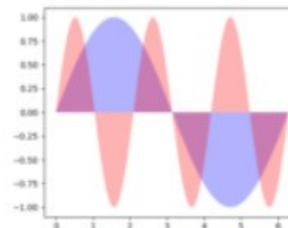
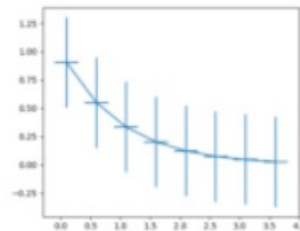
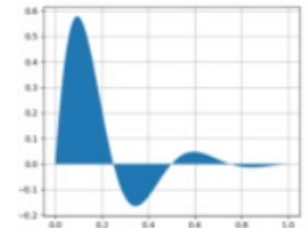
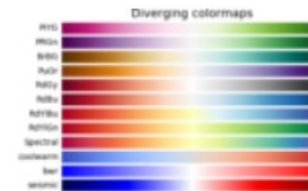
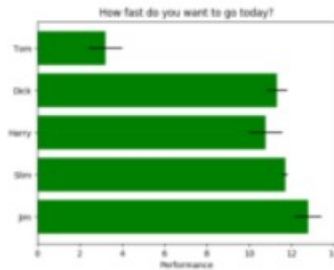
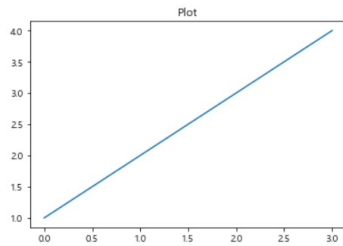


■ Pyplot

- 시각화 패키지 Matplotlib의 서브 패키지
- 간단한 시각화 프로그램을 만드는 경우 사용
- Line / Bar / Shape / Statistical / Pie 등 다양한 차트 사용 가능



<https://matplotlib.org/gallery.html>

■ Pyplot

● pyplot 사용 전 모듈 import

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

● 한글 사용 시 글자 깨짐 현상 해결 방법

```
from matplotlib import font_manager, rc  
font_name = font_manager.FontProperties(fname="c:/Windows/Fonts/malgun.ttf").get_name()  
rc('font', family=font_name)
```

```
from matplotlib import font_manager, rc  
font_name =  
font_manager.FontProperties(fname='c:/Windows/Fonts/malgun.ttf').get_name()  
rc('font', family=font_name)
```

● plot 생성 후 제어 가능

```
%matplotlib nbagg
```

● plot 생성 후 제어 불가 (기본값)

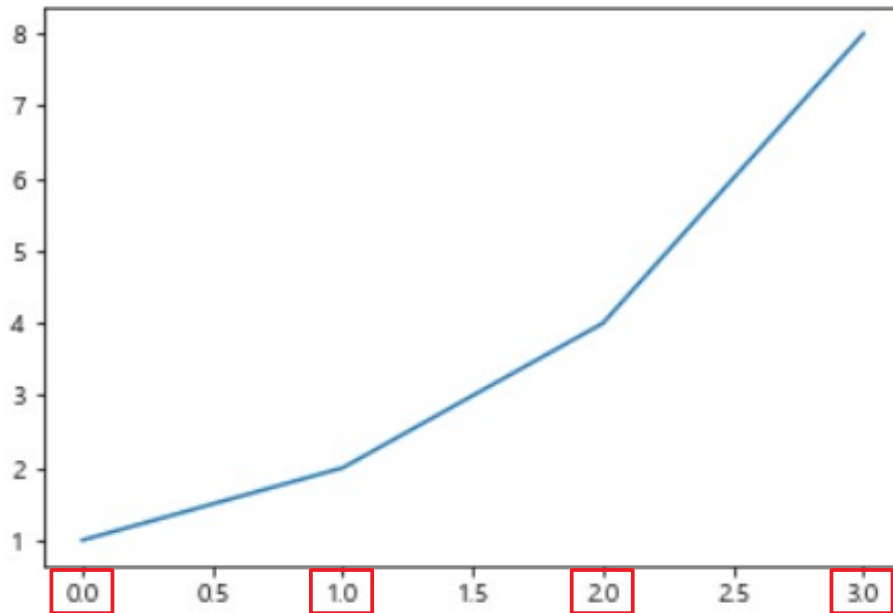
```
%matplotlib inline
```

■ Line Plot

- 리스트 또는 NumPy의 ndarray 를 이용하여 데이터 삽입

– X 축에 나타내는 자료의 위치(tick)은 자동으로 0, 1, 2, 3으로 설정

```
plt.plot([1, 2, 4, 8])  
plt.show()
```

