

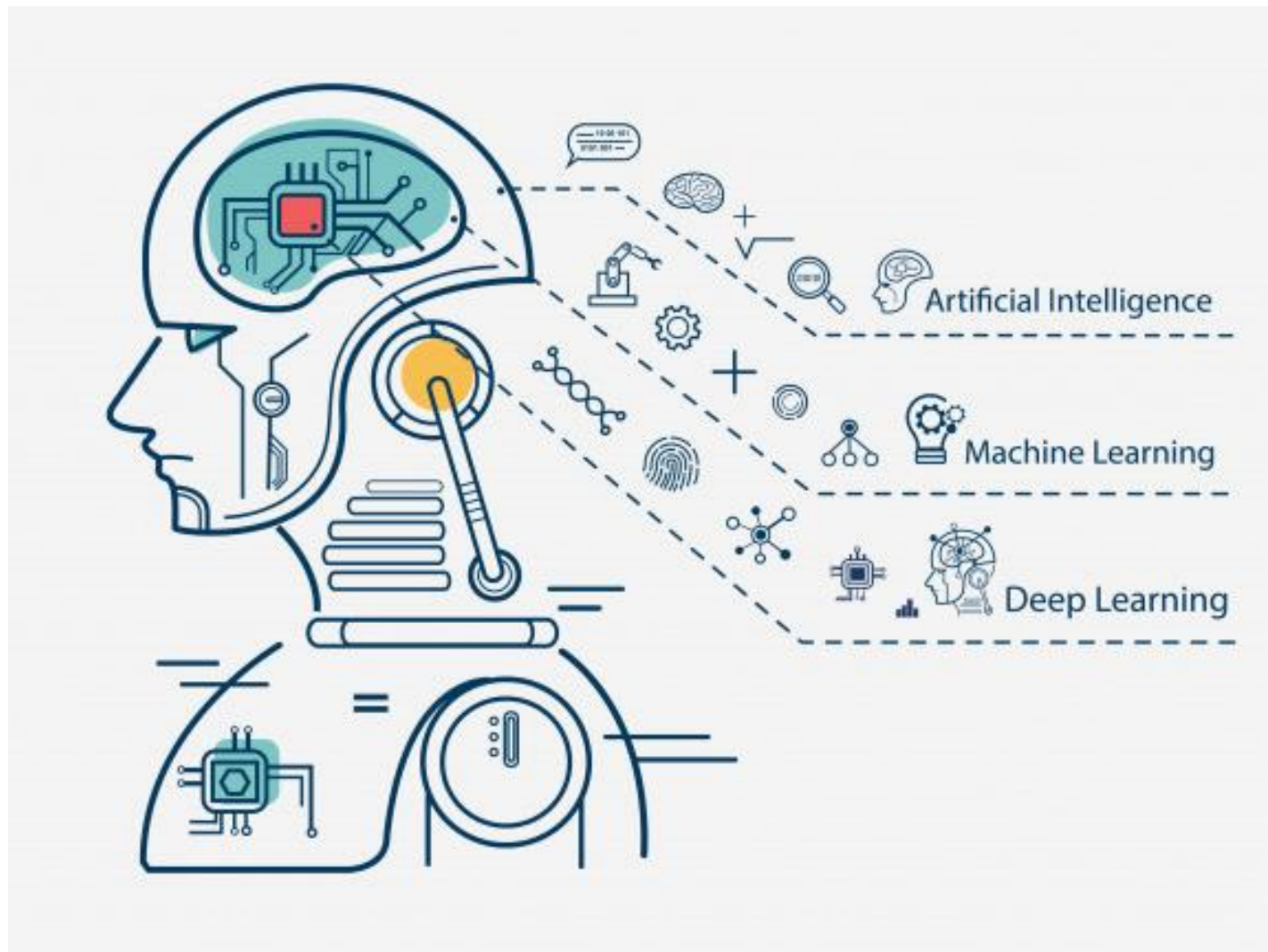
파이썬을 이용한 데이터수집 파이썬을 이용한 시각화

Jaekwan Ahn



목차

1. matplotlib
2. seaborn
3. 기타

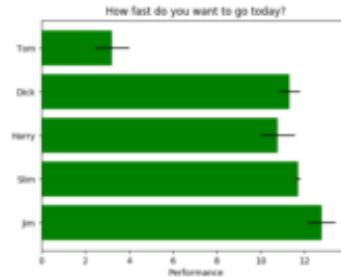


Matplotlib

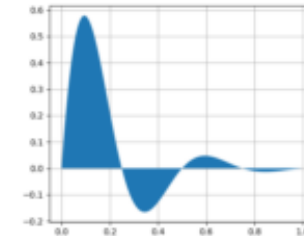
- 맷플롯라이브(Matplotlib)는 파이썬에서 자료를 차트(chart)나 플롯(plot)으로 시각화하는 패키지
맷플롯라이브는 다음과 같은 정형화된 차트나 플롯 이외에도 저수준 API를 사용한 다양한 시각화 기능을 제공

- 매우 많은 기능을 제공하므로 documentation 참고 추천

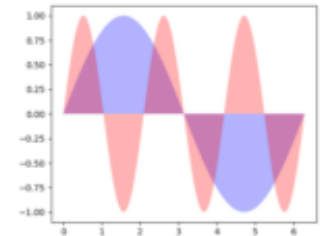
- [Matplotlib: Python plotting — Matplotlib 3.4.3 documentation](#)
- 시각화 예시는 <http://matplotlib.org/gallery.html> 참조



barh_demo



fill_demo



fill_demo_features

- 맷플롯라이브 패키지에는 pyplot 라는 서브패키지가 존재
이 pyplot 서브패키지는 매트랩(matlab) 이라는 수치해석 소프트웨어의 시각화 명령을 거의 그대로 사용할 수 있도록 맷플롯라이브의 하위 API를 포장(wrapping)한 명령어 집합으로 간단한 시각화 프로그램을 만드는 경우에는 pyplot 서브패키지의 명령만으로도 충분
- 주로 `import matplotlib.pyplot as plt` 나 `from matplotlib import pyplot as plt` 를 통해 plt라는 별칭으로 활용
 - * 주피터 노트북에서는 “%matplotlib inline”이라는 코드를 볼 수 있는데 이는 노트북을 실행한 브라우저에서 바로 그림을 볼 수 있게 해주는 것

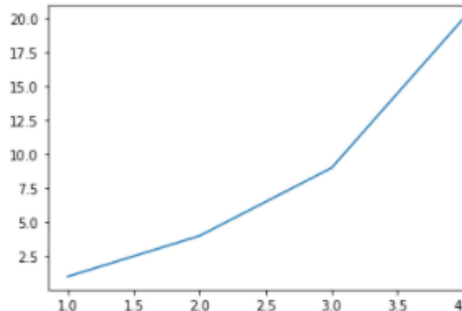
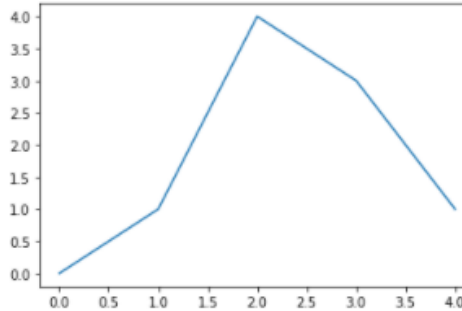
Matplotlib

