Python Programming

시각화

matplotlib.pyplot

Yoonjoong Kim

Department of Computer Engineering, Hanbat National University

yjkim@hanbat.ac.kr

내용

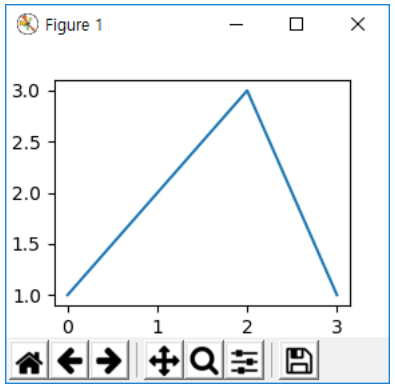
1. Figure and subplot
    1. Figure and subplot
    2. 하나의 그림에 다수의 하위그림 그리기 방법 3가지
2. Plot 속성
    1. Color, marker, linestyle,
    2. 범례, 축 눈금
    3. Line 속성(setp)
    4. Subplot 간격제어 축 공유
3. Plotting 종류
    - Bar, 산포도(scatter), 시계열데이터의 그래프, 산포도1, 히스토그램, 도형그리기
4. 수학식 다루기
5. 주석 다루기
6. Examples

1. Figure and subplot

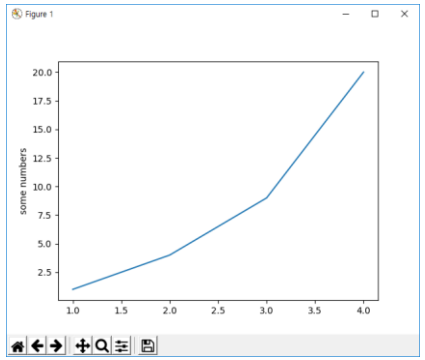
- matplotlib.pyplot
  - matplotlib가 MATLAB처럼 작동하도록 하는 명령 스타일 함수의 모음입니다.
  - pyplot 모듈은 그림을 만들고 그림에 플로팅 영역을 만들고 플로팅 영역에 일부 선을 플로팅하고 레이블로 플롯을 장식하는 등의 함수들의 모음입니다.

- Figure 와 subplot
  - matplotlib.pyplot는 figure라는 그림에 다수의 plot을 그릴 수 있도록 지원한다. figure(그림)에 들어가는 plot를 subplot(하위그림)이라고 한다.
  - 하나의 figure에 하나의 subplot은 기본

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




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



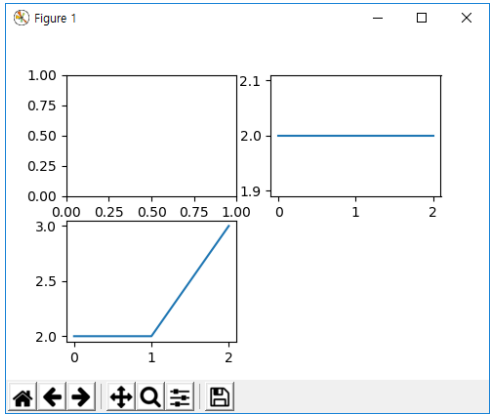
1. Figure and subplot(cont.)

- 하나의 그림에 여러 개의 하위 그림 그리기
  - plt.subplot(221)
    - 그림생성(기본)하고
    - 하위그림들을 생성하고 활성그림에 그리기
    - 활성그림은 마지막 지정된 하나만 존재한다.

```
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.
```

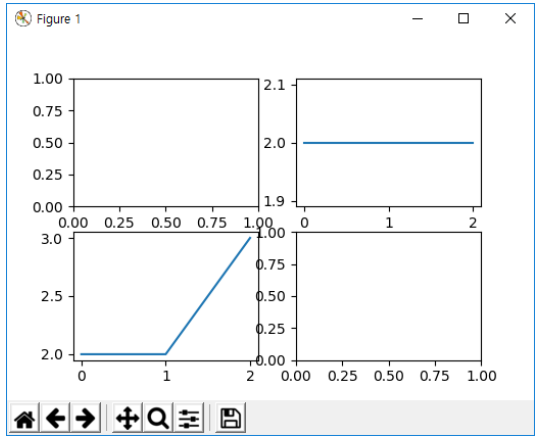
  - fig=plt.figure()
    ax=fig.add_subplot(221)
    - 그림을 생성하고
    - 하위그림을 추가하고
      하위그림의 이름을 참조하여 그린다.

```
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.
```



  - fig, axes=plt.subplots(2,2)
    - 그림과 하위 그림들의 축을 일괄하여 생성한다.
    - 축을 참조하여 그린다.