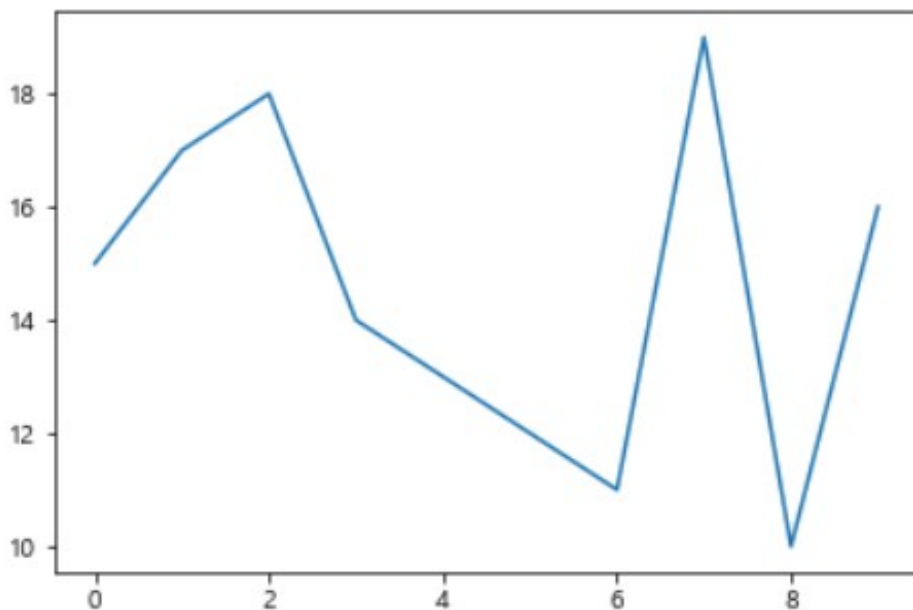


■ Line Plot

- 리스트 또는 NumPy의 ndarray 를 이용하여 데이터 삽입

– X / Y 축 지정

```
plt.plot(range(0, 10), [15, 17, 18, 14, 13, 12, 11, 19, 10, 16])  
plt.show()
```

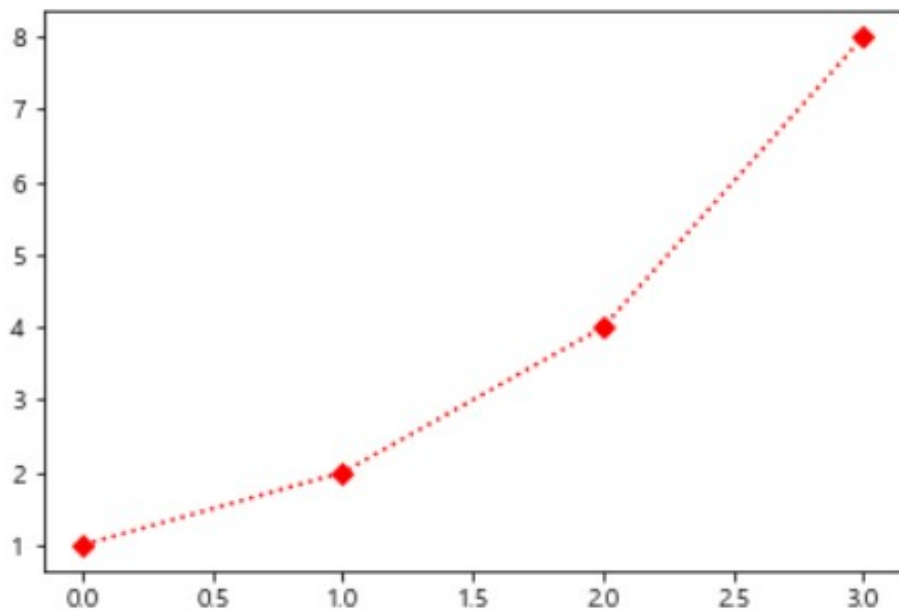


■ Line Plot

● 스타일 지정









– 색상, 마커, 스타일 순으로 지정

```
plt.plot([1, 2, 4, 8], 'rD:') # red, diamond, dotted  
plt.show()
```























■ Line Plot

● 색상

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





■ Line Plot

● 마커

약자	의미	표현	약자	의미	표현
.	point		D	diamond	
,	pixel		d	thin diamond	
o	circle		*	star	
s	square		+	plus	
p	pentagon		x	x	
1	tri_down		v	triangle down	
2	tri_up		^	triangle up	
3	tri_left		<	triangle left	
4	tri_right		>	triangle left	
h	hexagon1		H	hexagon2	