- Figure and subplot

- Pyplot에는 기본적으로 figure라는 그림에 다수의 plot을 그릴 수 있도록 지원한다.
- Figure(그림)에 들어가는 plot을 subplot(하위그림)이라고 하는데, 하나의 그림은 figure로 여러 개의 그림은 subplot으로 진행한다.
- 몇몇의 변경 코드가 그래서 조금씩 다른 경우가 있다.

```
import matplotlib.pyplot as plt
import numpy as np
plt.figure(1) #생략가능 기본값
plt.subplot(111) #생략가능 기본값
plt.plot([0,1,4,3,1])
#plt.plot([0,1,2,3,4],[0,1,4,3,1])
plt.show()
```

```
plt.plot([1, 2, 3, 4], [1, 4, 9, 20])
```



```
import matplotlib.pyplot as plt

# figure 크기 설정
plt.figure(figsize=(15,7))

# grid 설정
plt.grid(True)

# title 설정
plt.title()

# x, y축 라벨 설정
plt.xlabel()
plt.ylabel()

# x, y축 범위 설정
plt.xlim()
plt.ylim()

# x, y축 눈금 설정
plt.xticks()
plt.yticks()

# 범례
plt.legend()

# 그리기
plt.plot()

import matplotlib.pyplot as plt

# figure 크기 설정
fig = plt.figure(figsize=(15,7))
# subplot 추가
ax = fig.add_subplot(111)

# grid 설정
ax.grid(True)

# title 설정
ax.set_title()

# x, y축 라벨 설정
ax.set_xlabel()
ax.set_ylabel()

# x, y축 범위 설정
ax.set_xlim()
ax.set_ylim()

# x, y축 눈금 설정
ax.set_xticks()
ax.set_yticks()

# x, y축 눈금 라벨 설정
ax.set_xticklabels()
ax.set_yticklabels()

# 범례
ax.legend()

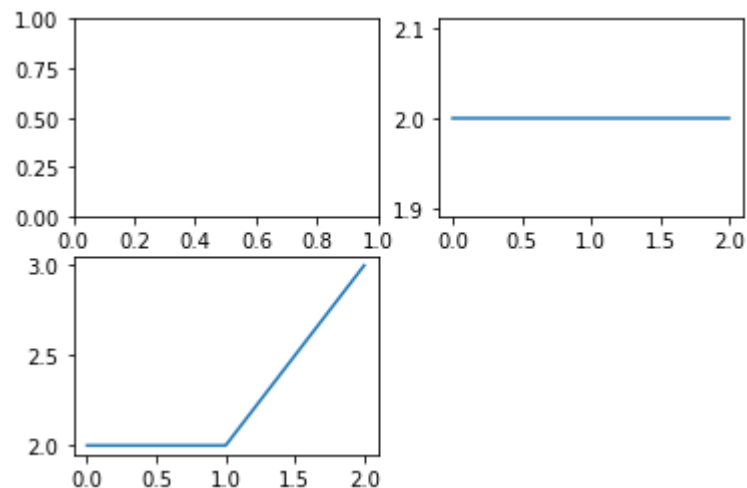
# 그리기
ax.plot()
```

Matplotlib

- Figure and subplot

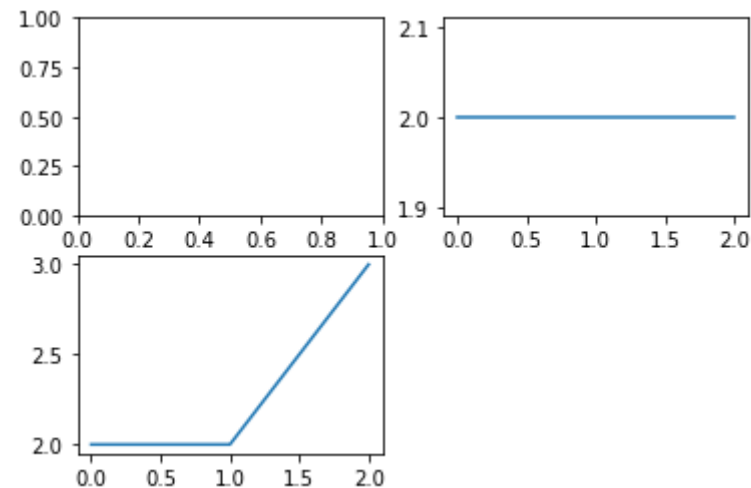
- 하나의 그림에 여러 개의 하위 그림 그리기
큰 기본 그림 생성하고 하위그림들을 생성
활성그림은 마지막 지정된 하나만 존재

```
plt.figure(1)      #생략가능 기본값  
plt.subplot(221)   #하위그림 1 정의  
plt.subplot(222)   #하위그림 2 정의  
plt.plot([2,2,2])  #활성그림 2에 그린다.  
plt.subplot(223)   #하위그림 3 정의  
plt.plot([2,2,3])  #활성그림 3에 그린다.
```



fig=plt.figure()로 큰 그림 정의하고 하위 그림의 이름을 지정하며 fig와 연동.

```
fig=plt.figure(1)      #그림 생성  
ax1=fig.add_subplot(221)#하위그림 1 정의  
ax2=fig.add_subplot(222)#하위그림 2 정의  
ax3=fig.add_subplot(223)#하위그림 3 정의  
  
ax2.plot([2,2,2])      #그림 2에 그린다.  
ax3.plot([2,2,3])      #그림 3에 그린다.
```



Matplotlib

- Figure and subplot의 기본값(default) 변경
 - rcParams 함수를 이용하여 그래프의 기본값을 변경할 수 있음

[Customizing Matplotlib with style sheets and rcParams — Matplotlib 3.4.3 documentation](#) 참조

```
plt.rcParams['figure.figsize'] = 20, 10
plt.rcParams['axes.grid'] = True
plt.rcParams['font.size'] = 20
plt.rcParams['axes.titlesize'] = 27
plt.rcParams['axes.labelsize'] = 24
```

Matplotlib

- Plot 속성

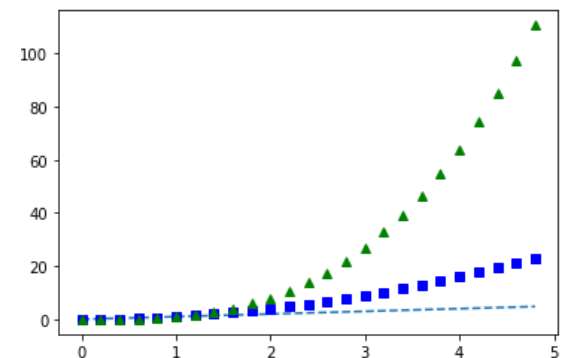
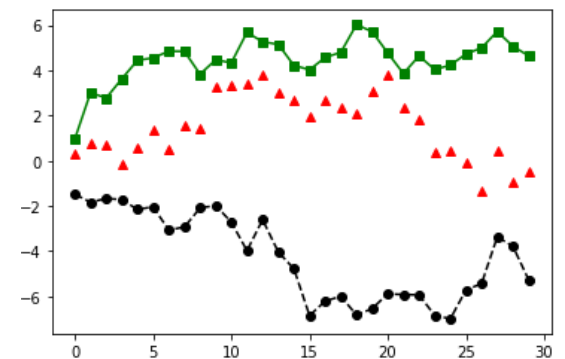
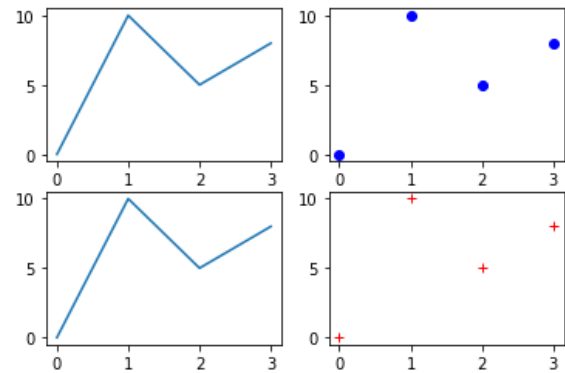
- Color, marker, linestyle

```
x=[0,1,2,3]
y=[0,10,5,8]
fig,axes=plt.subplots(2,2)
axes[0,0].plot(x, y)           # plot x and y using default line style and color
axes[0,1].plot(x, y, 'bo')    # plot x and y using blue circle markers
axes[1,0].plot(y)             # plot y using x as index array 0..N-1
axes[1,1].plot(y, 'r+')       # ditto, but with red plusses
```

다른 방식의 코드

```
plt.plot(np.random.randn(30).cumsum(),color='k',linestyle='--',marker='o')
plt.plot(np.random.randn(30).cumsum(),'r^') #o,s,x,^
plt.plot(np.random.randn(30).cumsum(),'gs-') #o,x,^
```

```
import matplotlib.pyplot as plt
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^') #한 줄에 3개의 그래프
plt.show()
```



Matplotlib

- Plot 속성

- Color, marker, linestyle

“[matplotlib.pyplot.plot — Matplotlib 2.1.2 documentation](https://matplotlib.org/2.1.2/pyplot-api.html)”

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

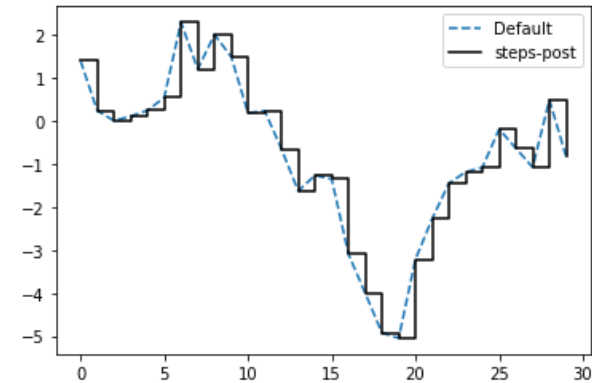
character	description
'_'	solid line style
'--'	dashed line style
'-.'	dash-dot line style
':'	dotted line style
'.'	point marker
','	pixel marker
'o'	circle marker
'v'	triangle_down marker
'^'	triangle_up marker
'<'	triangle_left marker
'>'	triangle_right marker
'1'	tri_down marker
'2'	tri_up marker
'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

