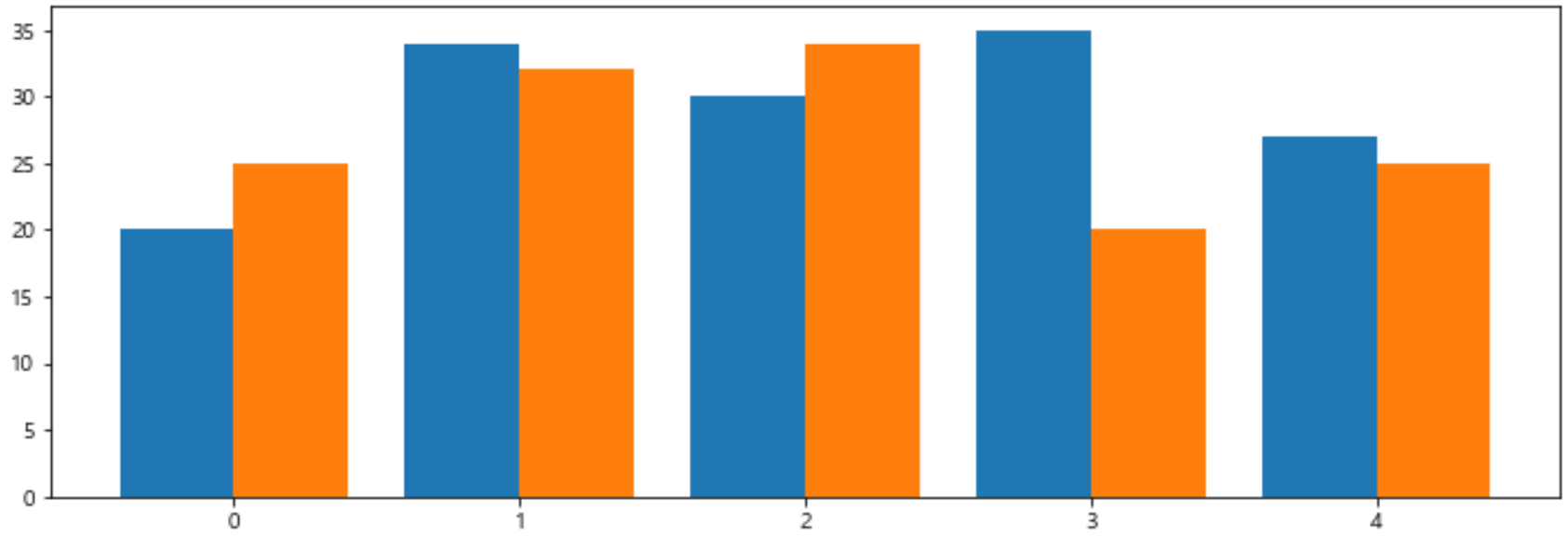
Matplotlib – 기본 Bar 차트 그리기

- 두개의 bar 그래프를 그리기 위해, X 위치를 계산합니다.

```
바너비 = 0.4

print(X)
print(X - 바너비/2)
print(X + 바너비/2)
```

```
[0 1 2 3 4]
[-0.2  0.8  1.8  2.8  3.8]
[0.2 1.2 2.2 3.2 4.2]
```

```
plt.figure(figsize=(12,4))
plt.bar(X - 바너비/2, 광고별_남성_클릭횟수, width=바너비)
plt.bar(X + 바너비/2, 광고별_여성_클릭횟수, width=바너비)
plt.show()
```



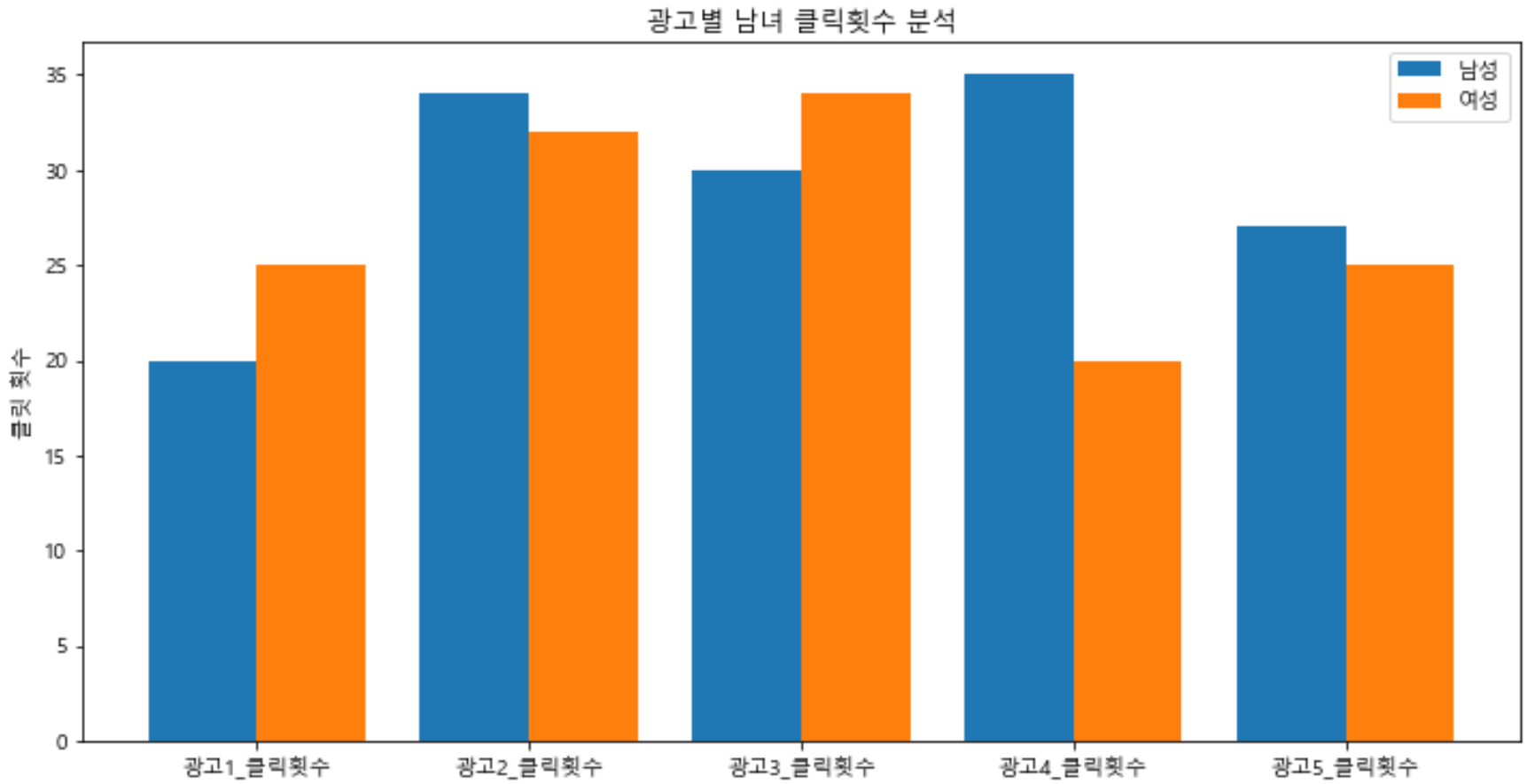
Matplotlib – 기본 Bar 차트 그리기

- bar()안의 label과 plt.legend()로 범례를 출력합니다.
- plt.title()로 차트 제목을 입력합니다.
- plt.ylabel()로 Y축 제목을 입력합니다.
- plt.xticks()로 X 값 및 X 값 레이블을 입력합니다.

```
바너비 = 0.4

plt.figure(figsize=(12,6))
plt.bar(X - 바너비/2, 광고별_남성_클릭횟수, width=바너비, label='남성')
plt.bar(X + 바너비/2, 광고별_여성_클릭횟수, width=바너비, label='여성')

plt.title('광고별 남녀 클릭횟수 분석')
plt.ylabel('클릿 횟수')
plt.xticks(X, labels=X축_레이블)
plt.legend()

plt.show()
```