■ Line Plot

● 선 스타일

약자	의미	표현
—	solid	
--	dashed	
-. .	dash-dot	
:	dotted	

■ Line Plot

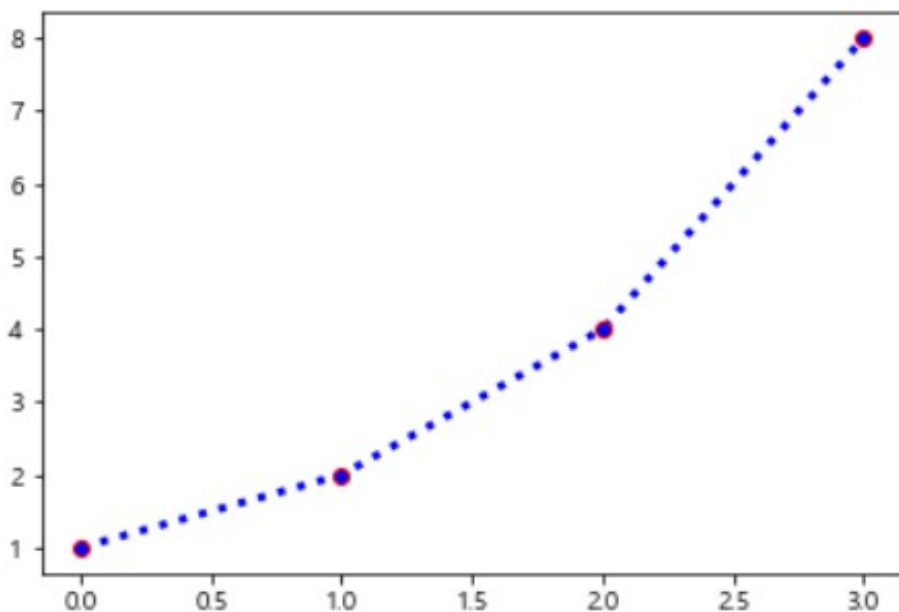
● 기타 스타일

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

■ Line Plot

● 여러가지 스타일 적용

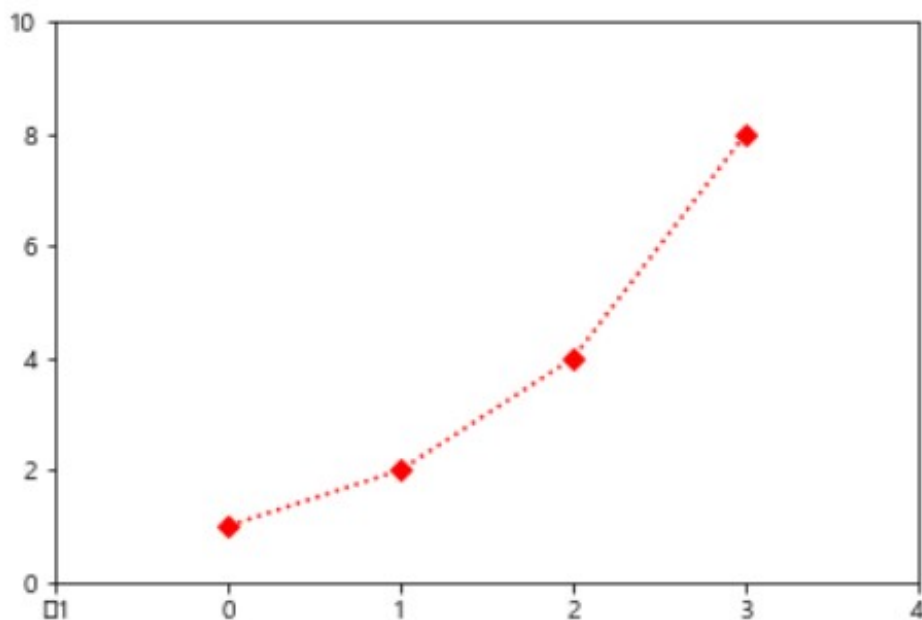
```
plt.plot([1, 2, 4, 8], color='blue', lw=3, ls=':', marker='o', mec='r')  
plt.show()
```



■ Line Plot

● 차트 표현 범위 지정

```
plt.plot([1, 2, 4, 8], 'rD:')  
plt.xlim(-1, 4)  
plt.ylim(0, 10)  
plt.show()
```

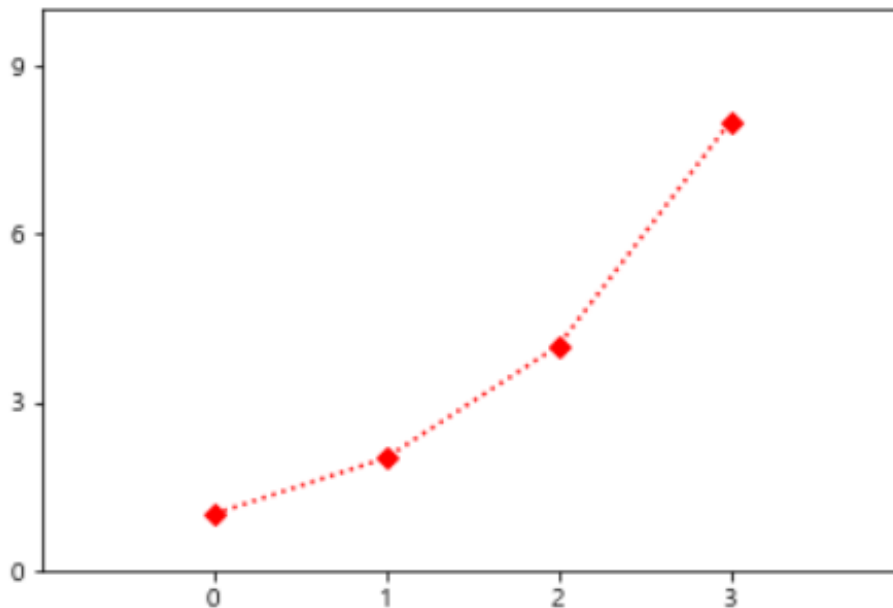


■ Line Plot

● X / Y 축 상의 위치 표시 지점 (tick)

- X 축 : plot에 입력된 데이터의 위치
- Y 축 : 데이터의 위치를 가늠하기 위한 지표

```
plt.plot([1, 2, 4, 8], 'rD:')  
plt.xlim(-1, 4)  
plt.ylim(0, 10)    lim 범위가 tick 보다 적은 경우 무시  
plt.xticks([0, 1, 2, 3])    ticks의 개수와 데이터가 다른 경우 상황에 따라 동적으로 표현  
plt.yticks([0, 3, 6, 9])  
plt.show()
```