Matplotlib

- Plot 속성

- 범례

```
data=np.random.randn(30).cumsum()
plt.plot(data,'k--',label='Default')
plt.plot(data,'k-',drawstyle='steps-post',label='steps-post')
plt.legend(loc='best')
```

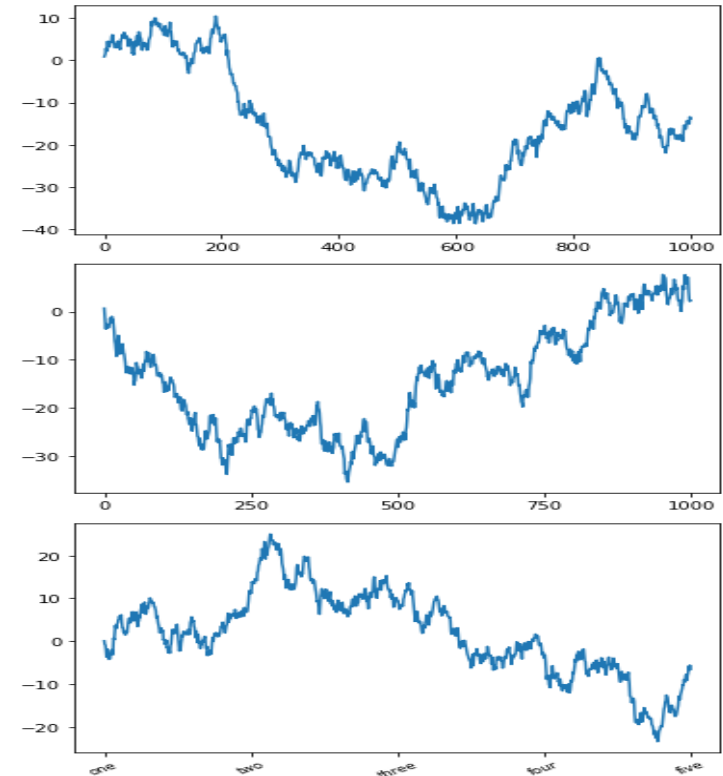


- 축눈금 설정, 눈금이름(text) 설정

```
fig=plt.figure()
ax=fig.add_subplot(1,1,1)
data=np.random.randn(1000).cumsum()
ax.plot(data)

fig=plt.figure()
ax=fig.add_subplot(1,1,1)
data=np.random.randn(1000).cumsum()
ax.plot(data)
ticks=ax.set_xticks([0,250,500,750,1000])

fig=plt.figure()
ax=fig.add_subplot(1,1,1)
data=np.random.randn(1000).cumsum()
ax.plot(data)
ticks=ax.set_xticks([0,250,500,750,1000])
labels=ax.set_xticklabels(['one','two','three','four','five'],rotation=30,fontsize='small')
```



Matplotlib

- 플롯팅 종류

- 바(bar) 그래프 산점도(scatter) 그래프

범주형 변수를 사용하여 여러가지 종류의 플롯을 생성 가능

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

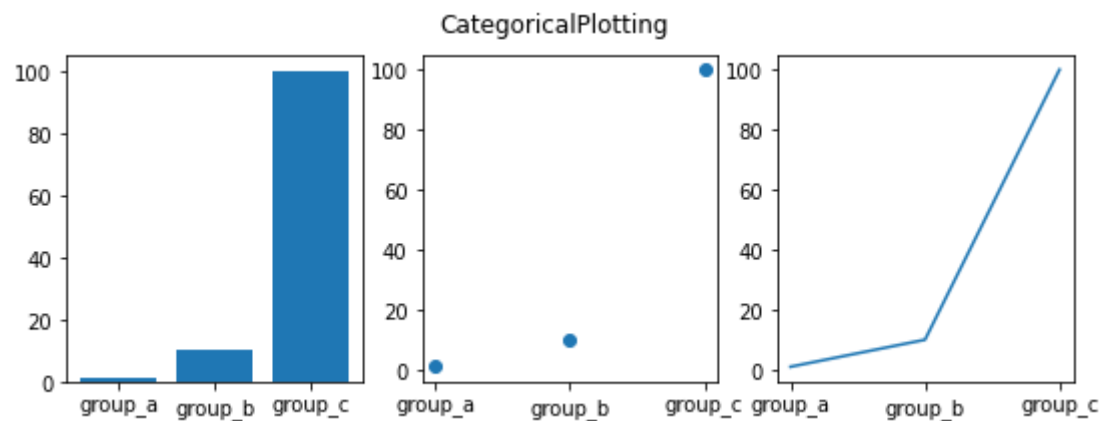
names = ['group_a', 'group_b', 'group_c']
values = [1, 10, 100]

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

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

- 플롯팅 종류

- 산점도(scatter) 상세

numpy.recarray나 pd.DataFrame문자열을 사용하여 특정 변수에 액세스 할 수 있는 형식의 데이터의 각각 data 키워드 인수를 가져올 수 있다.

Scatter 함수에서 data=' '는 이런 기능을 제공

```
x= [ 0, 1, 2, 3, 4]
y= [ 0, 1, -2, 2, 3]
color=[0, 50, 30, 20, 40]
scale=[100, 200, 400,800,1600]
plt.scatter(x, y, c=color,s=scale)
plt.show()
```

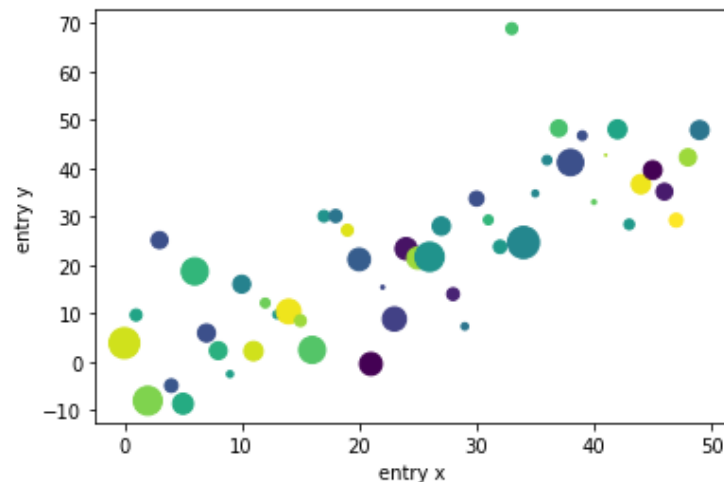
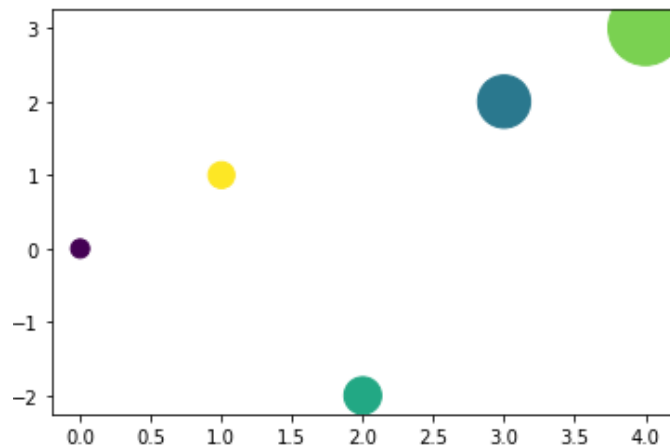
```
data={'x' :[ 0, 1, 2, 3, 4],
      'y' :[ 0, 1, -2, 2, 3],
      'color':[0, 50, 30, 20, 40],
      'scale':[100, 200, 400,800,1600]}
plt.scatter('x', 'y', c='color',s='scale',data=data)
plt.show()
```

```
import pandas as pd
```

```
data=pd.DataFrame()
data['x']=np.arange(50)
data['y']=data['x']+10*np.random.randn(50)
data['color']=np.random.randint(0, 50, 50)
data['scale']=np.abs(np.random.randn(50)) * 100

plt.scatter('x', 'y', c='color', s='scale', data=data)
plt.xlabel('entry x')
plt.ylabel('entry y')
plt.show()
```

	x	y	color	scale
0	0	4.483724	15	76.544530
1	1	0.272306	4	162.272907
2	2	6.575563	42	6.341110
3	3	6.859992	49	35.107901
4	4	1.002152	34	28.927000



Matplotlib

- 플롯팅 종류

- 히스토그램(Histogram)