Matplotlib – 기본 Bar 차트 그리기

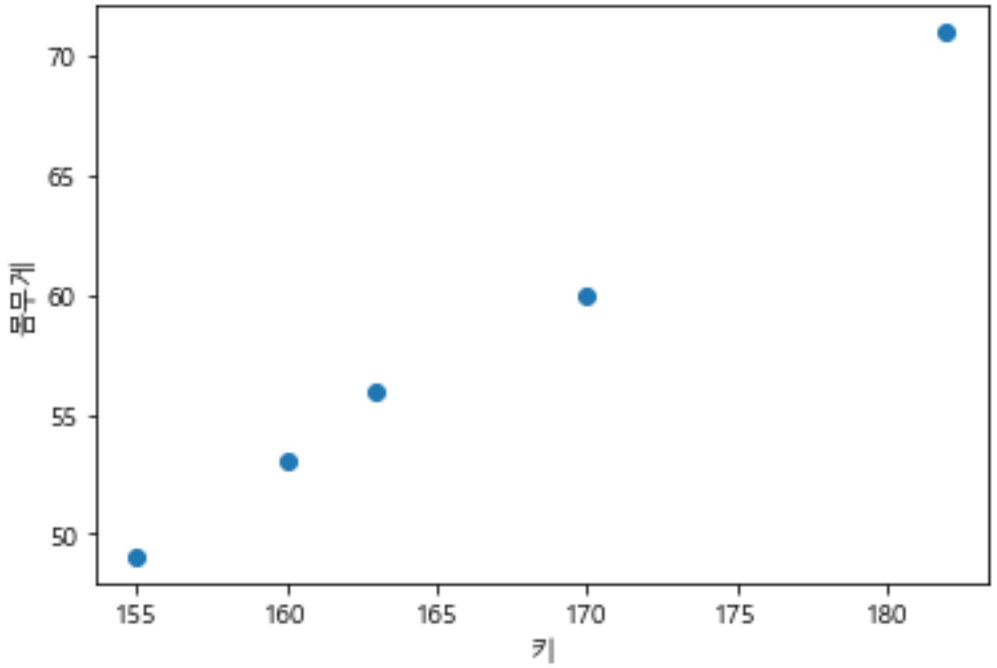
- 최종 완성된 Bar 차트



Matplotlib – 산점도 (Scatter) 차트 그리기

- X와 Y의 관계를 살펴보는 산점도 차트

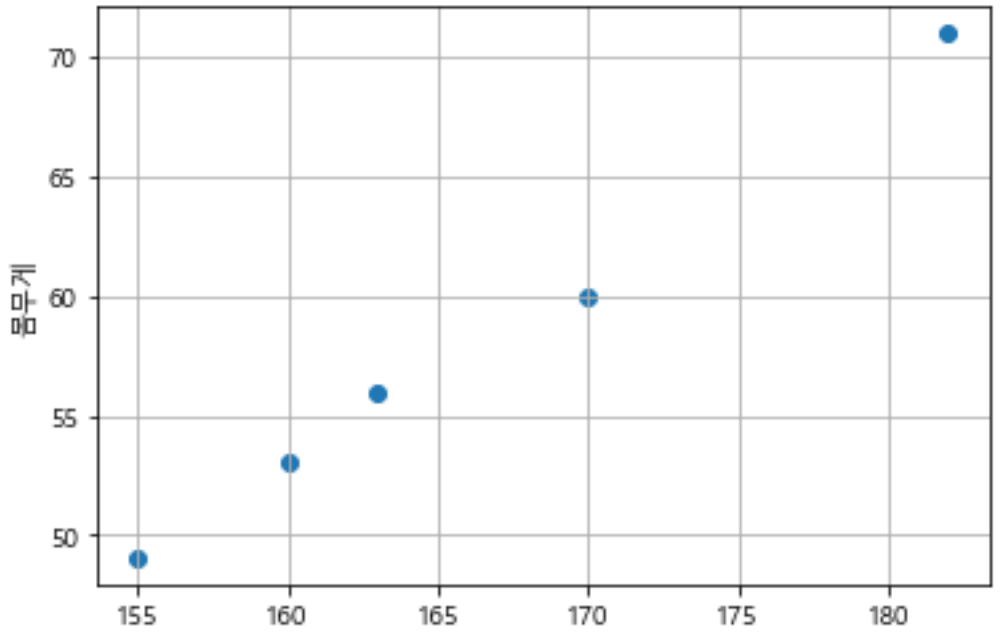
```
X축_데이터 = [170, 182, 155, 160, 163] # 키
Y축_데이터 = [60, 71, 49, 53, 56]      # 몸무게

plt.scatter(X축_데이터, Y축_데이터)
plt.xlabel('키')
plt.ylabel('몸무게')

plt.show()
```



Matplotlib – 산점도 (Scatter) 차트 그리기

- plt.grid()로 격자 출력하기

```
X축_데이터 = [170, 182, 155, 160, 163] # 키
Y축_데이터 = [60, 71, 49, 53, 56]      # 몸무게

plt.scatter(X축_데이터, Y축_데이터)
plt.xlabel('키')
plt.ylabel('몸무게')
plt.grid()

plt.show()
```



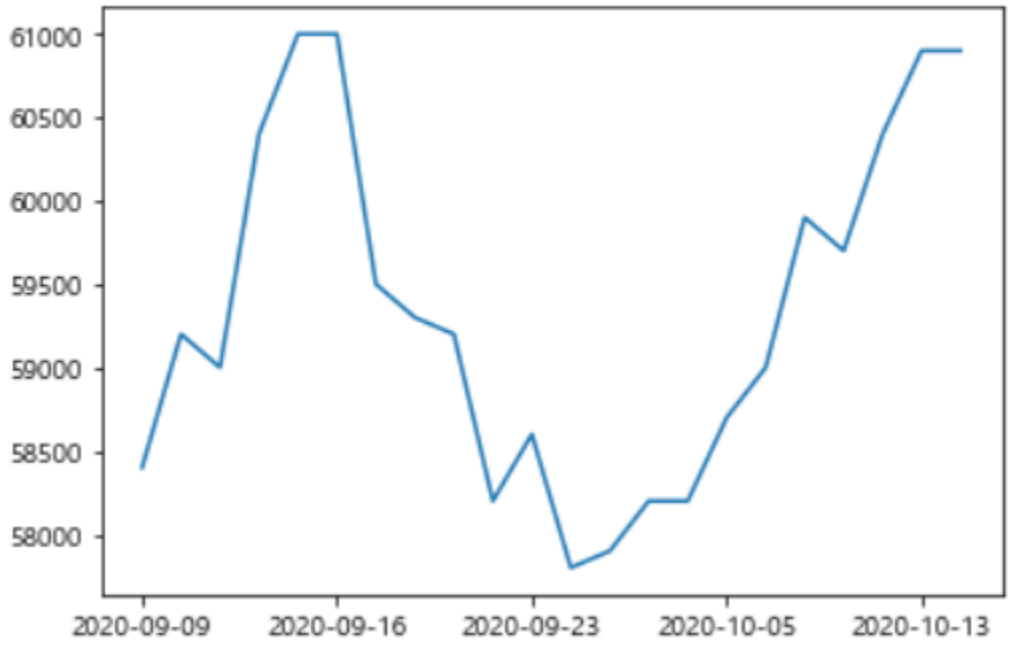
Matplotlib – pandas 내장 plot

- pandas에는 matplotlib으로 **차트를 그리는 기능이 내장**되어 있습니다.
- Series.plot(), DataFrame.plot() ← plot() 함수의 인자로, 아주 다양한 옵션들이 존재

```
my_df
```