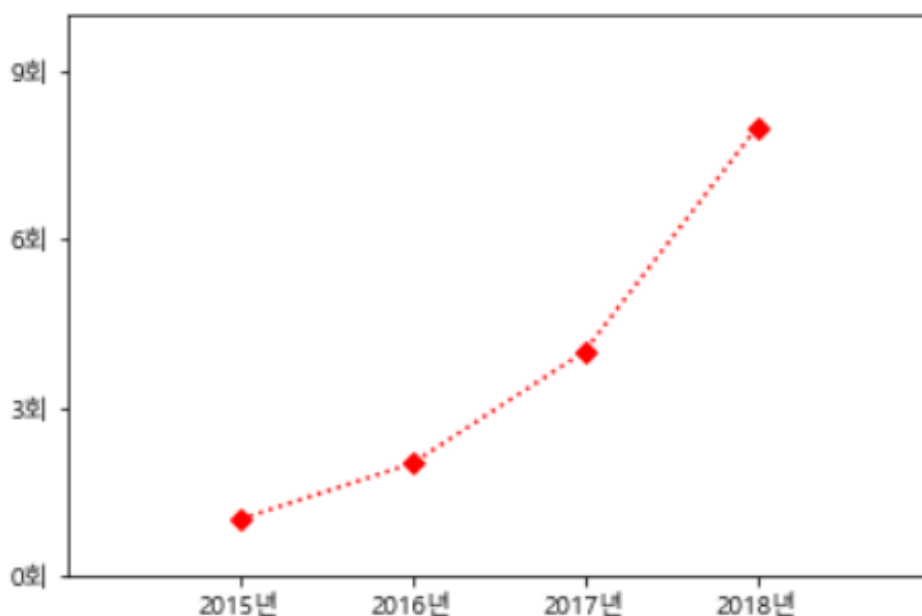


■ Line Plot

● X / Y 축 상의 위치 표시 지점 (tick)

– 문자열 표현

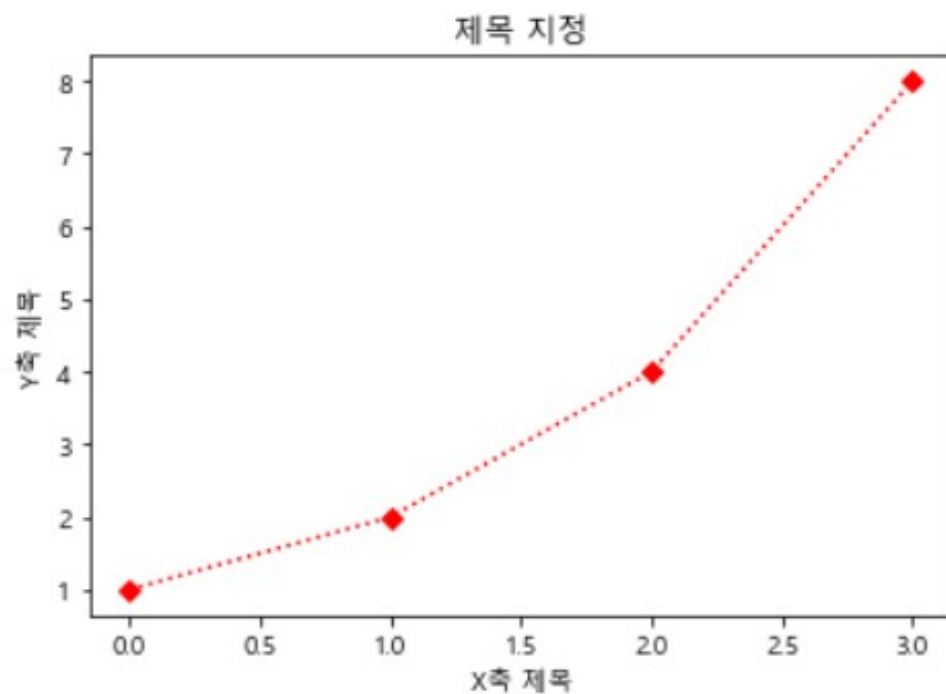
```
plt.plot([1, 2, 4, 8], 'rD:')  
plt.xlim(-1, 4)  
plt.ylim(0, 10)  
plt.xticks([0, 1, 2, 3], ['2015년', '2016년', '2017년', '2018년'])  
plt.yticks([0, 3, 6, 9], ['0회', '3회', '6회', '9회'])  
plt.show()
```



■ Line Plot

● title / label 지정

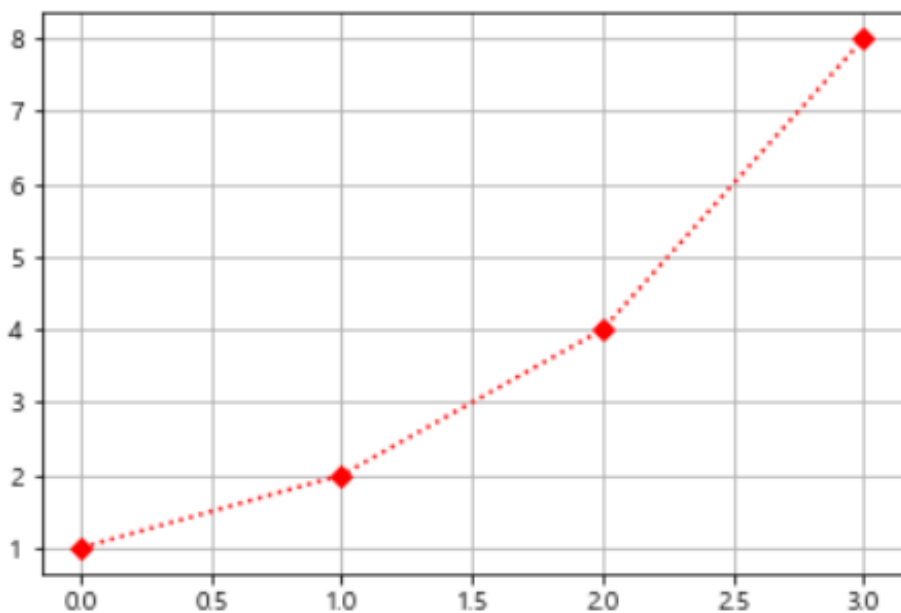
```
plt.plot([1, 2, 4, 8], 'rD:')  
plt.title('제목 지정')  
plt.xlabel('X축 제목')  
plt.ylabel('Y축 제목')  
plt.show()
```