범주 형 변수를 사용하여 여러가지 종류의 플롯을 생성 가능

입력 데이터 시퀀스 $x = [55, 50, 57, 80, 90, 100]$, $nb_bins = 3$

• $bins = [50, 66.67, 83.33, 100]$,

$b_0 = 50$, $b_1 = 50 + \frac{100-50}{3} = 66.67$, $b_2 = 50 + \frac{100-50}{3} * 2 = 83.3$, $b_3 = 100$

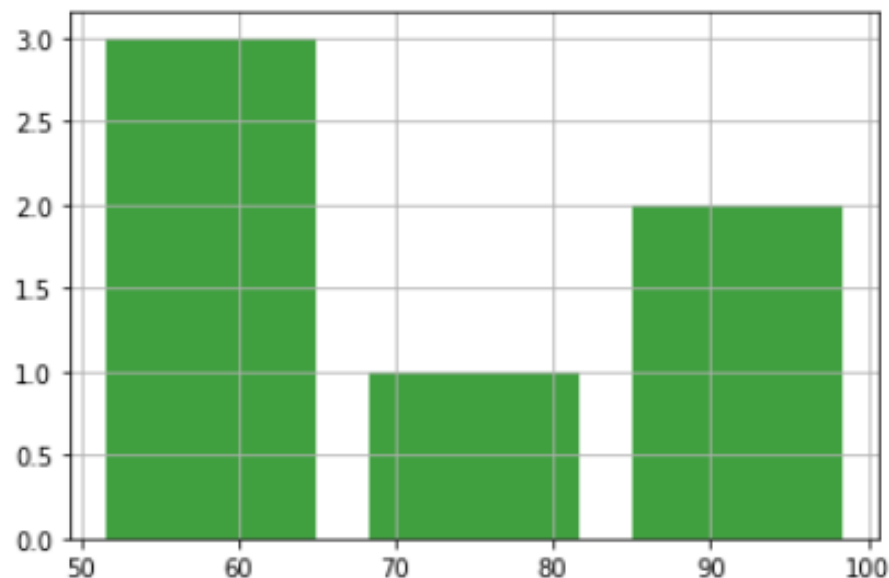
• $n = [3, 1, 2]$,

$n_0 = count(50 \leq x < 66.67) = 3$,

$n_1 = count(66.67 \leq x < 83.33) = 1$,

$n_2 = count(83.33 \leq x < 100) = 2$

```
x=[50,55,57,80,90,100]
n,bins,patches=plt.hist(
    x=x,                # 입력 데이터
    bins=3,             # 구분할 바구니(구간)의 수
    histtype='bar',     # 타입 or step으로 하면 모양이 바뀐.
    orientation='vertical', # or horizontal
    rwidth=0.8,         # 바의 폭 비율 1.0일 경우 꼭 채움
    facecolor='g',      # bar의 색상
    alpha=0.75)         # 투명도
plt.grid()
plt.show()
```



Matplotlib

- 플롯팅 종류

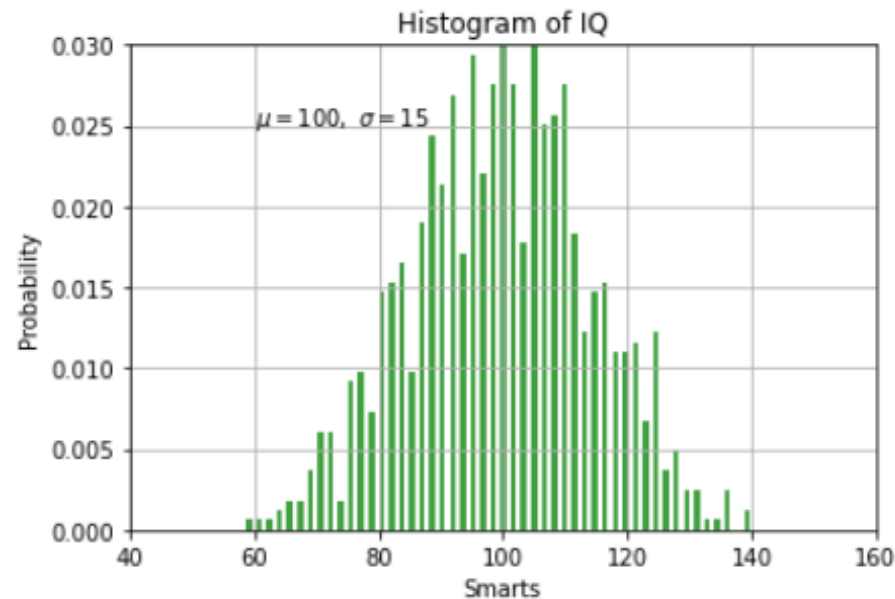
- 히스토그램(Histogram)과 text

text () 명령은 임의의 위치에 텍스트를 추가하는 데 사용될 수 있으며 xlabel(), ylabel() 및 title()은 표시된 위치에 텍스트를 추가하는 데 사용

```
import matplotlib.pyplot as plt
import numpy as np
mu, sigma = 100, 15
x = mu + sigma * np.random.randn(1000)

# the histogram of the data
n, bins, patches = plt.hist(x, 50, density=1, width=0.9, 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()
```



matplotlib는 모든 텍스트 표현식에서 TeX 방정식 표현식을 허용. Ex) “ $\sigma = 15$ ” 표현식을 쓰려면 \$ 기호로 묶은 TeX 표현식 문자열 앞의 r은 매우 중요. 문자열이 원시 문자열이며 백 슬래시를 파이썬 이스케이프로 취급하지 않음

Matplotlib

- 주석 시각화

- Annotating text

위의 기본 `text()` 명령을 사용하면 텍스트가 Axes의 임의 위치에 배치된다. 텍스트는 일반적으로 플롯의 일부 특성에 주석을 다는 것이고, `annotate()` 명령어는 주석을 쉽게 보여지도록 도우미 기능을 제공한다.

주석에서 고려해야 할 두 가지:

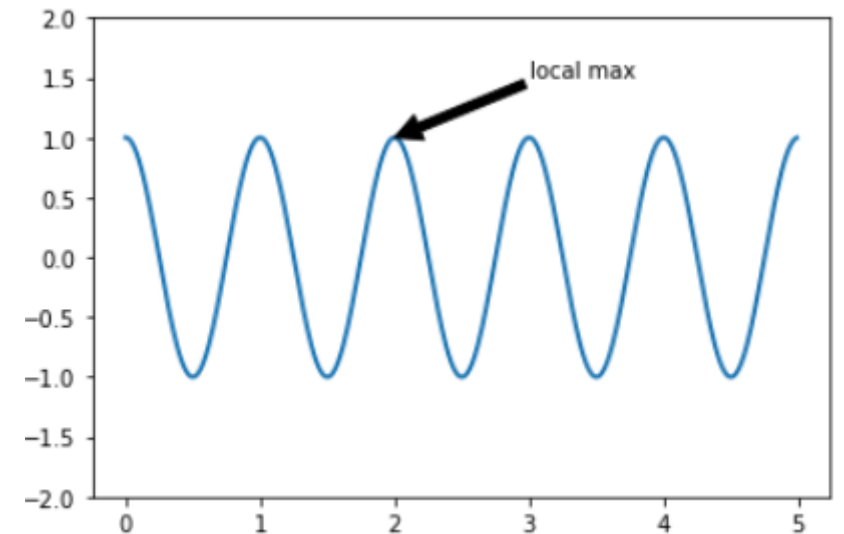
- 1) 플롯의 특성의 위치 `xy`(arrow tip)
- 2) 주석의 문자열의 위치 `xytext`(text location)
→ 이 두 인수는 (x, y) 튜플이다.