Symbol	A005930	A005940	A005950	A005960	A005980	A005990
2020-09-09	58400.0	9190.0	9080.0	11000.0	671.0	7990.0
2020-09-10	59200.0	9280.0	9280.0	11150.0	671.0	7960.0
2020-09-11	59000.0	9270.0	9360.0	11050.0	671.0	8120.0
2020-09-14	60400.0	9370.0	9400.0	11200.0	671.0	8200.0
2020-09-15	61000.0	9470.0	9390.0	11350.0	671.0	8180.0
2020-09-16	61000.0	9530.0	9080.0	11250.0	671.0	8180.0
2020-09-17	59500.0	9320.0	8880.0	11000.0	671.0	8000.0
2020-09-18	59300.0	9320.0	9070.0	10850.0	671.0	8020.0
2020-09-21	59200.0	9200.0	8880.0	10700.0	671.0	7970.0
2020-09-22	58200.0	9040.0	8580.0	10600.0	671.0	7890.0
2020-09-23	58600.0	9060.0	8500.0	10750.0	671.0	7890.0
2020-09-24	57800.0	8940.0	8280.0	10500.0	671.0	7710.0
2020-09-25	57900.0	9070.0	8300.0	10500.0	671.0	7860.0
2020-09-28	58200.0	9080.0	8530.0	10800.0	671.0	7940.0
2020-09-29	58200.0	9140.0	8670.0	10900.0	671.0	7950.0
2020-10-05	58700.0	9400.0	8880.0	11050.0	671.0	8010.0
2020-10-06	59000.0	9280.0	11500.0	10950.0	671.0	7990.0
2020-10-07	59900.0	9370.0	11550.0	10950.0	671.0	8010.0
2020-10-08	59700.0	9390.0	11800.0	10900.0	671.0	8010.0
2020-10-12	60400.0	9510.0	11700.0	11200.0	671.0	8010.0
2020-10-13	60900.0	9450.0	12300.0	11200.0	671.0	7980.0
2020-10-14	60900.0	9350.0	11600.0	11150.0	671.0	7890.0

```
my_df.loc[:,'A005930'].plot();  # Series 그래프
```



wanted

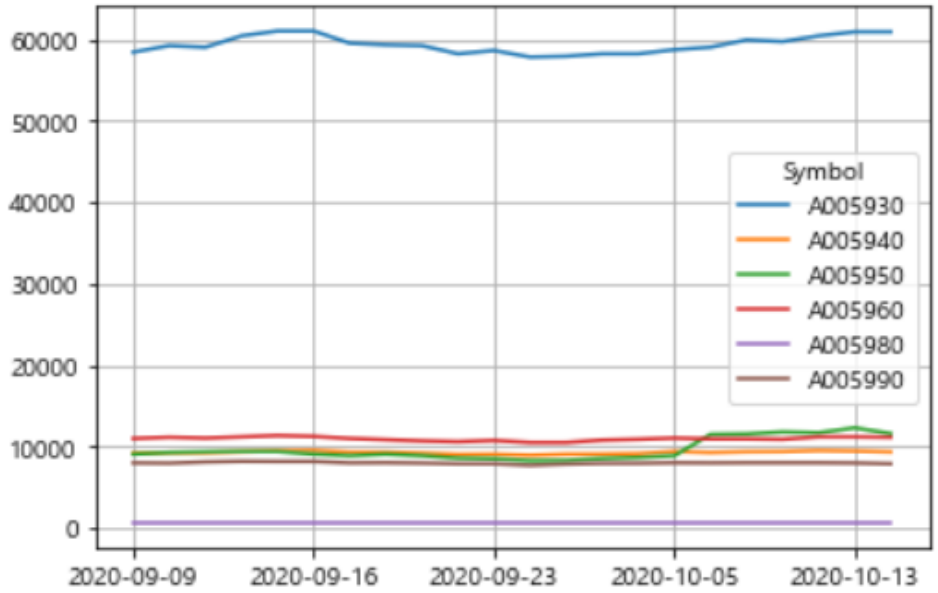
Matplotlib – pandas 내장 plot

- plot의 주요 옵션들

인자	역할
figsize	그래프 크기
kind	그래프 종류
title	제목
xlabel	x축 제목
ylabel	y축 제목
legend	범례
grid	격자
rot	눈금 회전
subplots	복수 그래프

- grid를 옵션으로 주기

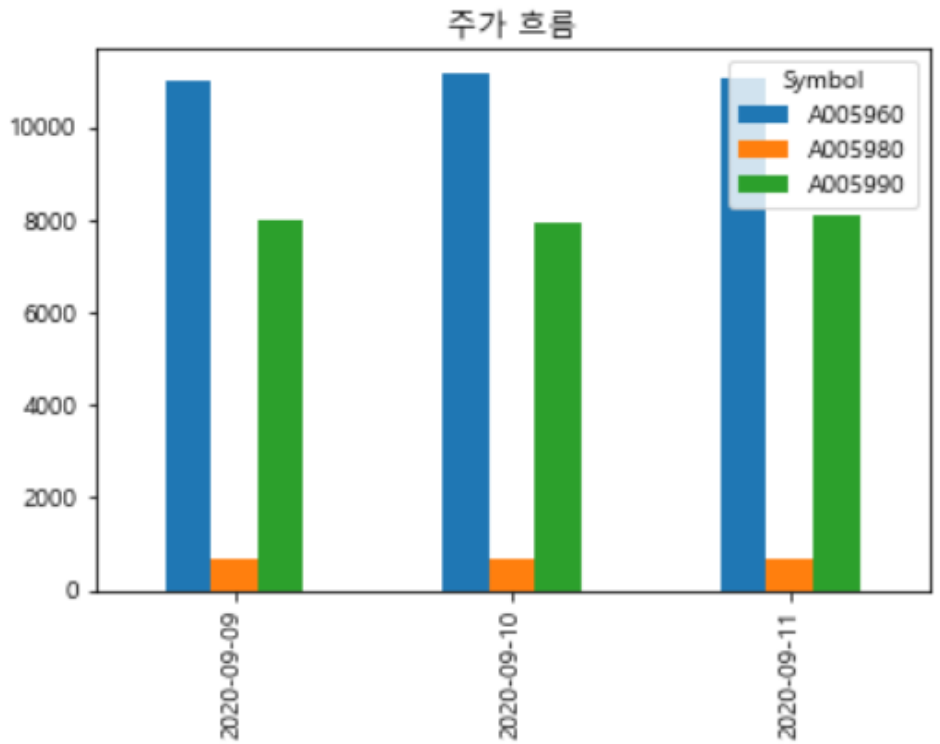
```
my_df.plot(grid=True);  # DataFrame 그래프
```



Matplotlib – pandas 내장 plot

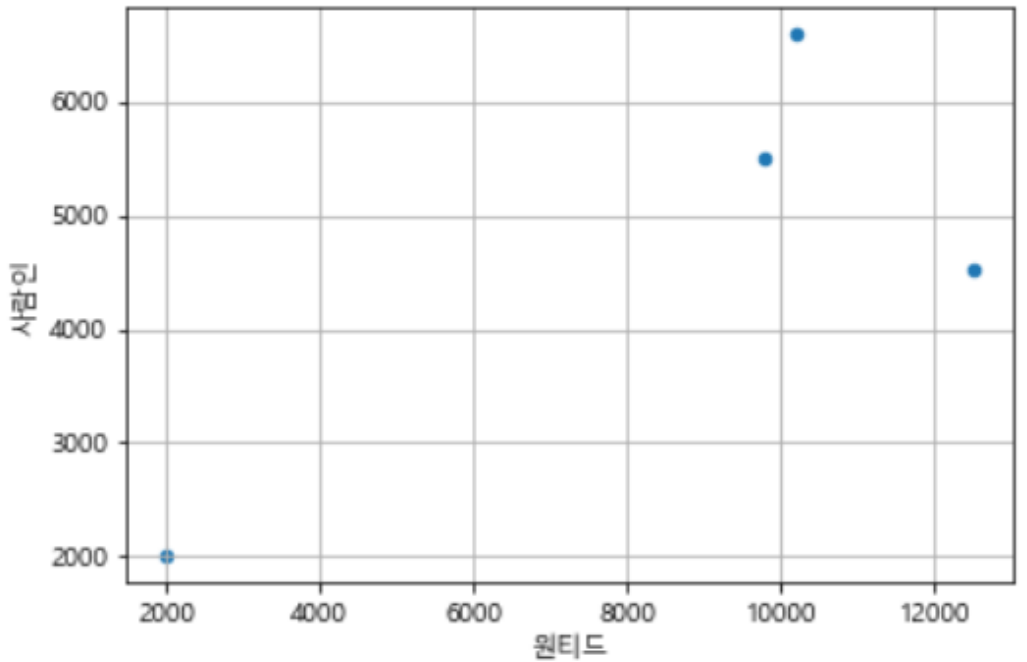
- 차트 종류 변경하기

```
my_df_2.plot(kind='bar', title='주가 흐름');
```



- 데이터를 column 명으로 입력하기

```
trafic_data.plot(kind='scatter', grid=True,
                 x='원티드', y='사람인');
```