■ Line Plot

● grid 지정

```
plt.plot([1, 2, 4, 8], 'rD:')  
plt.grid(True)  
plt.show()
```

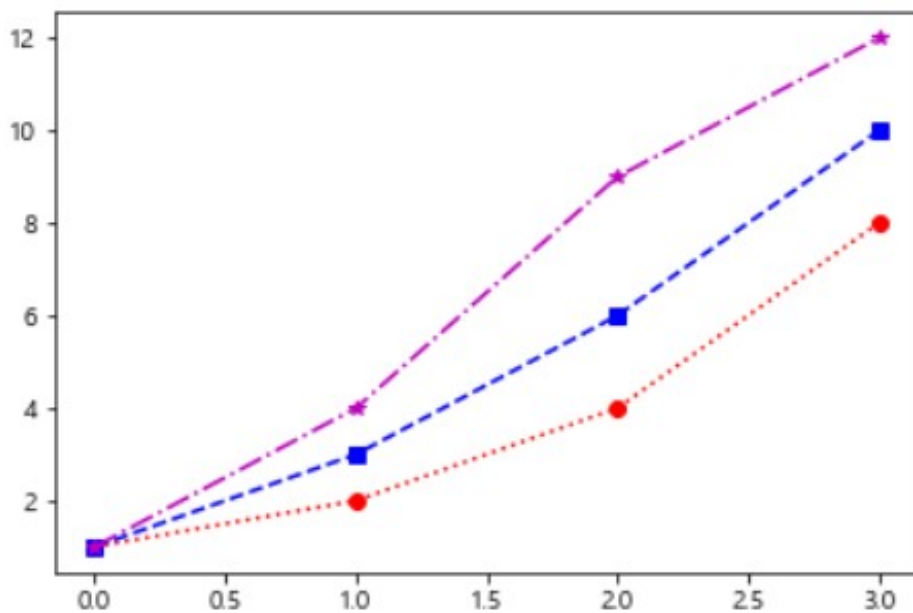


■ Line Plot

● 여러개 라인 그리기

```
data1 = [1, 2, 4, 8]  
data2 = [1, 3, 6, 10]  
data3 = [1, 4, 9, 12]
```

```
plt.plot(data1, 'ro:', data2, 'bs--', data3, 'm*-')  
plt.show()
```



■ Line Plot

● 범례

- 여러개의 라인을 그리는 경우 각 선이 어떤 자료를 나타내는지 표시
- 기본적으로 위치는 자동으로 정해지지만 수동으로 설정하고 싶을때
loc 인수 사용 (문자열 또는 숫자)

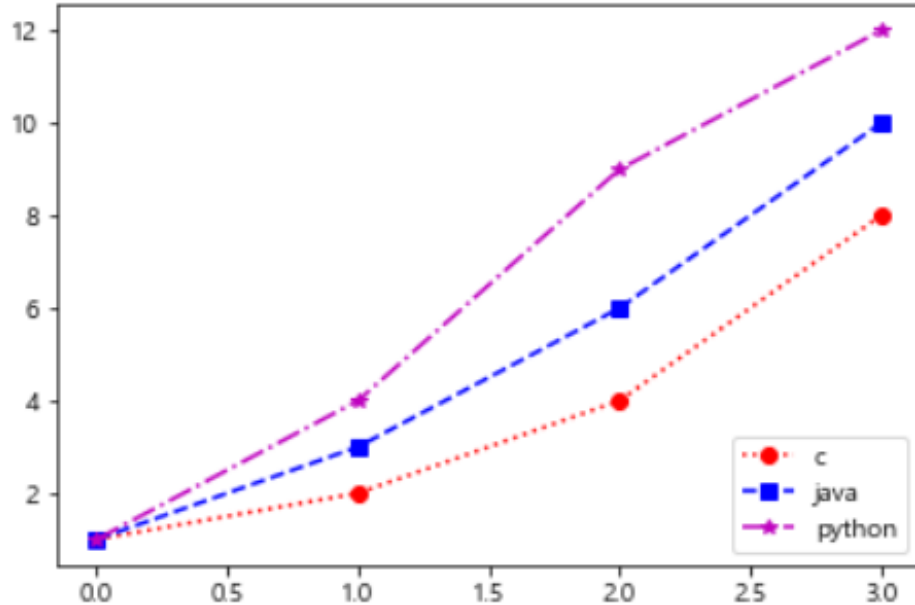
문자열	숫자	문자열	숫자
best	0	center left	6
upper right	1	center right	7
upper left	2	lower center	8
lower left	3	upper center	9
lower right	4	center	10
right	5		

