In [1]:
```
import matplotlib.pyplot as plt #그래프 패키지 모듈 등록
%matplotlib inline
#그래프는 show()함수를 통해서 독립창에서 실행되는 것이 원칙
#그래프를 콘솔에서 바로 작도되록 하는 설정
```

In [2]:
```
# from IPython.core.interactiveshell import InteractiveShell
# InteractiveShell.ast_node_interactivity="all"
```

In [3]:
```
import pandas as pd
import numpy as np
```

In [4]:
```
# 한글 문제
# matplotlit의 기본 폰트에서 한글 지원되지 않기 때문에
# matplotlib의 폰트 변경 필요
import platform

from matplotlib import font_manager, rc
plt.rcParams['axes.unicode_minus'] = False

if platform.system() == 'Darwin':  # 맥OS
    rc('font', family='AppleGothic')
elif platform.system() == 'Windows':  # 윈도우
    path = "c:/Windows/Fonts/malgun.ttf"
    font_name = font_manager.FontProperties(fname=path).get_name()
    rc('font', family=font_name)
else:
    print('Unknown system...  sorry~~~')
```

Matplotlib :

- 시각화 패키지
- 파이썬 표준 시각화 도구로 불림
- 2D 평면 그래프에 관한 다양한 포맷과 기능 지원
- 데이터 분석 결과를 시각화 하는데 필요한 다양한 기능을 제공

패키지 사용 법

1.matplotlib 주 패키지 사용시

- import matplotlib as mpl

2.pylab 서브 패키지 사용시 : 주로 사용 한다.

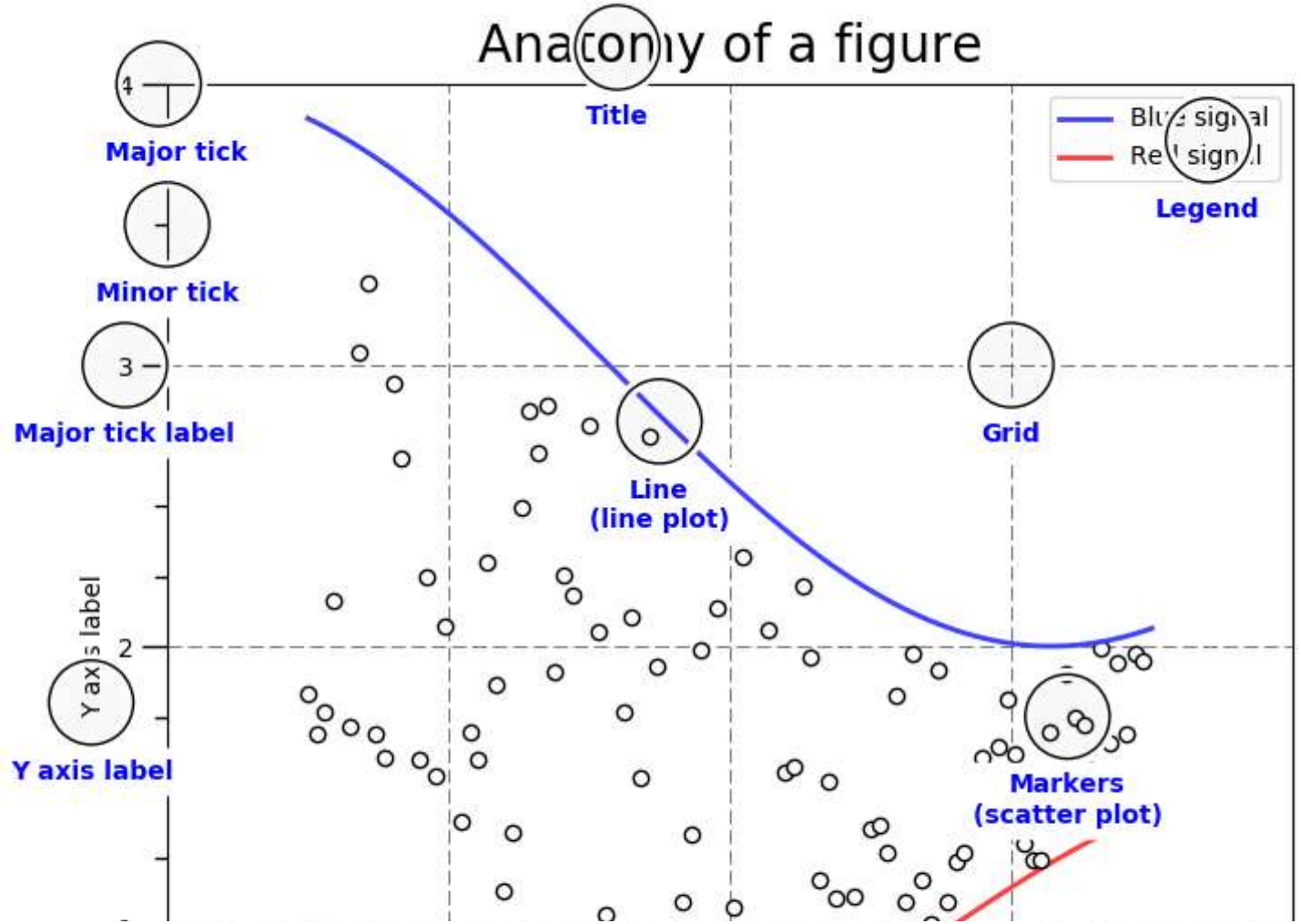
- import matplotlib.pyplot as plt

- 매직 명령어 %matplotlib inline
  - 주피터 노트북 사용시 노트북 내부에 그림을 표시하도록 지정하는 명령어

- 지원 되는 플롯 유형

- 라인플롯(line plot) : plot()
- 바 차트(bar chart) : bar()
- 스캐터플롯(scatter plot) : scatter()
- 히스토그램(histogram) : hist()
- 박스플롯(box plot) : boxplot()
- 파이 차트(pie chart) : pie()
- 기타 다양한 유형의 차트/플롯을 지원 : 관련 홈페이지를 참고

그래프 용어 정리



1. 라인 플롯 : plot()

함수설명 : plot()

- 기본으로 선을 그리는 함수
- 데이터가 시간, 순서 등에 따라 변화를 보여주기 위해 사용

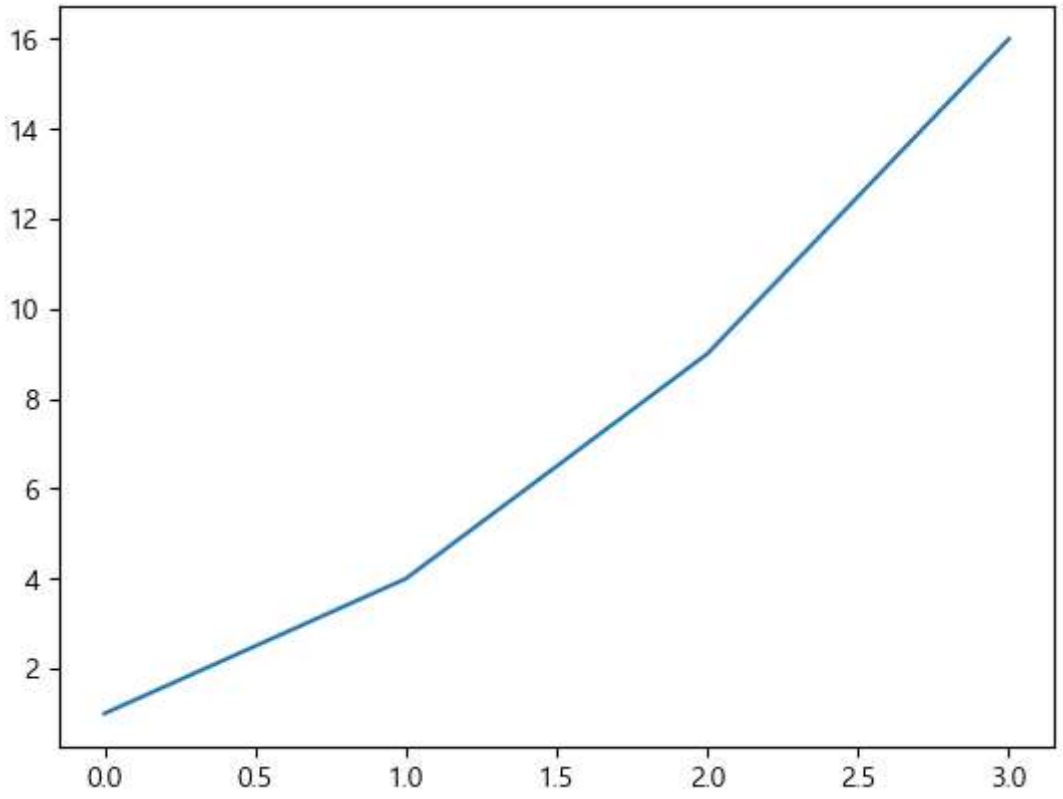
- show()

    - 각화명령(그래프 그리는 함수) 후 실제로 차트로 렌더링 하고 마우스 이벤트등의 지시를 기다리는 함수
    - 주피터 노트북 에서는 셀 단위로 플롯 명령을 자동으로 렌더링 주므로 show 명령이 필요 없지만
    - 일반 파이썬 인터프리터(파이참)로 가동되는 경우를 대비해서 항상 마지막에 실행하도록 한다 .

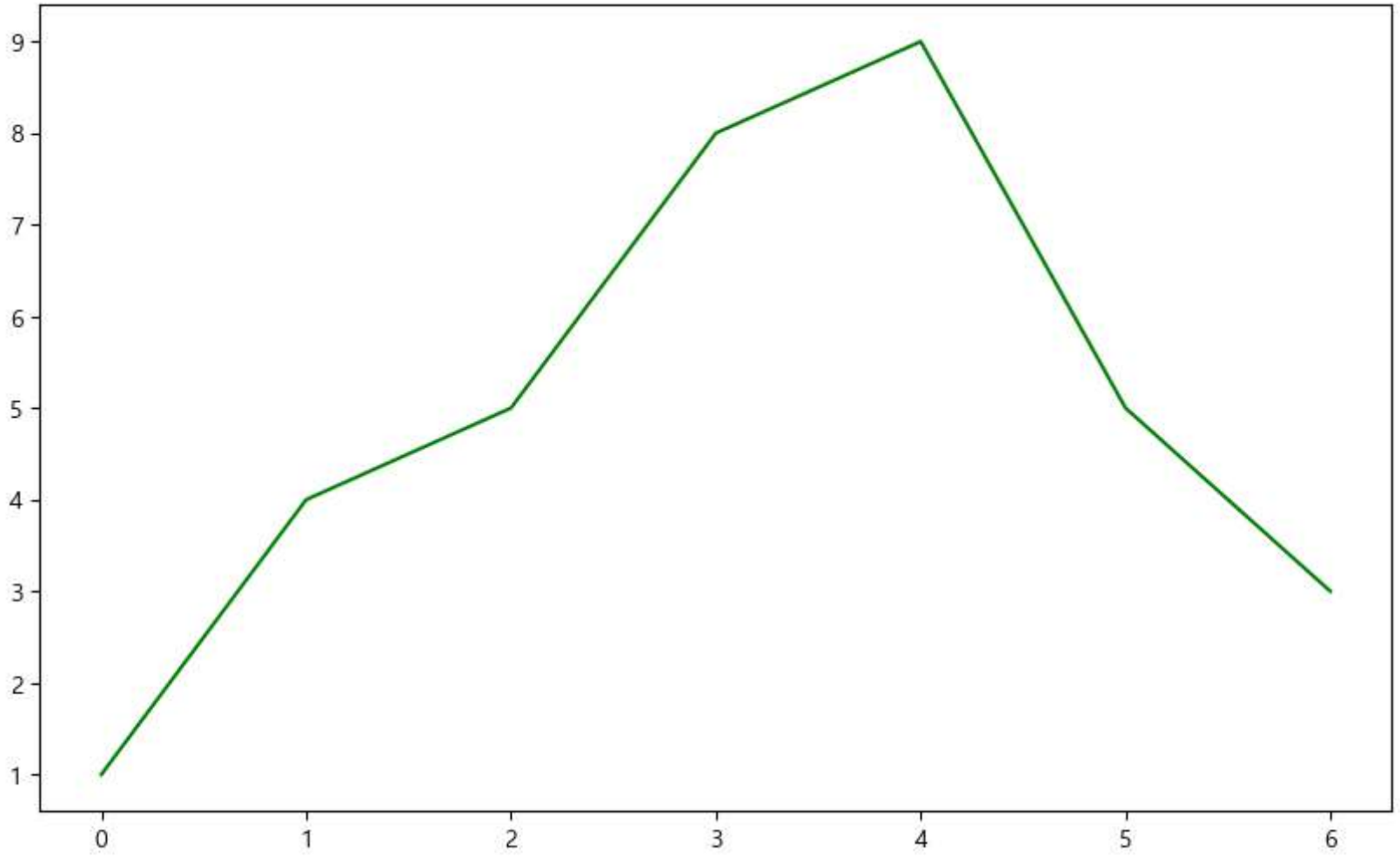
- 관련 함수 및 속성
    - figure(x,y) : 그래프 크기 설정 : 단위 인치
    - title() : 제목 출력
    - xlim(범위값): x 축 범위
    - ylim(범위값) : y 축 범위
    - xticks():yticks() : 축과 값을 잇는 실선
    - legend() : 범례
    - xlabel() : x축라벨(값)
    - ylabel() : y축라벨(값)
    - grid() : 그래프 배경으로 grid 사용 결정 함수

- line plot 에서 자주 사용되는 스타일 속성(약자로도 표기 가능)
    - color:c(선색깔)
    - linewidth : lw(선 굵기)
    - linestyle: ls(선스타일)
    - marker:마커 종류
    - markersize : ms(마커크기)
    - markeredgecolor:mec(마커선색깔)
    - markeredgewidth:mew(마커선굵기)
    - markerfacecolor:mfc(마커내부색깔)

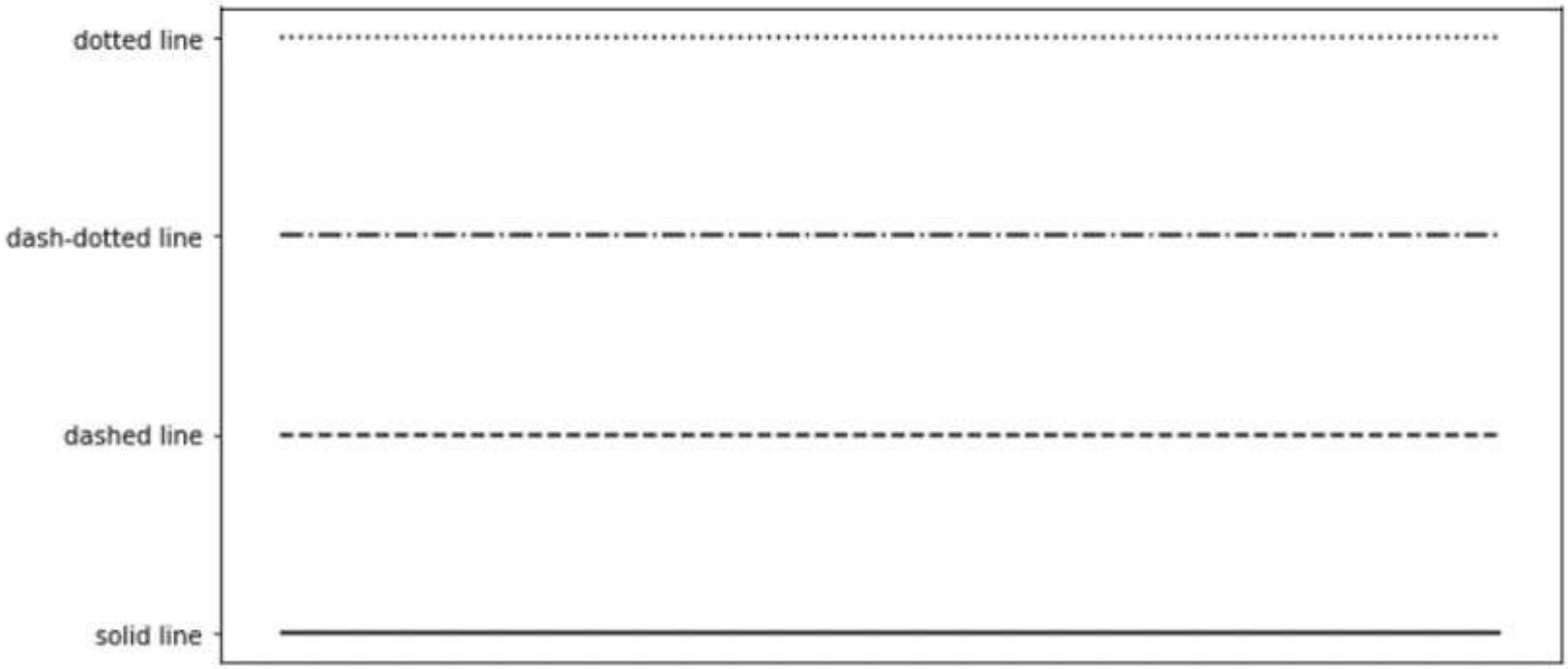
In [5]:
```
#plt.plot([]) 기본 문법 : []에 y 축값, x축값은 자동 생성
plt.plot([1,4,9,16]) # x축값 [0,1,2,3] 자동으로 생성
plt.show()
```



In [6]:
```
# 그래프 크기설정 및 선 색상설정
#색상은 단어로 지정 : color='green'

t=[0,1,2,3,4,5,6]
y=[1,4,5,8,9,5,3]
plt.figure(figsize=(10,6)) # 단위: 인치(가로,세로)
plt.plot(t,y,color='green')
plt.show()
```



- linestyle =



- marker =

지정자	마커 유형
'+'	플러스 기호
'o'	원
'*'	별표
'.'	점
'x'	십자
's'	정사각형
'd'	다이아몬드

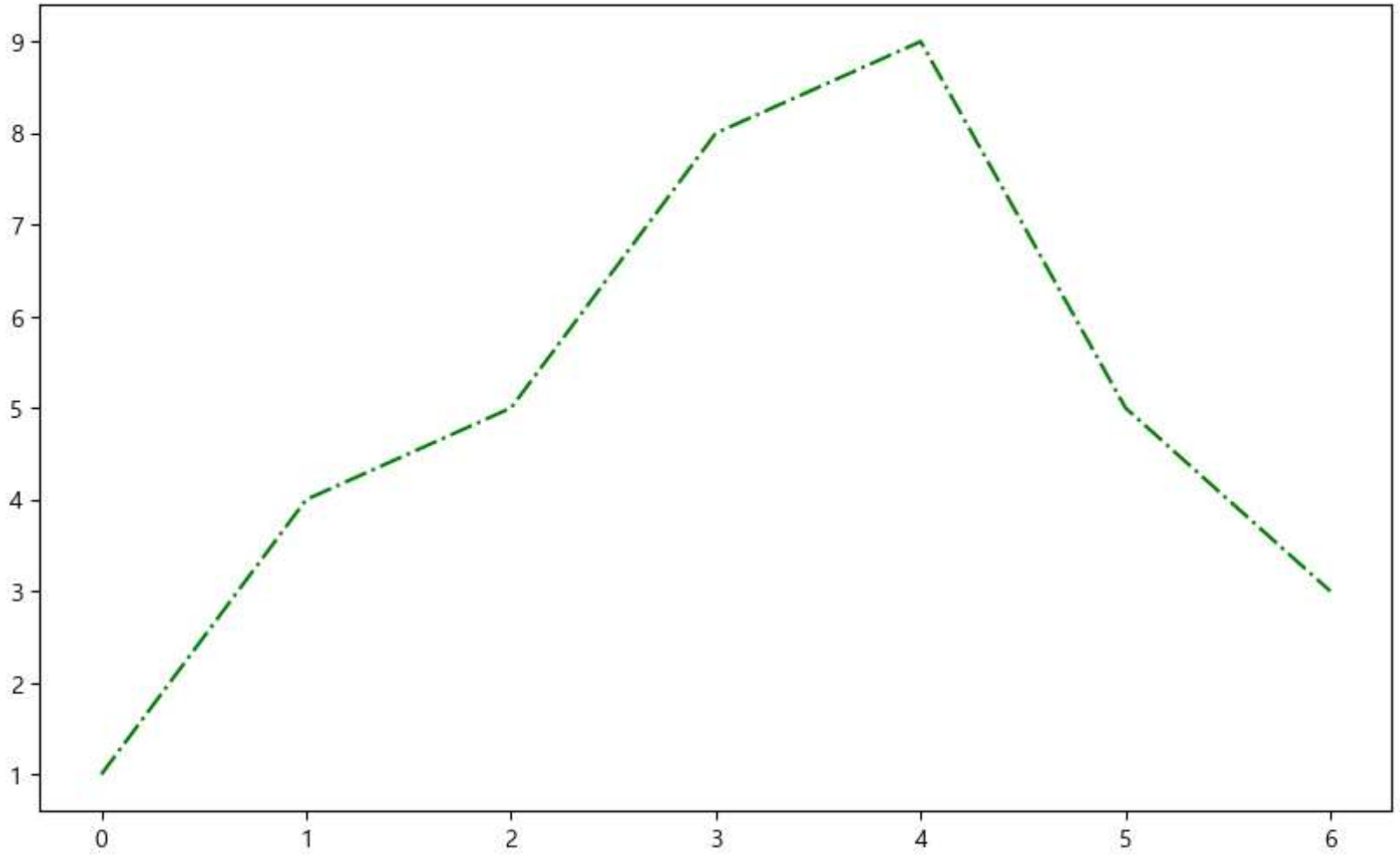
- 라인스타일 기호 지정

실선 (solid)