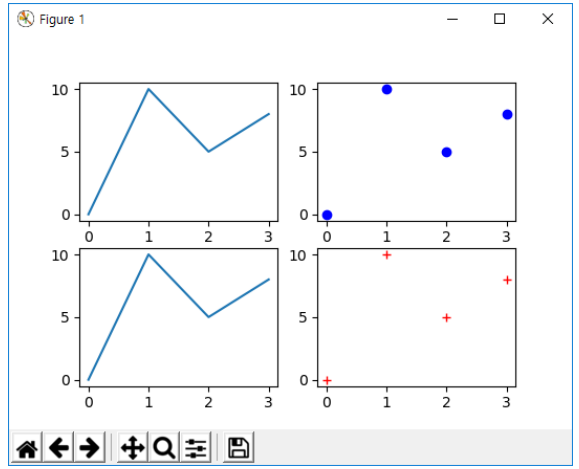
```
fig,axes=plt.subplots(2,2) #하위그림일괄생성
axes[0,1].plot([2,2,2])    #그림(0,1)에 그린다.
axes[1,0].plot([2,2,3])    #그림(1,0)에 그린다.
```



2. Plot 속성

- Color, marker , linestyle

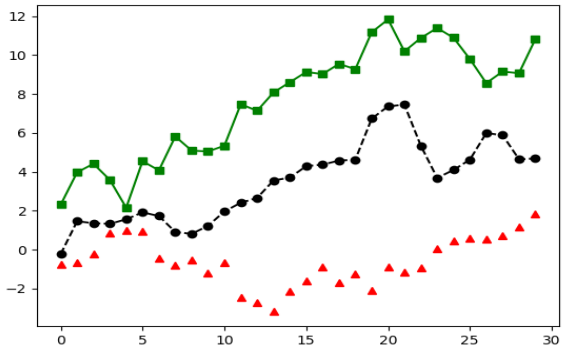
```
x=[0,1,2,3]
y=[0,10,5,8]
fig,axes=plt.subplots(2,2)
axes[0,0].plot(x, y)        # plot x and y using default line style and color
axes[0,1].plot(x, y, 'bo')  # plot x and y using blue circle markers
axes[1,0].plot(y)           # plot y using x as index array 0..N-1
axes[1,1].plot(y, 'r+')     # ditto,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



- Exaqmples using matplotlib.pyplot.plot
  - [link]

- Examples

```
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,^
```



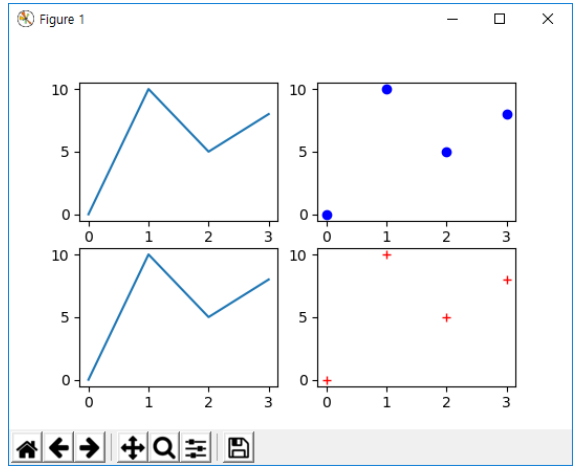
character	color
'b'	blue
'g'	green
'r'	red
'c'	cyan
'm'	magenta
'y'	yellow
'k'	black
'w'	white

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

2. Plot 속성(cont.)

- Color, marker , linestyle

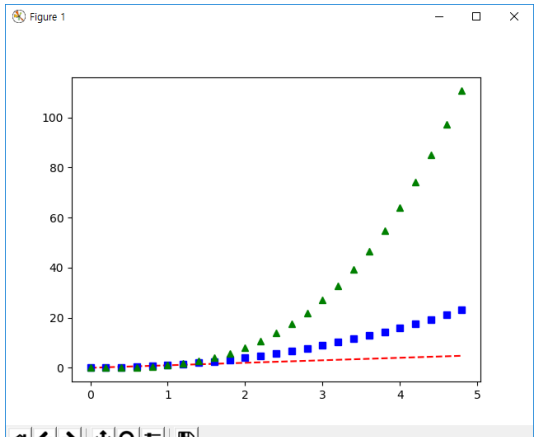
```
x=[0,1,2,3]
y=[0,10,5,8]
fig,axes=plt.subplots(2,2)
axes[0,0].plot(x, y)        # plot x and y using default line style and color
axes[0,1].plot(x, y, 'bo')  # plot x and y using blue circle markers
axes[1,0].plot(y)           # plot y using x as index array 0..N-1
axes[1,1].plot(y, 'r+')     # ditto,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



- Exaqmples using matplotlib.pyplot.plot
  - [link]

- 한 줄에 3개의그래프 그리기

```
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()
```



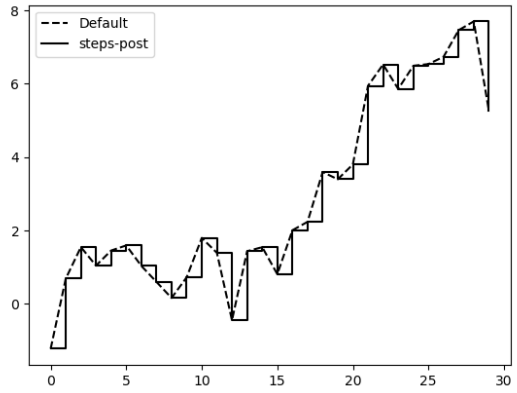
character	color
'b'	blue
'g'	green
'r'	red
'c'	cyan
'm'	magenta
'y'	yellow
'k'	black
'w'	white