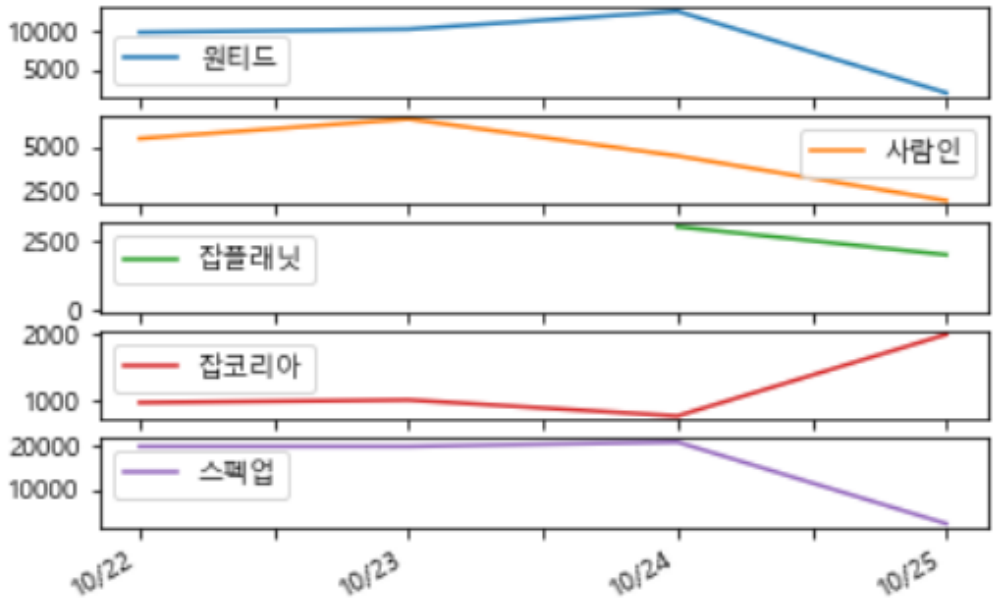


wanted

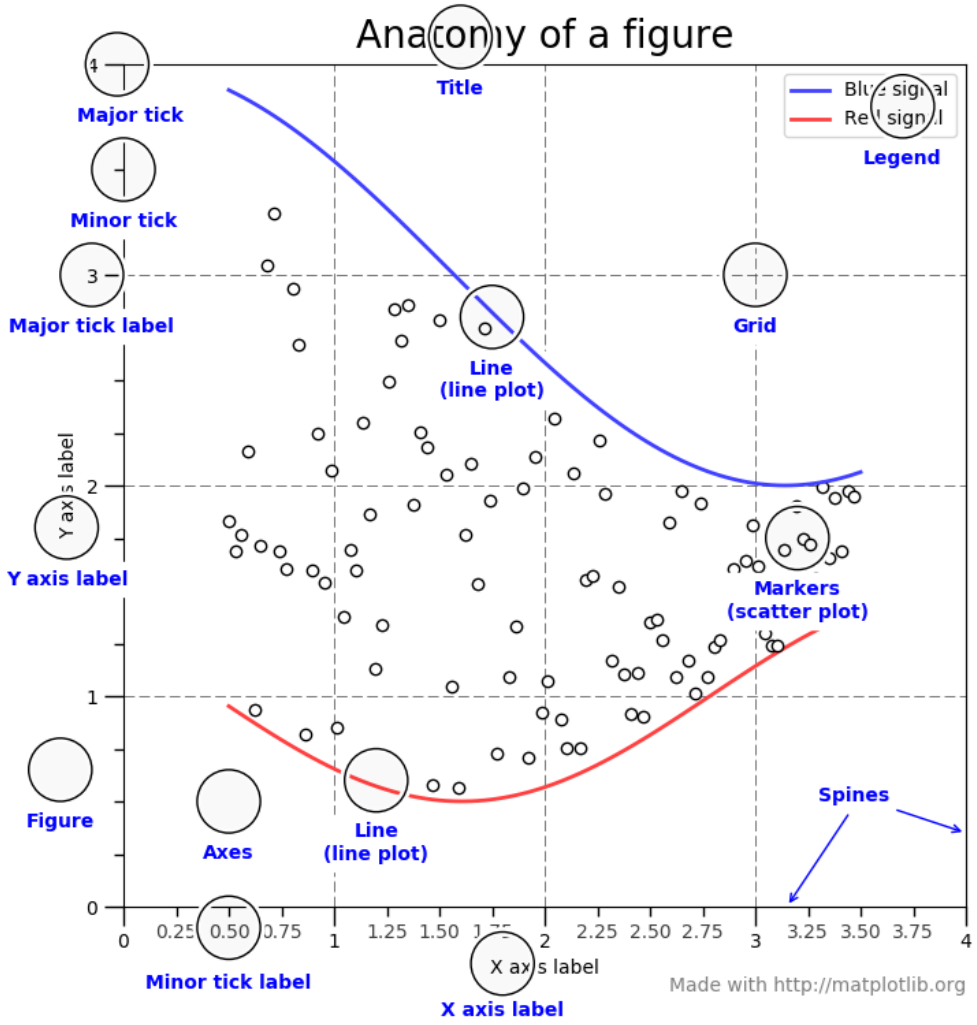
Matplotlib – pandas 내장 plot

- subplots를 이용하여 제각기 차트 그리기

```
trafic_data.plot(subplots=True);
```



Matplotlib 고급 – Figure 객체 구조

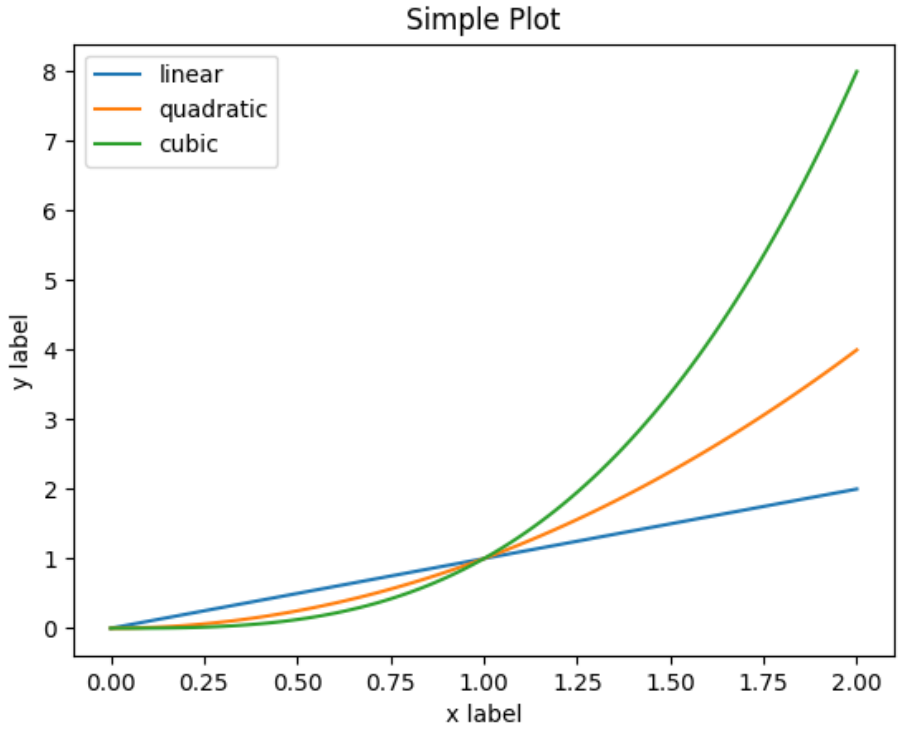


- 주요 구성 요소들
  - Figure : 차트가 그려지는 큰 도화지
  - Axes : 도화지 안에 하나의 그래프를 그릴 공간
  - xlabel, xticks : X축의 제목 및 눈금
  - ylabel, yticks : Y축의 제목 및 눈금
  - Title : Axes에 그려진 그래프의 제목
  - Grid : 격자
  - Legend : 범례

Matplotlib 고급 – 코딩 스타일 “OO”