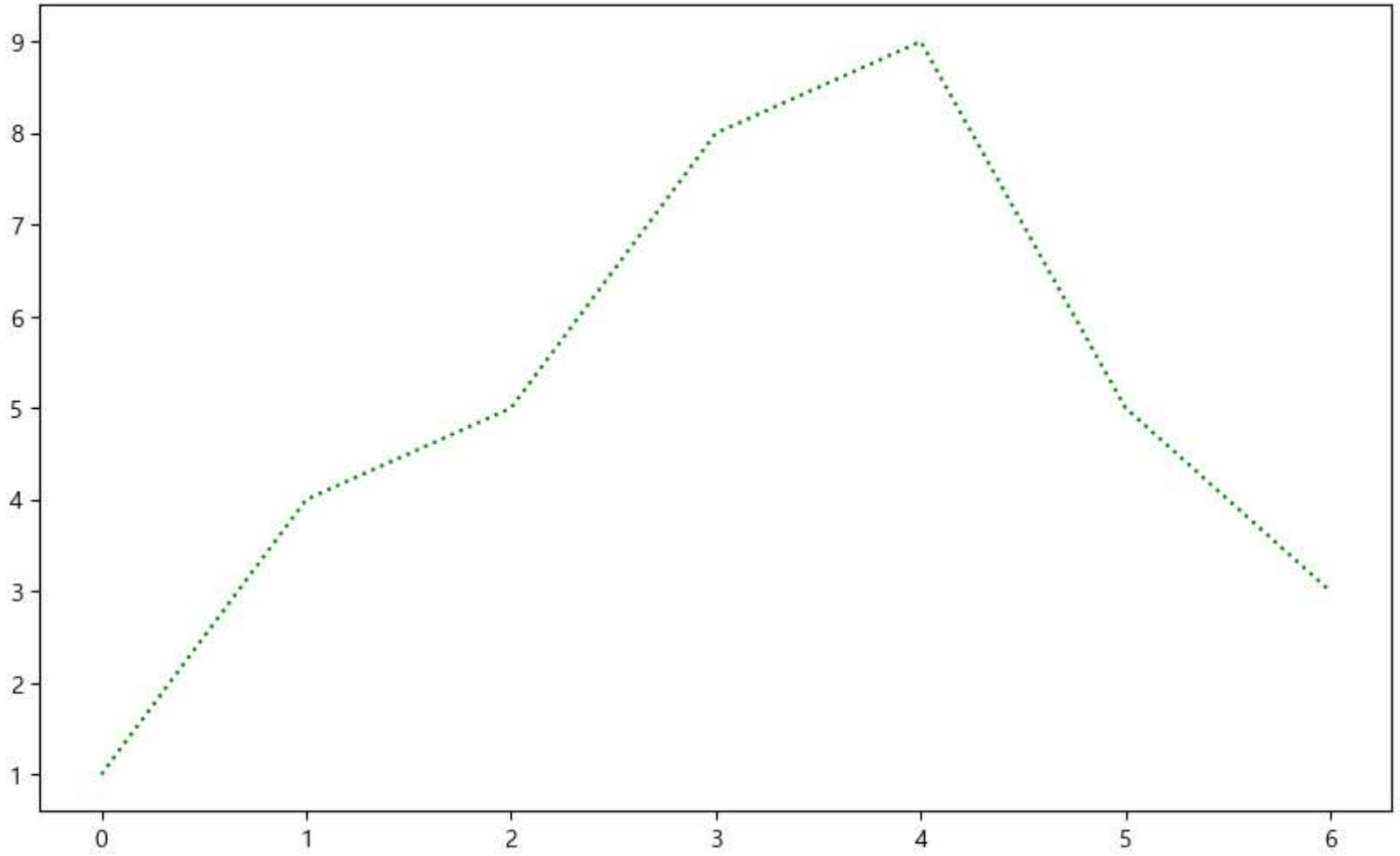
파선 (dashed)

지정자	선 스타일

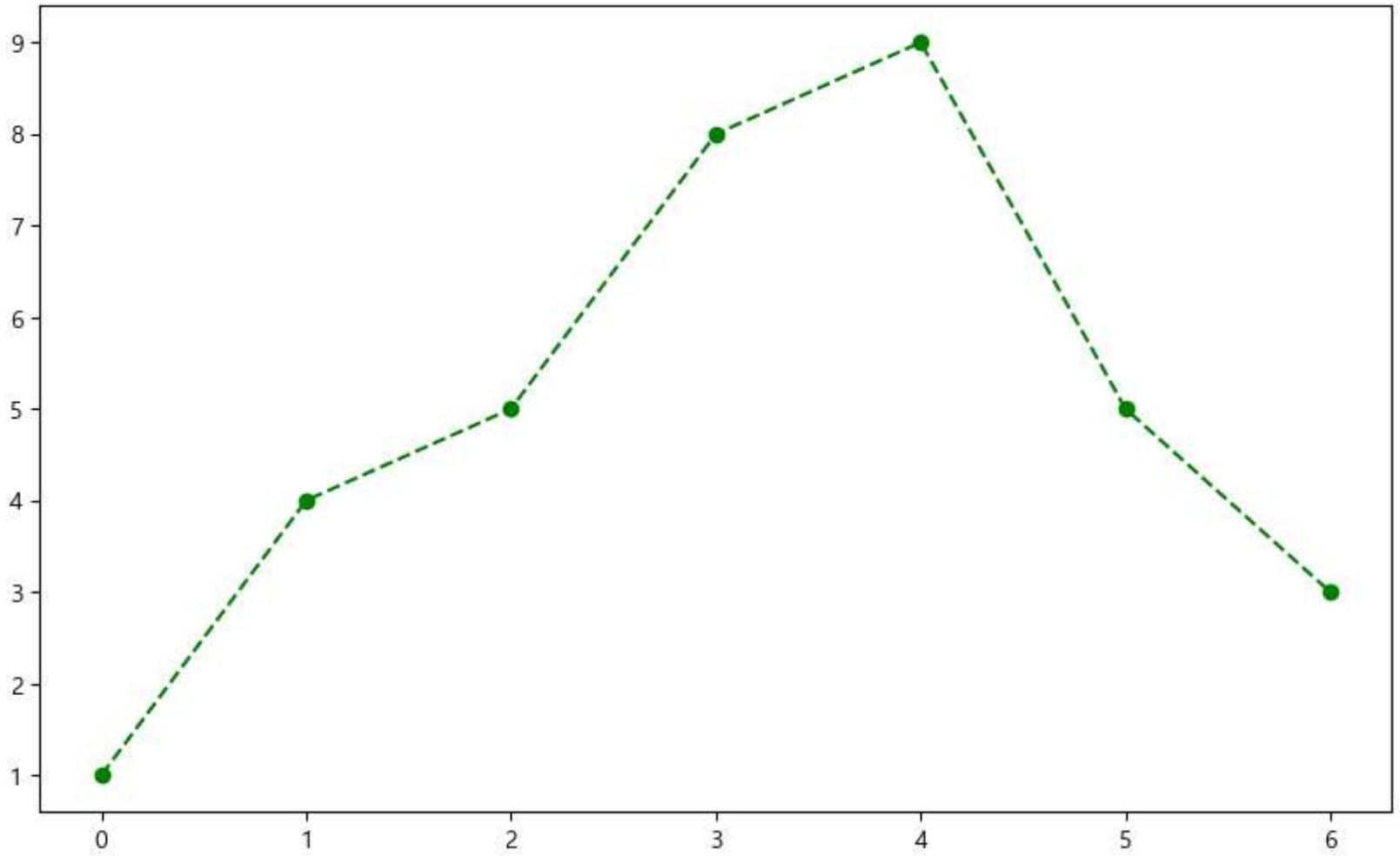
In [7]:
```
# 선 스타일 설정
#색상은 단어로 지정 : color='green'
t=[0,1,2,3,4,5,6]
y=[1,4,5,8,9,5,3]
plt.figure(figsize=(10,6))
plt.plot(t,y,color='green',linestyle='dashdot')
plt.show()
```



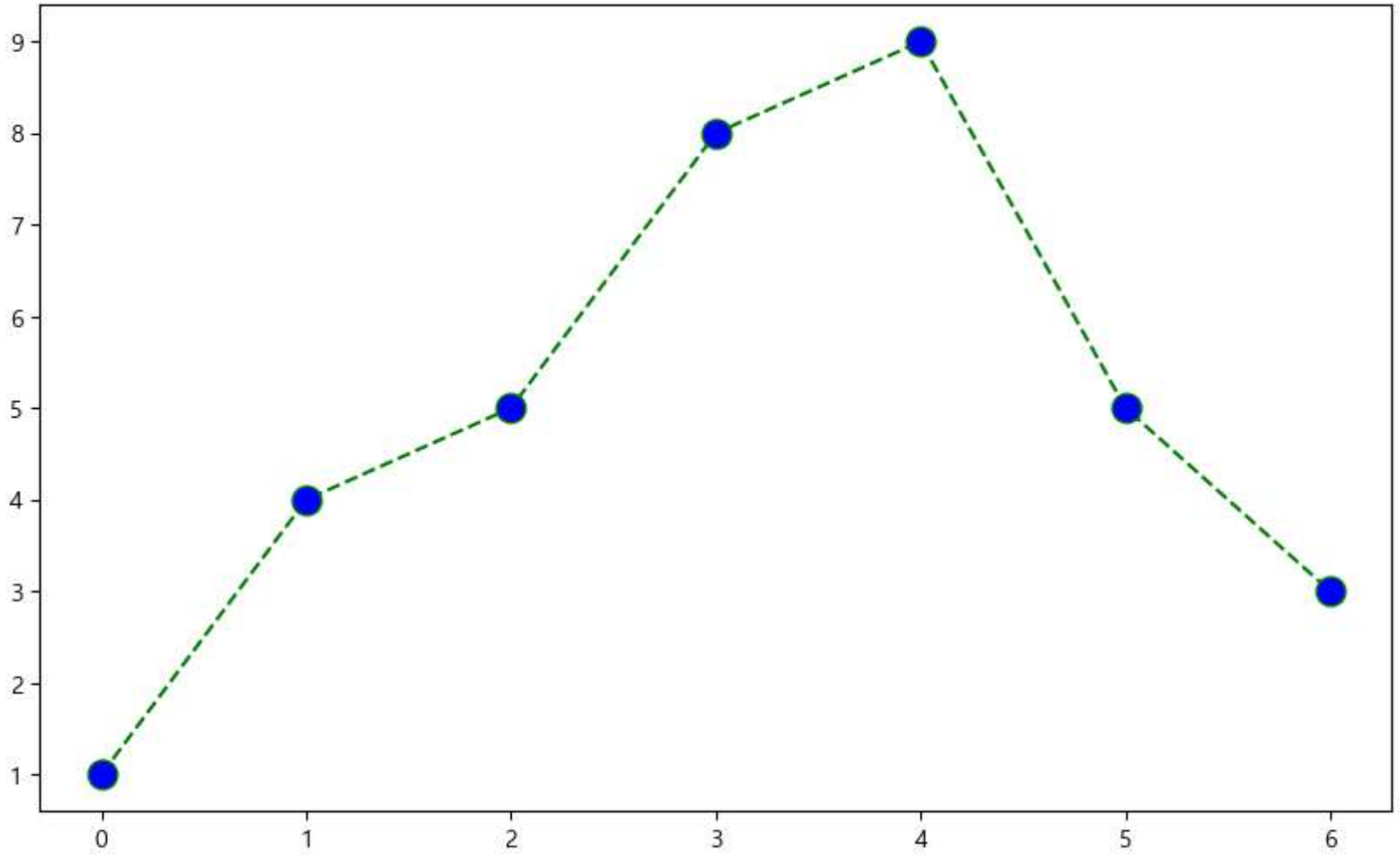
In [8]:
```
# 선 스타일 설정
#색상은 단어로 지정 : color='green'
t=[0,1,2,3,4,5,6]
y=[1,4,5,8,9,5,3]
plt.figure(figsize=(10,6))
plt.plot(t,y,color='green',linestyle='dotted')
plt.show()
```



In [9]:
```
#색상은 단어로 지정 : color='green'
t=[0,1,2,3,4,5,6]
y=[1,4,5,8,9,5,3]
plt.figure(figsize=(10,6))
plt.plot(t,y,color='green',linestyle='dashed',marker='o')
plt.show()
```

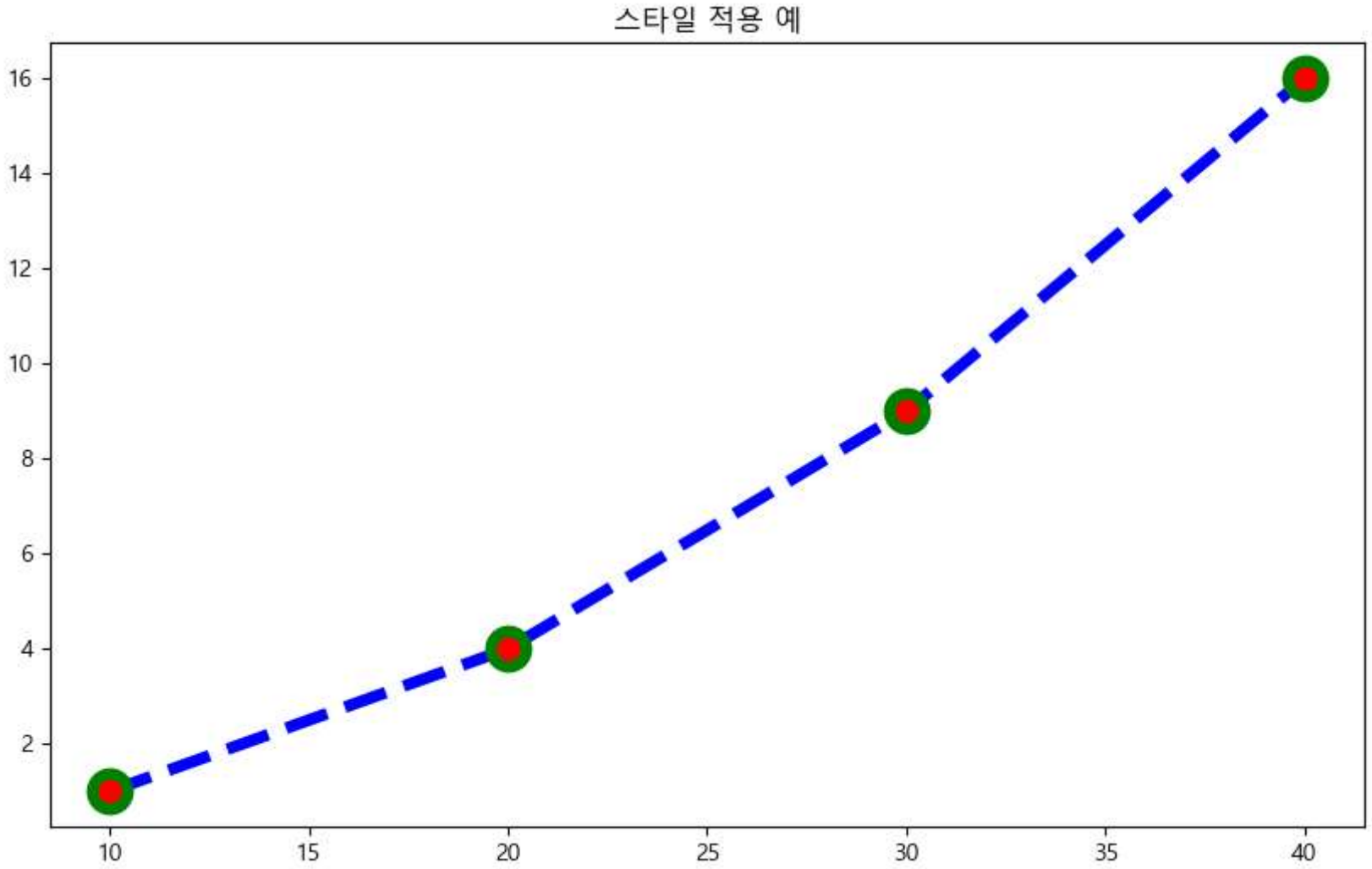


In [10]:
```
#색상은 단어로 지정 : color='green'
#markerfacecolor : 마커 색상, markersize : 마커 크기
t=[0,1,2,3,4,5,6]
y=[1,4,5,8,9,5,3]
plt.figure(figsize=(10,6))
plt.plot(t,y,color='green',linestyle='dashed',marker='o',
        markerfacecolor='blue',markersize=12)
plt.show()
```



In [11]:
```
#스타일을 약자로 표시
plt.figure(figsize=(10,6))
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")
```

Out[11]: Text(0.5, 1.0, '스타일 적용 예')

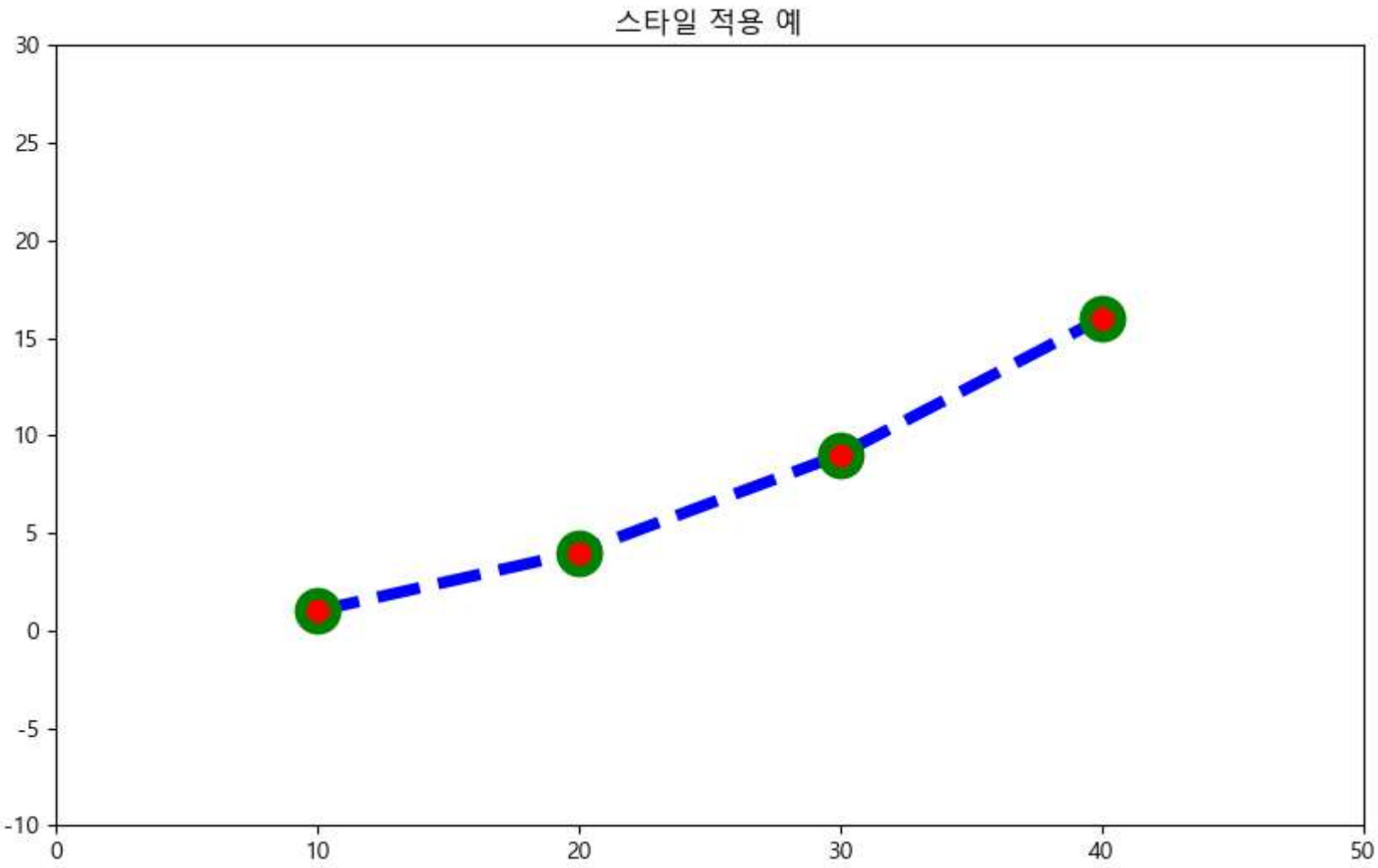


- color(c) : 선색깔

- linewidth(lw) : 선굵기
- linestyle(ls) : 선스타일

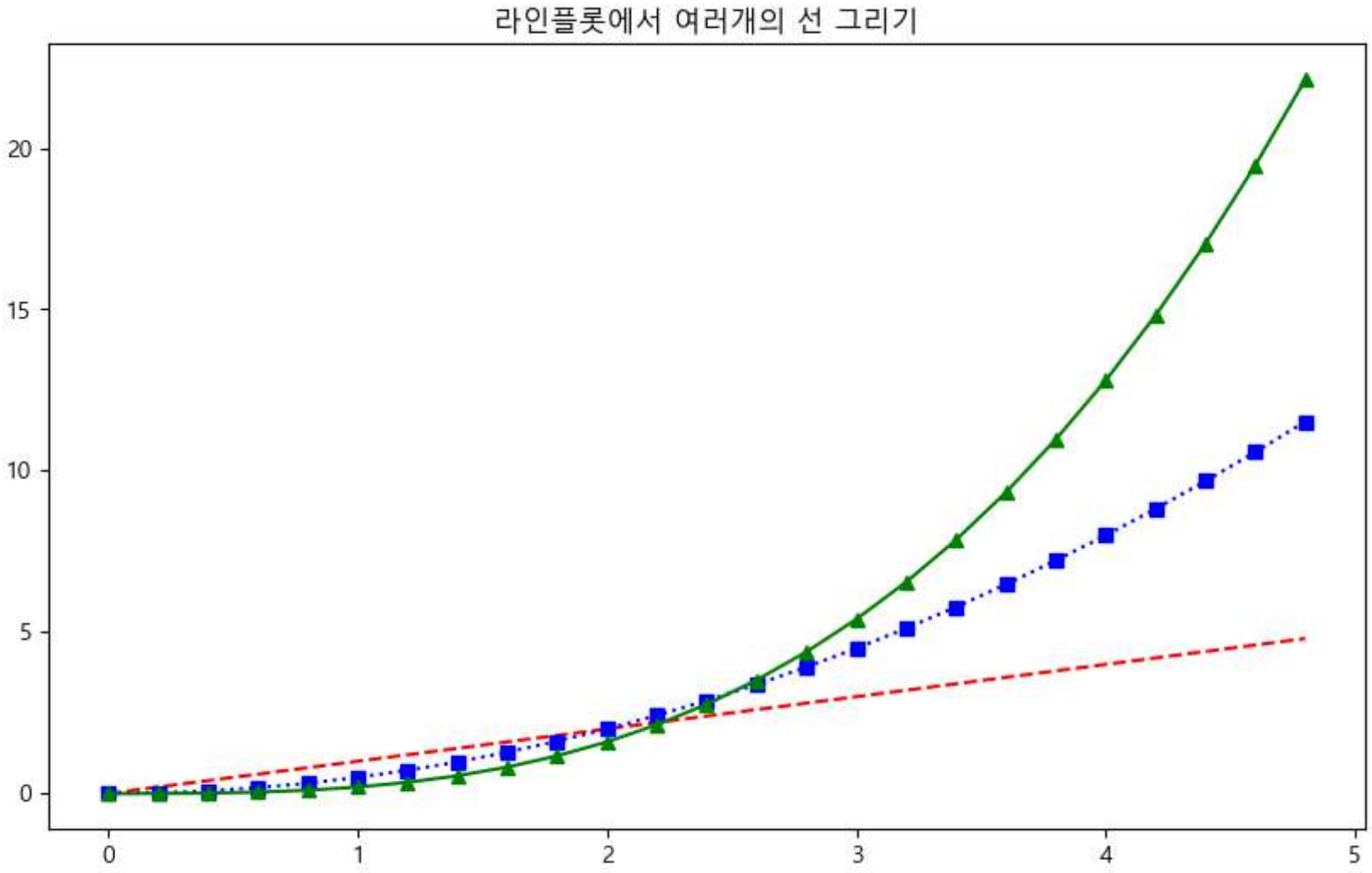
- marker : 마커의 종류
- markersize(ms) : 마커의 크기
- markeredgecolor(mec) : 마커 선 색깔
- markeredgewidth(mew) : 마커 선 굵기
- markerfacecolor(mfc) : 마커 내부 색깔