■ Line Plot

● 범례

```
data1 = [1, 2, 4, 8]
data2 = [1, 3, 6, 10]
data3 = [1, 4, 9, 12]

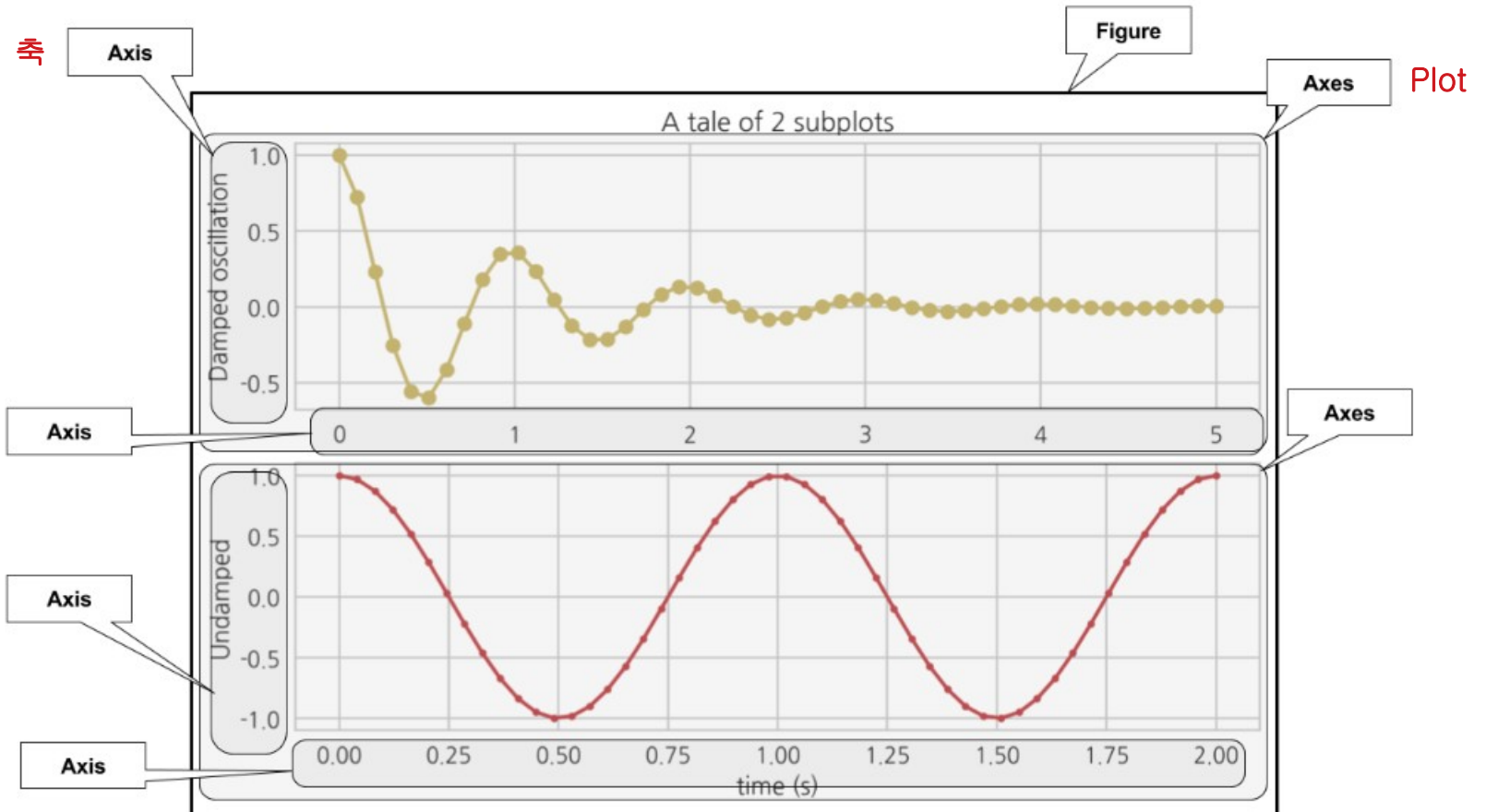
plt.plot(data1, 'ro:', label='c')
plt.plot(data2, 'bs—', label='java')
plt.plot(data3, 'm*-.', label='python')
plt.legend(loc='lower right') # loc=4
plt.show()
```



■ Line Plot

● Plot 구조

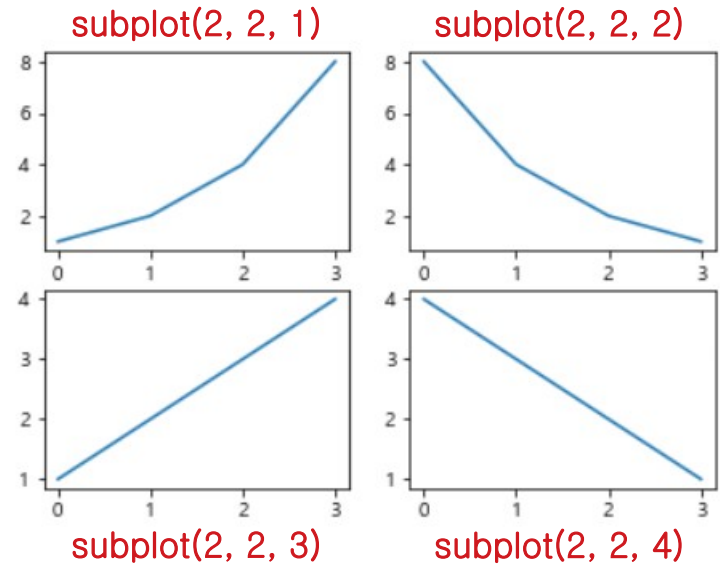
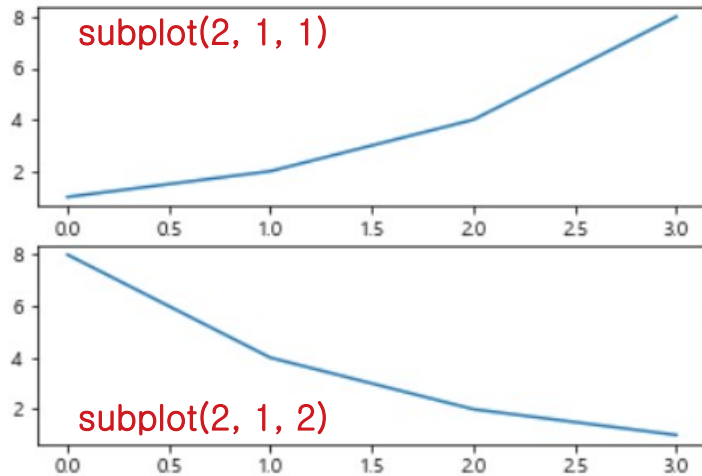
Plot이 그려지는 공간