* 다양한 좌표시스템은 아래 참조

[Annotating Plots — Matplotlib 3.1.2 documentation](#)

```
import matplotlib.pyplot as plt
ax = plt.subplot(111)
t = np.arange(0.0, 5.0, 0.01)
y = np.cos(2*np.pi*t)
plt.plot(t, y, lw=2)          #line width=2

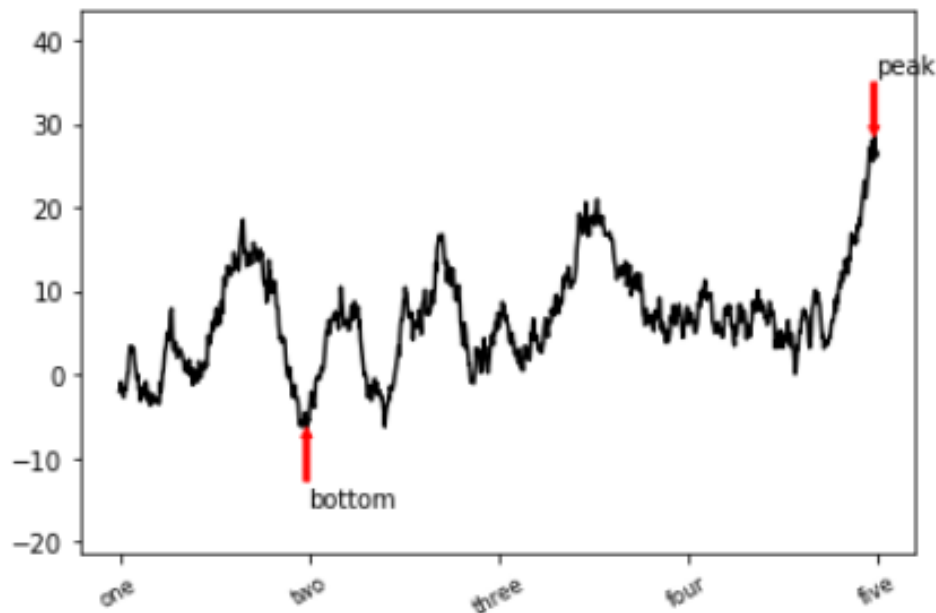
plt.annotate('local max',    #출력할 문자열
             xy=(2, 1),      #화살표 끝의 위치
             xytext=(3, 1.5), #화살표 시작, 문자열위치
             arrowprops=dict(facecolor='black', shrink=0.00) #화살표 설정
            )
plt.ylim(-2, 2)
plt.show()
```



Matplotlib

- 주식 시각화

- 주식 과 이름(text) 추가하기



```
fig=plt.figure()
ax=fig.add_subplot(1,1,1)
data=np.random.randn(1000).cumsum()
ax.plot(data,'k-')
ticks=ax.set_xticks([0,250,500,750,1000]) # x축 눈금
labels=ax.set_xticklabels(['one','two','three','four','five'], # x축 눈금 문자열
rotation=30,fontsize='small')

max_y=data.max(); max_x=list(data).index(max_y) # maximum point의 x,y 값
min_y=data.min(); min_x=list(data).index(min_y) # minimum point의 x,y 값
ax.annotate('peak', # maximum point의 문자열주석,
            xy=(max_x,max_y), # maximum point의 위치
            xytext=(max_x,max_y+5), # 문자열주석의 위치
            arrowprops=dict(color='red',headwidth=4,width=2,headlength=4),
            horizontalalignment='left',verticalalignment='top'
            )
ax.annotate('bottom', # minimum point의 문자열주석,
            xy=(min_x,min_y), # maximum point의 위치
            xytext=(min_x,min_y-5), # 문자열주석의 위치
            arrowprops=dict(color='red',headwidth=4,width=2,headlength=4),
            horizontalalignment='left',verticalalignment='bottom'
            )
ax.set_ylim([min_y-10,max_y+10]) # y축의 출력 범가지정
```

Seaborn

- Seaborn은 향상된 데이터 시각화를 위해 만들어진 Python 라이브러리로 효과적인 데이터 시각화 및 비데이터전문가와의 의사 소통에 용이
다양한 색상 테마와 통계용 차트 기능을 추가한 시각화 패키지로 기본적인 시각화 기능은 Matplotlib 패키지에 의존하며 통계 기능은 Statsmodels 패키지에 의존
- 매우 많은 기능을 제공하므로 documentation 참고 추천
 - <http://seaborn.pydata.org/>
- Seaborn이 ‘joint plots’이라고 하는 방법은 분산 플롯과 축의 분산 플롯의 각 변수 분포와 쌍을 이루고, R에도 있는 패키지이지만, seaborn에서는 한 줄의 코드로 데이터 혹은 피처의 관계성을 시각화 할 수 있다.
- 주로 `import seaborn as sns`를 통해 sns라는 별칭으로 활용

Seaborn

- 플롯 스타일

- Seaborn을 임포트하면 색상 등을 Matplotlib에서 제공하는 기본 스타일이 아닌 Seaborn에서 지정한 기본 스타일로 바꾼다. 자세한 내용은 다음 문서를 참조한다.

- <http://seaborn.pydata.org/tutorial/aesthetics.html>

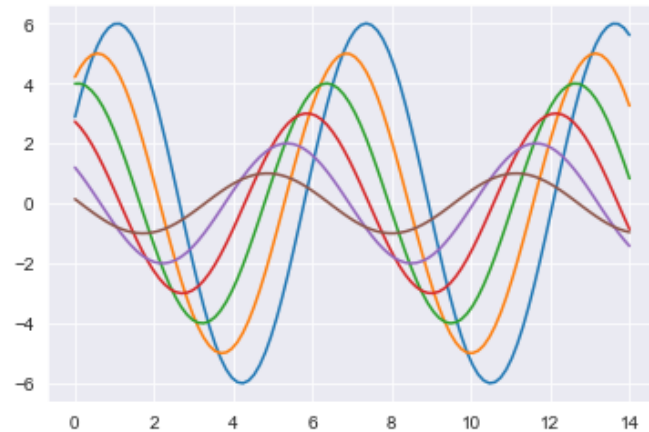
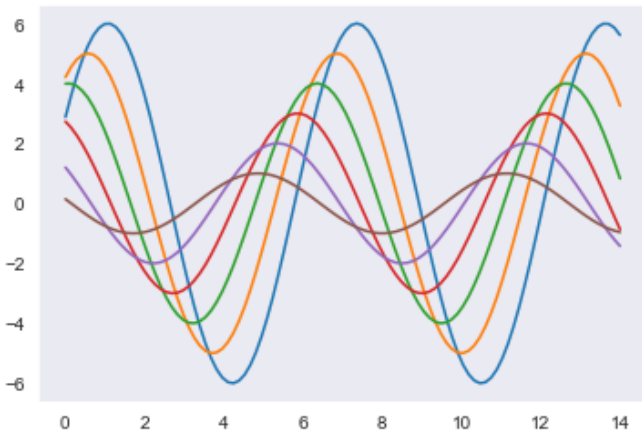
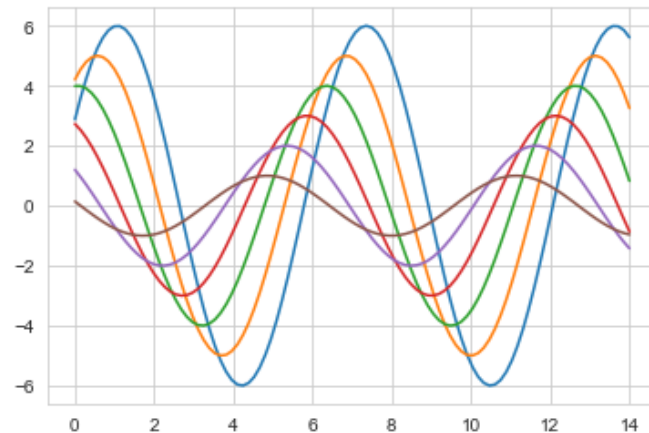
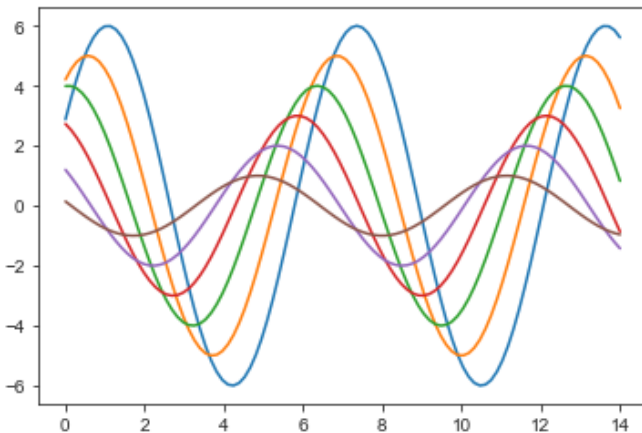
```
def sinplot(flip=1):  
    x = np.linspace(0, 14, 100)  
    for i in range(1, 7):  
        plt.plot(x, np.sin(x + i * .5) * (7 - i) * flip)
```

```
sns.set_style("white")  
sinplot()
```

```
sns.set_style("dark")  
sinplot()
```

```
sns.set_style("whitegrid")  
sinplot()
```

```
sns.set_style("darkgrid")  
sinplot()
```



Seaborn

- 1차원 실수 플롯

- 실수 분포 플롯은 자료의 분포를 묘사하기 위한 것으로 Matplotlib의 단순한 히스토그램과 달리 커널 밀도(kernel density) 및 러그(rug) 표시 기능 및 다차원 복합 분포 기능 등을 제공한다. 1차원 실수 분포 플롯 명령어로는 rugplot, kdeplot, distplot이 있다.

rugplot: 러그(rug) 플롯은 데이터 위치를 x축 위에 작은 선분(rug)으로 나타내어 실제 데이터들의 위치를 보여준다.