In [12]:
```
#스타일을 약자로 표시
plt.figure(figsize=(10,6))
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) # x축 범위
plt.ylim(-10,30) # y축 범위
plt.title("스타일 적용 예")
plt.show()
```



- 여러 데이터를 하나의 그래프에 여러 선 으로 표현
  - plot() 여러번 사용 가능

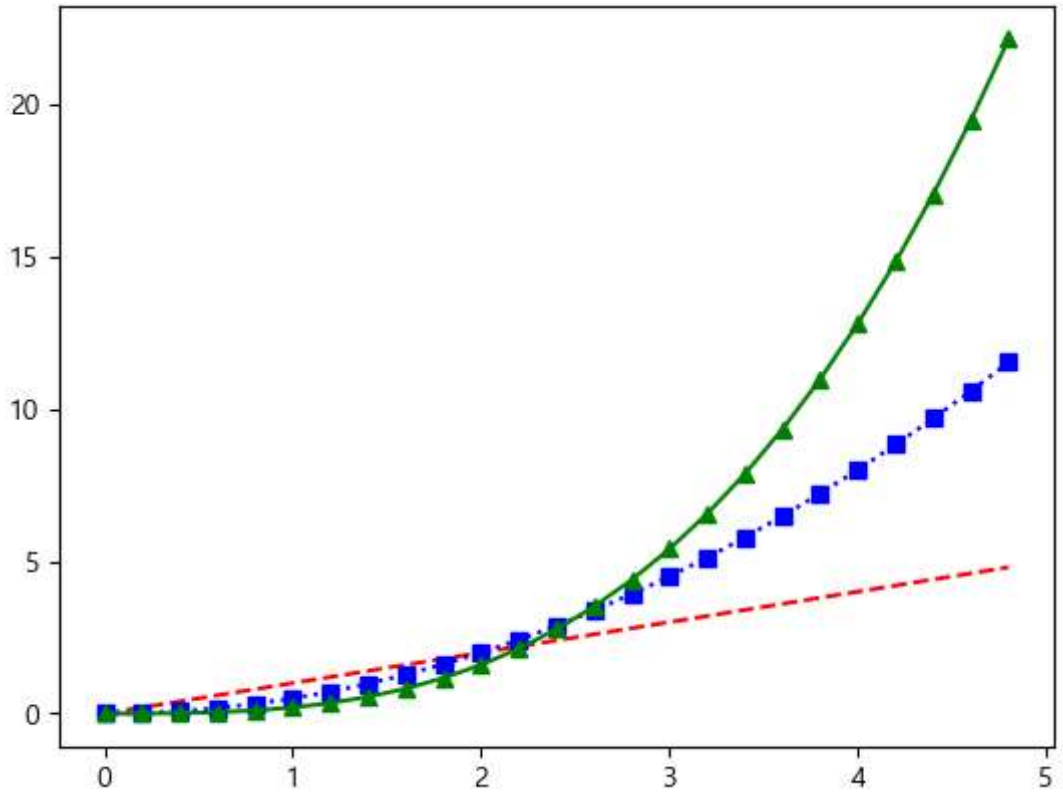
In [13]:
```
#여러개의 선 그리기 : plot() 여러번 사용

t=np.arange(0.,5.,0.2)
t
plt.figure(figsize=(10,6))
plt.title('라인플롯에서 여러개의 선 그리기')
plt.plot(t,t,'r--')#r(red),--(dashed line style)
plt.plot(t,0.5*t**2,'bs:') #b(blue),s(square marker,),:(dot line style)
plt.plot(t,0.2*t**3,'g^-') #g(green),^(triangle_up marker),-(solid lin style)
plt.show()
```



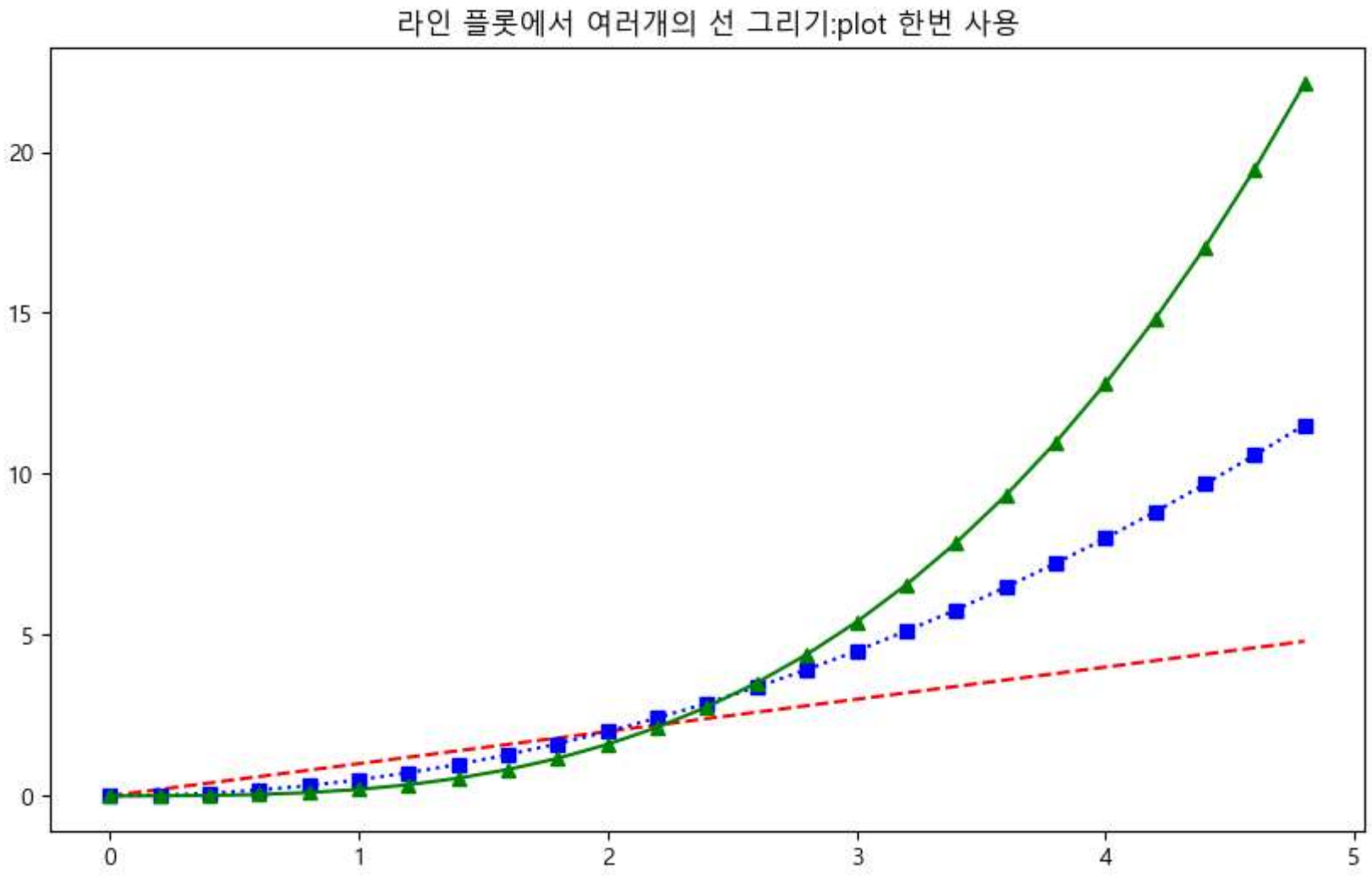
- 위 그래프 코드를 plot() 하나로 한번에 표현하기

In [14]:
```
plt.plot(t,t,'r--',t,0.5*t**2,'bs:',t,0.2*t**3,'g^-')
```

Out[14]:
```
[<matplotlib.lines.Line2D at 0x194938612b0>,
 <matplotlib.lines.Line2D at 0x194938610d0>,
 <matplotlib.lines.Line2D at 0x194938617f0>]
```



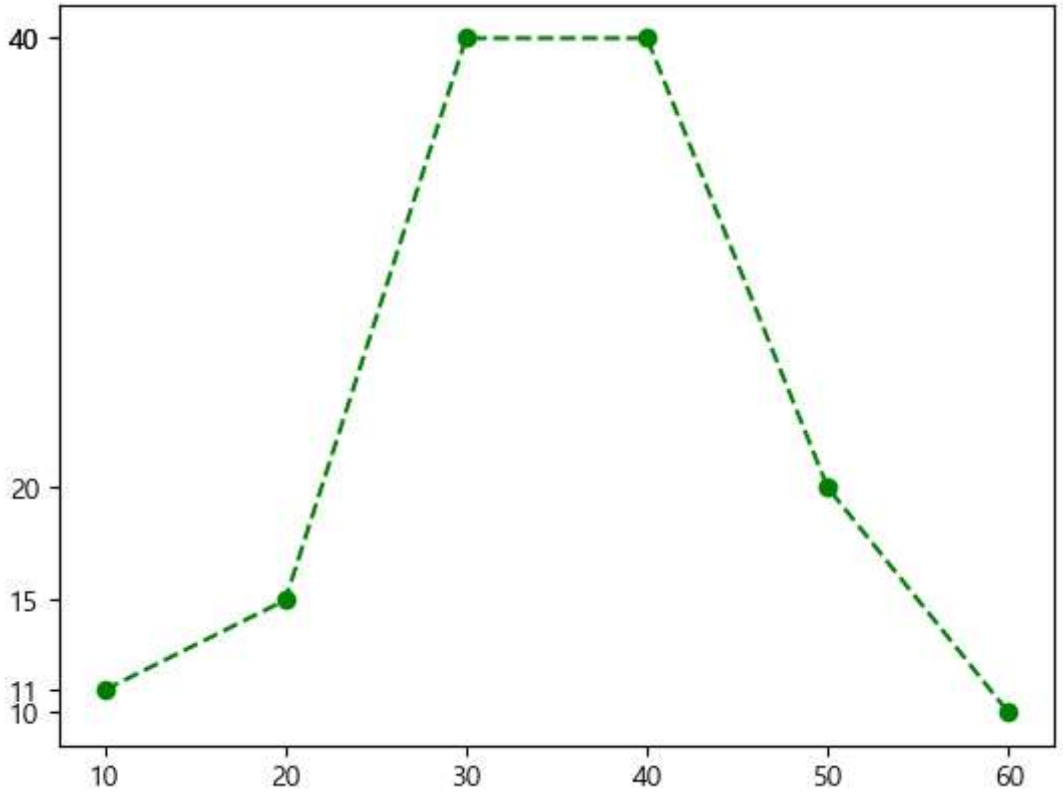
In [15]:
```
#여러개의 선 그리기 : plot() 한번사용 사용
t=np.arange(0.,5.,0.2)
t
plt.figure(figsize=(10,6))
plt.title('라인 플롯에서 여러개의 선 그리기:plot 한번 사용')
plt.plot(t,t,'r--',t,0.5*t**2,'bs:',t,0.2*t**3,'g^-')
plt.show()
```



tick 설정

- tick은 축상의 위치 표시 지점-축에 간격을 구분하기 위해 표시하는 눈금
- xticks([x축값1,x축값2,...]) #튜플,리스트등 이용해서 축 값(위치 나열)
- yticks([y축값1,y축값2,...]) #튜플,리스트등 이용해서 축 값(위치 나열)
- tick_params()
- tick label(눈금 레이블) : tick에 써진 숫자 혹은 글자

In [16]:
```
x=[10,20,30,40,50,60]
y=[11,15,40,40,20,10]
plt.plot(x,y,color='green',linestyle='dashed',marker='o')
plt.xticks(x)
plt.yticks(y)
plt.show()
```

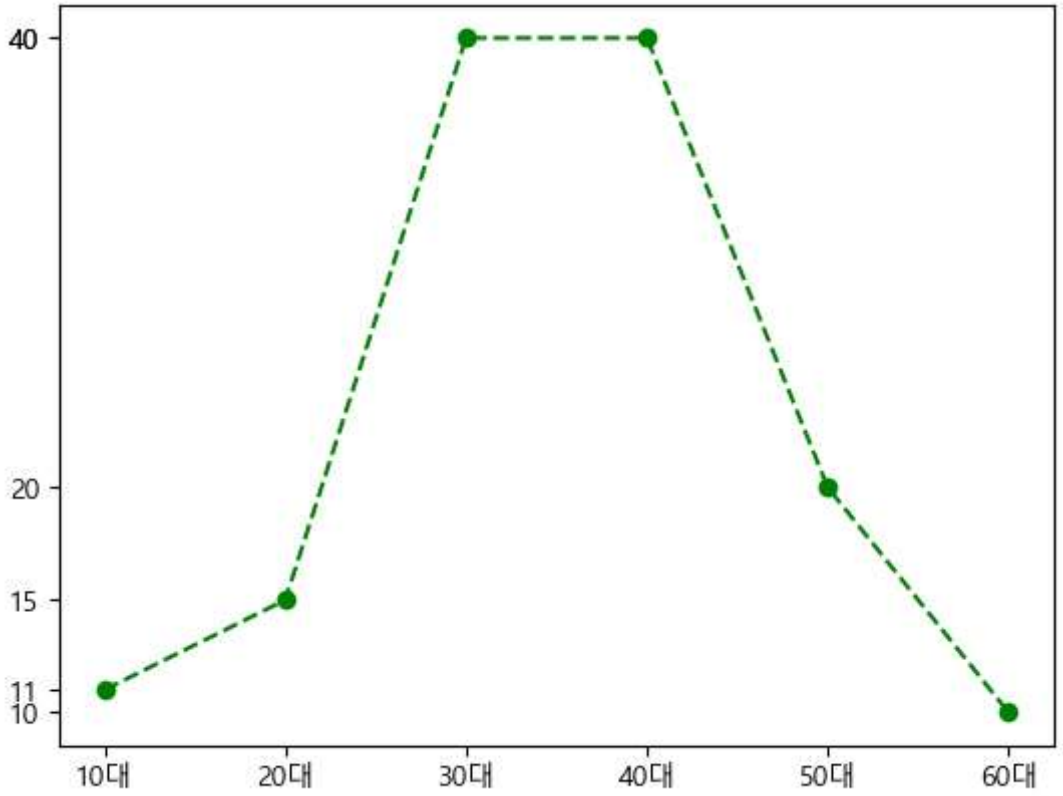


- 눈금 레이블 지정

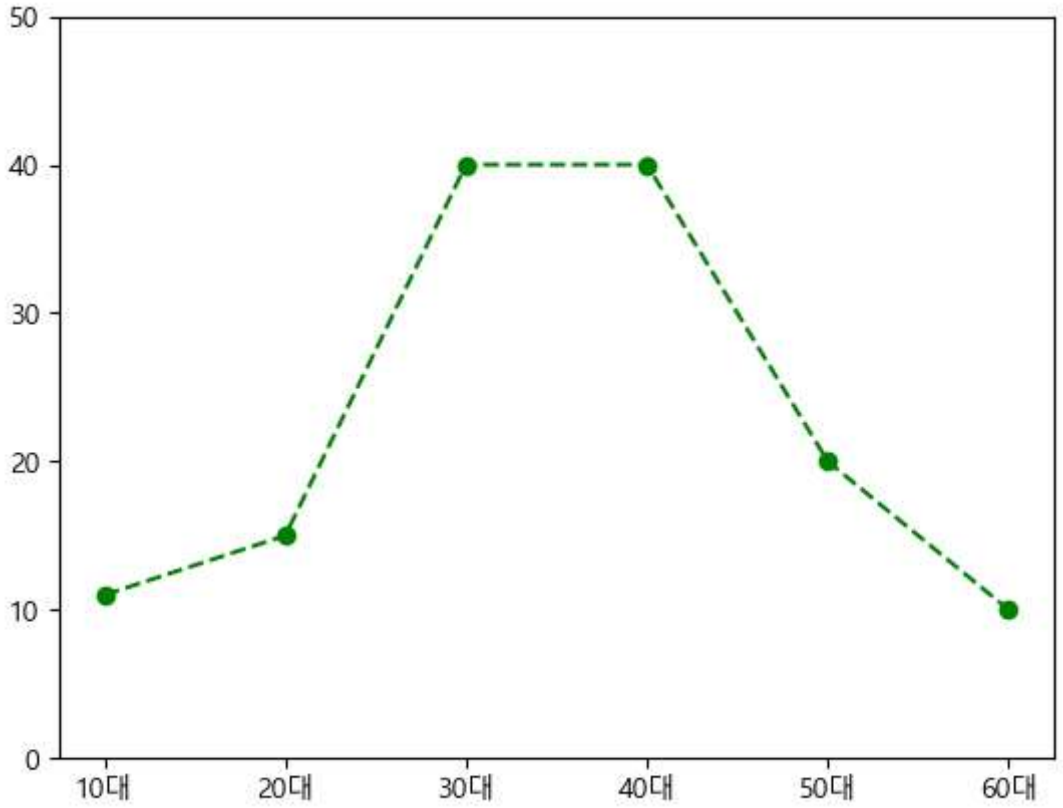
In [17]:
```
[y[i] for i in range(6)]
# y 리스트값 한번씩 추출하는 내포 for문
```

Out[17]:
```
[11, 15, 40, 40, 20, 10]
```

In [18]:
```
# 눈금 레이블 지정

x=[10,20,30,40,50,60]
y=[11,15,40,40,20,10]
plt.plot(x,y,color='green',linestyle='dashed',marker='o')
plt.xticks(x,['10대','20대','30대','40대','50대','60대'])
plt.yticks(y,[y[i] for i in range(6)])
plt.show()
```



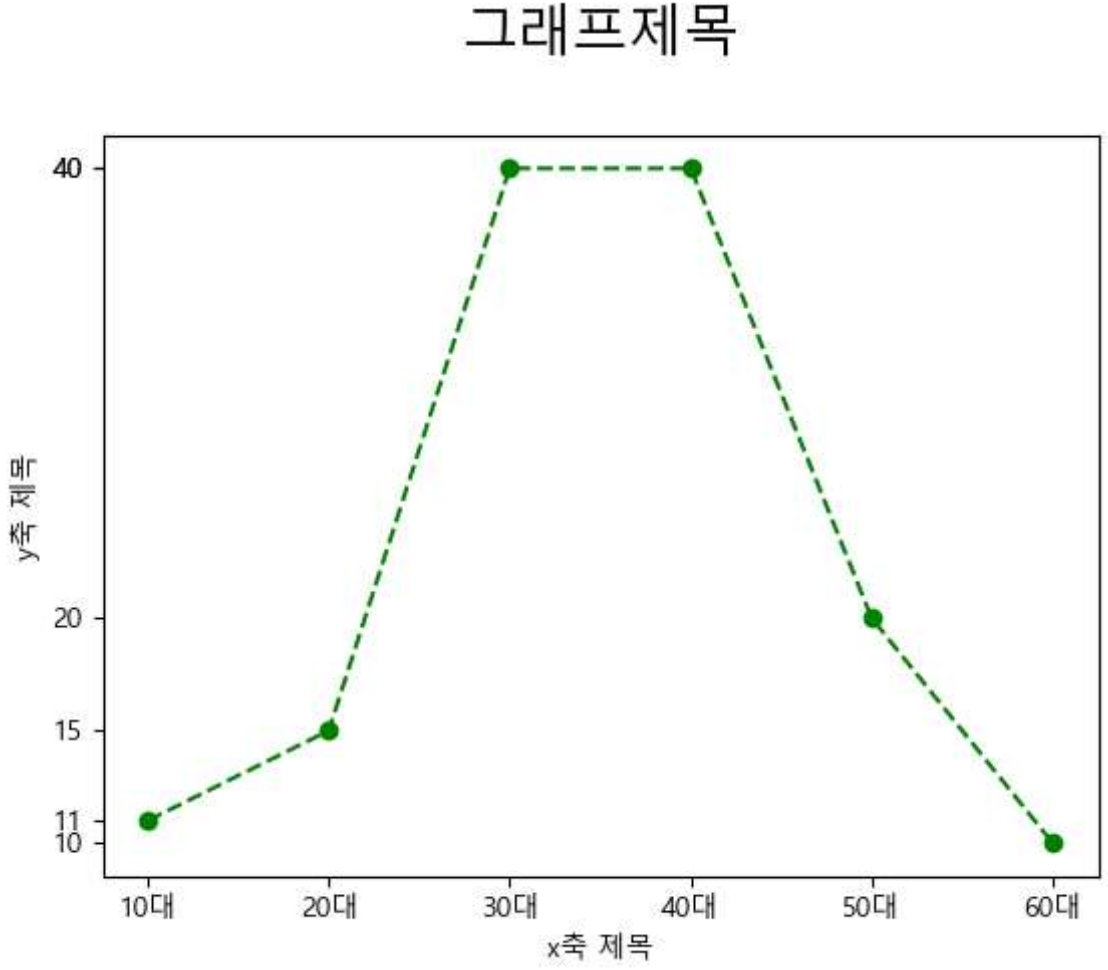
In [19]:
```
# 눈금 레이블 지정

x=[10,20,30,40,50,60]
y=[11,15,40,40,20,10]
plt.plot(x,y,color='green',linestyle='dashed',marker='o')
plt.xticks(x,['10대','20대','30대','40대','50대','60대'])
plt.yticks([0,10,20,30,40,50])
plt.show()
```



- 그래프 제목 및 축 레이블 설정
  - plot.title(data,loc=, pad=, fontsize=)
    - loc= 'right'|'left'| 'center'| 'right'로 설정할 수 있으며 디폴트는 'center'
    - pad=point 은 타이틀과 그래프와의 간격 (오프셋)을 포인트(숫자) 단위로 설정