■ Line Plot

● 여러개 Plot 그리기 (subplot)

- 그리드 형태의 Axes 객체 생성
- Figure = 행렬(matrix) / Axes = 행렬의 요소(plot)
- `tight_layout()` 으로 자동 간격 맞춤



■ Line Plot

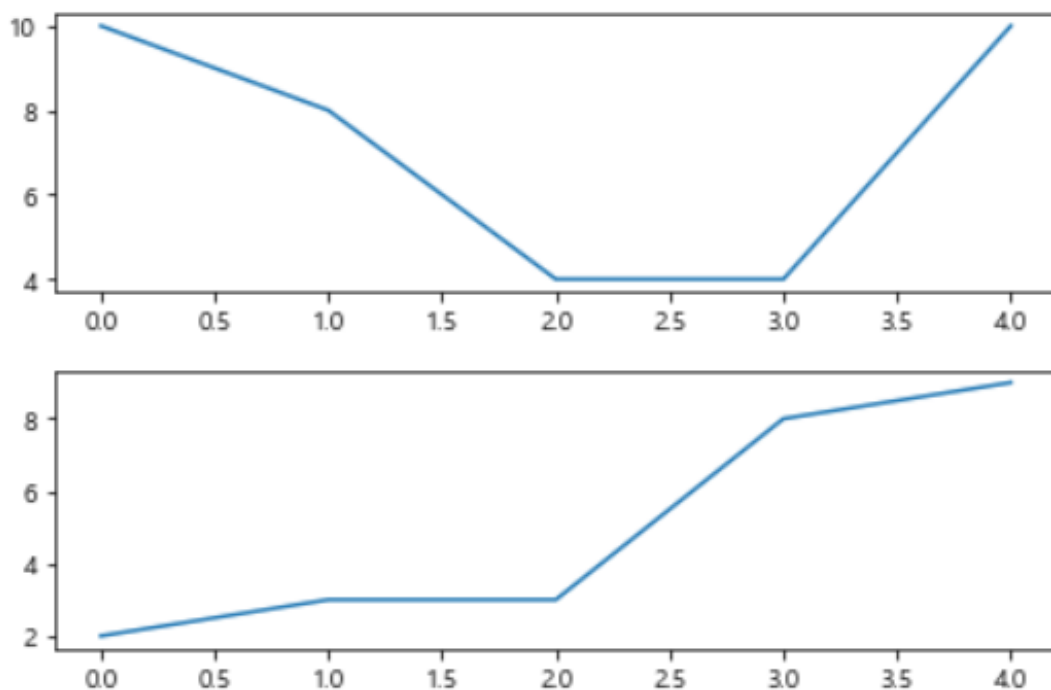
● 2개 Plot 그리기 (subplot)

```
ax1 = plt.subplot(2, 1, 1)
plt.plot([random.randint(0, 10) for i in range(5)])

ax2 = plt.subplot(2, 1, 2)
plt.plot([random.randint(0, 10) for i in range(5)])

plt.tight_layout()

plt.show()
```



■ Line Plot

● 4개 Plot 그리기 (subplot)

```
import random

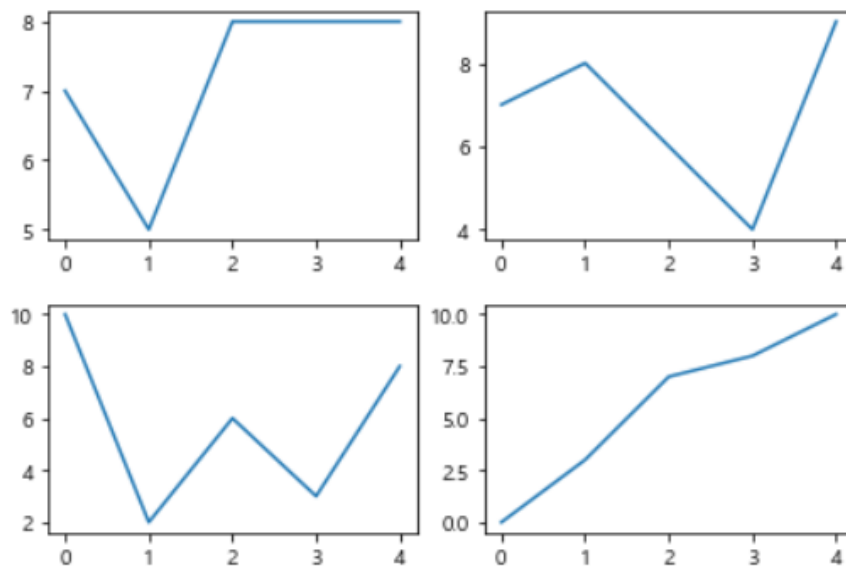
ax1 = plt.subplot(221)
plt.plot([random.randint(0, 10) for i in range(5)])

ax2 = plt.subplot(222)
plt.plot([random.randint(0, 10) for i in range(5)])

ax3 = plt.subplot(223)
plt.plot([random.randint(0, 10) for i in range(5)])

ax4 = plt.subplot(224)
plt.plot([random.randint(0, 10) for i in range(5)])

plt.show()
```