kdeplot: 커널 밀도(kernel density)는 커널이라는 함수를 겹치는 방법으로 히스토그램보다 부드러운 형태의 분포 곡선을 보여주는 방법이다.

distplot: distplot은 러그와 커널 밀도를 같이 표시해주기때문에 matplotlib의 hist 명령어보다 많이 사용된다.

```
import seaborn as sns
iris = sns.load_dataset("iris")      # 붓꽃 데이터
titanic = sns.load_dataset("titanic") # 타이타닉호 데이터
tips = sns.load_dataset("tips")     # 팁 데이터
flights = sns.load_dataset("flights") # 여객운송 데이터

x = iris.petal_length.values
fig, axes=plt.subplots(1,3, figsize=(12,6))

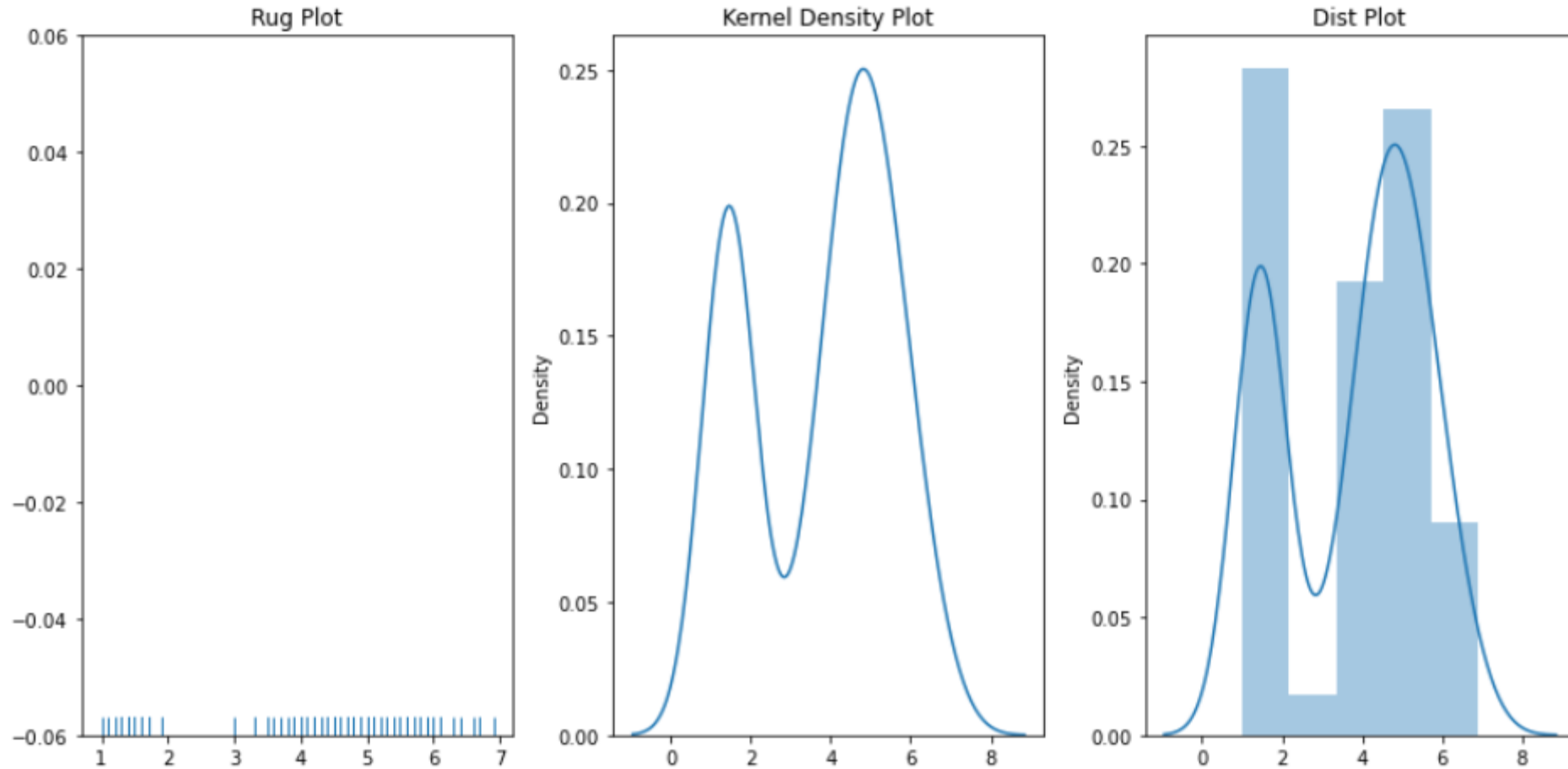
sns.rugplot(x, ax=axes[0])
sns.kdeplot(x, ax=axes[1])
sns.distplot(x, ax=axes[2])

axes[0].set_title("Rug Plot")
axes[1].set_title("Kernel Density Plot")
axes[2].set_title("Dist Plot")

fig.tight_layout()
plt.show()
```

Seaborn

- 1차원 실수 플롯



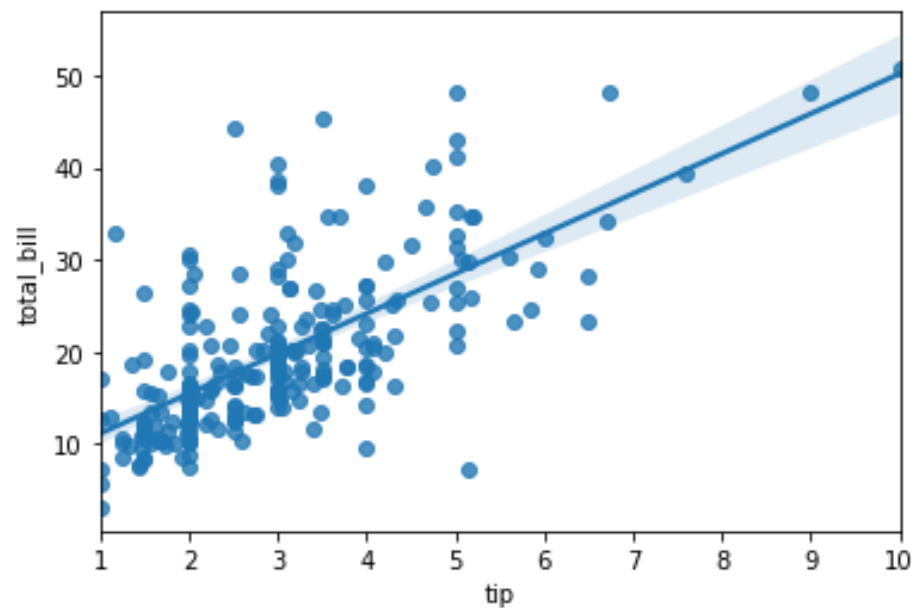
Seaborn

- 1차원 실수 플롯

- Linear regression도 간단히 그려준다.

```
1 sns.regplot(x = 'tip', y = 'total_bill', data = tips)
```

```
<AxesSubplot:xlabel='tip', ylabel='total_bill'>
```

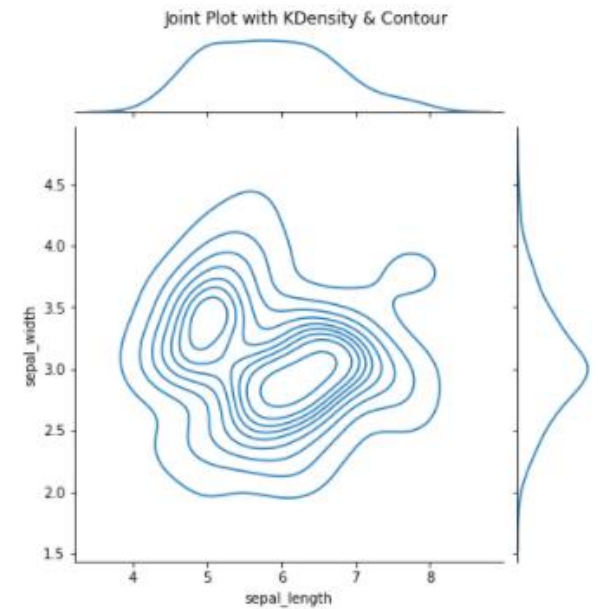
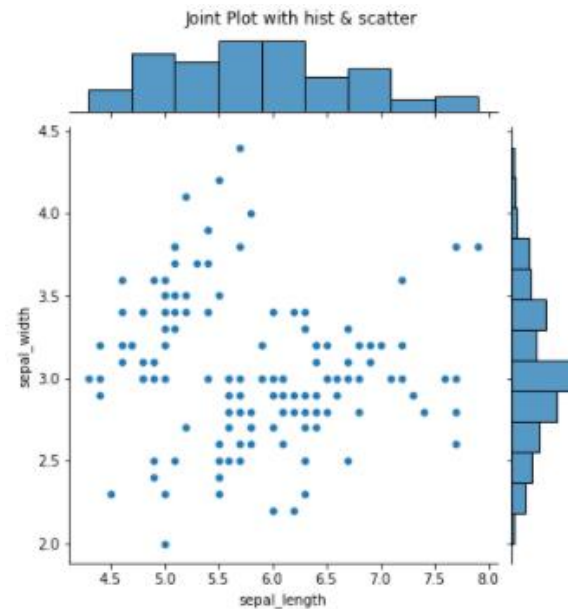