- Object Oriented 방식으로, 언급된 구성요소들에 접근하여 플로팅 하는 Code-Style

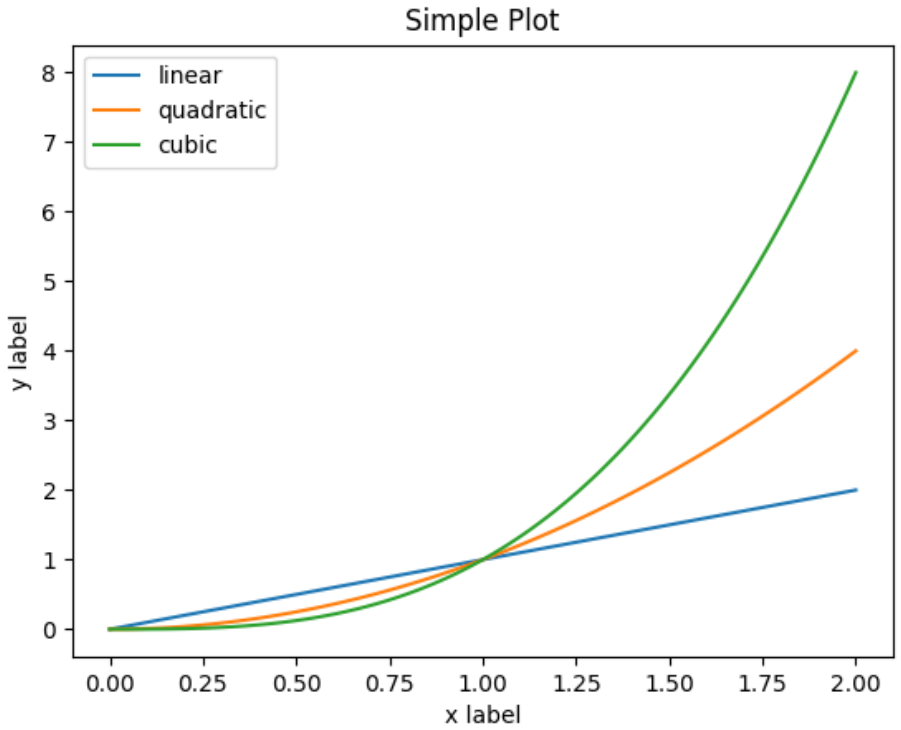
```
x = np.linspace(0, 2, 100)

# Note that even in the OO-style, we use `.pyplot.figure` to create the figure.
fig, ax = plt.subplots()  # Create a figure and an axes.
ax.plot(x, x, label='linear')  # Plot some data on the axes.
ax.plot(x, x**2, label='quadratic')  # Plot more data on the axes...
ax.plot(x, x**3, label='cubic')  # ... and some more.
ax.set_xlabel('x label')  # Add an x-label to the axes.
ax.set_ylabel('y label')  # Add a y-label to the axes.
ax.set_title("Simple Plot")  # Add a title to the axes.
ax.legend()  # Add a legend.
```



Matplotlib 고급 – 코딩 스타일 “pyplot”

- pyplot 방식으로, 현재 그려지고 있는 요소에 플로팅이 적용되는 Code-Style

```
x = np.linspace(0, 2, 100)

plt.plot(x, x, label='linear')  # Plot some data on the (implicit) axes.
plt.plot(x, x**2, label='quadratic')  # etc.
plt.plot(x, x**3, label='cubic')
plt.xlabel('x label')
plt.ylabel('y label')
plt.title("Simple Plot")
plt.legend()
```



Matplotlib 고급 – subplots과 ax로 그래프 그리기

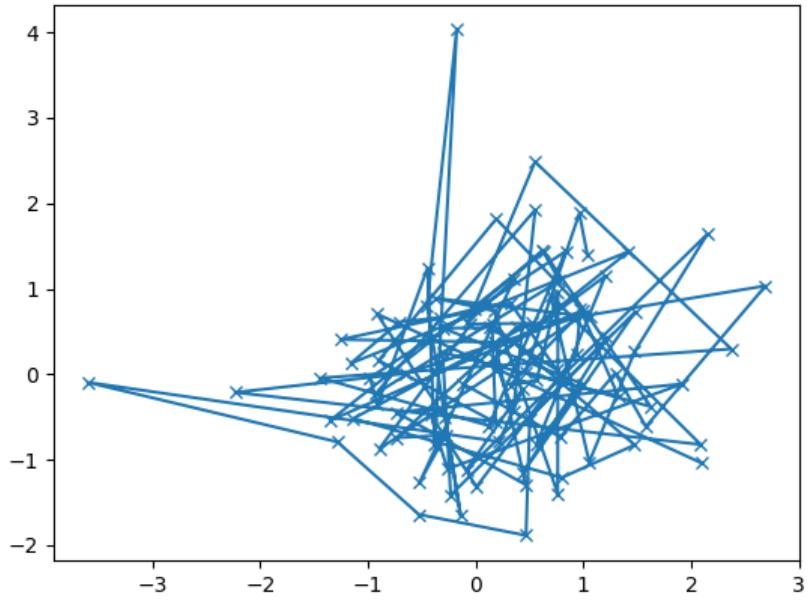
- subplots는 그래프를 그릴 수 있는 실질적인 축 ax를 요구한 개수만큼 생성합니다
- 생성된 ax에 그래프를 입력하여 플로팅을 수행합니다

```python
def my_plotter(ax, data1, data2, param_dict):
    """
    A helper function to make a graph

    Parameters
    ----------
    ax : Axes
        The axes to draw to

    data1 : array
       The x data

    data2 : array
       The y data

    param_dict : dict
       Dictionary of kwargs to pass to ax.plot

    Returns
    -------
    out : list
        list of artists added
    """
    out = ax.plot(data1, data2, **param_dict)
    return out
```

ax에 그래프를 그리는 함수

<하나의 ax를 생성하고 플로팅>

```python
data1, data2, data3, data4 = np.random.randn(4, 100)
fig, ax = plt.subplots(1, 1)
my_plotter(ax, data1, data2, {'marker': 'x'})
```



wanted

Matplotlib 고급 – subplots과 ax로 그래프 그리기

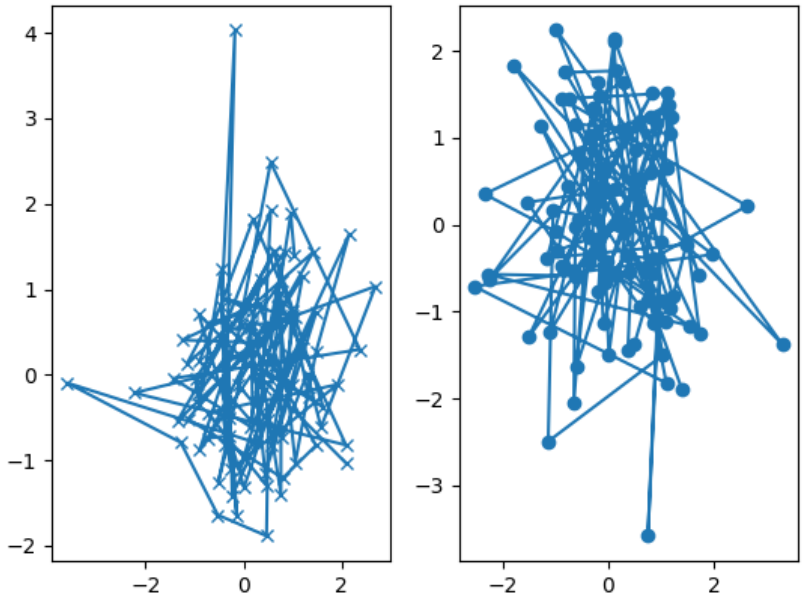
- subplots는 그래프를 그릴 수 있는 실질적인 축 ax를 요구한 개수만큼 생성합니다
- 생성된 ax에 그래프를 입력하여 플로팅을 수행합니다

```python
def my_plotter(ax, data1, data2, param_dict):
    """
    A helper function to make a graph

    Parameters
    ----------
    ax : Axes
        The axes to draw to

    data1 : array
       The x data

    data2 : array
       The y data

    param_dict : dict
       Dictionary of kwargs to pass to ax.plot

    Returns
    -------
    out : list
        list of artists added
    """
    out = ax.plot(data1, data2, **param_dict)
    return out
```

ax에 그래프를 그리는 함수

<두개의 ax를 생성하고 플로팅>

```python
fig, (ax1, ax2) = plt.subplots(1, 2)
my_plotter(ax1, data1, data2, {'marker': 'x'})
my_plotter(ax2, data3, data4, {'marker': 'o'})
```

