데이터사이언스

시각화 패키지 Matplotlib

자료를 차트(chart)나 플롯(plot)으로 시각화(visulaization)하는 패키지이다.

- http://matplotlib.org

```
import matplotlib as plt
%matplotlib inline
```

라인 플롯

선을 그리는 라인 플롯(line plot)
라인 플롯은 데이터가 시간, 순서 등에 따라 변화를 보여주기 위해 사용

만약 데이터가 1, 4, 9, 16 으로 변화하였다면

```
plt.title("Plot")
plt.plot([1,4,9,16])
plt.show()
```

x 축의 자료 위치는 자동으로 0, 1, 2, 3 이 된다.

x tick 위치를 별도로 명시하고 싶다면

```
plt.title("x축의 tick 위치를 명시")
plt.plot([10,20,30,40],[1,4,9,16])
plt.show()
```

☞ 실제로 차트로 렌더링(rendering)

스타일 지정

스타일 문자열은 색깔(color), 마커(marker), 선 종류(line style)의 순서로 지정

```
plt.title("'rs--' 스타일의 plot ")
plt.plot([10,20,30,40],[1,4,9,16],'rs--')
plt.show()
```

색

색이름, 약자, #RGB코드 사용한다.

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

마커 : 데이터 위치를 나타내는 기호

.	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

선 스타일

-	solid line style	--	dashed line style
-.	dash-dot line style	:	dotted line style

기타 스타일

라인 플롯에서 자주 사용되는 기타 스타일

color	c	선 색깔
linewidth	lw	선 굵기
linestyle	ls	선 스타일
marker		마커 종류
markersize	ms	마커 크기
markeredgecolor	mec	마커 선 색깔
markeredgewidth	mew	마커 선 굵기
markerfacecolor	mfc	마커 내부 색깔

```
plt.plot([10,20,30,40],[1,4,9,16], c="b",
lw=5, ls="--", marker="o", ms=15,
mec="g", mew=5, mfc="r")
plt.title("스타일 적용 예")
plt.show()
```

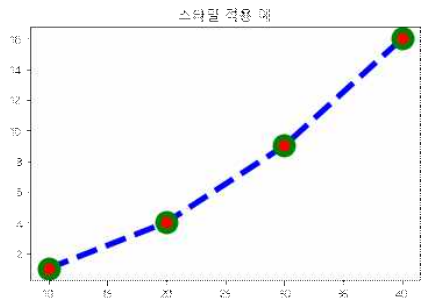
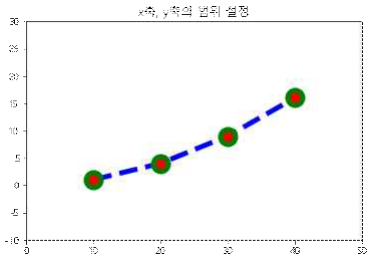


그림 범위 xlim, ylim 명령

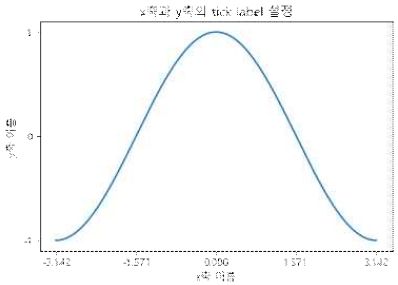
```
plt.title("x축, y축의 범위 설정")
plt.plot([10,20,30,40],[1,4,9,16],
         c="b",lw=5,ls="--",marker="o",
         ms=15,mec="g",mew=5,mfc="r")
plt.xlim(0,50)   ☞ x축의 최소, 최대값
plt.ylim(-10,30)  ☞ y축의 최소, 최대값
plt.show()
```



틱 xticks, yticks 명령

플롯이나 차트에서 축상의 위치 표시 지점을 틱(tick)
이 틱에 써진 숫자 혹은 글자를 틱 라벨(tick label)

```
X=np.linspace(-np.pi,np.pi,256)
C=np.cos(X)
plt.title("x축과 y축의 tick label 설정")
plt.plot(X,C)
plt.xlabel("x축 이름")
plt.ylabel("y축 이름")
plt.xticks([-np.pi,-np.pi/2,0,np.pi/2,np.pi])
plt.yticks([-1,0,+1])
plt.show()
```



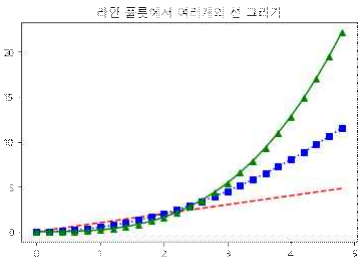
틱 라벨 문자열에는 $$ 사이에 LaTeX 수학 문자식을 넣을 수도 있다.(참고)

```
X=np.linspace(-np.pi,np.pi,256)
C=np.cos(X)
plt.title("LaTeX, 문자열로 tick label 정의")
plt.plot(X,C)
plt.xticks([-np.pi,-np.pi/2,0,np.pi/2,np.pi],
           [r'$-\pi$',r'$-\pi/2$',r'$0$',r'$+\pi/2$',r'$+\pi$'])
plt.yticks([-1,0,1],["Low","Zero","High"])
plt.grid(False)  ☜ 틱위치 보여주는 그리드 안 보이게 함
plt.show()
```