Seaborn

- 다차원 실수 플롯

- 만약 데이터가 2차원이고 모두 연속적인 실수값이라면 스캐터 플롯(scatter plot)을 사용할 수 있는데, Seaborn에서 스캐터 플롯을 그리기 위해서 `jointplot` 명령어를 사용
- `jointplot` 명령어는 스캐터 플롯 뿐만 아니라 차트의 가장자리에 각 변수의 히스토그램도 그린다.
→ histogram 대신 Kdensity를 넣을 수 있다.

```
sns.jointplot(x="sepal_length", y="sepal_width", data=iris, kind="scatter")  
plt.suptitle("Joint Plot with hist & scatter", y=1.02)  
plt.show()  
  
sns.jointplot(x="sepal_length", y="sepal_width", data=iris, kind="kde")  
plt.suptitle("Joint Plot with KDensity & Contour", y=1.02)  
plt.show()
```



Seaborn

- 다차원 복합 데이터 플롯

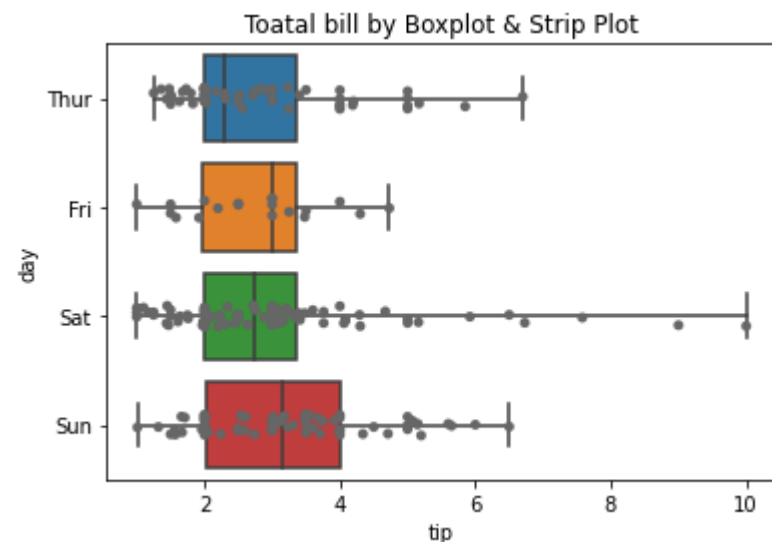
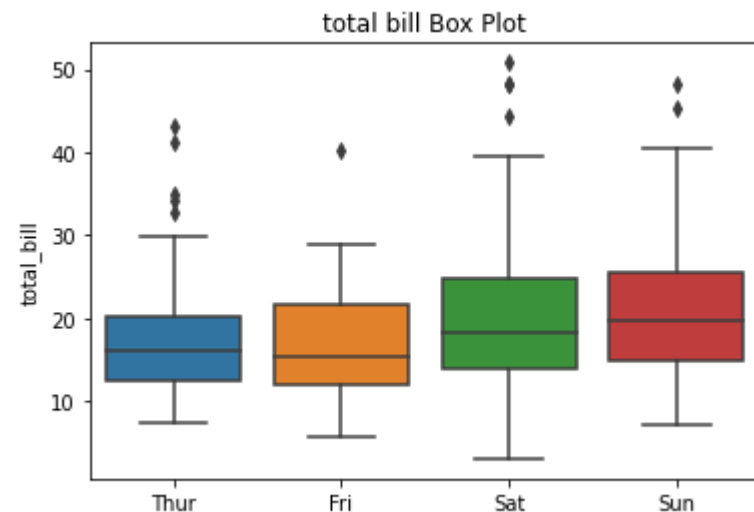
- 데이터가 2차원이고 실수 값, 카테고리 값이 섞여 있다면 기존의 플롯 이외에도 다음과 같은 descriptive 분포 플롯들을 이용할 수 있다.

Barplot
boxplot
pointplot
violinplot
stripplot
swarmplot

- 또 위 여러 종류의 차트를 겹쳐서 표시할 수도 있다.

```
sns.boxplot(x="day", y="total_bill", data=tips)
plt.title("total bill Box Plot")
plt.show()
```

```
plt.title("Toatal bill by Boxplot & Strip Plot")
sns.boxplot(x="tip", y="day", data=tips, whis=np.inf)
sns.stripplot(x="tip", y="day", data=tips, jitter=True, color="0.4")
plt.show()
```



기타; tqdm

- Tqdm은 파이썬으로 어떤 작업을 수행중일 때, 프로그램이 내가 의도한 데로 돌아가고 있는 중인가 진행상황을 시각화해주는 모듈이다.
- 주로 `from tqdm import tqdm`으로 호출
 - 1) Wrap `tqdm()` around any iterable:
어느 이터러블이든 `tqdm()`로 감싼다.
 - 2) `Progress_apply`문에 적용하기 위해서는 `pandas` 전용 `tqdm.pandas()`를 선언해준 후 똑같이 하면 된다.

```
1 import numpy as np
2 from tqdm import tqdm
3
4 i=0
5 for i in tqdm(np.random.rand(10000000)):
6     i = i**2
```

25%|██████ | 2515210/10000000 [00:01<00:04, 1521790.47it/s]

```
1 import pandas as pd
2 from tqdm import tqdm
3
4
5 test = pd.DataFrame({"col1":np.random.rand(1000000)})
```

```
1 tqdm.pandas()
2 test["col1"].progress_apply(lambda x : x*2)
```

62%|██████████ | 622893/1000000 [00:00<00:00, 1223231.58it/s]

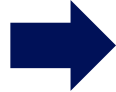
Matplotlib

- Pandas의 시각화 기능

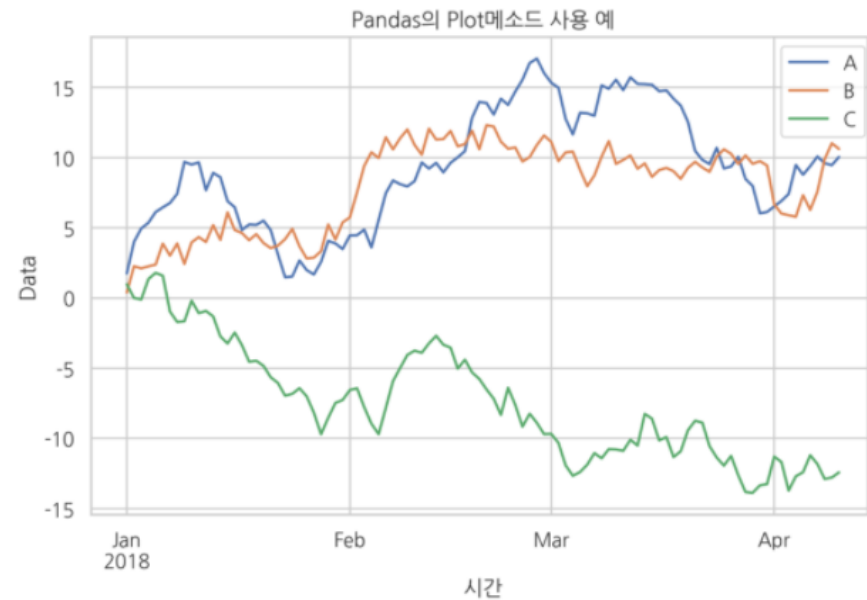
- Pandas의 시리즈나 데이터프레임은 'plot'이라는 시각화 메서드를 내장하고 있다. 'plot'은 matplotlib를 내부에서 임포트하여 사용한다.

```
np.random.seed(0)
df1 = pd.DataFrame(np.random.randn(100, 3),
                    index=pd.date_range('1/1/2018', periods=100),
                    columns=['A', 'B', 'C']).cumsum()
df1.tail()
```