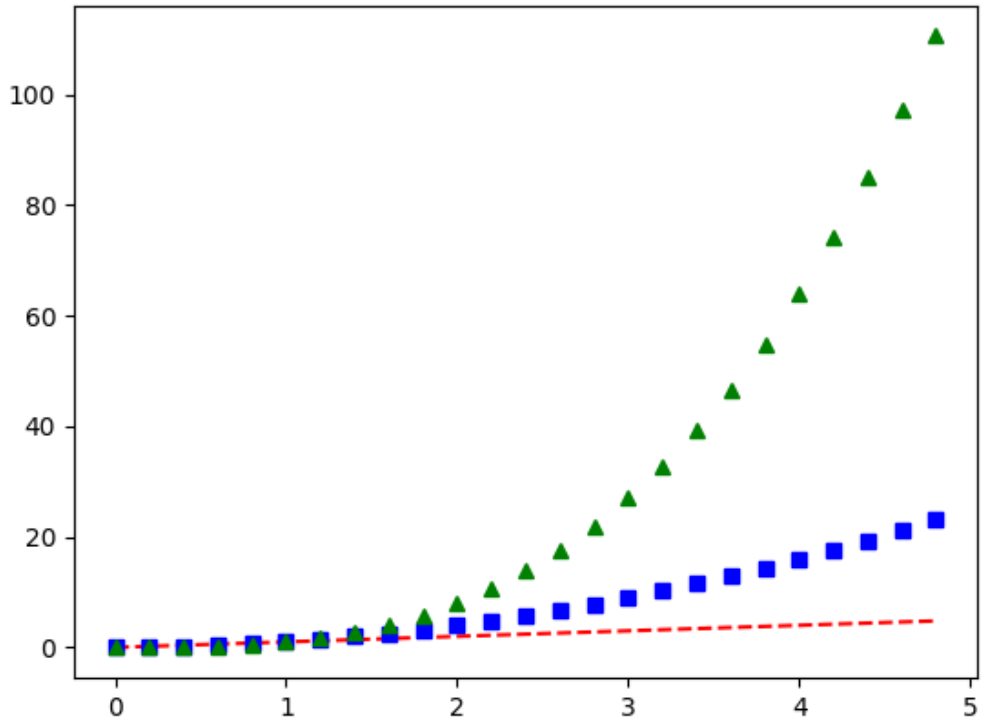


wanted

Matplotlib 고급 – 그래프 예시 1

- line 의 스타일 변경해보기

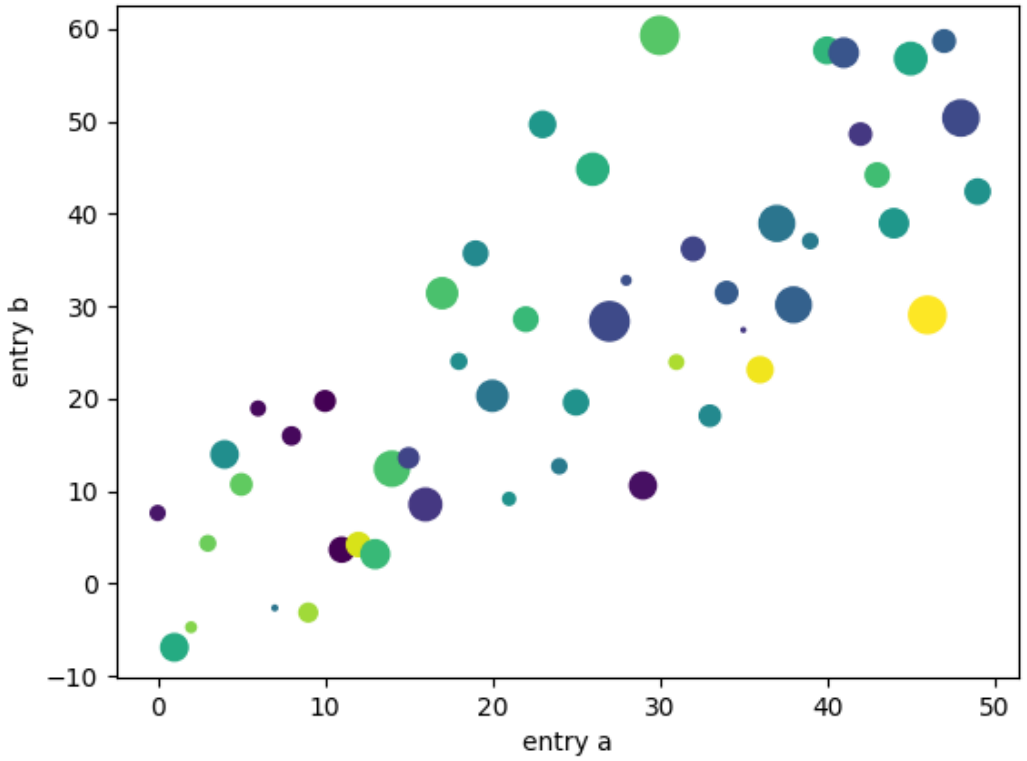
```
import numpy as np

# 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--', t, t**2, 'bs', t, t**3, 'g^')
plt.show()
```



Matplotlib 고급 – 그래프 예시 2

- scatter 차트에서 점의 크기를 활용하기

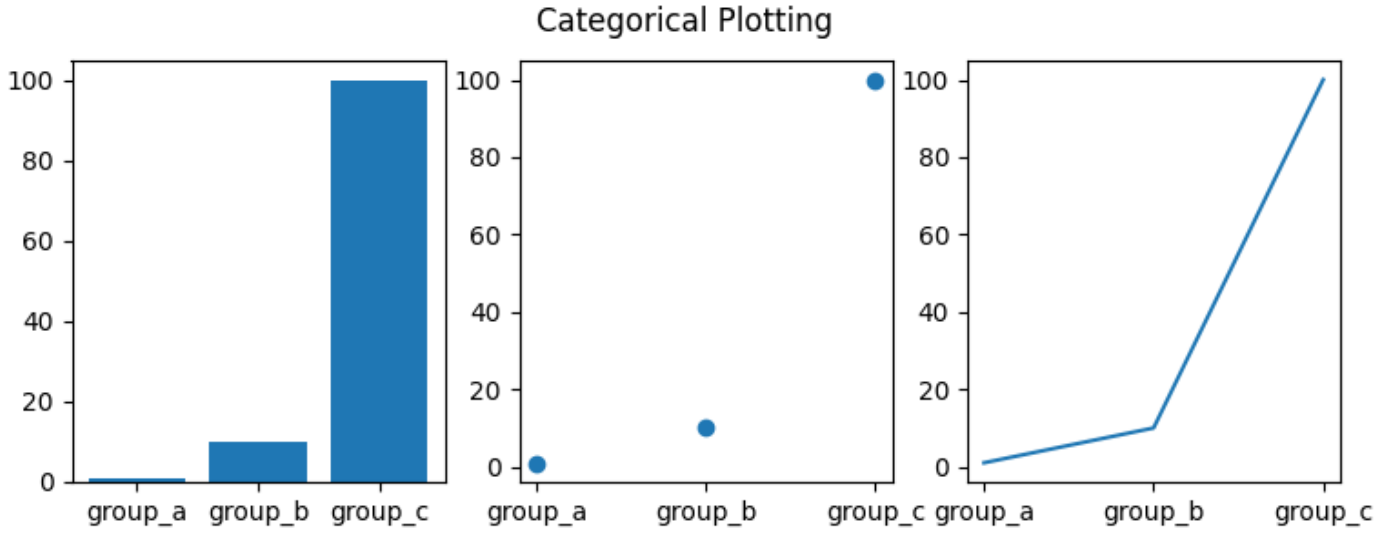
```
data = {'a': np.arange(50),
        'c': np.random.randint(0, 50, 50),
        'd': np.random.randn(50)}
data['b'] = data['a'] + 10 * np.random.randn(50)
data['d'] = np.abs(data['d']) * 100

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



Matplotlib 고급 – 그래프 예시 3

- 카테고리로 그래프 그려보기

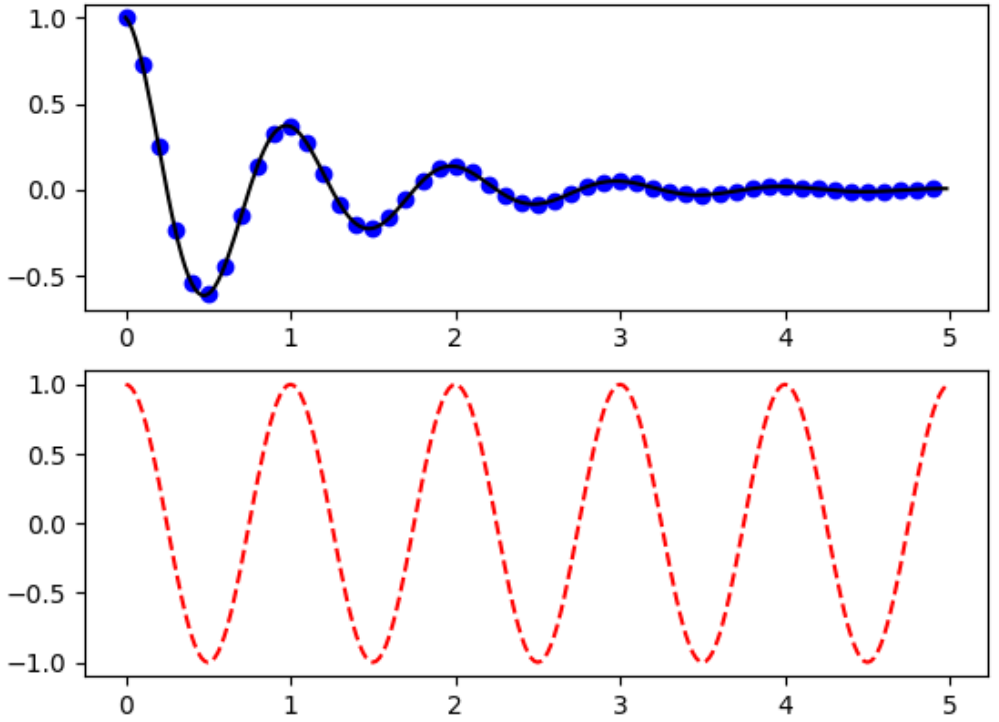
```
names = ['group_a', 'group_b', 'group_c']
values = [1, 10, 100]