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

2. Plot 속성(cont.)

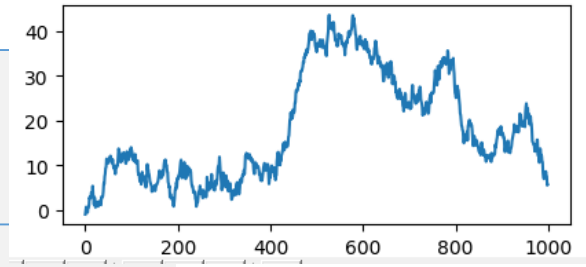
- 범례

```
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')
```

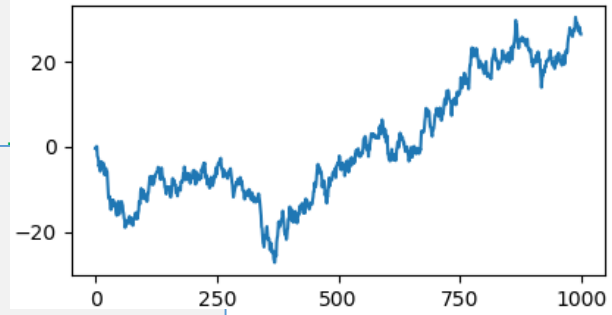


- 축눈금 설정, 눈금이름(text)설정

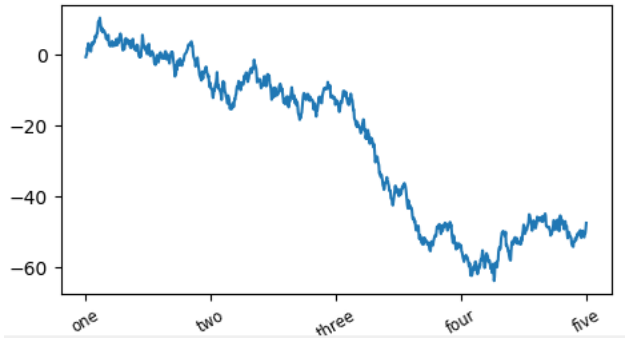
```
fig=plt.figure()
ax=fig.add_subplot(1,1,1)
data=np.random.randn(1000).cumsum()
ax.plot(data)
#ticks=ax.set_xticks([0,250,500,750,1000])
```



```
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])
```



```
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')
```



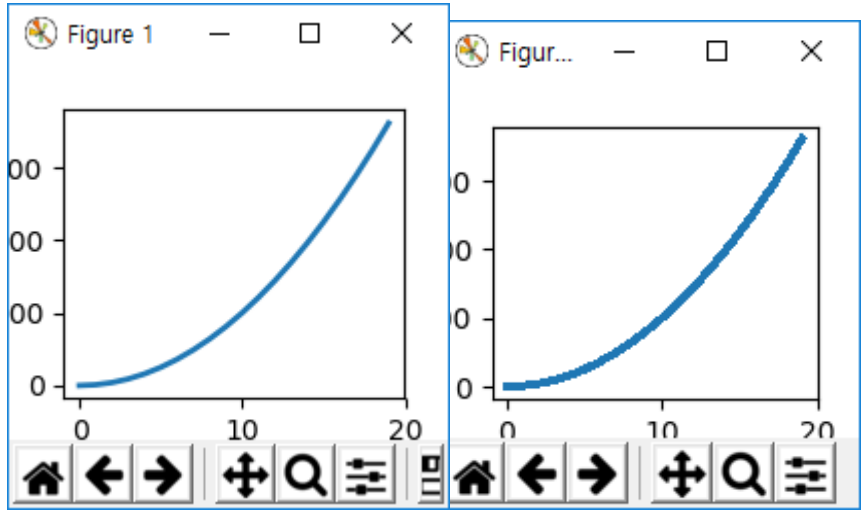
2. Plot 속성(cont.)

- line 속성(line property)
  - plot() 함수인자로 라인속성 설정하기
  - plot()함수로 반환된 line인스턴스 함수로 라인속성 설정하기
  - setp()함수를 이용하여 라인의 속성값을 점검하기

```
x=np.arange(20)
y=x**2
plt.plot(x,y,lw=2) #line width=2
plt.show()
```

```
line, = plt.plot(x, y, '-')
line.set_antialiased(False)
```

```
>>>lines = plt.plot(x, y, '-')
>>>plt.setp(lines)
 agg_filter: a filter function, ...
  alpha: float
  animated: bool
  antialiased: bool
  ...
```



- setp() 명령을 사용하여 속성 설정
  - setp는 객체 목록 또는 단일 객체와 투명하게 작동합니다.
  - 파이썬 키워드 인수 또는
  - MATLAB 스타일 문자열 / 값 쌍을 사용할 수 있습니다.