	A	B	C
2018-04-06	9.396256	6.282026	-11.198087
2018-04-07	10.086074	7.583872	-11.826175
2018-04-08	9.605047	9.887789	-12.886190
2018-04-09	9.469097	11.024680	-12.788465
2018-04-10	10.052051	10.625231	-12.418409



```
df1.plot()
plt.title("Pandas의 Plot메소드 사용 예")
plt.xlabel("시간")
plt.ylabel("Data")
plt.show()
```



Matplotlib

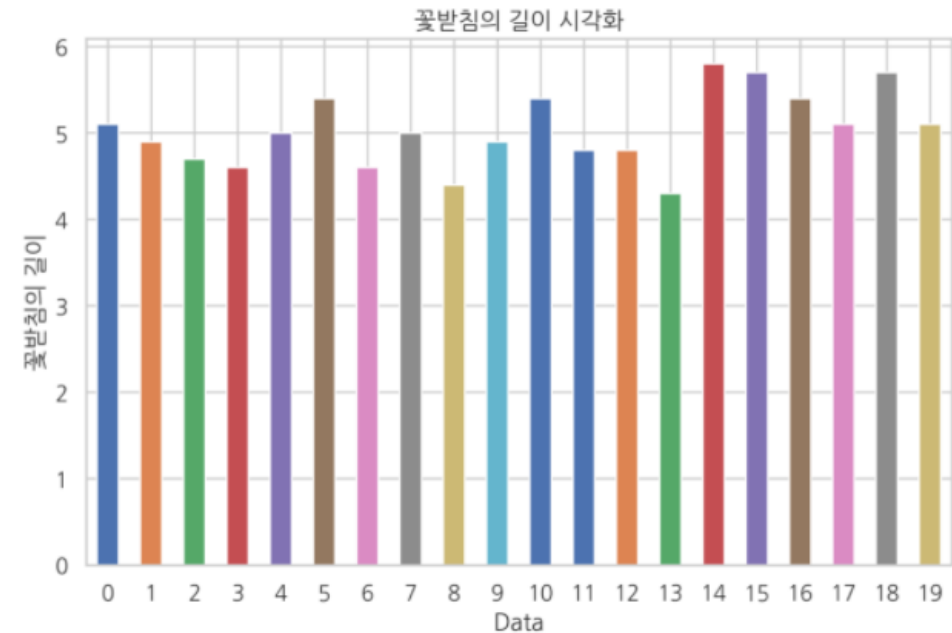
- Pandas의 시각화 기능

- 'plot' 메서드의 kind라는 인수를 바꾸면 여러가지 플롯을 그릴 수 있다.

bar
pie
hist
kde
box
scatter
area

* kind 인수 대신 plot.bar로 직접 메소드를 사용할 수도 있다.

```
iris = sns.load_dataset("iris") # 붓꽃 데이터  
titanic = sns.load_dataset("titanic") # 타이타닉호 데이터  
  
iris.sepal_length[:20].plot(kind='bar', rot=0)  
plt.title("꽃받침의 길이 시각화")  
plt.xlabel("Data")  
plt.ylabel("꽃받침의 길이")  
plt.show()
```



참고하기 좋은 사이트

- [Http://wikidocs.net/book/5011](http://wikidocs.net/book/5011)
[Matplotlib Tutorial - 파이썬으로 데이터 시각화하기 - WikiDocs](#)

검색어를 입력하세요.

Matplotlib Tutorial - 파이썬으로 데이터 시각화하기

00. Matplotlib 설치하기

01. Matplotlib 기본 사용

02. Matplotlib 숫자 입력하기

03. Matplotlib 축 레이블 설정하기

04. Matplotlib 범례 표시하기

05. Matplotlib 축 범위 지정하기

06. Matplotlib 선 종류 지정하기

07. Matplotlib 마커 지정하기

08. Matplotlib 색상 지정하기

09. Matplotlib 그래프 영역 채우기

10. Matplotlib 축 스케일 지정하기

11. Matplotlib 여러 곡선 그리기

12. Matplotlib 그리드 설정하기

13. Matplotlib 눈금 표시하기

14. Matplotlib 타이틀 설정하기

15. Matplotlib 수평선/수직선 표시하기

16. Matplotlib 막대 그래프 그리기

17. Matplotlib 수평 막대 그래프 그리기

18. Matplotlib 산점도 그리기

19. Matplotlib 3차원 산점도 그리기

20. Matplotlib 히스토그램 그리기

21. Matplotlib 에러바 표시하기

Matplotlib 소개



Version 3.2.1

Matplotlib은 데이터 시각화와 2D 그래프 플롯에 사용되는 파이썬 라이브러리입니다.

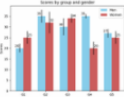
Matplotlib을 이용하면 아래 그림과 같이 다양한 유형의 그래프를 간편하게 그릴 수 있습니다.



Arctest



Stacked Bar Graph



Barchart



Horizontal bar chart



Broken Barh



Plotting categorical variables




Plotting the coherence of two signals



CSD Demo

참고하기 좋은 사이트

- [Http://datascienceschool.net](http://datascienceschool.net)
5.1 시각화 패키지 맷플롯리브 소개 — 데이터 사이언스 스쿨 (datascienceschool.net)



데이터 사이언스 스쿨

Search this book...

데이터 사이언스 스쿨

파이썬 편

소개의 글

1장 파이썬 설치와 설정

2장 파이썬 기초문법

3장 넘파이 배열 프로그래밍

4장 판다스 데이터 분석

5장 데이터 시각화

5.1 시각화 패키지 맷플롯리브 소개

Matplotlib의 여러가지 플롯

Matplotlib의 triangular grid 사용법

Seaborn을 사용한 데이터 분포 시각화

Pandas의 시각화 기능

수학 편

소개의 글

1장 수학 기호

2장 넘파이(NumPy)로 공부하는 선형대수

←

5.1 시각화 패키지 맷플롯리브 소개

맷플롯리브(Matplotlib)는 파이썬에서 자료를 차트(chart)나 플롯(plot)으로 시각화하는 패키지이다. 맷플롯리브는 다음과 같은 정형화된 차트나 플롯 이외에도 저수준 API를 사용한 다양한 시각화 기능을 제공한다.

- 라인 플롯(line plot)
- 스캐터 플롯(scatter plot)
- 컨투어 플롯(contour plot)
- 서피스 플롯(surface plot)
- 바 차트(bar chart)
- 히스토그램(histogram)
- 박스 플롯(box plot)

맷플롯리브를 사용한 시각화 예제들을 보고 싶다면 맷플롯리브 갤러리 웹사이트를 방문한다.

- <http://matplotlib.org/gallery.html>

pyplot 서브패키지

맷플롯리브 패키지에는 pyplot 라는 서브패키지가 존재한다. 이 pyplot 서브패키지는 매트랩(matlab)이라는 수치해석 소프트웨어의 시각화 명령을 거의 그대로 사용할 수 있도록 맷플롯리브의 하위 API를 포장(wrapping)한 명령어 집합을 제공한다. 간단한 시각화 프로그램을 만드는 경우에는 pyplot 서브패키지의 명령만으로도 충분하다. 다음에 설명할 명령어들도 별도의 설명이 없으면 pyplot 패키지의 명령이라고 생각하면 된다.

맷플롯리브 패키지를 사용할 때는 보통 다음과 같이 주 패키지는 mpl이라는 별칭(alias)으로 임포트하고 pyplot 서브패키지는 plt라는 다른 별칭으로 임포트하여 사용하는 것이 관례이므로 여기에서도 이러한 방법을 사용한다.

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

주피터 노트북을 사용하는 경우에는 다음처럼 %matplotlib 매직(magic) 명령으로 노트북 내부에 그림을 표시하도록 지정해야 한다.

```
%matplotlib inline
```

pyplot 서브패키지

Contents

- pyplot 서브패키지
- 라인 플롯
- 한글폰트 사용
- 스타일 지정
- 그림 범위 지정
- 틱 설정
- 그리드 설정
- 여러개의 선을 그리기
- 겹치그리기
- 범례
- x축, y축 라벨, 타이틀
- 그림의 구조

참고하기 좋은 사이트

- 시각화할 때, 폰트깨짐 현상이 발생할 때 아래 링크 참조.

[matplotlib/seaborn으로 시각화할 때 한글 폰트 깨짐현상 해결방법](#) ([teddylee777.github.io](#))

파이썬 시각화 한글/마이너스 깨짐 (윈도우) (velog.io)

03. Matplotlib 축 레이블 설정하기 - Matplotlib Tutorial - 파이썬으로 데이터 시각화하기 (wikidocs.net)

[강의 01 matplotlib 패키지 한글 깨짐 처리 - 토닥토닥 파이썬 - 데이터 시각화 \(wikidocs.net\)](https://wikidocs.net/101)

- [\[FAQ\] matplotlib 차트의 기본 크기 설정 | FinanceData](#)