plt.figure(figsize=(9, 3))

plt.subplot(131)
plt.bar(names, values)
plt.subplot(132)
plt.scatter(names, values)
plt.subplot(133)
plt.plot(names, values)
plt.suptitle('Categorical Plotting')
plt.show()
```



Matplotlib 고급 – 그래프 예시 4

- subplot으로 여러 개의 차트 그리기 (pyplot-style)

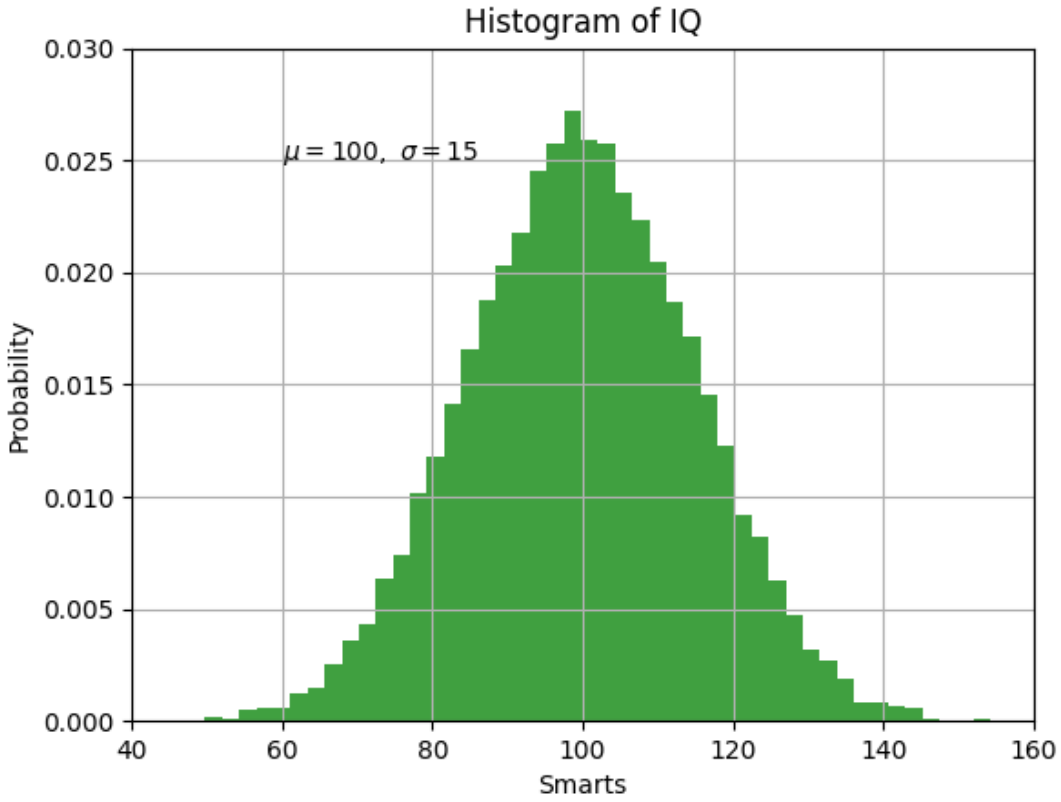
```
def f(t):
    return np.exp(-t) * np.cos(2*np.pi*t)

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()
```



Matplotlib 고급 – 그래프 예시 5

- 차트에 텍스트 넣기

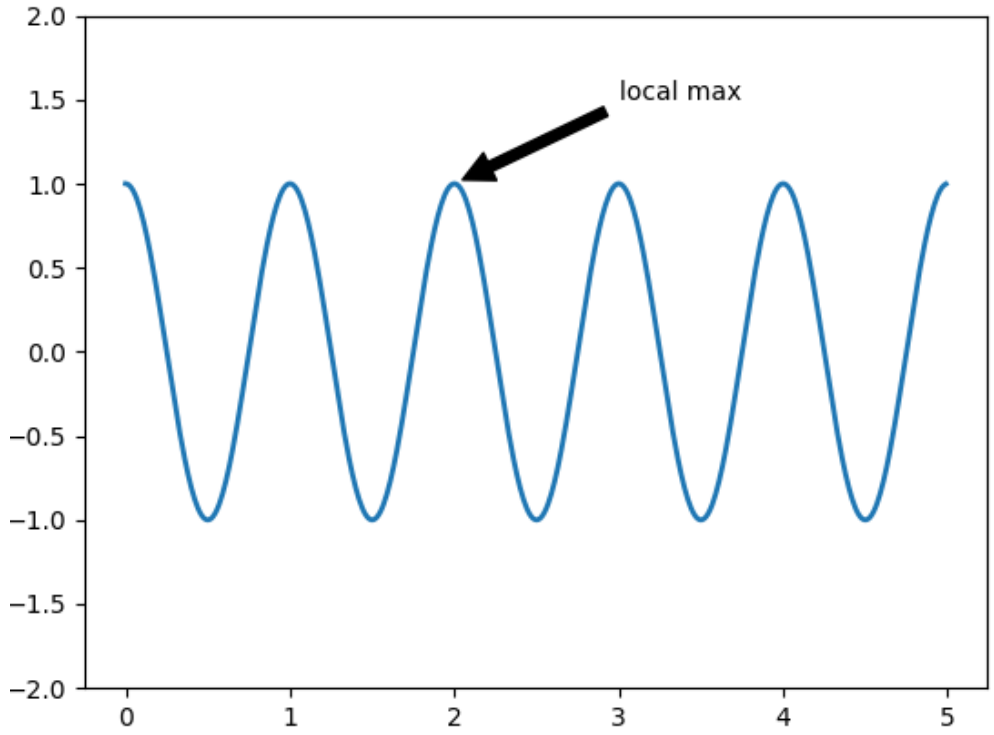
```
mu, sigma = 100, 15
x = mu + sigma * np.random.randn(10000)