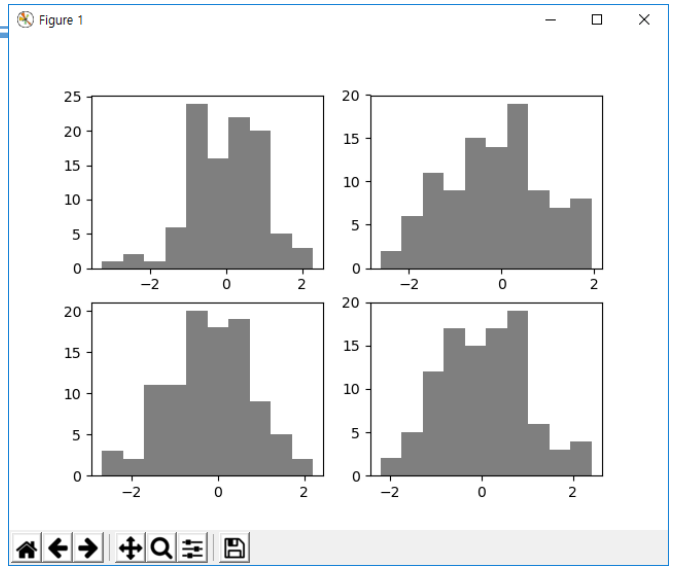
```
lines = plt.plot(x1, y1, x2, y2)
# use keyword args like python parameters
plt.setp(lines, color='r', linewidth=2.0)
# or MATLAB style string value pairs
plt.setp(lines, 'color', 'r', 'linewidth', 2.0)
```

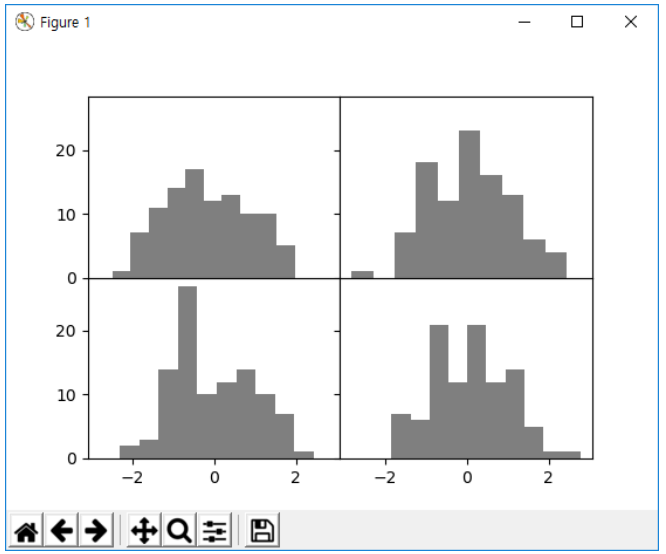
2. Plot 속성(cont.)

- subplot 간의 간격 조절

```
fig,axes=plt.subplots(2,2)#,sharex=True, sharey=True)
axes[0,0].hist(np.random.randn(100),bins=10,color='k',alpha=0.5)
axes[0,1].hist(np.random.randn(100),bins=10,color='k',alpha=0.5)
axes[1,0].hist(np.random.randn(100),bins=10,color='k',alpha=0.5)
axes[1,1].hist(np.random.randn(100),bins=10,color='k',alpha=0.5)
```

```
fig,axes=plt.subplots(2,2,sharex=True, sharey=True) #x,y축공유
axes[0,0].hist(np.random.randn(100),bins=10,color='k',alpha=0.5)
axes[0,1].hist(np.random.randn(100),bins=10,color='k',alpha=0.5)
axes[1,0].hist(np.random.randn(100),bins=10,color='k',alpha=0.5)
axes[1,1].hist(np.random.randn(100),bins=10,color='k',alpha=0.5)
plt.subplots_adjust(wspace=0,hspace=0) #subplot(좌표계)의 폭 및 높이 간격조절
```

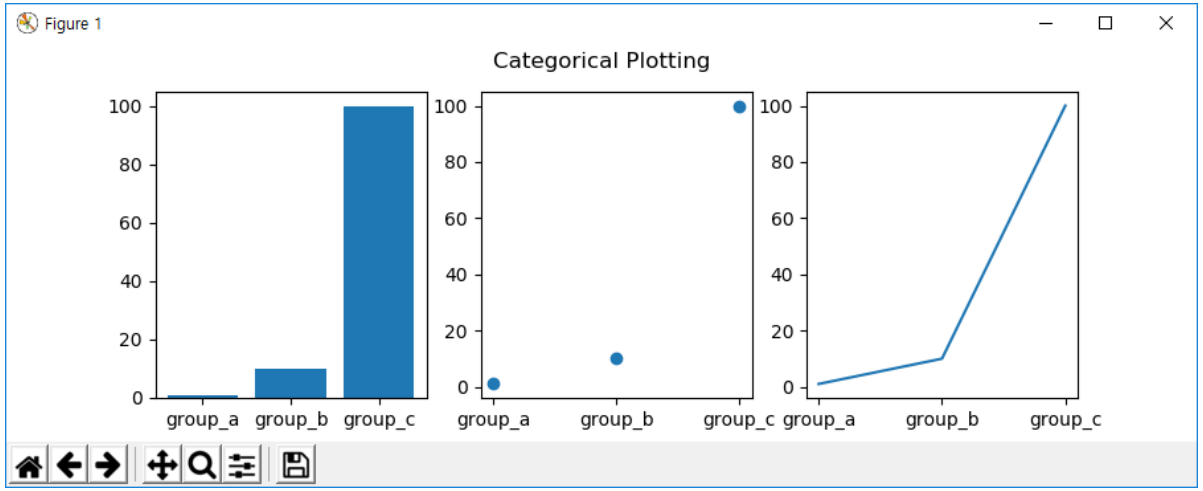




3. 프로팅 종류

- Bar, scatter (산점도),그래프
  - 범주 형 변수를 사용하여 플롯을 생성 할 수도 있습니다. Matplotlib를 사용하면 범주 형 변수를 여러 플로팅 함수에 직접 전달할 수 있습니다. 예를 들면 다음과 같습니다.