    - fontsize=제목폰트크기
- plot.xlabel()
- plot.ylabel()

In [20]:
```
# 그래프 제목 x축 y축 라벨

x=[10,20,30,40,50,60]
y=[11,15,40,40,20,10]
# plt.title('그래프제목')
# plt.title('그래프제목', loc='left')
# plt.title('그래프제목', loc='right', pad=30)
plt.title('그래프제목', pad=30, fontsize=20)
plt.plot(x,y,color='green', linestyle='dashed',marker='o')
plt.xticks(x,('10대','20대','30대','40대','50대','60대'))
plt.yticks(y,(y[i] for i in range(6)))
plt.xlabel('x축 제목')
plt.ylabel('y축 제목')

plt.show()
```

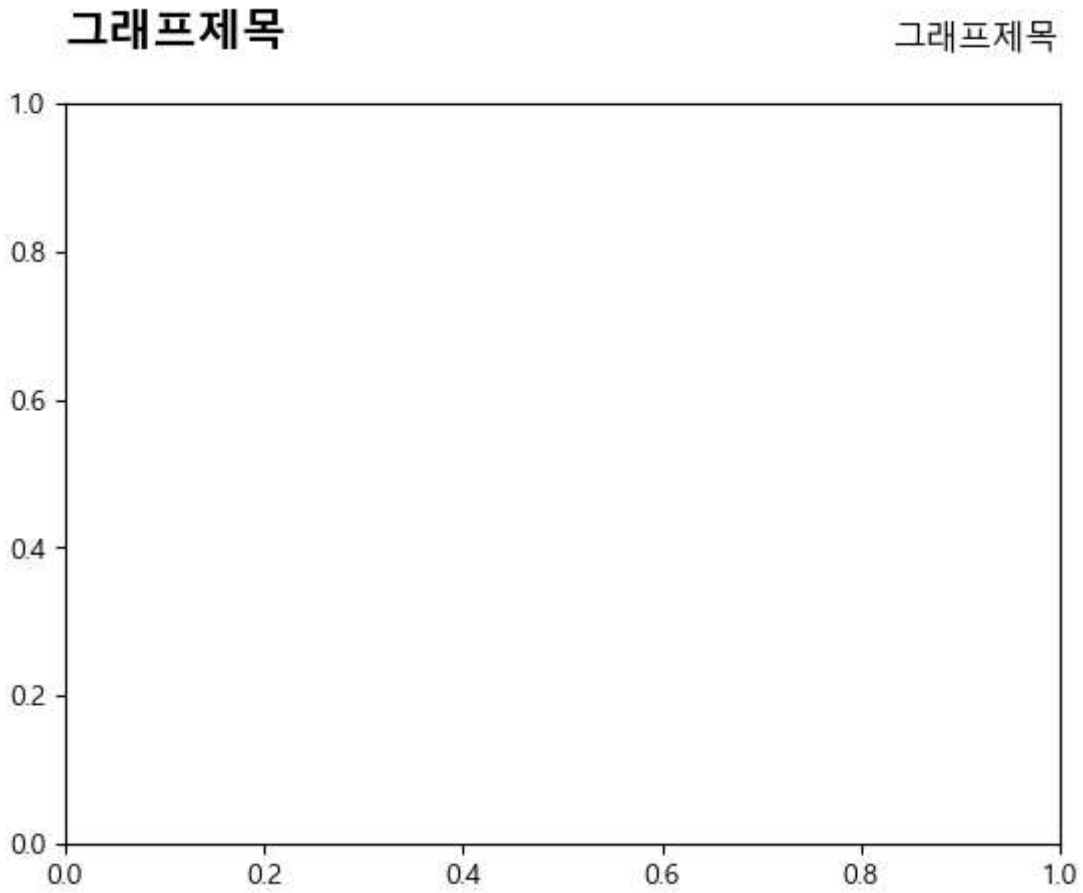


그래프 Title 폰트 관련 지정

- 딕셔너리형식으로 fontsize 및 fontwegith 등 지정 가능

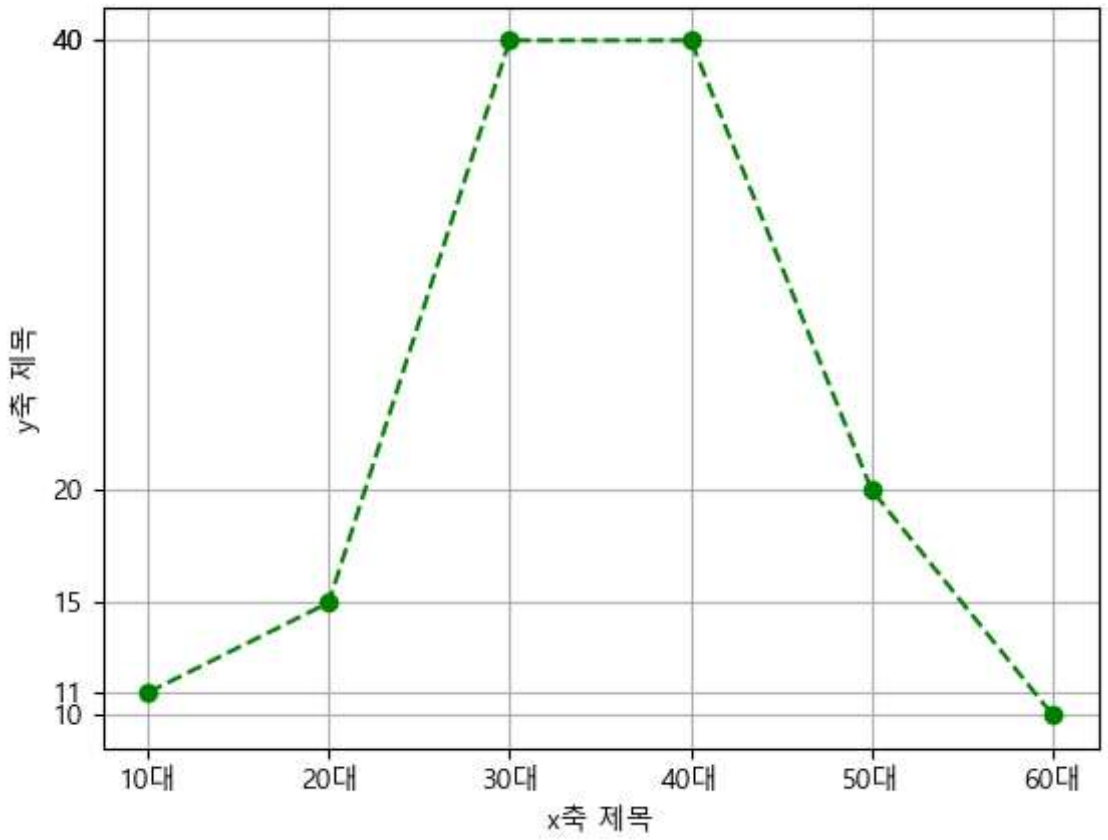
In [21]:
```
plt.title('그래프제목',loc='right', pad=20)

title_font = {
    'fontsize':16,
    'fontweight' : 'bold'
}
# fontdict
plt.title('그래프제목',fontdict=title_font, loc='left', pad=20)
plt.show
```

Out[21]:
```
<function matplotlib.pyplot.show(close=None, block=None)>
```



- plot.grid(True) : 배경 그리드 표시

In [22]:
```
# 배경 그리드 표현

x=[10,20,30,40,50,60]
y=[11,15,40,40,20,10]

plt.title('그래프제목', pad=30, fontsize=20)
plt.plot(x,y,color='green', linestyle='dashed',marker='o')
plt.xticks(x,('10대','20대','30대','40대','50대','60대'))
plt.yticks(y,(y[i] for i in range(6)))
plt.xlabel('x축 제목')
plt.ylabel('y축 제목')
plt.grid(True) # ticks 선을 연장해 grid를 표현
plt.show()
```



subplot() : 하나의 윈도우(figure)안에여러개의 플롯을 배열 형태로 표시

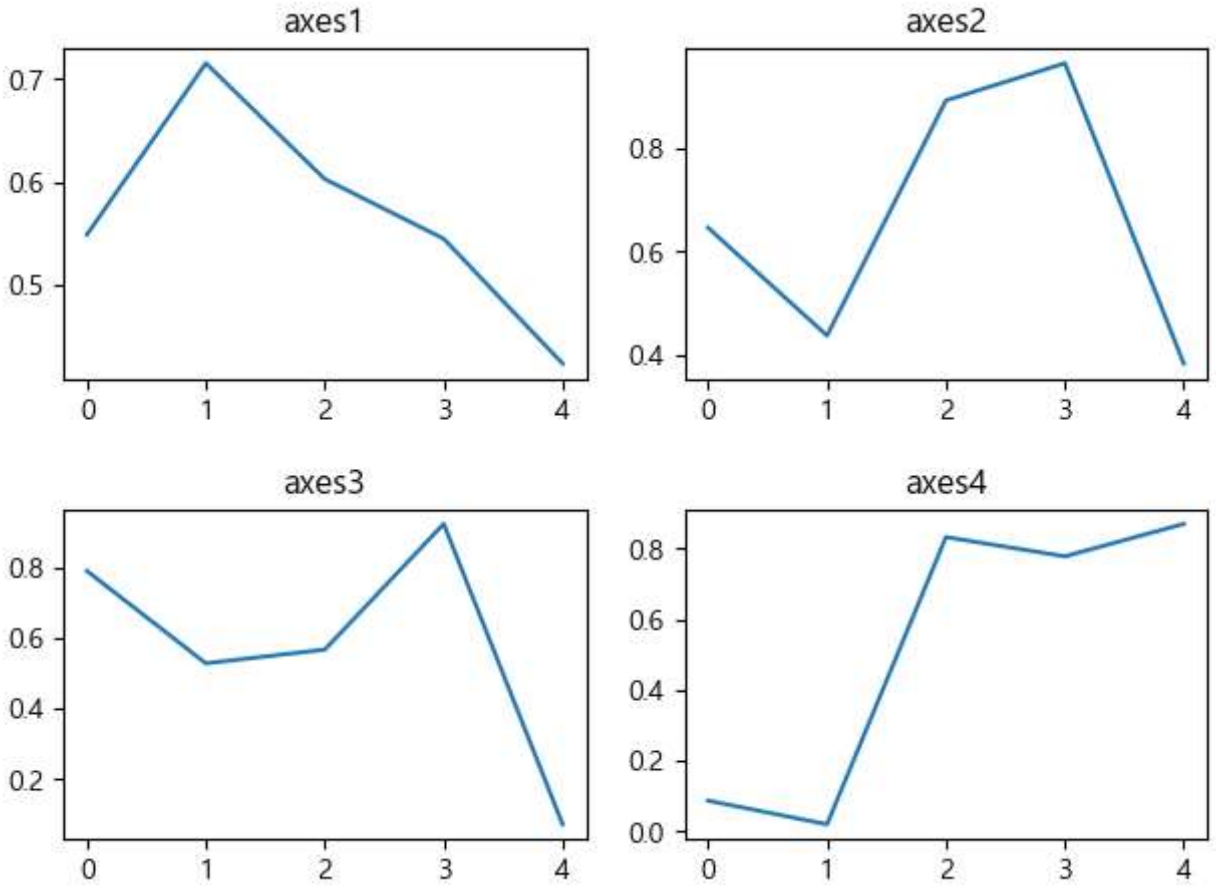
- 그리드 형태의 Axes 객체들 생성

- 형식 : subplot(인수1,인수2,인수3)
- 인수1 과 인수2는 전체 그리드 행렬 모양 지시
- 인수3 : 그래프 위치 번호

  - subplot(2,2,1) 가 원칙이나 줄여서 221로 쓸 수 있음
  - subplot(221) 2행 2열의 그리드에서 첫번째 위치
  - subplot(224) 2행 2열의 그리드에서 마지막 위치

- tight_layout(pad=) : 플롯간 간격을 설정

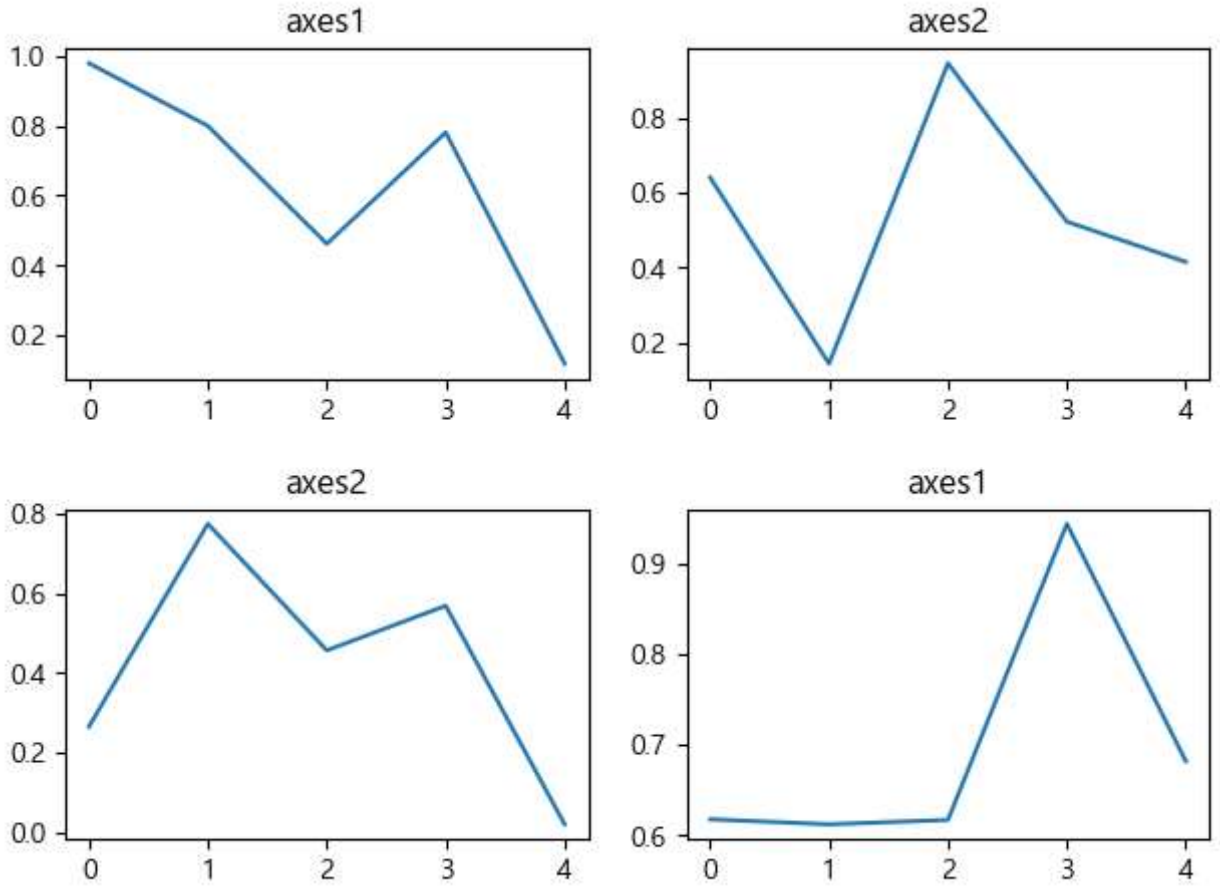
In [23]:
```
# 2 x 2 형태의 네개의 플롯

np.random.seed(0) # 항상 같은 난수가 발생

plt.subplot(221) # show() 함수 전에 생성되어야 함(위치 설정)
plt.plot(np.random.rand(5))
plt.title('axes1')

plt.subplot(222) # show() 함수 전에 생성되어야 함(위치 설정)
plt.plot(np.random.rand(5))
plt.title('axes2')

plt.subplot(223) # show() 함수 전에 생성되어야 함(위치 설정)
plt.plot(np.random.rand(5))
plt.title('axes3')

plt.subplot(224) # show() 함수 전에 생성되어야 함(위치 설정)
plt.plot(np.random.rand(5))
plt.title('axes4')

plt.tight_layout(pad=1.5)
```



- plt.subplots(행,열)
  - 여러개의 Axes 객체를 동시에 생성해주는 함수
  - 행렬 형태의 객체로 반환

- 두개의 반환값이 있음 :
  - 첫번 째 반환은 그래프 객체 전체 이름 - 거의 사용하지 않음
  - 두번 째 반환값에 Axes 객체를 반환 함
  - 두번 째 반환값이 필요하므로 반환 값 모두를 반환받아 두번 째 값을 사용해야 함
  - ex. fig, axes = plt.subplots(2,2)

In [24]:
```
#subplots() : 여러개의 Axes 객체 동시 생성해주는 함수
fig, ax = plt.subplots(2,2)

ax[0,0].plot(np.random.rand(5))
ax[0,0].set_title('axes1')

ax[0,1].plot(np.random.rand(5))
ax[0,1].set_title('axes2')

ax[1,0].plot(np.random.rand(5))
ax[1,0].set_title('axes2')

ax[1,1].plot(np.random.rand(5))
ax[1,1].set_title('axes1')

plt.tight_layout(pad=1.5)
plt.show()
```



plot 함수 응용 예제

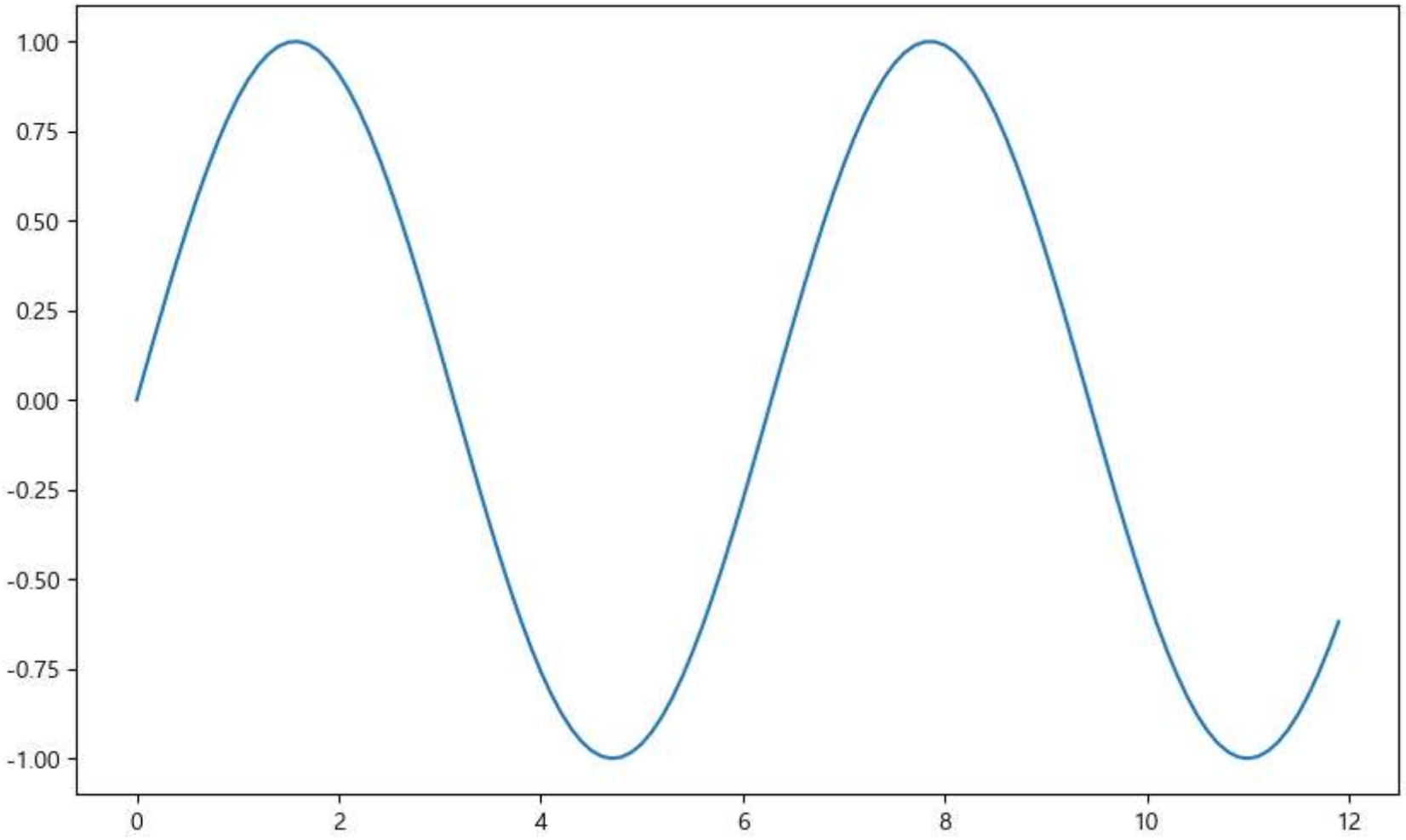
- numpy 모듈의 sin()함수를 이용하여 sin 곡선 그래프 그리기

In [25]:
```
# data 생성

t=np.arange(0,12,0.1)
#arange(시작, 끝값, 간격)
#시작값부터 끝값 - 간격까지 순서대로 수를 생성
t
```