# the histogram of the data
n, bins, patches = plt.hist(x, 50, density=1, facecolor='g', alpha=0.75)


plt.xlabel('Smarts')
plt.ylabel('Probability')
plt.title('Histogram of IQ')
plt.text(60, .025, r'$\mu=100,\ \sigma=15$')
plt.axis([40, 160, 0, 0.03])
plt.grid(True)
plt.show()
```



Matplotlib 고급 – 그래프 예시 6

- 차트에 고급 어노테이션 넣기

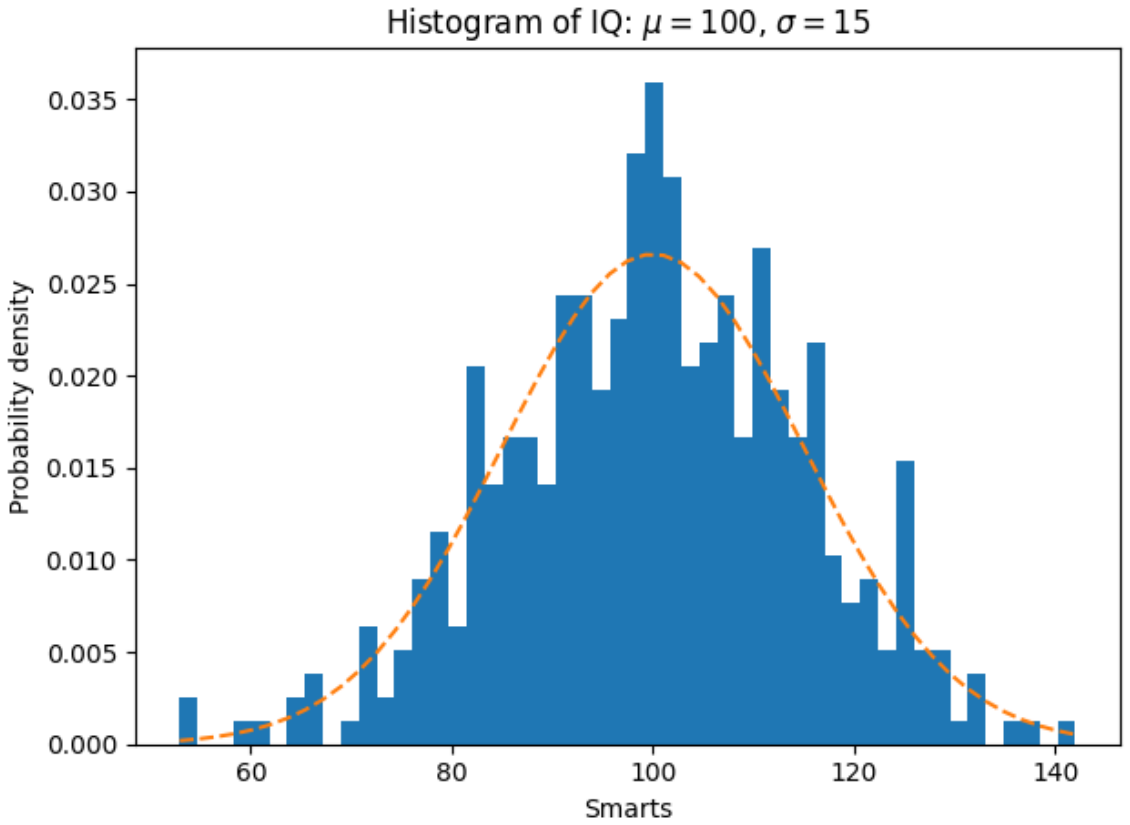
```
ax = plt.subplot()

t = np.arange(0.0, 5.0, 0.01)
s = np.cos(2*np.pi*t)
line, = plt.plot(t, s, lw=2)

plt.annotate('local max', xy=(2, 1), xytext=(3, 1.5),
             arrowprops=dict(facecolor='black', shrink=0.05),
             )

plt.ylim(-2, 2)
plt.show()
```



Matplotlib 고급 – 그래프 예시 7

- 여러 개의 그래프를 하나의 차트 안에 그리기 (hist, plot)

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

np.random.seed(19680801)

# example data
mu = 100  # mean of distribution
sigma = 15  # standard deviation of distribution
x = mu + sigma * np.random.randn(437)

num_bins = 50

fig, ax = plt.subplots()

# the histogram of the data
n, bins, patches = ax.hist(x, num_bins, density=True)

# add a 'best fit' line
y = ((1 / (np.sqrt(2 * np.pi) * sigma)) *
     np.exp(-0.5 * (1 / sigma * (bins - mu))**2))
ax.plot(bins, y, '--')
ax.set_xlabel('Smarts')
ax.set_ylabel('Probability density')
ax.set_title(r'Histogram of IQ: $\mu=100$, $\sigma=15$')

# Tweak spacing to prevent clipping of ylabel
fig.tight_layout()
plt.show()
```

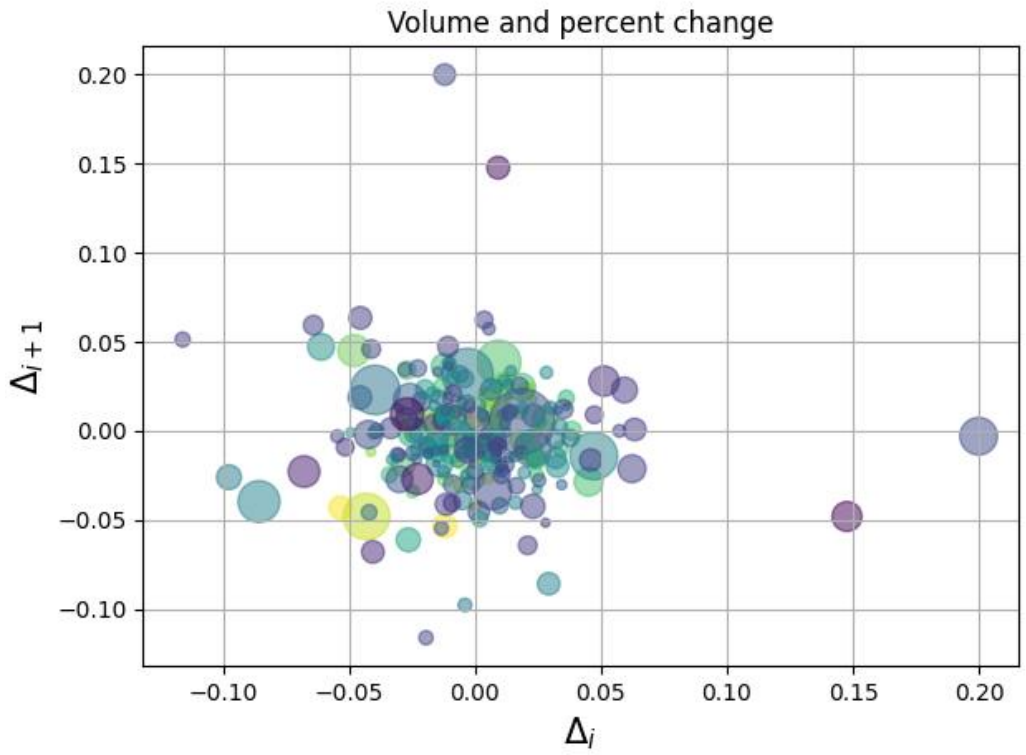


wanted

Matplotlib 고급 – 그래프 예시 8

- 고급 scatter plot

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

# Load a numpy record array from yahoo csv data with fields date, open, close,
# volume, adj_close from the mpl-data/example directory. The record array
# stores the date as an np.datetime64 with a day unit ('D') in the date column.
price_data = (cbook.get_sample_data('goog.npz', np_load=True)['price_data']
              .view(np.recarray))
price_data = price_data[-250:]  # get the most recent 250 trading days

delta1 = np.diff(price_data.adj_close) / price_data.adj_close[:-1]

# Marker size in units of points^2
volume = (15 * price_data.volume[:-2] / price_data.volume[0])**2
close = 0.003 * price_data.close[:-2] / 0.003 * price_data.open[:-2]

fig, ax = plt.subplots()
ax.scatter(delta1[:-1], delta1[1:], c=close, s=volume, alpha=0.5)

ax.set_xlabel(r'$\Delta_i$', fontsize=15)
ax.set_ylabel(r'$\Delta_{i+1}$', fontsize=15)
ax.set_title('Volume and percent change')

ax.grid(True)
fig.tight_layout()

plt.show()
```



Appendix. Reference로 배운 내용 마스터하기

- matplotlib 공식 홈페이지, tutorial
  https://matplotlib.org/tutorials/introductory/usage.html

Q & A

THANK YOU :)

wanted