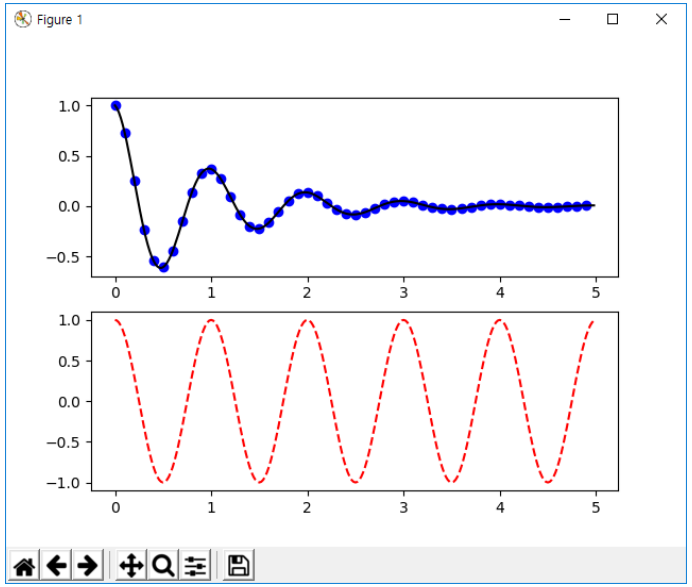
```
import matplotlib.pyplot as plt
import numpy as np

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()
```



3. 프로팅 종류(cont.)

- 시계열 데이터의 그래프

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

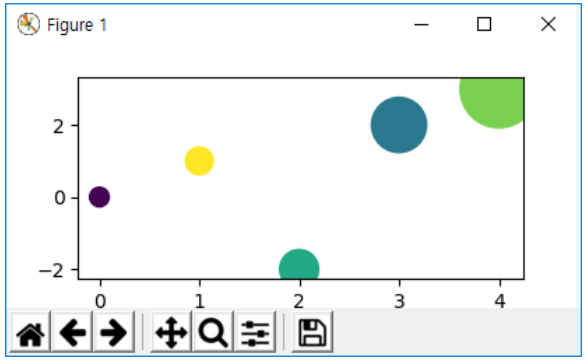


3. 프로팅 종류(cont.)

- 산점도(Scatter)
  - 예

```
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()
```

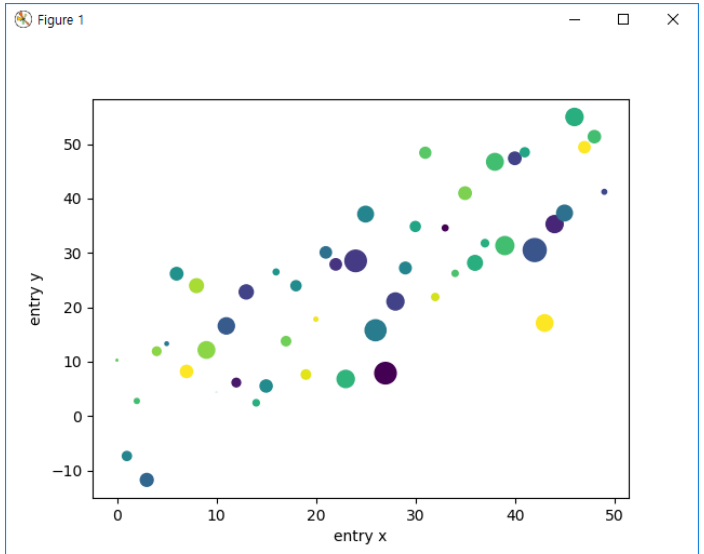


- 문자열을 사용하여 특정 변수에 액세스 할 수 있는 형식의 데이터가 있다.
  - 예를 들어, numpy.recarray 또는 pandas.DataFrame이 있다.
- Matplotlib를 사용하면 이러한 객체에 data 키워드 인수를 제공 할 수 있습니다. Scatter 함수의 파라메터 data는 이러한 기능을 제공한다. 즉 주어지는 문자열에 해당하는 사전형 변수의 키를 이용하여 프로팅을 작성할 수 있다.
- Simple example

```
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()
```

사전형을 이용한 문자열 변수 사용하기

```
>>> np.random.randint(10) #0,10,1
3
>>> np.random.randint(2,10) #2,10,1
5
>>> np.random.randint(2,10,5) #2,10,5
array([5, 9, 6, 2, 9])
>>> np.random.randn(2,3) #shape(2,3)
array([[-0.51797009, -0.07351125,  0.7998174 ],
       [ 1.26988673, -0.04959269, -1.05238163]])
```



3. 프로팅 종류(cont.)

- Histogram
  - 입력 데이터 시퀀스 $x = [55,50,57,80,90,100]$, $nb\_bins = 3$
    - $bins = [\ 50.\ \ 66.67\ \ 83.33\ 100]$,

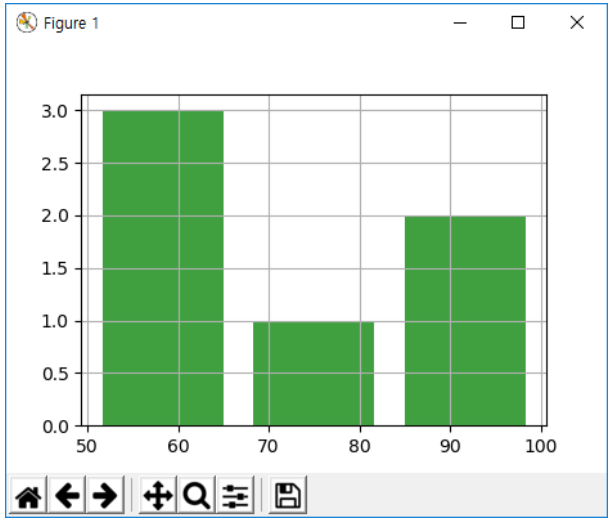
      $b_0 = 50, \quad b_1 = 50 + \frac{100-50}{3} = 66.67, \quad b_2 = 50 + \frac{100-50}{3} * 2 = 83.3\,, \quad b_3 = 100$
    - $n = [3,1,2]$,

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

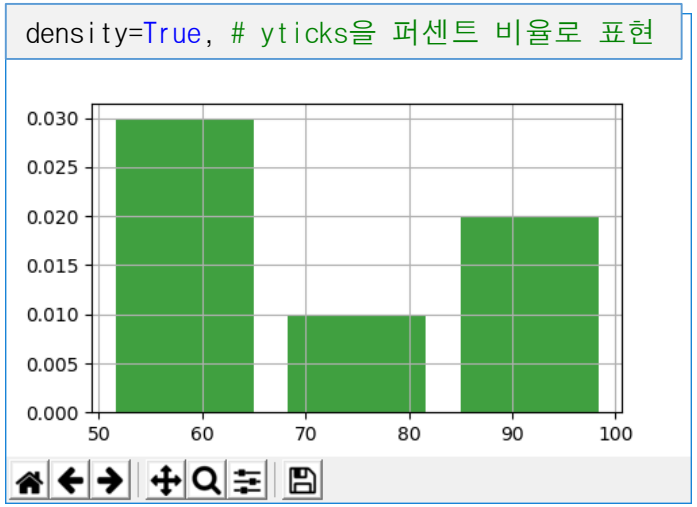
      $n_1 = count(66.67 \le x < 83.33) = 1,$

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

```
x=[50,55,57,80,90,100]
n,bins,patches=plt.hist(
x=x,                  # 입력 데이터
bins=3,               # 구분할 바구니(구간)의수
#density=True,        # yticks을 퍼센트 비율로 표현
histtype='bar' ,      # 타입. or step으로 하면 모양이 바뀜.
orientation='vertical', # or horizontal
rwidth=0.8,           # 바의 폭 비률 1.0일 경우 꽉 채움
facecolor='g' ,       # bar의 색상
alpha=0.75)           # 투명도
plt.grid()
plt.show()
```



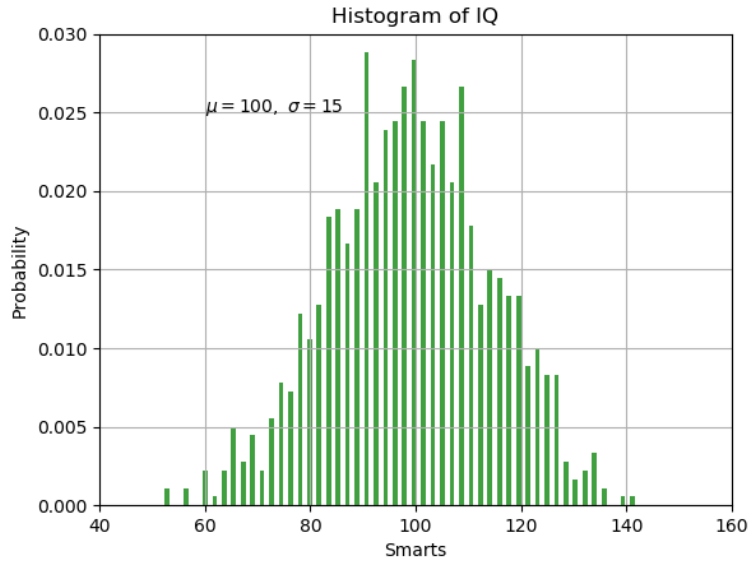

```
density=True, # yticks을 퍼센트 비율로 표현
```



3. 프로팅 종류(cont.)

- Histogram 과 text
  - text () 명령은 임의의 위치에 텍스트를 추가하는 데 사용될 수 있으며 xlabel (), ylabel () 및 title ()은 표시된 위치에 텍스트를 추가하는 데 사용됩니다 (자세한 예제는 Text in Matplotlib Plots 참조

```
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()
```



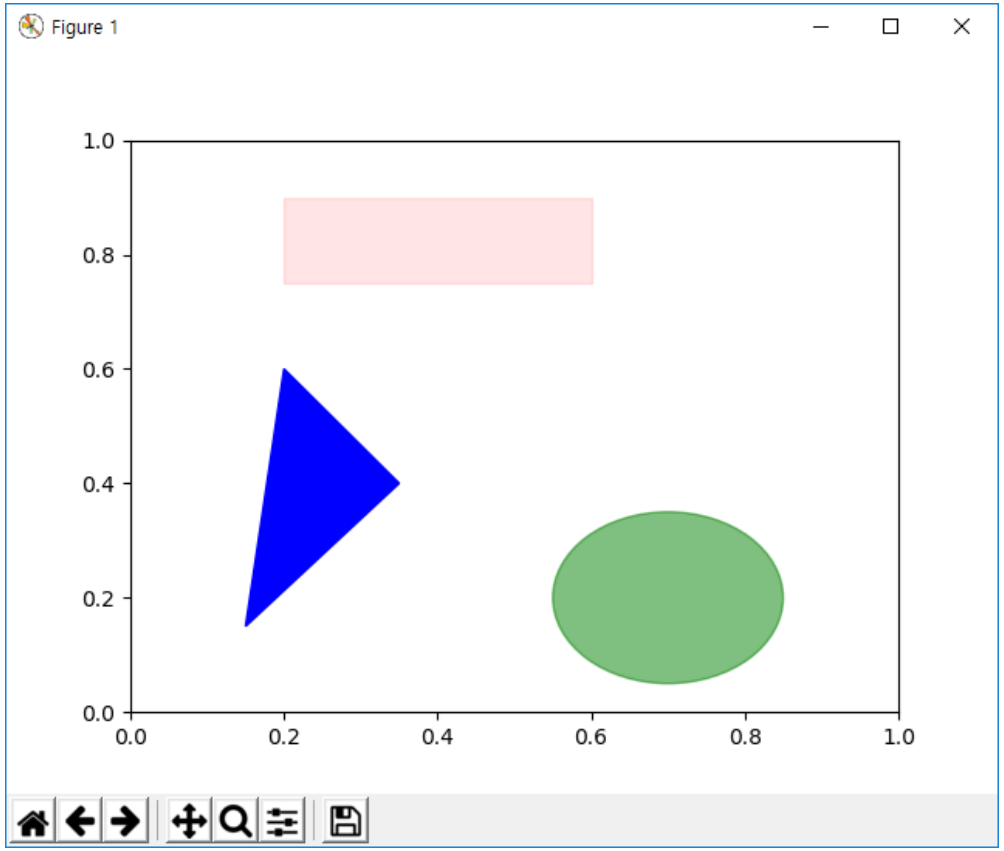
  - 모든 text () 명령은 matplotlib.text.Text 인스턴스를 반환합니다. 위의 행과 마찬가지로 키워드 인수를 텍스트 함수에 전달하거나 setp ()를 사용하여 속성을 사용자 정의 할 수 있습니다. (자세한내용은 Text properties and layout. 참조)

```
t = plt.xlabel('my data', fontsize=14, color='red')
```

3. 프로팅 종류(cont.)

- 도형 그리기

```
import matplotlib.pyplot as plt
import numpy as np
fig=plt.figure()
ax=fig.add_subplot(1,1,1)

rect    = plt.Rectangle((0.2,0.75),0.4,0.15,color='r',alpha=0.1)
circ    = plt.Circle((0.7,0.2),0.15,color='g',alpha=0.5)
pgon  = plt.Polygon([[0.15,0.15],[0.35,0.4],[0.2,0.6]],color='b',alpha=1.0)

ax.add_patch(rect)
ax.add_patch(circ)
ax.add_patch(pgon)

plt.show()
```



4. 수학식 다루기

- Using mathematical expressions in text
  - matplotlib는 모든 텍스트 표현식에서 TeX 방정식 표현식을 허용합니다. 예를 들어 제목에 $\sigma i = 15$ 표현식을 쓰려면 달러 기호로 묶은 TeX 표현식을 작성할 수 있습니다.

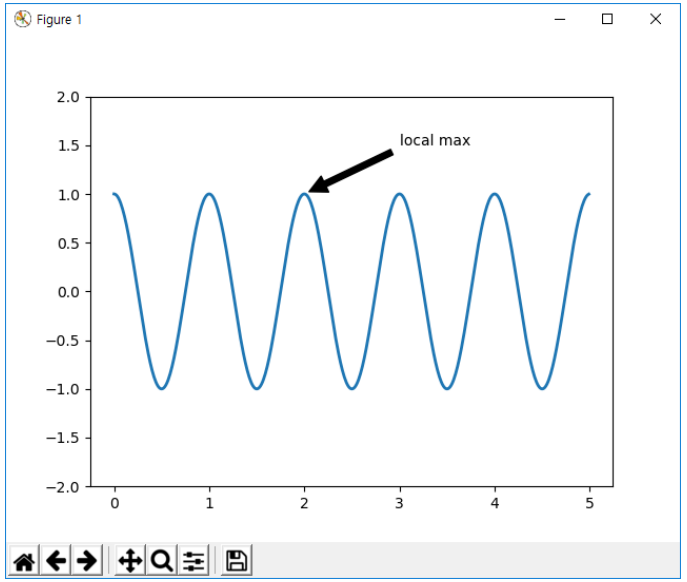
```
plt.text(60, .025, r'$₩mu=100,₩ ₩sigma=15$')
```