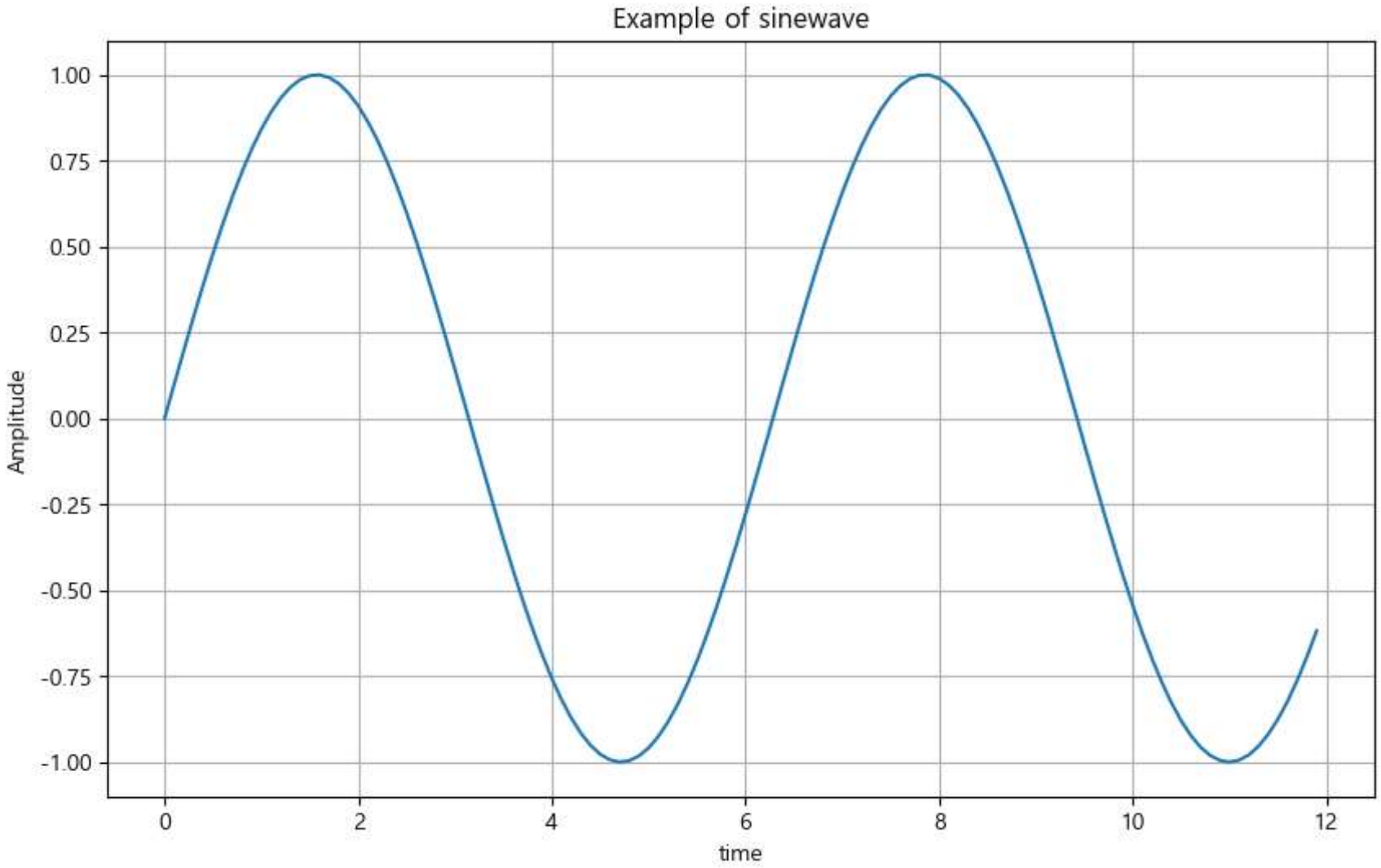
Out[25]:
```
array([ 0. ,  0.1,  0.2,  0.3,  0.4,  0.5,  0.6,  0.7,  0.8,  0.9,  1. ,
        1.1,  1.2,  1.3,  1.4,  1.5,  1.6,  1.7,  1.8,  1.9,  2. ,  2.1,
        2.2,  2.3,  2.4,  2.5,  2.6,  2.7,  2.8,  2.9,  3. ,  3.1,  3.2,
        3.3,  3.4,  3.5,  3.6,  3.7,  3.8,  3.9,  4. ,  4.1,  4.2,  4.3,
        4.4,  4.5,  4.6,  4.7,  4.8,  4.9,  5. ,  5.1,  5.2,  5.3,  5.4,
        5.5,  5.6,  5.7,  5.8,  5.9,  6. ,  6.1,  6.2,  6.3,  6.4,  6.5,
        6.6,  6.7,  6.8,  6.9,  7. ,  7.1,  7.2,  7.3,  7.4,  7.5,  7.6,
        7.7,  7.8,  7.9,  8. ,  8.1,  8.2,  8.3,  8.4,  8.5,  8.6,  8.7,
        8.8,  8.9,  9. ,  9.1,  9.2,  9.3,  9.4,  9.5,  9.6,  9.7,  9.8,
        9.9, 10. , 10.1, 10.2, 10.3, 10.4, 10.5, 10.6, 10.7, 10.8, 10.9,
       11. , 11.1, 11.2, 11.3, 11.4, 11.5, 11.6, 11.7, 11.8, 11.9])
```

In [26]:
```
# sin 값 생성
y=np.sin(t)
```

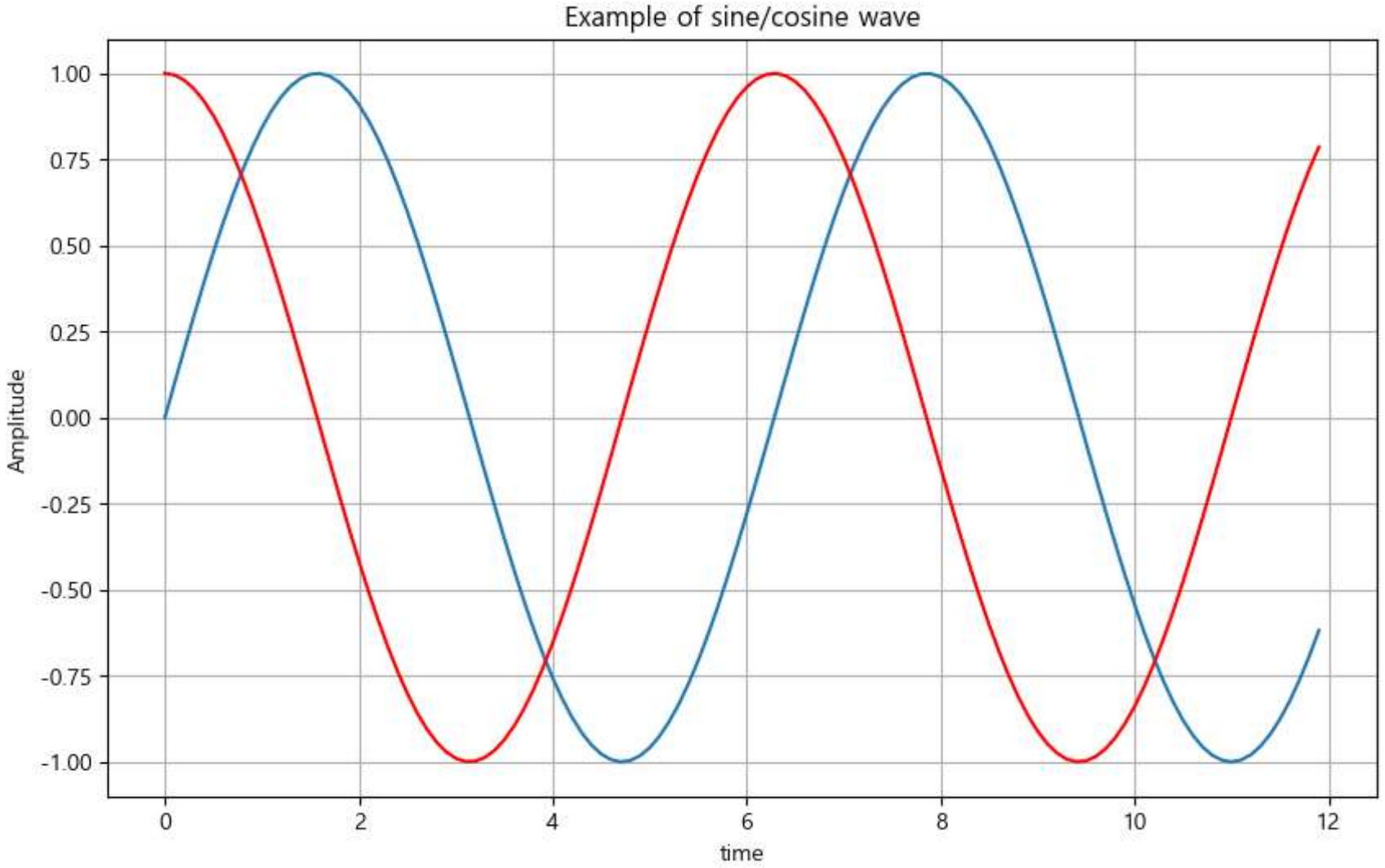
In [27]:
```
plt.figure(figsize=(10,6))
plt.plot(t,y)
plt.show()
```



In [28]:
```
#위 그래프에 x,y 축 제목, 그래프 제목, 격자무늬를 표시하시오.
#x축 : time
#y축 : Amplitude
#그래프 제목 : Example of sinewave

plt.figure(figsize=(10,6))
plt.title('Example of sinewave')
plt.plot(t,y)
plt.xlabel('time')
plt.ylabel('Amplitude')
plt.grid(True)
plt.show()
```


Example of sinewave
1.00
0.75
0.50
0.25
0.00
-0.25
-0.50
-0.75
-1.00
Amplitude
0
2
4
6
8
10
12
time

In [29]:
```
#np.sin() : sin 값을 계산후 반환
#np.cos() : cos 값을 계산 후 반환

#위쪽 그래프에 cos 곡선을 추가하는 코드를 작성하시오.
#cos 곡선의 색상은 빨간색으로 설정 할 것

plt.figure(figsize=(10,6))
plt.title('Example of sine/cosine wave')
plt.plot(t,y)
plt.plot(t,np.cos(t), c='red')
plt.xlabel('time')
plt.ylabel('Amplitude')
plt.grid()
plt.show()
```



범례(legend)표시

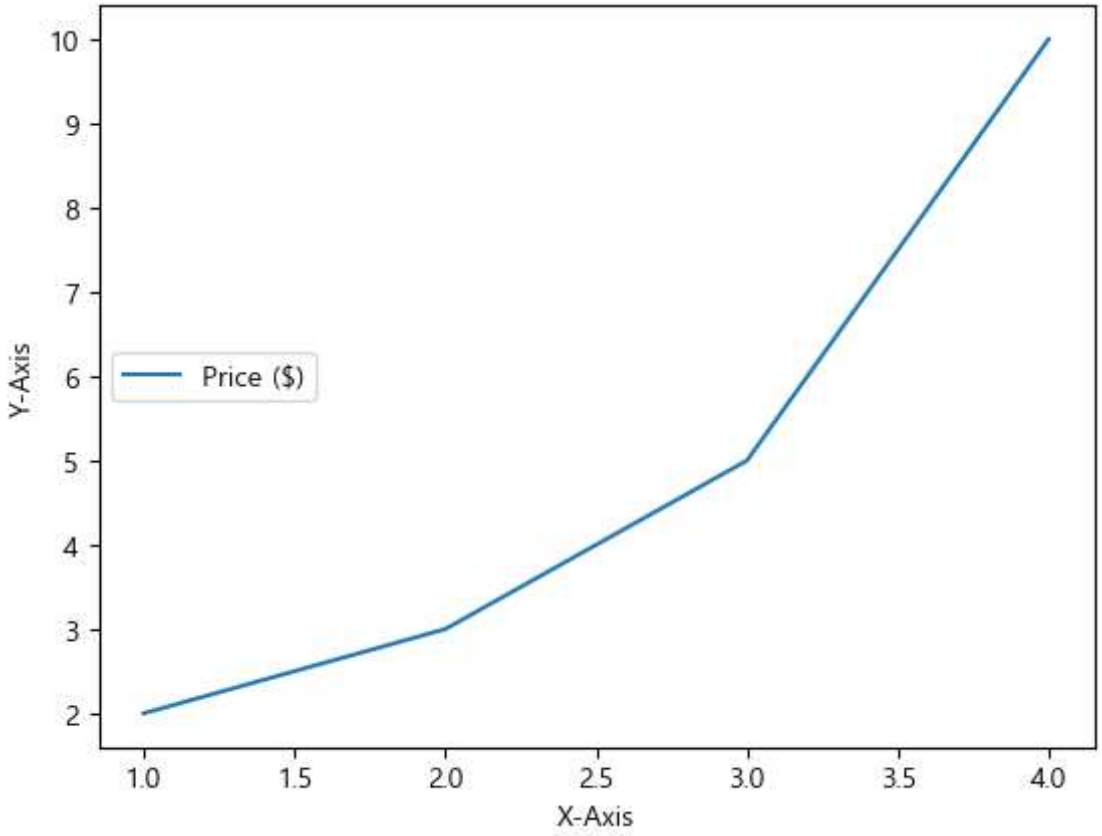
- plot에 label 속성이 추가되어 있어야 함
  - plt.plot(x,y,label='a')
- plt.legend(loc=, ncol=) #범례표시,
- loc = 1/2/3/4/5/6/7/8/9/10 # 범례표시 위치값

- loc = (x,y)
- ncol= 열갯수

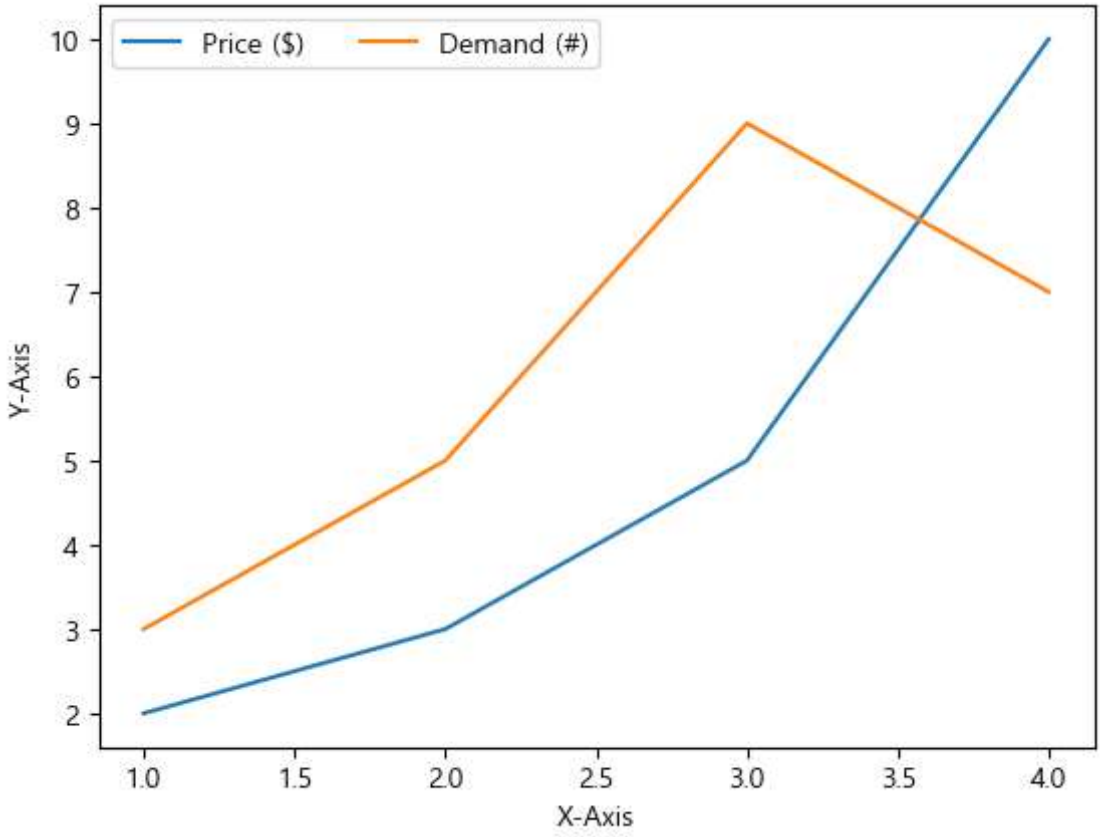
Location String	Location Code	설명
'best'	0	그래프의 최적의 위치에 표시합니다. (디폴트)
'upper right'	1	그래프의 오른쪽 위에 표시합니다.
'upper left'	2	그래프의 왼쪽 위에 표시합니다.
'lower left'	3	그래프의 왼쪽 아래에 표시합니다.
'lower right'	4	그래프의 오른쪽 아래에 표시합니다.
'right'	5	그래프의 오른쪽에 표시합니다.
'center left'	6	그래프의 왼쪽 가운데에 표시합니다.
'center right'	7	그래프의 오른쪽 가운데에 표시합니다.
'lower center'	8	그래프의 가운데 아래에 표시합니다.
'upper center'	9	그래프의 가운데 위에 표시합니다.
'center'	10	그래프의 가운데에 표시합니다.

In [30]:
```
import matplotlib.pyplot as plt

plt.plot([1, 2, 3, 4], [2, 3, 5, 10], label='Price ($)')
plt.xlabel('X-Axis')
plt.ylabel('Y-Axis')

plt.legend(loc=(0.0,0.0))
plt.legend(loc=(0.5,0.5))
plt.legend(loc=(1.0,1.0))
plt.legend(loc=0)
plt.legend(loc=10)
plt.legend(loc='center left')
```

Out[30]: <matplotlib.legend.Legend at 0x194940a9550>



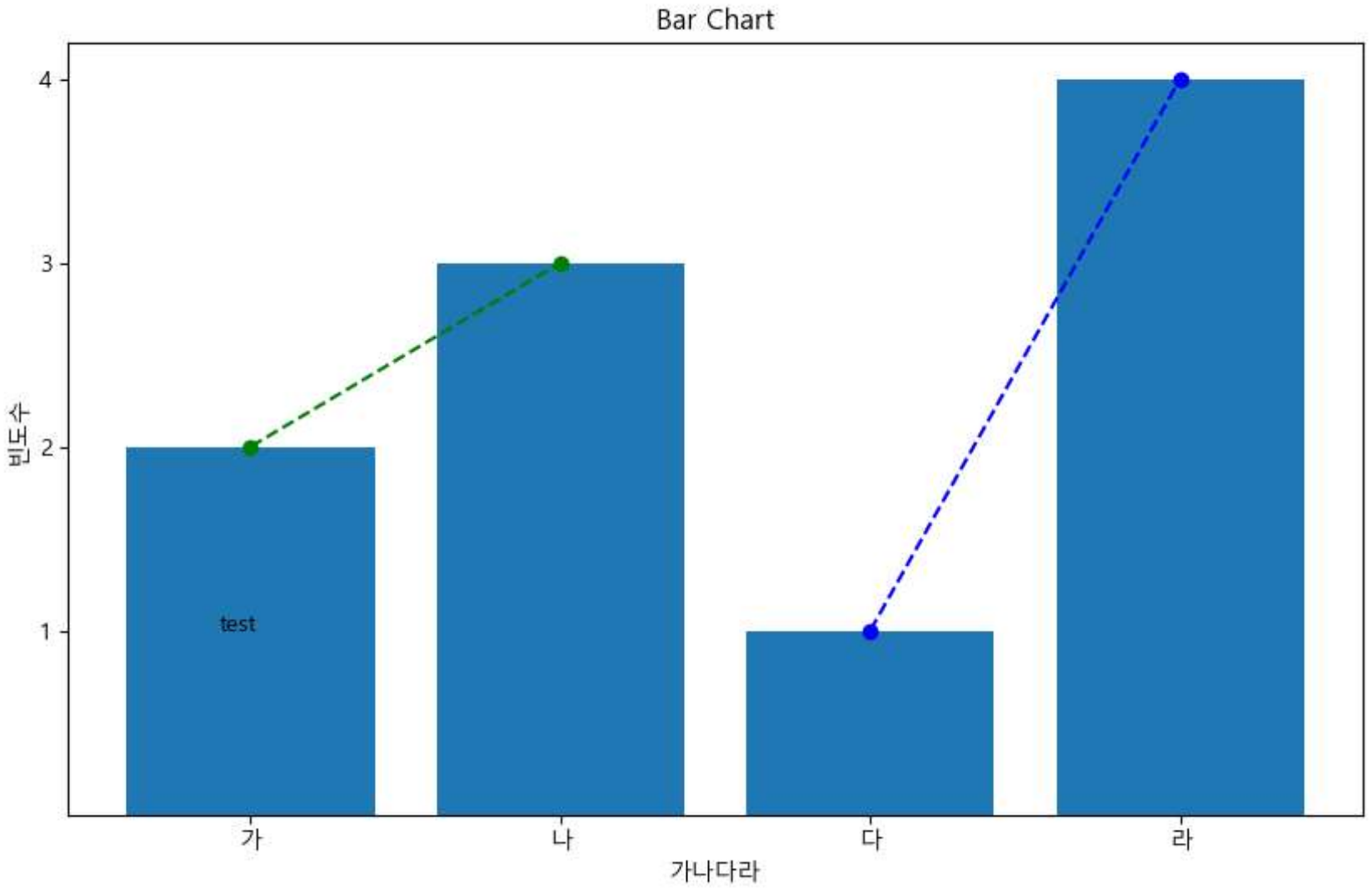
In [31]:
```
import matplotlib.pyplot as plt

plt.plot([1, 2, 3, 4], [2, 3, 5, 10], label='Price ($)')
plt.plot([1, 2, 3, 4], [3, 5, 9, 7], label='Demand (#)')
plt.xlabel('X-Axis')
plt.ylabel('Y-Axis')
plt.legend(loc='best') # 열 1개
plt.legend(loc='best', ncol=2) # 열 2개

plt.show()
```



- 세로 막대 그래프 그리기: bar()
  - bar(x,y,color=[],alpha=)
  - color = [] : 색상값 설정

- alpha = 투명도 설정

In [32]:
```
y=[2,3,1,4]
x=np.arange(len(y))
z=[2,3]
s=[0,1]

e=[1,4]
h=[2,3]

xlabel=['가','나','다','라']
plt.figure(figsize=(10,6))
plt.title('Bar Chart')
plt.bar(x,y)
plt.plot(s,z, color='green',linestyle='dashed',marker='o')
plt.plot(h,e, color='blue',linestyle='dashed',marker='o')
plt.xticks(x,xlabel)
plt.yticks(sorted(y))
plt.text(-0.1, 1, r'test')
plt.xlabel('가나다라')
plt.ylabel('빈도수')
plt.show()
x
```



Out[32]: array([0, 1, 2, 3])

In [33]:
```
np.random.seed(0)
people=['몽룡','춘향','방자','향단']
y=np.arange(len(people))
#np.arange(4) : 0-4사이의 정수를 순서적으로 추출
#0,1,2,3
#np.random.rand(4) : 0-1사이의 난수를 4개 추출
#array([0.5488135 , 0.71518937, 0.60276338, 0.54488318])
# a=np.random.rand(len(people))
# a=np.random.rand(4)
# a

performance = 3+ 10 * np.random.rand(len(people))
```

In [34]:
```
#seed 고정되었을 경우 rand()함수를 여러번 사용하면
#사용할때마다 다른 난수가 발생한다.
#단, 특정함수를 여러번 실행시켜도 결과는 동일하다.
np.random.seed(0)
print(np.random.rand(4))
print(np.random.rand(4))
print(np.random.rand(4))
```