■ Line Plot

● 두 개의 Y축(axis) 표현

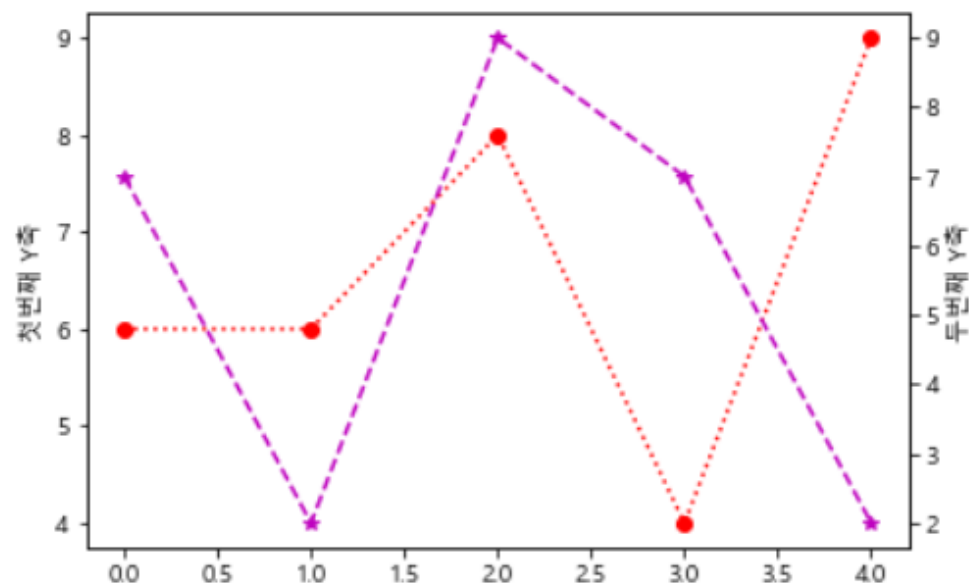
```
import random

figure, axis1 = plt.subplots()
axis2 = axis1.twinx()

axis1.plot([random.randint(0, 10) for i in range(5)], 'ro:')
axis1.set_ylabel('첫번째 Y축')

axis2.plot([random.randint(0, 10) for i in range(5)], 'm*--')
axis2.set_ylabel('두번째 Y축')

plt.show()
```

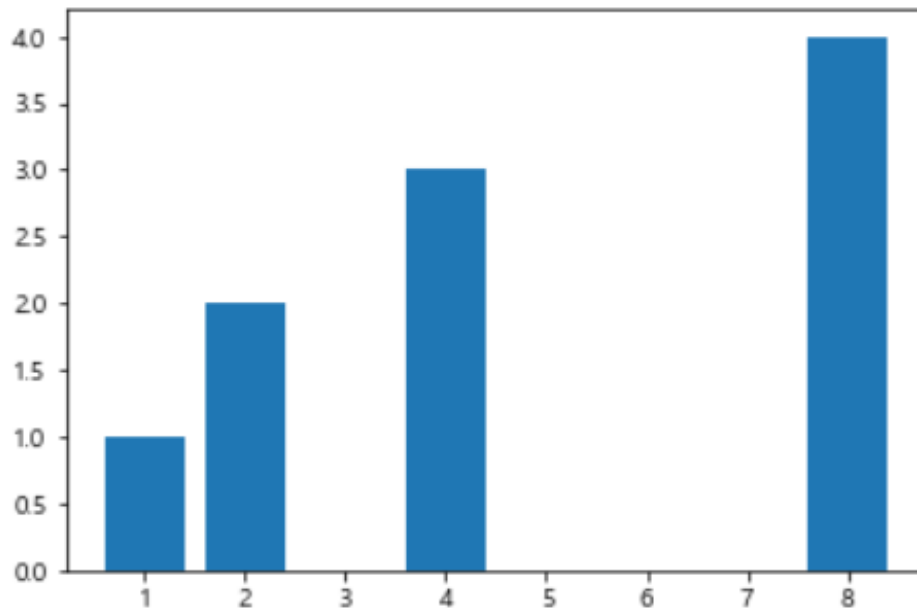


■ Bar Plot

- 리스트 또는 NumPy의 ndarray 를 이용하여 데이터 삽입

– X 축에 나타내는 자료의 위치(tick)은 자동으로 0, 1, 2, 3으로 설정

```
plt.bar([1, 2, 4, 8], [1, 2, 3, 4])  
plt.show()
```



■ Barh Plot

- 리스트 또는 NumPy의 ndarray 를 이용하여 데이터 삽입

– X 축에 나타내는 자료의 위치(tick)은 자동으로 0, 1, 2, 3으로 설정

```
plt.barh([1, 2, 4, 8], [1, 2, 3, 4])  
plt.show()
```