  - 문자열 앞의 r은 중요합니다. 문자열이 원시 문자열이며 백 슬래시를 파이썬 이스케이프로 취급하지 않음을 나타냅니다. matplotlib에는 TeX 표현식 파서 및 레이아웃 엔진이 내장되어 있으며 자체 수학 글꼴을 제공합니다. 자세한 내용은 Writing mathematical expressions을 참조하십시오. 따라서 TeX를 설치하지 않고도 여러 플랫폼에서 수학 텍스트를 사용할 수 있습니다. LaTeX 및 dvipng가 설치된 사용자의 경우 LaTeX를 사용하여 텍스트를 형식화하고 출력을 디스플레이 그림 또는 저장된 포스트 스크립트에 직접 통합 할 수 있습니다 (Text rendering With LaTeX 참조).

5 .주석 다루기

- Annotating text
  - 위의 기본 text () 명령을 사용하면 텍스트가 Axes의 임의 위치에 배치됩니다. 텍스트는 일반적으로 플롯의 일부 특성에 주석을 다는 것이고, annotate () 메서드는 주석을 쉽게 만들 수 있도록 도우미 기능을 제공합니다. 주석에서 고려해야 할 두 가지 점이 있습니다. 플롯의 특성의 위치 xy(arrow tip) 와 주석의 문자열의 위치 xytext(text location) 이다. 이 두 인수는 (x, y) 튜플입니다.

```
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()
```

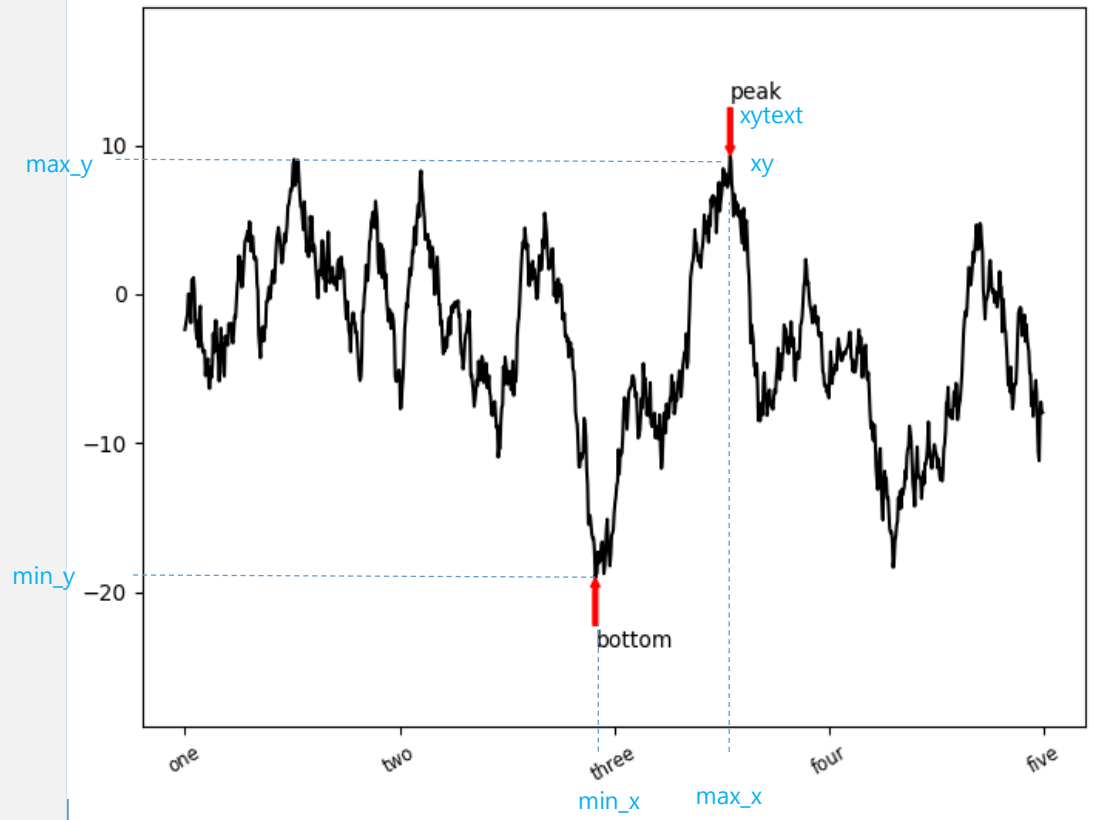


  - 다양한 좌표시스템(coordinate systems) 이있다(Basic annotation and Advanced Annotation). 다양한 사용예는 Annotating Plots를 참조

5 .주석 다루기(cont.)

- 주석과 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) # minmum 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 범위지정
```



Examples

1. 다음 그래프가 출력되는 코드를 완성하시오.

```
plt.plot(np.random.randn(30).cumsum(),
```

2. 다음 그래프가 출력되는 코드를 완성하시오.

```
fig=plt.figure(1)
ax=fig.add_subplot(211)

x=np.arange(50)
y=np.random.randn(50)




ax1=fig.add_subplot(212)
x1=x
y1=np.cos(2*np.pi*0.1*x1)
```