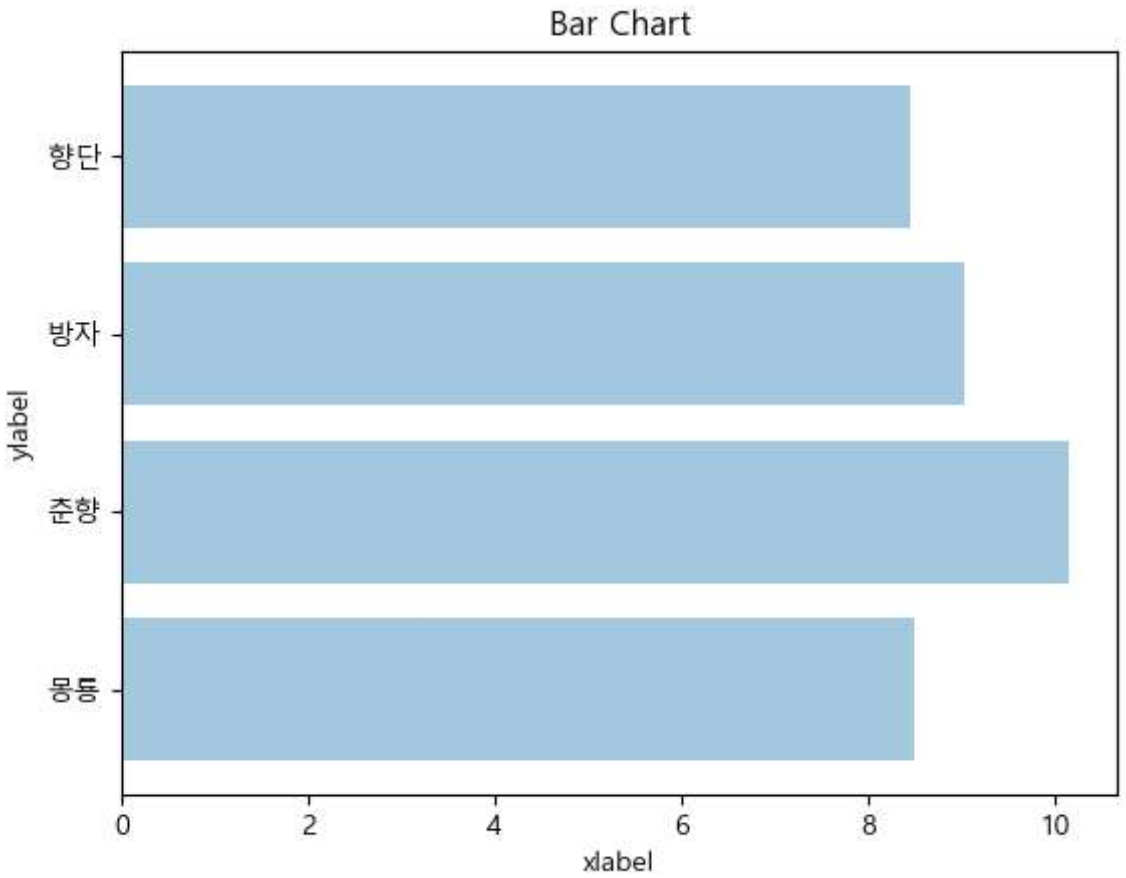
```
[0.5488135  0.71518937 0.60276338 0.54488318]
[0.4236548  0.64589411 0.43758721 0.891773  ]
[0.96366276 0.38344152 0.79172504 0.52889492]
```

- 가로 막대 그래프 그리기
  - barh(x,y,color=[], alpha=)

In [35]:
```
np.random.seed(0)
people=['몽룡','춘향','방자','향단']
y=np.arange(len(people))
performance = 3+ 10 * np.random.rand(len(people))
print(y)
print(performance)

plt.title('Bar Chart')
plt.barh(y,performance,alpha=0.4) # 주의 : 첫번째 데이터 y가 y축으로 표현됨
plt.yticks(y,people)
plt.xlabel('xlabel')
plt.ylabel('ylabel')
```

```
[0 1 2 3]
[ 8.48813504 10.15189366  9.02763376  8.44883183]
```

Out[35]:
```
Text(0, 0.5, 'ylabel')
```


Bar Chart
향단
방자
춘향
몽룡
ylabel
0
2
4
6
8
10
xlabel

데이터 프레임으로 바 그래프 그리기

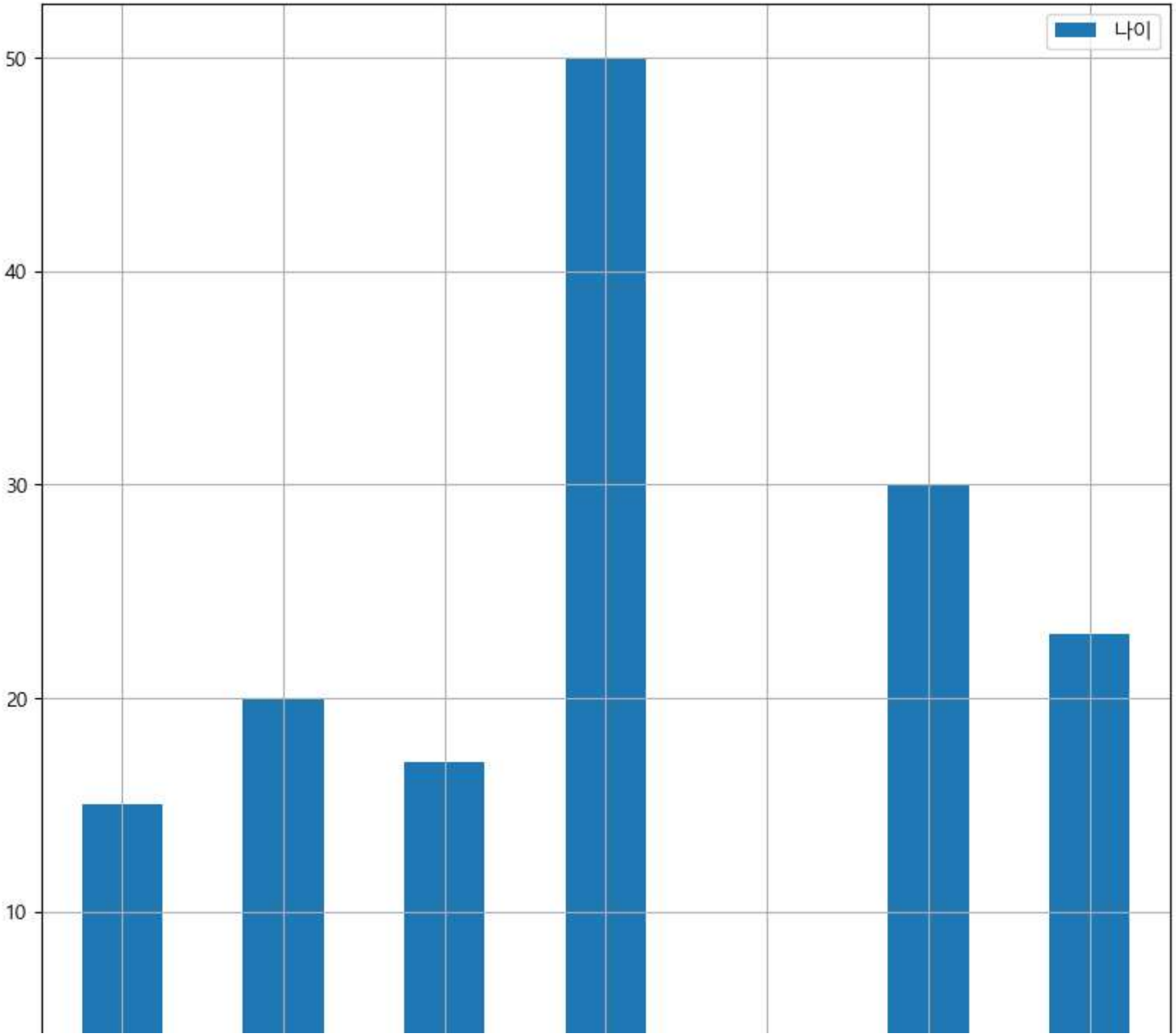
In [36]:
```
#데이터 프레임으로 바 그래프 그리기 1
df1 = pd.DataFrame({
    '나이':[15,20,17,50,2,30,23],
    '이름':['둘리','도우너','또치','길동','희동','마이클','영희']
},columns=['나이','이름'])
# df1
x=[0,1,2,3,4,5,6,7] #xticks 시 위치 표시에 사용할 변수

df1
```

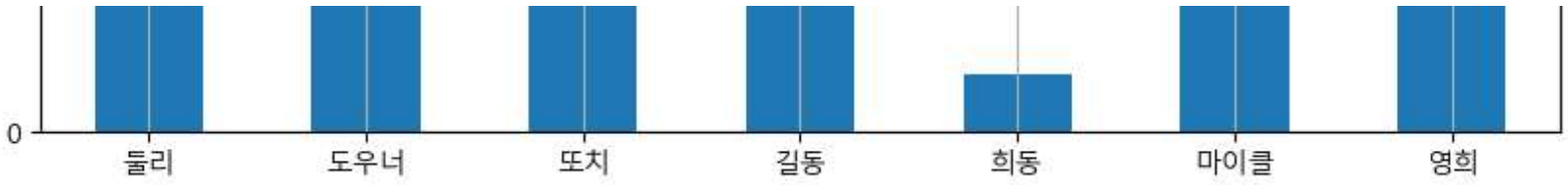
Out[36]:

	나이	이름
0	15	둘리
1	20	도우너
2	17	또치
3	50	길동
4	2	희동
5	30	마이클
6	23	영희

In [37]:
```
# 바그래프 그리기 - plot() 이용
x=[0,1,2,3,4,5,6] # x tick의 실제 위치값으로 사용
df1.plot(kind='bar', grid=True,figsize=(10,10))
plt.xticks(x,df1.이름,rotation='horizontal')
```

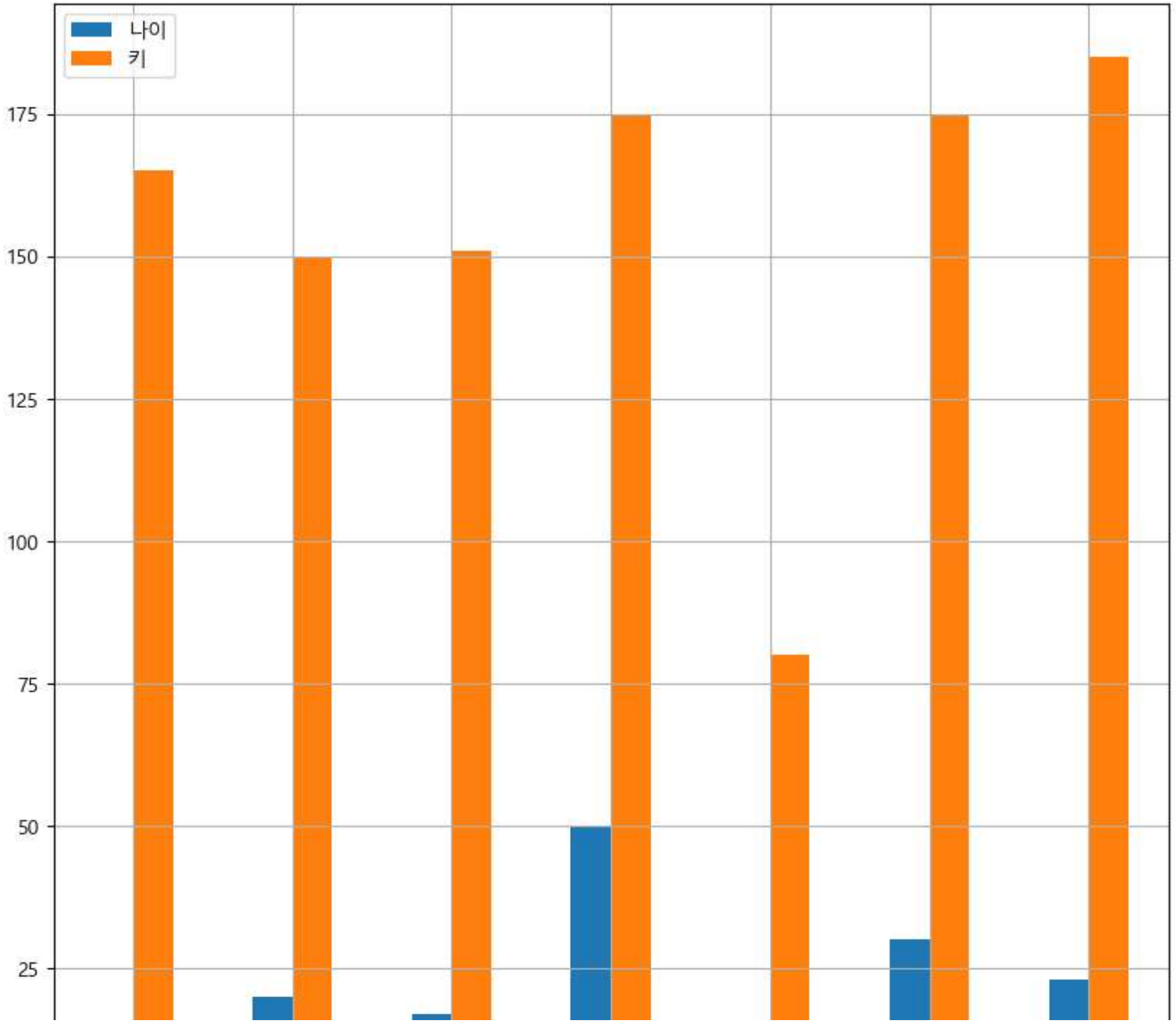
Out[37]:
```
([<matplotlib.axis.XTick at 0x194943b35e0>,
  <matplotlib.axis.XTick at 0x194943b35b0>,
  <matplotlib.axis.XTick at 0x194943dbc10>,
  <matplotlib.axis.XTick at 0x194944429a0>,
  <matplotlib.axis.XTick at 0x1949444a130>,
  <matplotlib.axis.XTick at 0x1949444a880>,
  <matplotlib.axis.XTick at 0x194944500a0>],
 [Text(0, 0, '둘리'),
  Text(1, 0, '도우너'),
  Text(2, 0, '또치'),
  Text(3, 0, '길동'),
  Text(4, 0, '희동'),
  Text(5, 0, '마이클'),
  Text(6, 0, '영희')])
```


나이
50
40
30
20
10


0
둘리
도우너
또치
길동
희동
마이클
영희

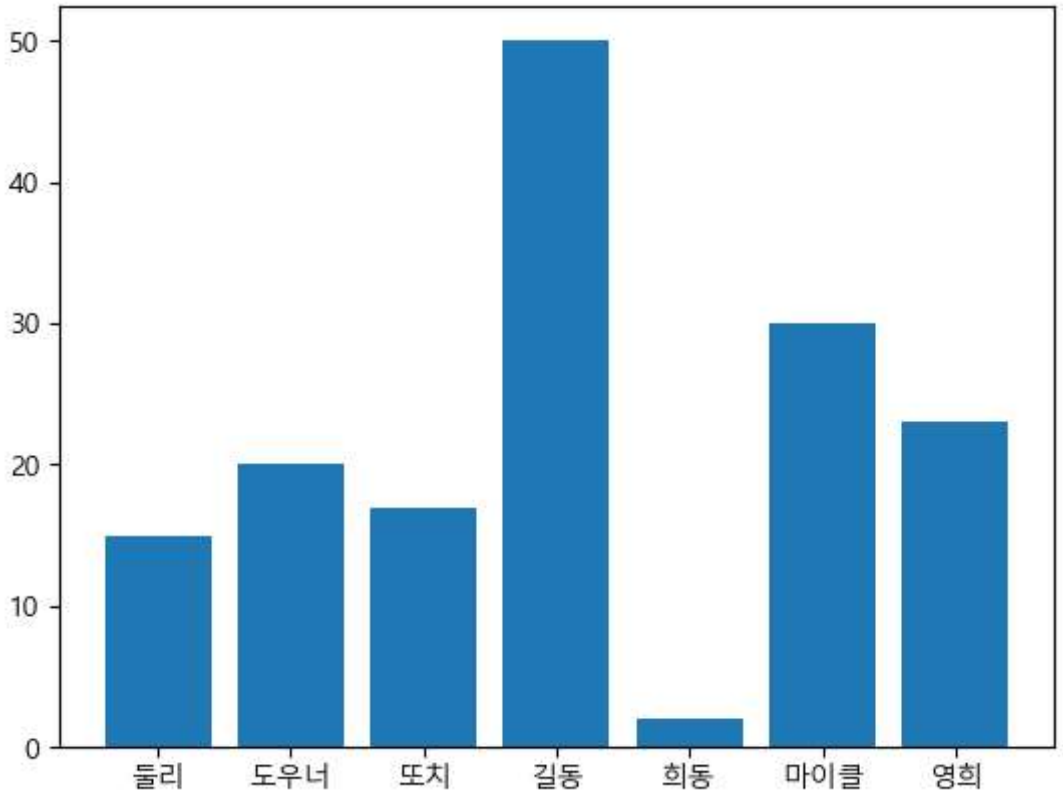
In [38]:
```
#데이터 프레임으로 바 그래프 그리기 1-1
#열 지정없이 그래프를 그리면
#수치 데이터가 있는 모든 필드를 이용해서 묶음 막대 그래프를
#그리게 된다.
df1 = pd.DataFrame({
    '나이':[15,20,17,50,2,30,23],
    '키' : [165,150,151,175,80,175,185],
    '이름':['둘리','도우너','또치','길동','희동','마이클','영희']
},columns=['나이','키','이름'])
# df1

x=[0,1,2,3,4,5,6] #xticks 시 위치 표시에 사용할 변수

#df로 막대그래프 그리기 첫번째 방법

df1.plot(kind='bar', grid=True,figsize=(10,10))
plt.xticks(x,df1.이름,rotation='horizontal')
```

Out[38]:
```
([<matplotlib.axis.XTick at 0x1949411b580>,
  <matplotlib.axis.XTick at 0x1949411b280>,
  <matplotlib.axis.XTick at 0x19493de6e20>,
  <matplotlib.axis.XTick at 0x194940b6580>,
  <matplotlib.axis.XTick at 0x194940b65b0>,
  <matplotlib.axis.XTick at 0x194940b6eb0>,
  <matplotlib.axis.XTick at 0x194940bcaf0>],
 [Text(0, 0, '둘리'),
  Text(1, 0, '도우너'),
  Text(2, 0, '또치'),
  Text(3, 0, '길동'),
  Text(4, 0, '희동'),
  Text(5, 0, '마이클'),
  Text(6, 0, '영희')])
```

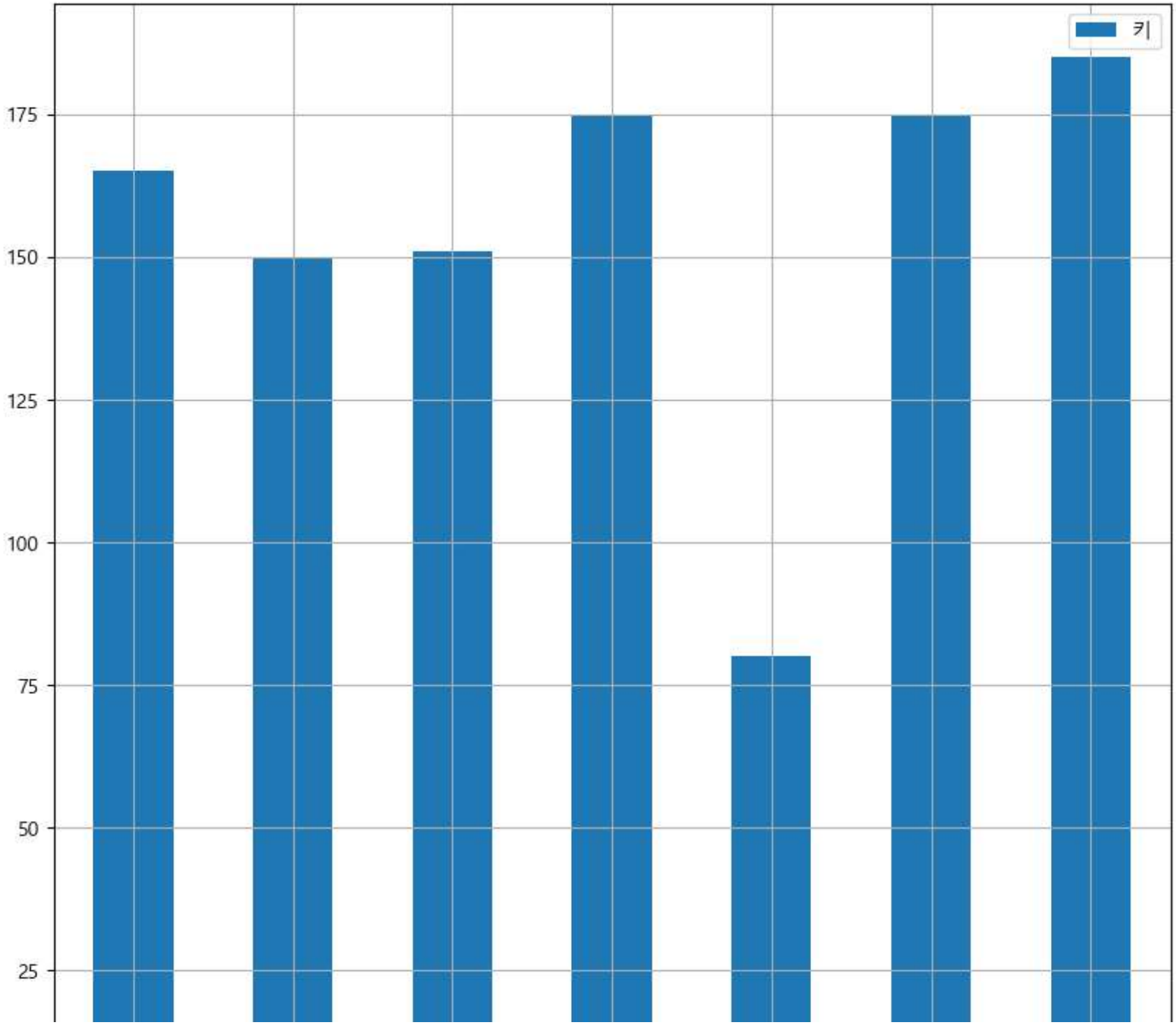

나이
키
175
150
125
100
75
50
25

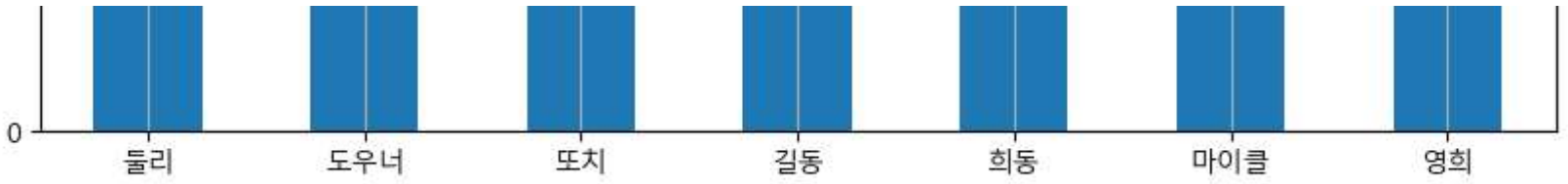

0
둘리
도우너
또치
길동
희동
마이클
영희

In [39]:
```
#데이터프레임으로 바 그래프 그리기 1-2
df1 = pd.DataFrame({
    '나이':[15,20,17,50,2,30,23],
    '키' : [165,150,151,175,80,175,185],
    '이름':['둘리','도우너','또치','길동','희동','마이클','영희']
},columns=['나이','키','이름'])

plt.bar(df1.이름, df1.나이)
plt.show()
```