여러개의 선을 그리기

x 데이터, y 데이터, 스타일 문자열을 반복 생략 불가

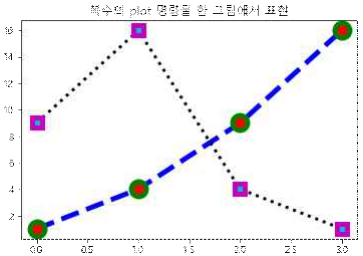
```
t=np.arange(0.,5.,0.2)
plt.title("라인 플롯 여러개 선 그리기")
plt.plot(t,t,'r--',t,0.5*t**2,
         'bs:',t,0.2*t**3,'g^-')
plt.show()
```



홀드 명령

복수의 plot 명령을 하나의 그림에 겹쳐서 그릴 수도 있다.
hold(True) 명령 : 그림 겹쳐 그리기 hold(False) 명령 : 겹치기를 종료

```
plt.title("복수의 plot 명령을 한 그림에서 표현")
plt.plot([1,4,9,16],
    c="b",lw=5,ls="--",marker="o",
    ms=15,mec="g",mew=5,mfc="r")
plt.plot([9,16,4,1], c="k",lw=3,ls=":",ms=10,
    marker="s" ,mec="m",mew=5,mfc="c")
plt.show()
```



범례 legend 명령

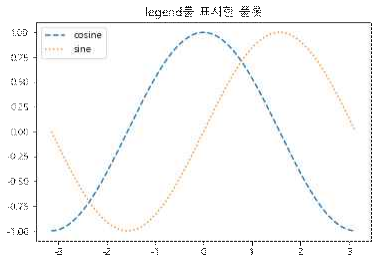
범례의 위치 : 수동으로 설정하고 싶으면 loc 인수를 사용

loc문자열	숫자
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

```
X=np.linspace(-np.pi,np.pi,256)
C,S=np.cos(X),np.sin(X)
plt.title("legend를 표시한 플롯")
plt.plot(X,C,ls="--",label="cosine")
plt.plot(X,S,ls=":",label="sine")
plt.legend(loc=2)
plt.show()
```



그림의 구조

Figure(그림이 그려지는 캔버스나 종이) 객체는 한 개 이상의 Axes 객체를 포함
Axes(하나의 플롯) 객체는 두 개 이상의 Axis(가로축, 세로축 등의 축) 객체 포함

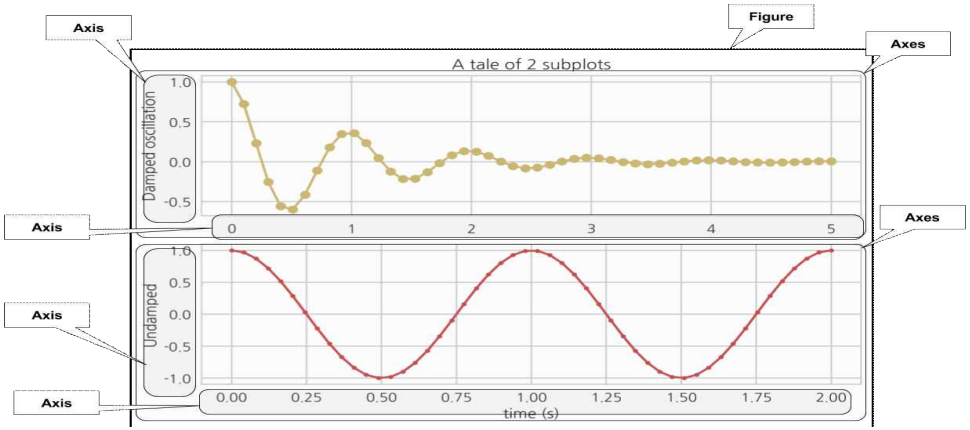
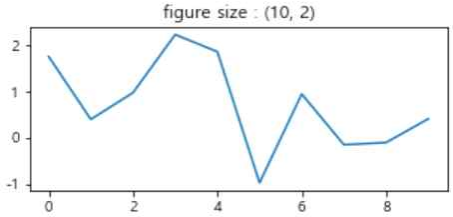


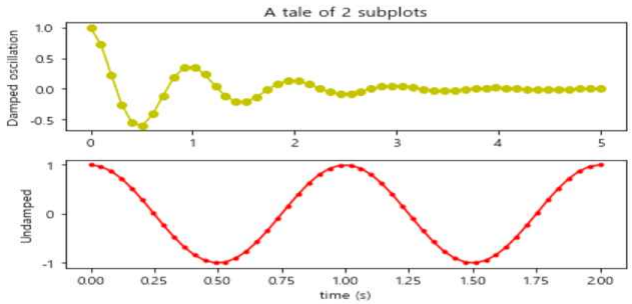
Figure 객체

```
np.random.seed(0)
f1=plt.figure(figsize=(5,2))
plt.title("figure size : (5, 2)")
plt.plot(np.random.randn(10))
plt.show()
f2=plt.gcf()
print(f1,id(f1))
print(f2,id(f2))
```




```
Figure(360x144) 1476148956104
Figure(432x288) 1476153309320

<Figure size 432x288 with 0 Axes>
```

Axes 객체와 subplot 명령

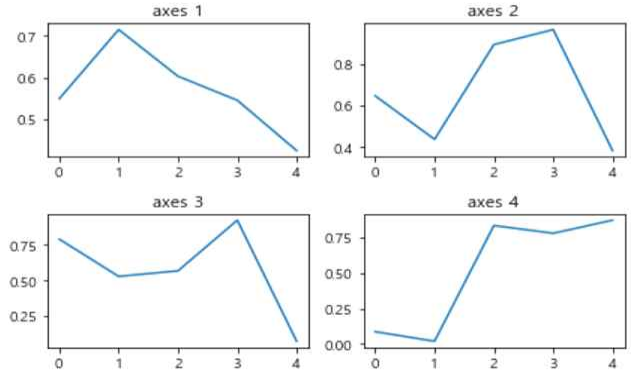



```
x1=np.linspace(0.0,5.0)
x2=np.linspace(0.0,2.0)

y1=np.cos(2*np.pi*x1)*np.exp(-x1)
y2=np.cos(2*np.pi*x2)

ax1=plt.subplot(2,1,1)
plt.plot(x1,y1,'yo-')
plt.title('A tale of 2 subplots')
plt.ylabel('Damped oscillation')

ax2=plt.subplot(2,1,2)
plt.plot(x2,y2,'r.-')
plt.xlabel('time (s)')
plt.ylabel('Undamped')

plt.tight_layout()
plt.show()

print(ax1)
print(ax2)
```

2x2 형태의 네 개의 플롯

subplot 의 인수는 (2,2,1)를 줄여서 221 라는 하나의 숫자로 표시
Axes의 위치는 위에서부터 아래로, 왼쪽에서 오른쪽으로 카운트한다.

```
np.random.seed(0)

plt.subplot(221)
plt.plot(np.random.rand(5))
plt.title("axes 1")

plt.subplot(222)
plt.plot(np.random.rand(5))
plt.title("axes 2")

plt.subplot(223)
plt.plot(np.random.rand(5))
plt.title("axes 3")

plt.subplot(224)
plt.plot(np.random.rand(5))
plt.title("axes 4")

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

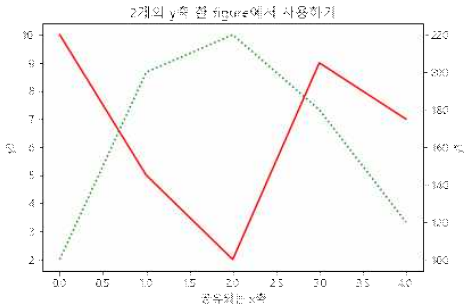


subplots 명령으로 복수의 Axes 객체를 동시에 생성할 수도 있다.
이때는 2차원 ndarray 형태로 Axes 객체가 반환된다.

```
fig,axes=plt.subplots(2,2)
np.random.seed(0)
axes[0,0].plot(np.random.rand(5))
axes[0,0].set_title("axes 1")
axes[0,1].plot(np.random.rand(5))
axes[0,1].set_title("axes 2")
axes[1,0].plot(np.random.rand(5))
axes[1,0].set_title("axes 3")
axes[1,1].plot(np.random.rand(5))
axes[1,1].set_title("axes 4")

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

Axis 객체와 축

twinx 명령 : 복수의 y 축을 가진 플롯
x 축을 공유하는 새로운 Axes 객체를 만든다.




```
fig,ax0=plt.subplots()
ax1=ax0.twinx()
ax0.set_title("2개의 y축 한 figure에서 사용하기")
ax0.plot([10,5,2,9,7],'r-',label="y0")
ax0.set_ylabel("y0")
ax0.grid(False)
ax1.plot([100,200,220,180,120],'g:',label="y1")
ax1.set_ylabel("y1")
ax1.grid(False)
ax0.set_xlabel("공유되는 x축")
plt.show()
```