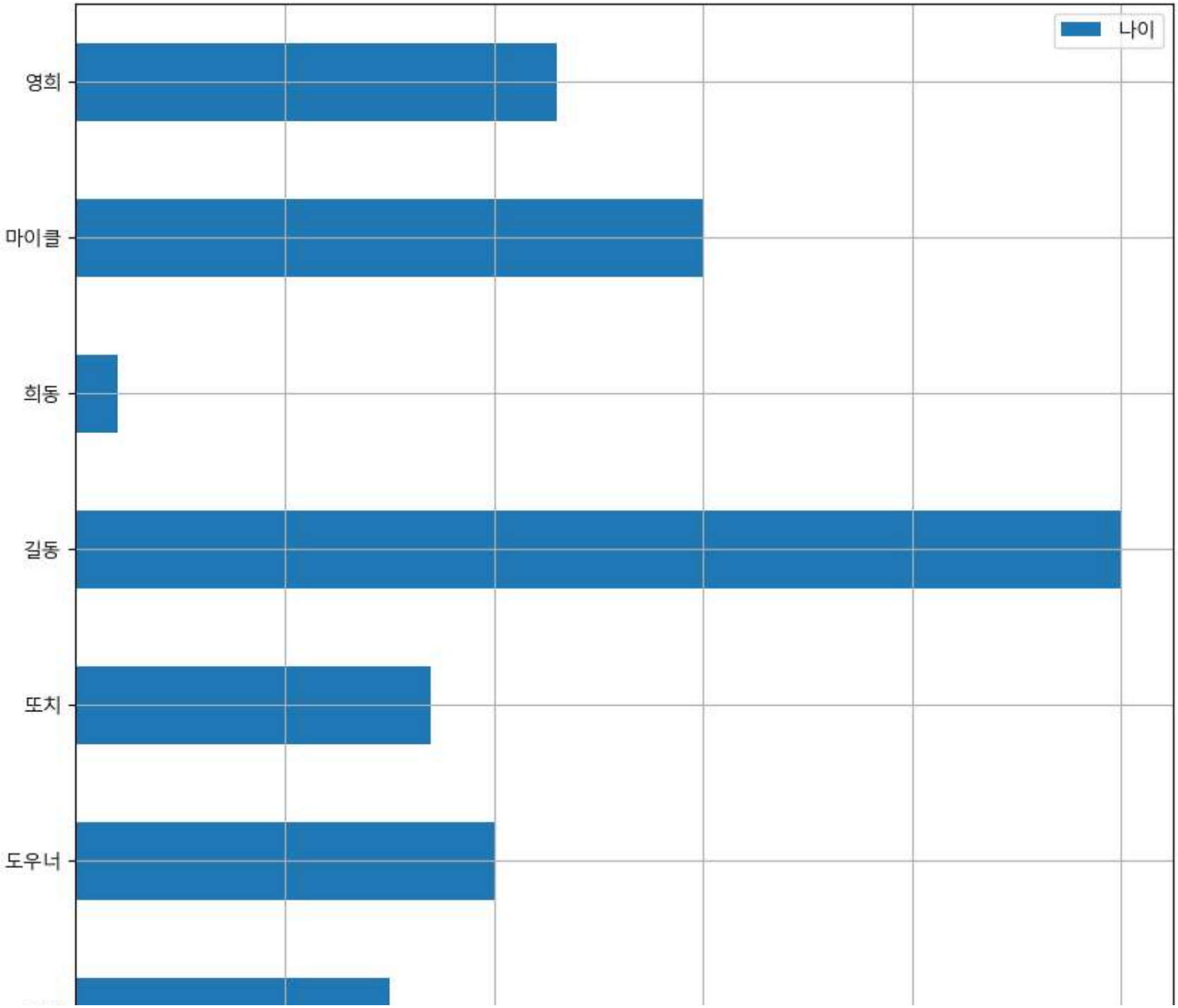


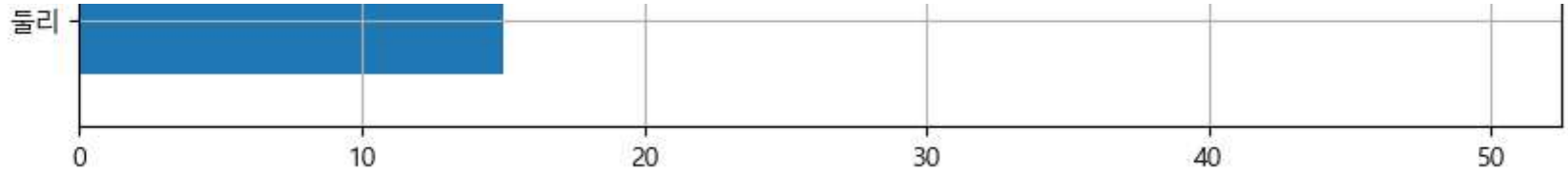
In [40]:
```
#데이터프레임의 일부 필드를 데이터프레임으로 추출해서
#그래프 그리기
df1 = pd.DataFrame({
    '나이':[15,20,17,50,2,30,23],
    '키' : [165,150,151,175,80,175,185],
    '이름':['둘리','도우너','또치','길동','희동','마이클','영희']
},columns=['나이','키','이름'])
# df1

x=[0,1,2,3,4,5,6] #xticks 시 위치 표시에 사용할 변수

df1[['키']].plot(kind='bar',grid=True, figsize=(10,10))
plt.xticks(x,df1.이름,rotation='horizontal')
plt.show()
```


키
175
150
125
100
75
50
25


0
둘리
도우너
또치
길동
희동
마이클
영희

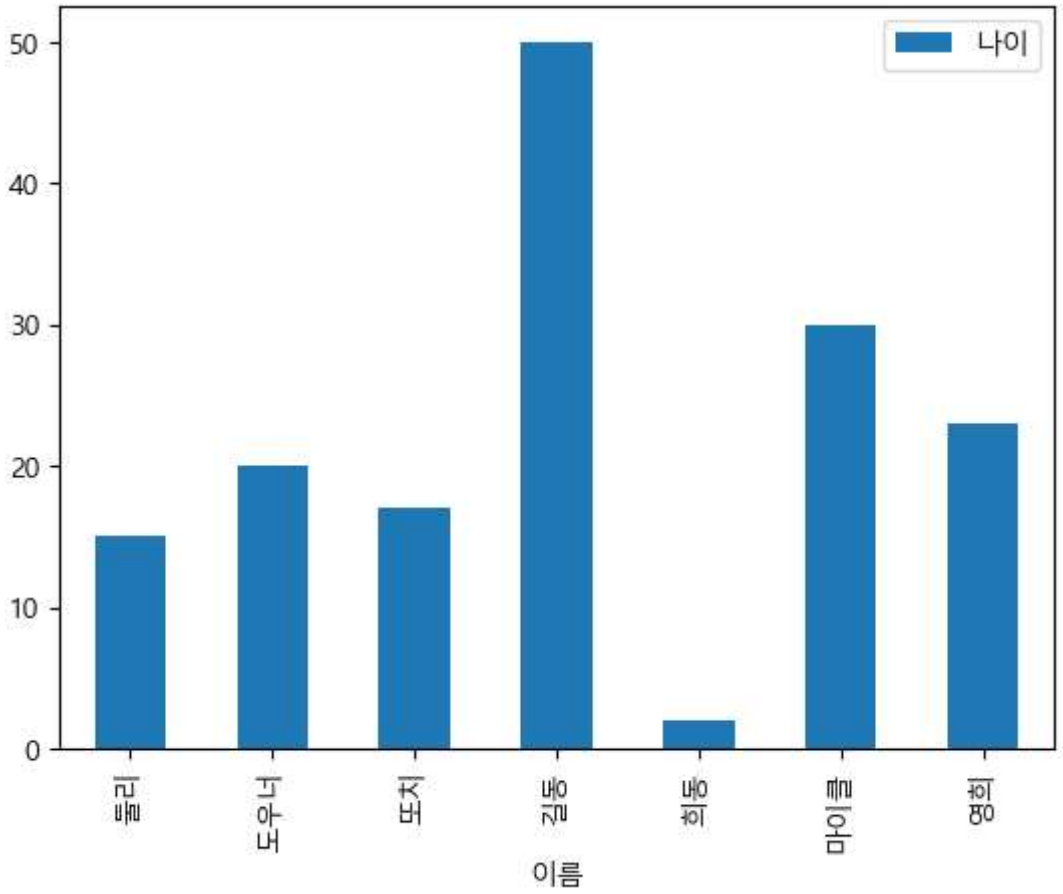
In [41]:
```
## 가로막대 그래프
#데이터프레임의 일부 필드를 데이터프레임으로 추출해서
#그래프 그리기
df1 = pd.DataFrame({
    '나이':[15,20,17,50,2,30,23],
    '이름':['둘리','도우너','또치','길동','희동','마이클','영희']
},columns=['나이','이름'])
# df1

x=[0,1,2,3,4,5,6] #xticks 시 위치 표시에 사용할 변수

df1[['나이']].plot(kind='barh',grid=True, figsize=(10,10))
plt.yticks(x,df1.이름,rotation='horizontal')
plt.show()
```


나이
영희
마이클
희동
길동
또치
도우너


들리
0
10
20
30
40
50

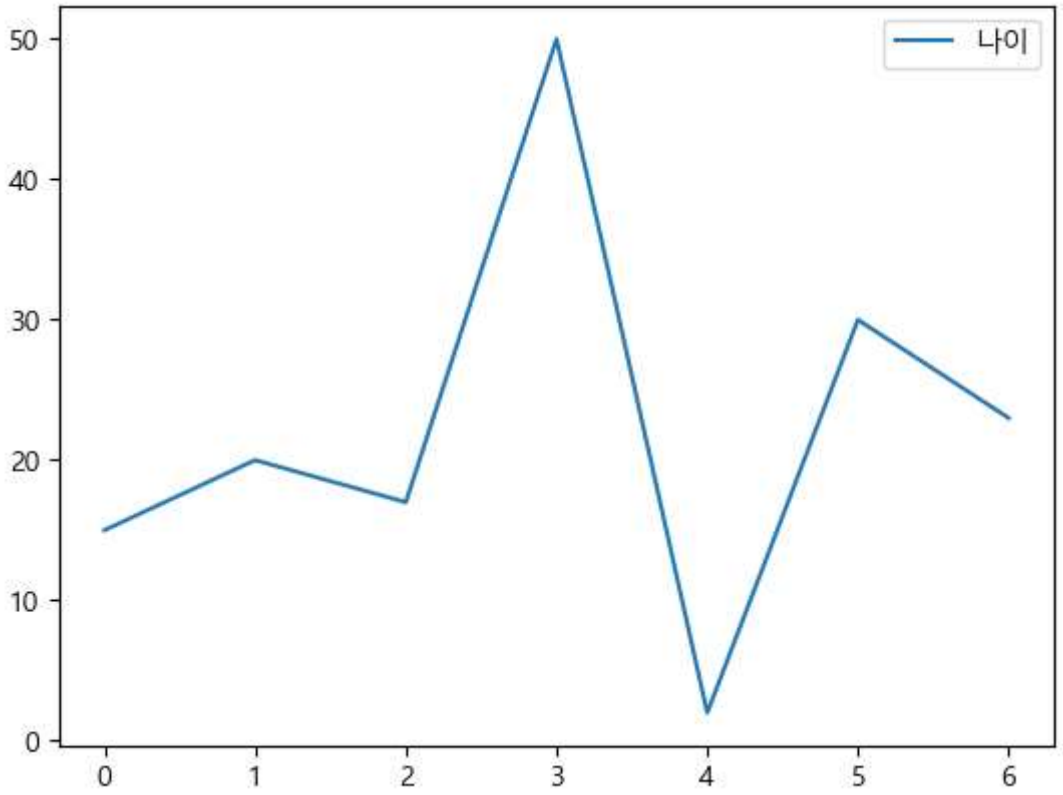
In [42]:
```
# 데이터프레임의 특정 필드만 추출해서 그래프 그리기
df1 = pd.DataFrame({
    '나이':[15,20,17,50,2,30,23],
    '이름':['둘리','도우너','또치','길동','희동','마이클','영희']
},columns=['나이','이름'])

df1.plot('이름','나이',kind='bar')
```

Out[42]: <AxesSubplot:xlabel='이름'>



In [43]:
```
df1.plot(y='나이').plot(grid=True,figsize=(5,5))
```

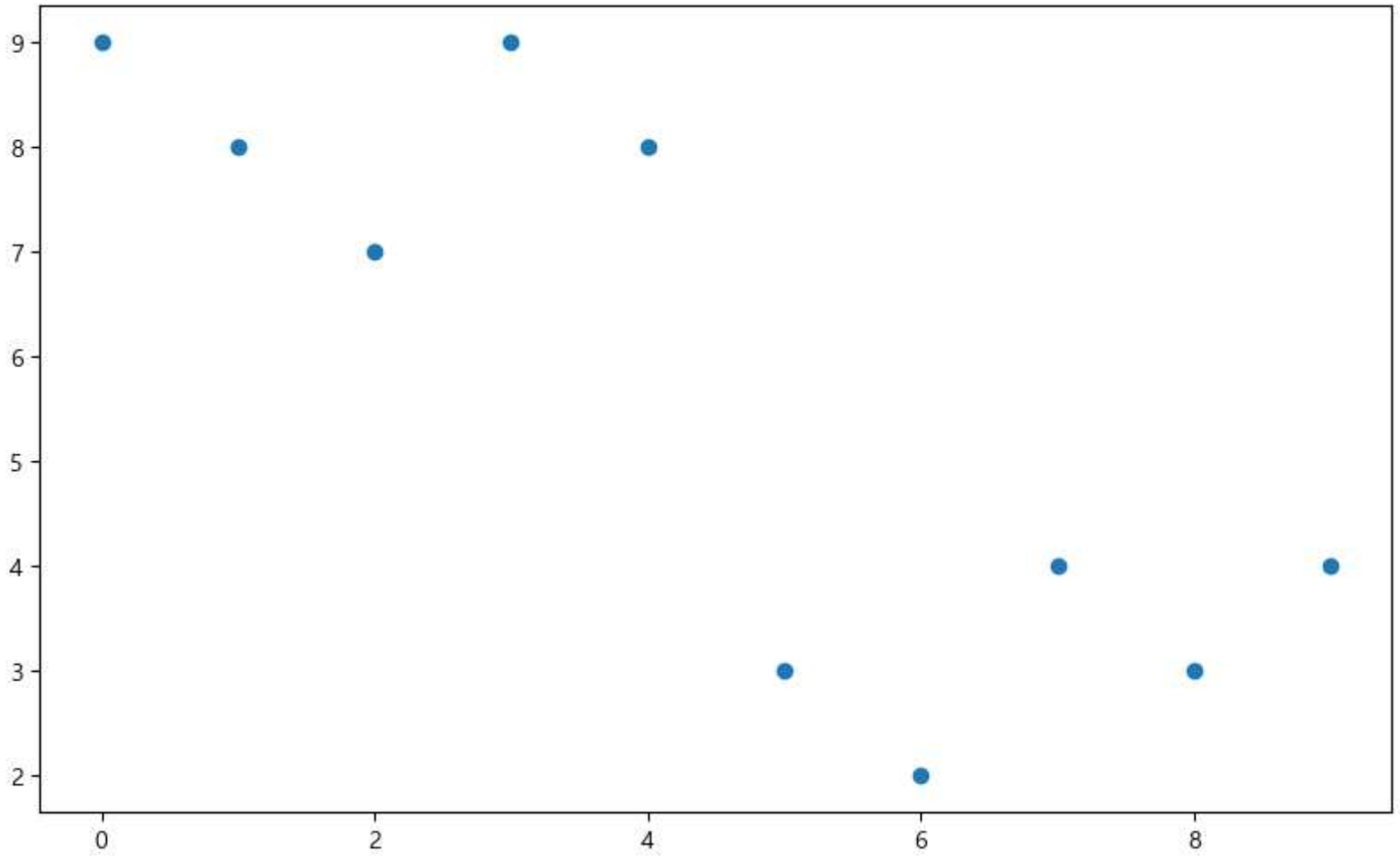
Out[43]: []



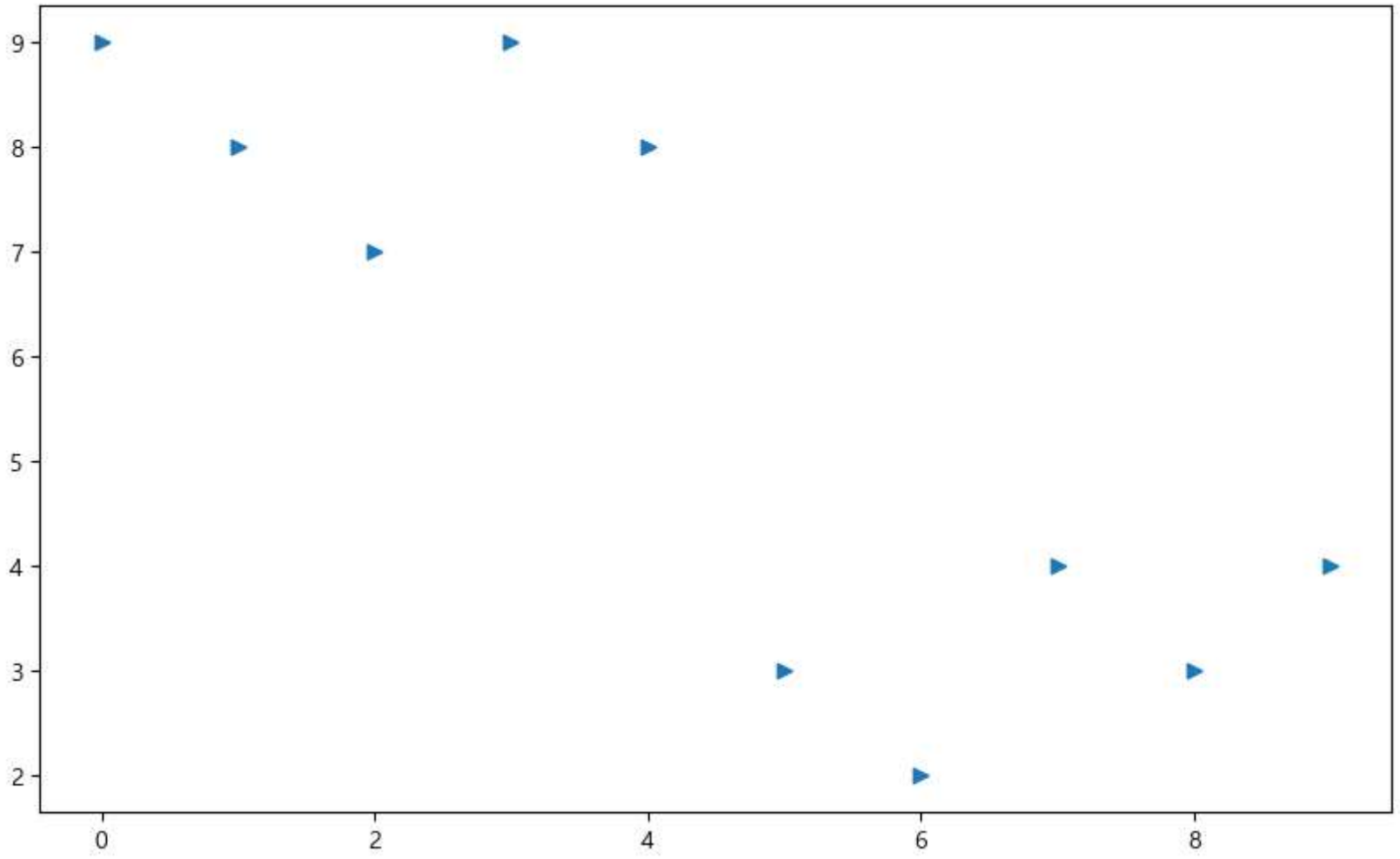
스캐터 플롯(scatter plot) : scatter()

In [44]:
```
#분산형 그래프
t = np.array([0,1,2,3,4,5,6,7,8,9])
y = np.array([9,8,7,9,8,3,2,4,3,4])
```

In [45]:
```
plt.figure(figsize=(10,6))
plt.scatter(t,y)
plt.show()
```

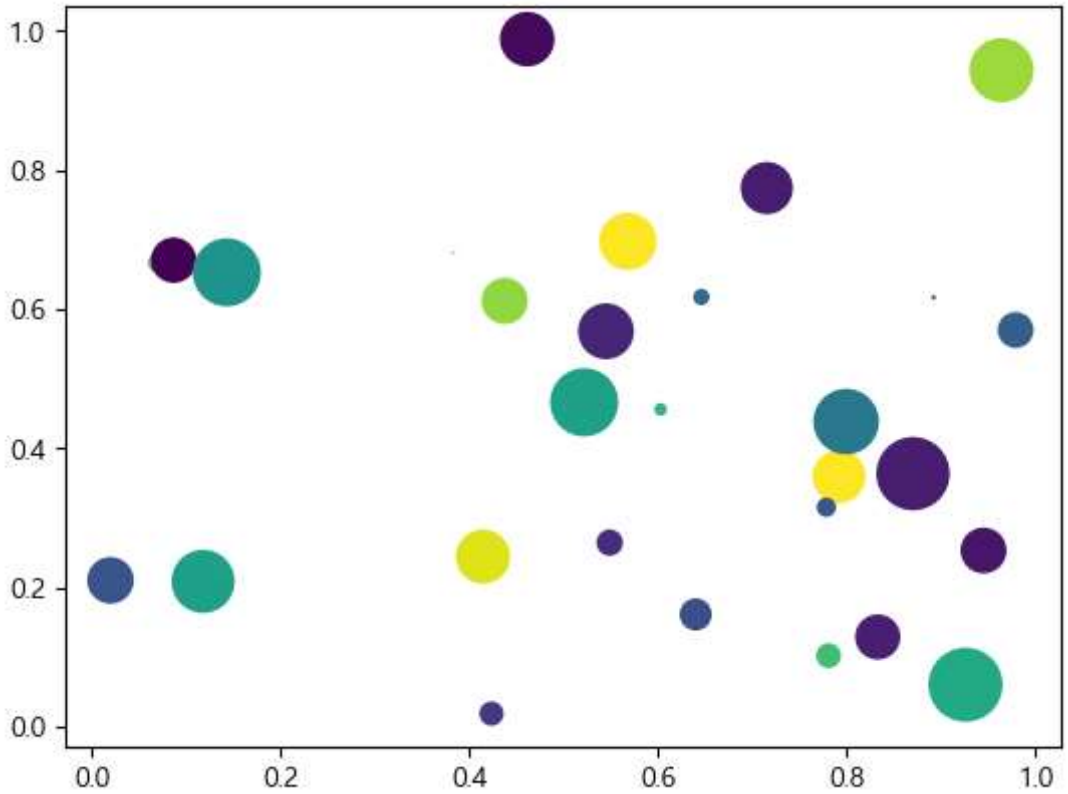


In [46]:
```
plt.figure(figsize=(10,6))
plt.scatter(t,y,marker='>')
plt.show()
```

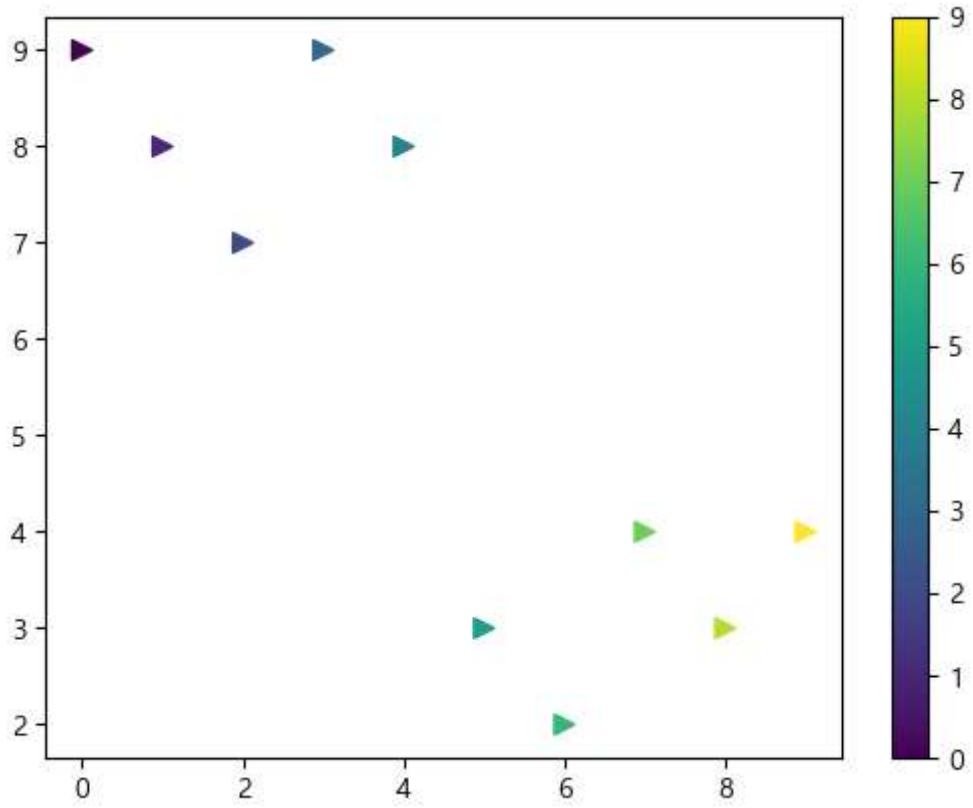


In [47]:
```
#버블차트 : 점 하나의 크기 또는 색상을 이용해서 서로 다른 데이터 값을 표시하는 그래프
#s 인수 : size
#c 인수 : color
N=30
np.random.seed(0)
x=np.random.rand(N)
y1 =np.random.rand(N)
y2 =np.random.rand(N)
y3=np.pi *(15 * np.random.rand(N))**2
plt.scatter(x,y1,c=y2,s=y3)
```

Out[47]: <matplotlib.collections.PathCollection at 0x19493ca11f0>

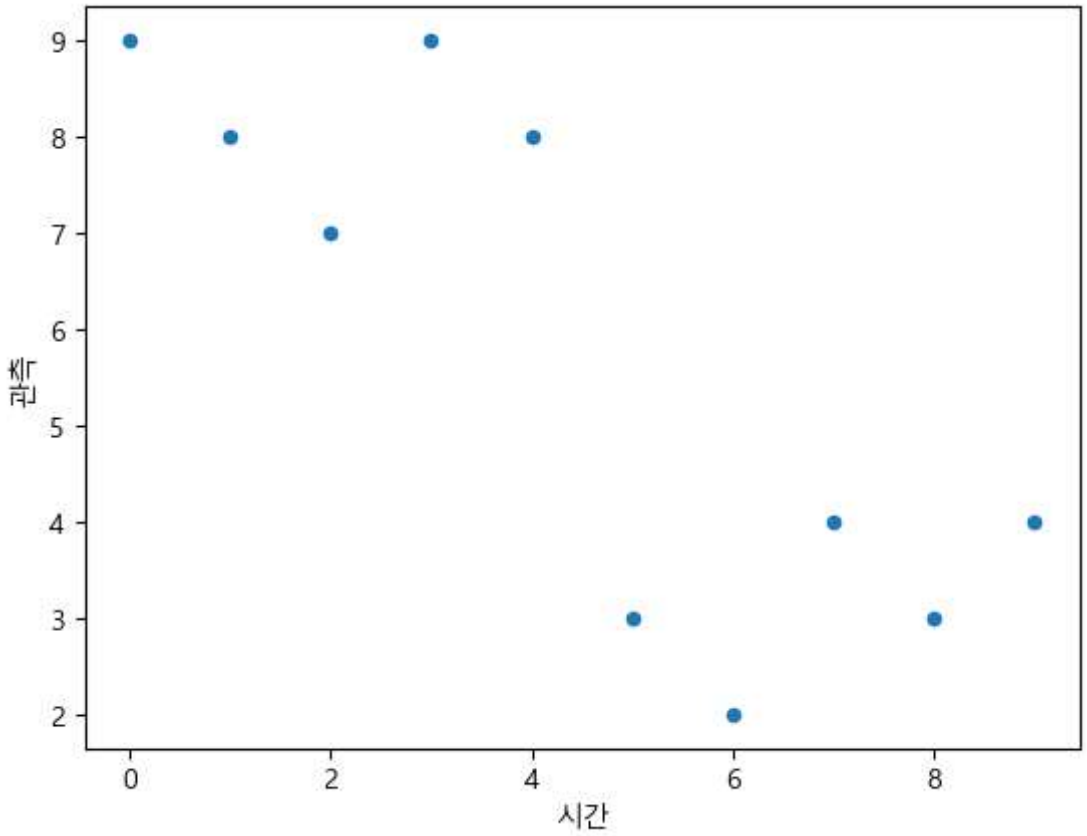


In [48]:
```
#color map 을 이용해서 그래프 그리기
colormap = t

plt.scatter(t,y,s=50,c=colormap,marker='>')

plt.colorbar() # 색상값의 가중치를 bar로 출력
plt.show()
```



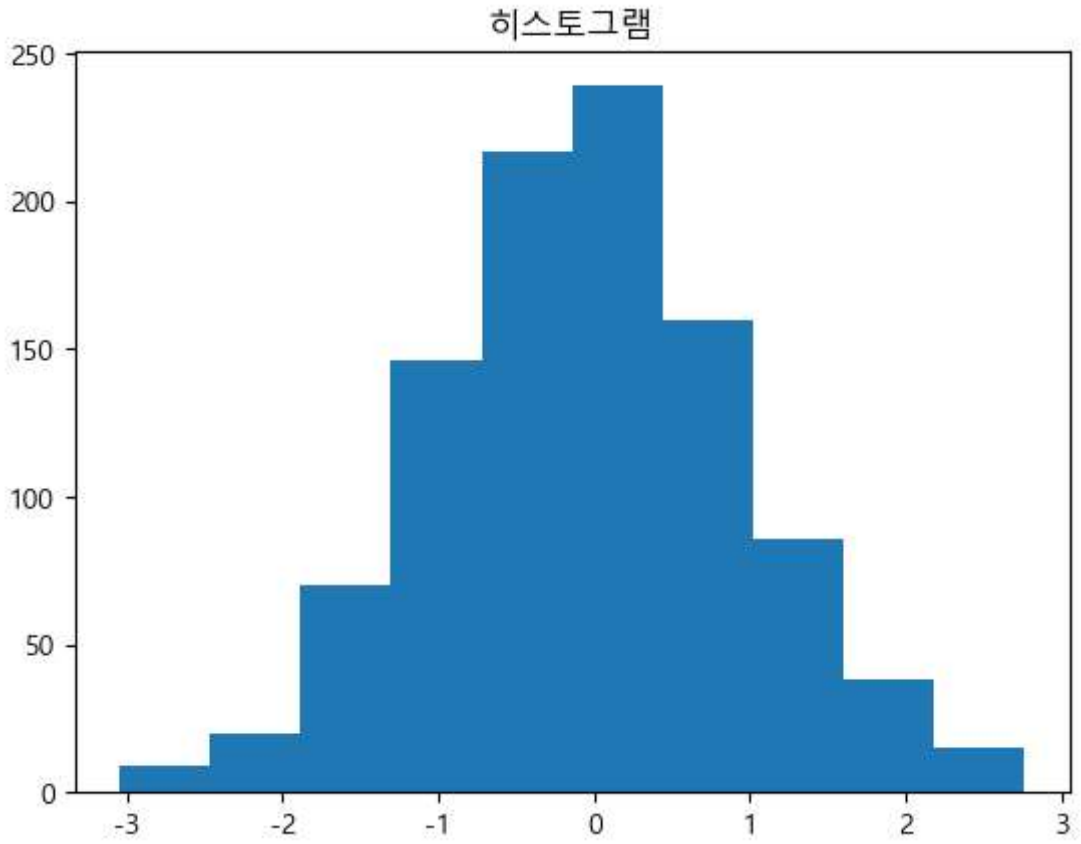
In [49]:
```
df2 = pd.DataFrame({'시간':t,
                    '관측':y})
df2

# df로 scatter 그릴때는 x,y 값을 명시
df2.plot('시간','관측',kind='scatter')
```

Out[49]: <AxesSubplot:xlabel='시간', ylabel='관측'>



4. 히스토그램 : hist()

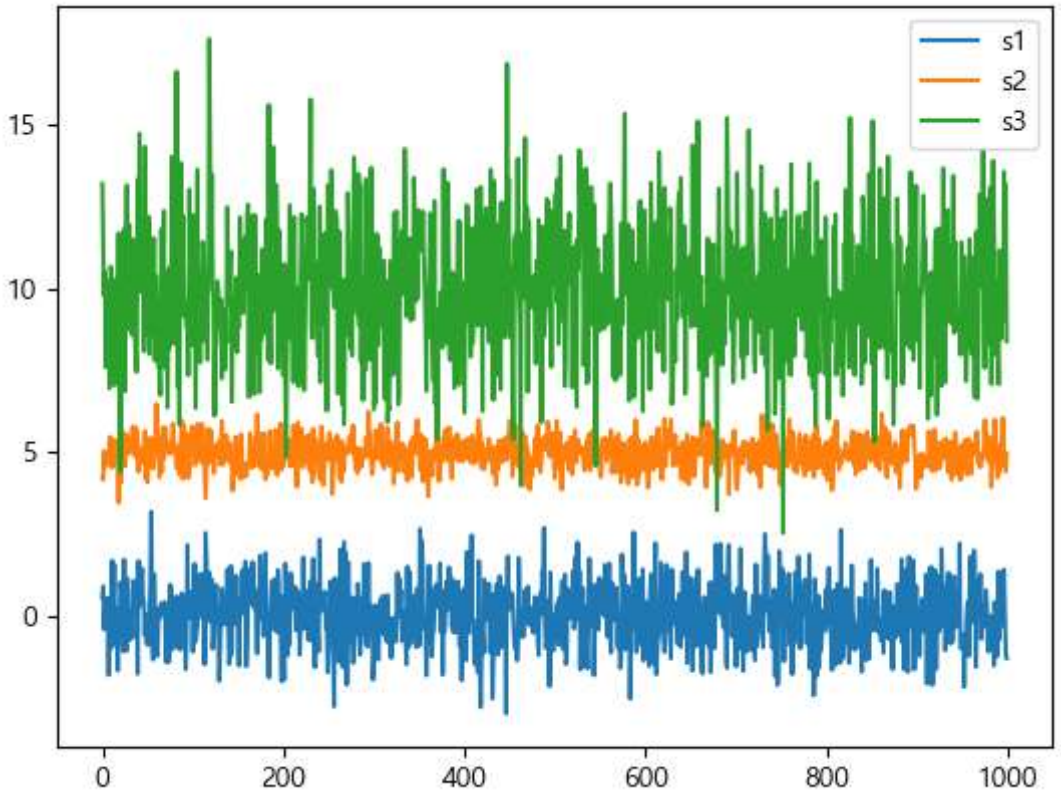
In [50]:
```
np.random.seed(0)
x=np.random.randn(1000) #난수 1000개 발생
plt.title('히스토그램')
plt.hist(x)
plt.show()
```



5.박스플롯 : boxplot()

In [51]:
```
#다차원 array 형태로 무작위 샘플을 생성
#np.random.normal(정규분포평균,표준편차,(행열) or 개수)
#정규분포 확률 밀도에서 표본 추출해주는 함수

#데이터 3개 생성
s1=np.random.normal(loc=0,scale=1,size=1000)
s2=np.random.normal(loc=5,scale=0.5,size=1000)
s3=np.random.normal(loc=10,scale=2,size=1000)
```

In [52]:
```
#line 그래프 이용해서 데이터 차이 확인
plt.plot(s1, label='s1')
plt.plot(s2, label='s2')
plt.plot(s3, label='s3')
plt.legend()
plt.show()
```

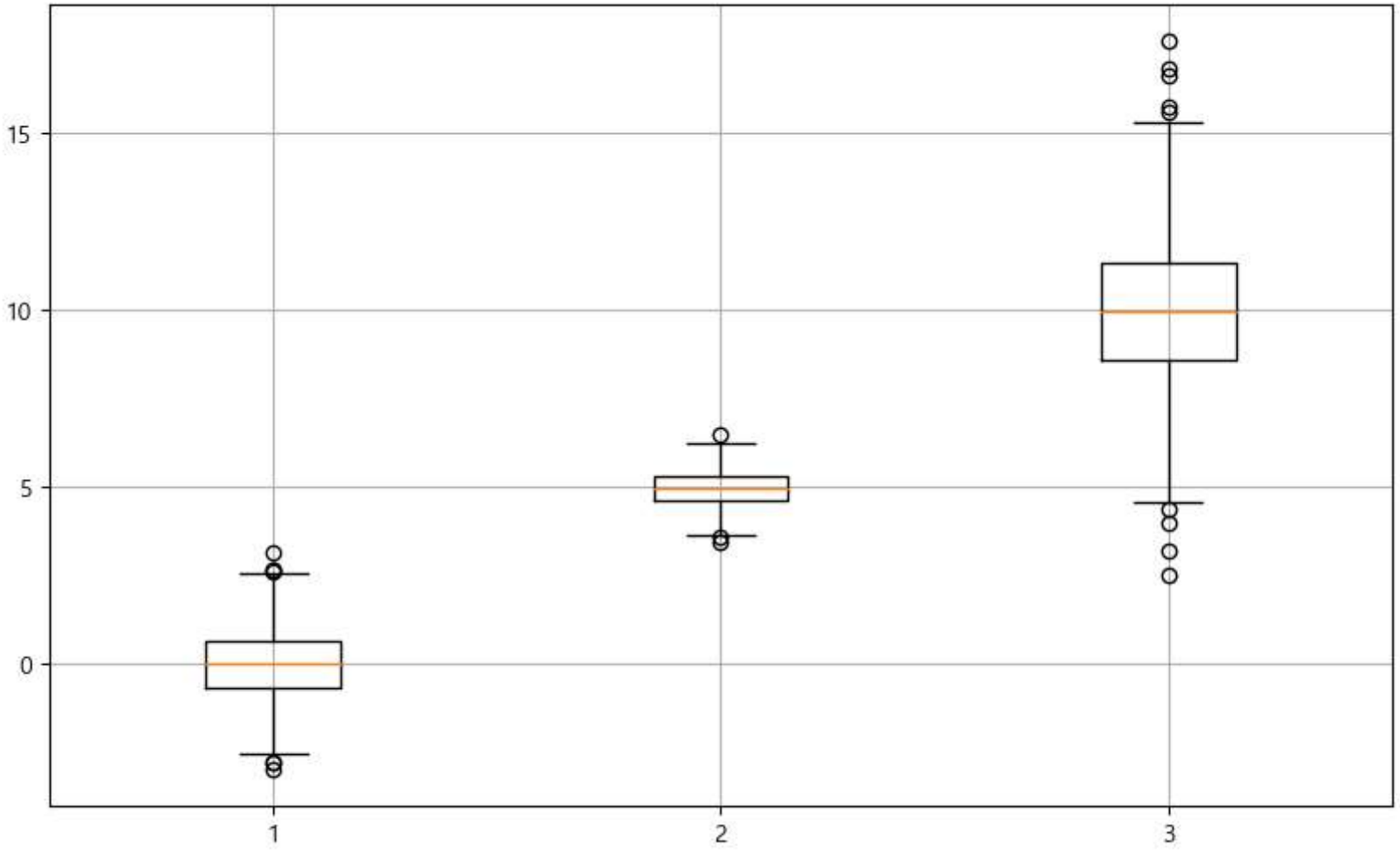


In [53]:
```
plt.figure(figsize=(10,6))
```

Out[53]:
```
<Figure size 1000x600 with 0 Axes>
```

```
<Figure size 1000x600 with 0 Axes>
```

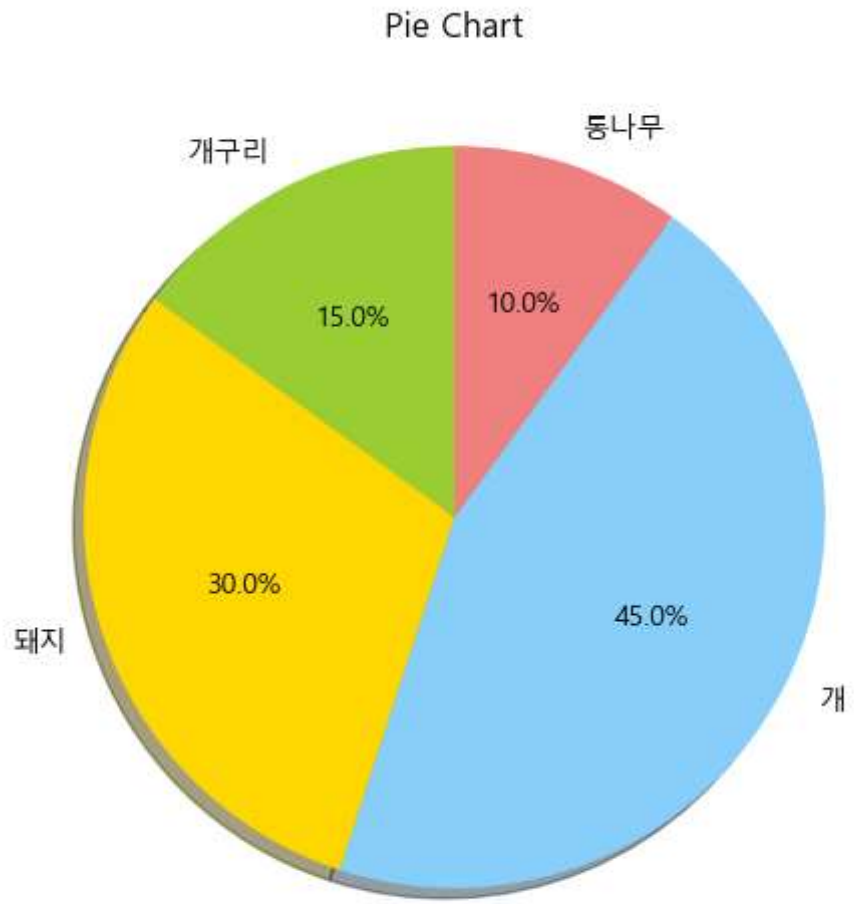
In [54]:
```
#박스 그래프
plt.figure(figsize=(10,6))
plt.boxplot([s1,s2,s3])
plt.grid()
plt.show()
```



6.파이차트:pie()

In [55]:
```
#카테고리 별 값의 상대적인 비교를 할때 주로 사용하는 차트
#원의 형태를 유지할 수 있도록 다음 명령을 실행해야 함.
#콘솔에서는 별 다른 변화 없음. plot 창에서는 필요함
#plt.axis('equal')

#예제 데이터 생성
labels=['개구리','돼지','개','통나무']
size=[15,30,45,10]
colors=['yellowgreen','gold','lightskyblue','lightcoral']
explode=(0,0.4,0,0.5)


plt.figure(figsize=(10,6))
plt.title('Pie Chart')
plt.pie(size,  labels=labels, colors=colors,
       autopct= '%1.1f%%', shadow=True,startangle=90)
```

Out[55]:
```
([<matplotlib.patches.Wedge at 0x19494496220>,
  <matplotlib.patches.Wedge at 0x19494496370>,
  <matplotlib.patches.Wedge at 0x19493e81b50>,
  <matplotlib.patches.Wedge at 0x19493e26850>],
 [Text(-0.4993895680663527, 0.9801071672559598, '개구리'),
  Text(-1.0461621424642782, -0.3399187721714579, '돼지'),
  Text(0.9801072140121813, -0.4993894763020948, '개'),
  Text(0.33991864973549485, 1.0461621822461364, '통나무')],
 [Text(-0.2723943098543742, 0.5346039094123416, '15.0%'),
  Text(-0.5706338958896062, -0.18541023936624976, '30.0%'),
  Text(0.5346039349157352, -0.27239425980114257, '45.0%'),
  Text(0.1854101725829972, 0.5706339175888016, '10.0%')])
```



In [ ]:

In [ ]:

In [ ]: