Python

라이브러리

김선녕(sykim.lecture@gmail.com)


Numpy
Matplotlib
Pandas

Numpy 패키지

- 과학 계산을 위한 라이브러리
- 다차원 배열 처리
- numpy 배열
  - rank : 차원
  - shape : 차원의 크기를 튜플로 표시

```
[1]  1 import numpy as np
     2 list = [1,2,3,4]
     3 a = np.array(list)
     4 print(a.shape)
```

```
(4,)
```

```
[37]  1 print(type(a))
```

```
<class 'numpy.ndarray'>
```

```
[2]  1 b = np.array([[1,2,3],[4,5,6]])
     2 print(b.shape)
     3 print(b[0,0])
```

```
(2, 3)
1
```

배열생성


```
[20]  1 a = np.zeros((2,2))
      2 print(a)
```

```
[[0. 0.]
 [0. 0.]]
```

```
[4]   1 a = np.ones((2,3))
      2 print(a)
```

```
[[1. 1. 1.]
 [1. 1. 1.]]
```

```
[5]   1 a = np.full((2,3), 5)
      2 print(a)
```

```
[[5 5 5]
 [5 5 5]]
```

```
[6]   1 a = np.eye(3)
      2 print(a)
```

```
[[1. 0. 0.]
 [0. 1. 0.]
 [0. 0. 1.]]
```

reshape() 함수(1/3)


```
[9]   1 a = np.ones([3,4,5])
      2 print(a)
```

```
[[[1. 1. 1. 1. 1.]
  [1. 1. 1. 1. 1.]
  [1. 1. 1. 1. 1.]
  [1. 1. 1. 1. 1.]]

 [[1. 1. 1. 1. 1.]
  [1. 1. 1. 1. 1.]
  [1. 1. 1. 1. 1.]
  [1. 1. 1. 1. 1.]]

 [[1. 1. 1. 1. 1.]
  [1. 1. 1. 1. 1.]
  [1. 1. 1. 1. 1.]
  [1. 1. 1. 1. 1.]]]
```

```
[16]   1 a.reshape((6,10))
```

```
array([[1., 1., 1., 1., 1., 1., 1., 1., 1., 1.],
       [1., 1., 1., 1., 1., 1., 1., 1., 1., 1.],
       [1., 1., 1., 1., 1., 1., 1., 1., 1., 1.],
       [1., 1., 1., 1., 1., 1., 1., 1., 1., 1.],
       [1., 1., 1., 1., 1., 1., 1., 1., 1., 1.],
       [1., 1., 1., 1., 1., 1., 1., 1., 1., 1.]])
```

reshape() 함수 (2/3)


```
[17]  1 a.reshape(3,-1)
```

```
array([[1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.,
        1., 1., 1., 1.],
       [1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.,
        1., 1., 1., 1.],
       [1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.,
        1., 1., 1., 1.]])
```

```
[18]  1 a.reshape(4,3,-1)
```

```
array([[[1., 1., 1., 1., 1.],
        [1., 1., 1., 1., 1.],
        [1., 1., 1., 1., 1.]],

       [[1., 1., 1., 1., 1.],
        [1., 1., 1., 1., 1.],
        [1., 1., 1., 1., 1.]],

       [[1., 1., 1., 1., 1.],
        [1., 1., 1., 1., 1.],
        [1., 1., 1., 1., 1.]],

       [[1., 1., 1., 1., 1.],
        [1., 1., 1., 1., 1.],
        [1., 1., 1., 1., 1.]]])
```

reshape() 함수(3/3)


```
[19]  1 a.reshape(13,2,-1)
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-19-aa315102c221> in <module>()
----> 1 a.reshape(13,2,-1)

ValueError: cannot reshape array of size 60 into shape (13,2,newaxis)
```

SEARCH STACK OVERFLOW

```
[7]  1 a = np.array(range(20)).reshape((4,5))
     2 print(a)
```

```
[[ 0  1  2  3  4]
 [ 5  6  7  8  9]
 [10 11 12 13 14]
 [15 16 17 18 19]]
```

slicing

[30]
```
list = [
    [1, 2, 3],
    [4, 5, 6],
    [7, 8, 9]
]
n_arr = np.array(list)

# [start:end] or [start:end:step]
a = n_arr[0:2, 0:2]
print(a)
```

```
[[1 2]
 [4 5]]
```

[33]
```
b = n_arr[1:, 1:] # [1:, 0:1]
print(b)
```

```
[[5 6]
 [8 9]]
```

[34]
```
arr = np.array([1, 2, 3, 4, 5, 6, 7])
print(arr[1:5:2])
```

```
[2 4]
```

[35]
```
arr = np.array([[1, 2, 3, 4, 5], [6, 7, 8, 9, 10]])
print(arr[1, 1:4])
```

```
[7 8 9]
```

indexing(1/2)

[40]
```
1 arr = np.array([1, 2, 3, 4])
2 print(arr[1])
```

```
2
```

[42]
```
1 arr = np.array([[1,2,3,4,5], [6,7,8,9,10]])
2
3 print('2nd element on 1st dim: ', arr[0, 1])
4 print('5th element on 2nd dim: ', arr[1, 4])
```

```
2nd element on 1st dim:  2
5th element on 2nd dim:  10
```

[44]
```
1 arr = np.array([[1,2,3,4,5], [6,7,8,9,10]])
2 print('Last element from 2nd dim: ', arr[1, -1])
```

```
Last element from 2nd dim:  10
```

indexing(2/2)


```
[43]  1 arr = np.array([[[1, 2, 3], [4, 5, 6]], [[7, 8, 9], [10, 11, 12]]])
      2 print(arr[0, 1, 2])
```

```
6
```

[[[1, 2, 3], [4, 5, 6]], [[7, 8, 9], [10, 11, 12]]]

```
[
 [
  [1, 2, 3],
  [4, 5, 6]
 ],
 [
  [7, 8, 9],
  [10, 11, 12]
 ]
]
```

numpy 연산


```
[51]  1 x = [1, 2, 3, 4]
      2 y = [4, 5, 6, 7]
      3
      4 a = np.add(x, y)
      5 b = np.subtract(x, y)
      6 c = np.multiply(x, y)
      7 d = np.divide(x, y)
      8
      9 print(a, b, c, d)
```

```
[ 5  7  9 11] [-3 -3 -3 -3] [ 4 10 18 28] [0.25       0.4        0.5        0.57142857]
```

dot() 함수 - vector or matrix 연산

[47]
```
list1 = [
    [1,2,3],
    [3,4,6]
]

list2 = [
    [5,6],
    [7,8],
    [9,10]
]
a = np.array(list1)
b = np.array(list2)

c = np.dot(a, b)
print(c)
```

```
[[ 46  52]
 [ 97 110]]
```

numpy 연산

[57]
```
1 x = [1, 2, 3, 4]
2 y = [4, 5, 6, 7]
3
4 print(np.add(x, y))
5 print(np.subtract(x, y))
6 print(np.multiply(x, y))
7 print(np.divide(x, y))
```

```
[ 5  7  9 11]
[-3 -3 -3 -3]
[ 4 10 18 28]
[0.25       0.4        0.5        0.57142857]
```

[58]
```
1 x = np.array([[1,2],[3,4]])
2
3 print(np.sum(x))
4 print(np.sum(x, axis=0))
5 print(np.sum(x, axis=1))
6 print(np.prod(x))
```

```
10
[4 6]
[3 7]
24
```

random()함수(1/2)


```
[76]  1 x = np.random.randint(100, size=(3, 5))
      2 print(x)
```

```
[[49 70 28 39 24]
 [90 88 63 97 62]
 [65 17  6 17  0]]
```

```
[78]  1 x = np.random.rand(5)
      2 print(x)
```

```
[0.15584415 0.30411971 0.46081787 0.87030141 0.26021401]
```

```
[79]  1 x = np.random.rand(3, 5)
      2 print(x)
```

```
[[0.72743018 0.03926854 0.31199742 0.53518991 0.65116507]
 [0.19850554 0.46463453 0.49819918 0.56540146 0.26763839]
 [0.20754275 0.94399626 0.67460862 0.12842712 0.09176041]]
```

random()함수(2/2)

[80]
```
1 x = np.random.choice([3, 5, 7, 9])
2 print(x)
```

```
7
```

[81]
```
1 x = np.random.choice([3, 5, 7, 9], size=(3, 5))
2 print(x)
```

```
[[7 3 7 3 5]
 [3 7 5 3 9]
 [3 3 7 9 5]]
```

[86]
```
1 mu, sigma = 10, 0.1 # mean and standard deviation
2 x = np.random.normal(mu, sigma, size=(3, 5))
3 print(x)
```

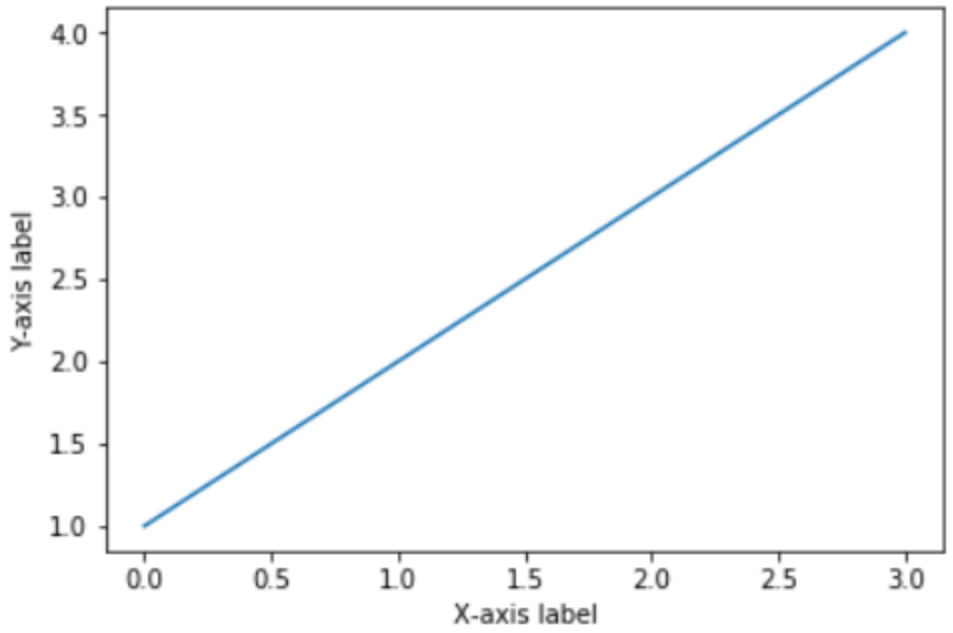
```
[[10.00148785 10.16209655 10.07034586 10.00561144 10.21317226]
 [ 9.86556907  9.89766168 10.0744619  10.17792387  9.74979853]
 [ 9.85765947 10.24484865  9.96300179  9.82233445  9.97044581]]
```


Numpy
Matplotlib
Pandas

기본 그래프

In [2]:

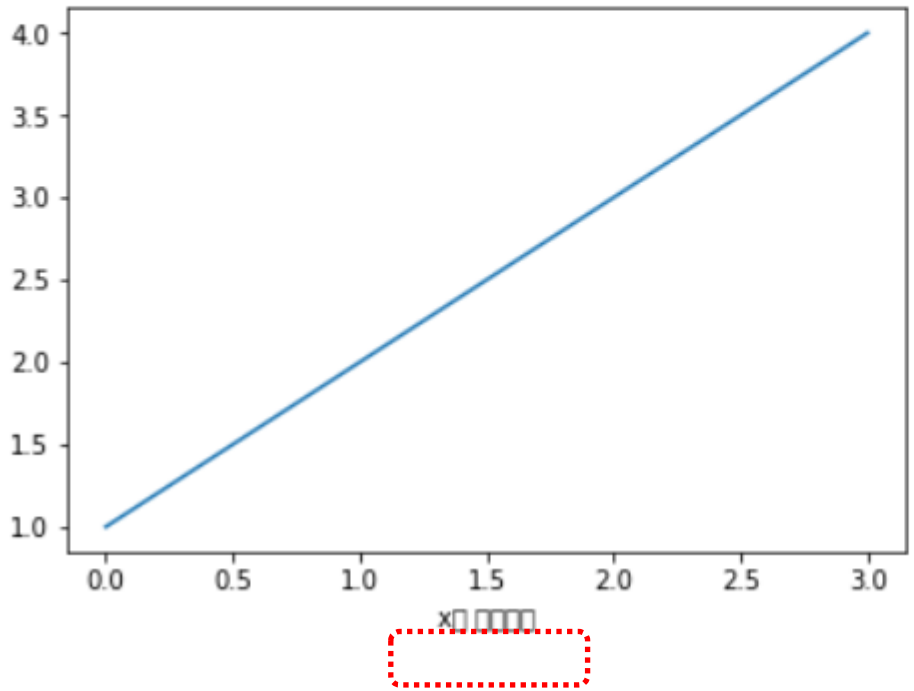
```
1 import matplotlib.pyplot as plt
2 plt.plot([1,2,3,4])
3 plt.xlabel("X-axis label")
4 plt.ylabel("Y-axis label")
5 plt.show()
```



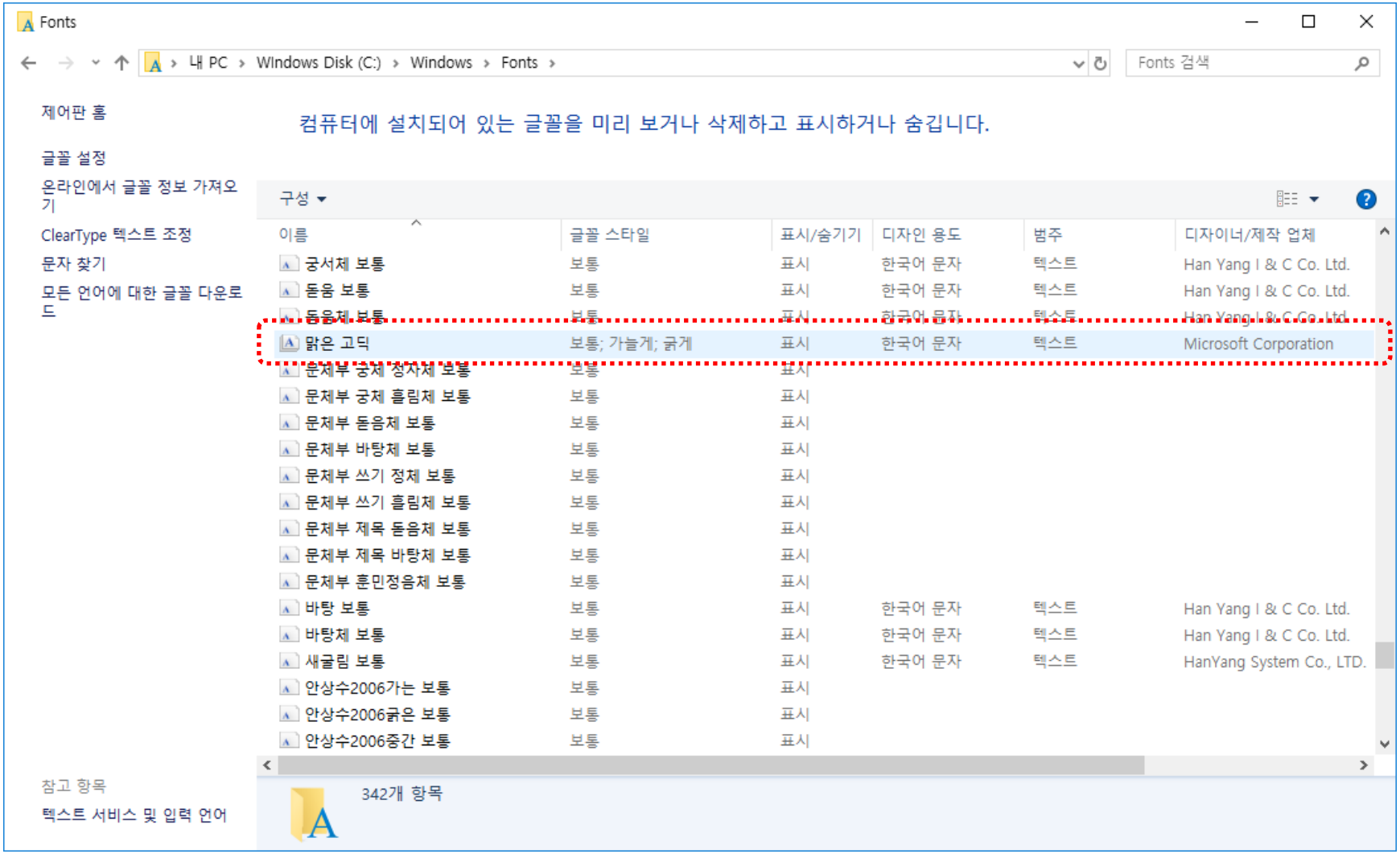
한글처리

In [8]:
```
1 plt.plot([1,2,3,4])
2 plt.xlabel('x축 한글표시')
3 plt.show
```

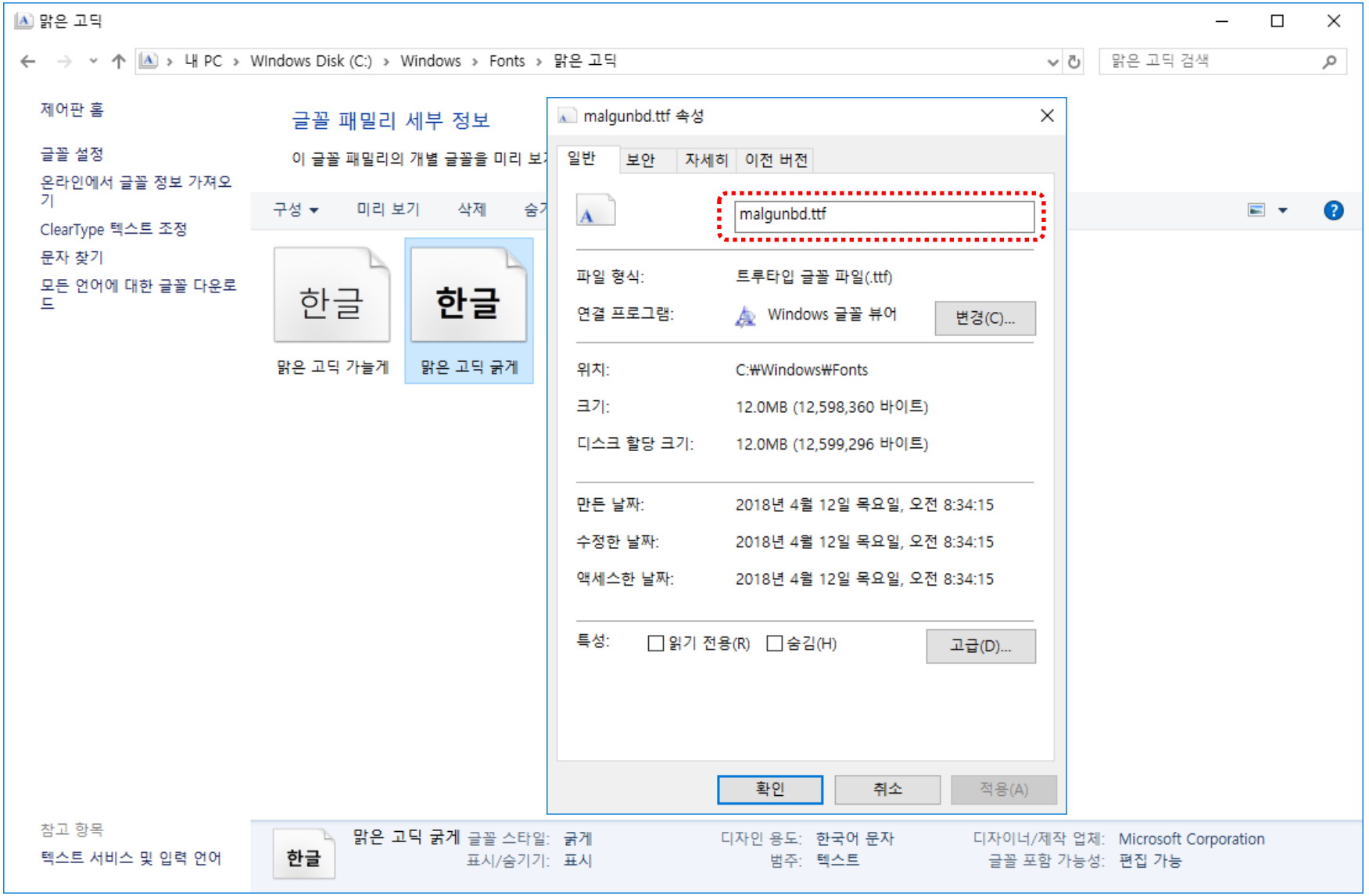
Out[8]: <function matplotlib.pyplot.show(*args, **kw)>



한글처리


Fonts
내 PC › Windows Disk (C:) › Windows › Fonts ›
Fonts 검색
제어판 홈
글꼴 설정
온라인에서 글꼴 정보 가져오기
ClearType 텍스트 조정
문자 찾기
모든 언어에 대한 글꼴 다운로드
컴퓨터에 설치되어 있는 글꼴을 미리 보거나 삭제하고 표시하거나 숨깁니다.
구성
이름 글꼴 스타일 표시/숨기기 디자인 용도 범주 디자이너/제작 업체
궁서체 보통 보통 표시 한국어 문자 텍스트 Han Yang I & C Co. Ltd.
돋움 보통 보통 표시 한국어 문자 텍스트 Han Yang I & C Co. Ltd.
돋움체 보통 보통 표시 한국어 문자 텍스트 Han Yang I & C Co. Ltd.
맑은 고딕 보통; 가늘게; 굵게 표시 한국어 문자 텍스트 Microsoft Corporation
문체부 궁체 정자체 보통 보통 표시
문체부 궁체 흘림체 보통 보통 표시
문체부 돋음체 보통 보통 표시
문체부 바탕체 보통 보통 표시
문체부 쓰기 정체 보통 보통 표시
문체부 쓰기 흘림체 보통 보통 표시
문체부 제목 돋음체 보통 보통 표시
문체부 제목 바탕체 보통 보통 표시
문체부 훈민정음체 보통 보통 표시
바탕 보통 보통 표시 한국어 문자 텍스트 Han Yang I & C Co. Ltd.
바탕체 보통 보통 표시 한국어 문자 텍스트 Han Yang I & C Co. Ltd.
새굴림 보통 보통 표시 한국어 문자 텍스트 HanYang System Co., LTD.
안상수2006가는 보통 보통 표시
안상수2006굵은 보통 보통 표시
안상수2006중간 보통 보통 표시
342개 항목
참고 항목
텍스트 서비스 및 입력 언어

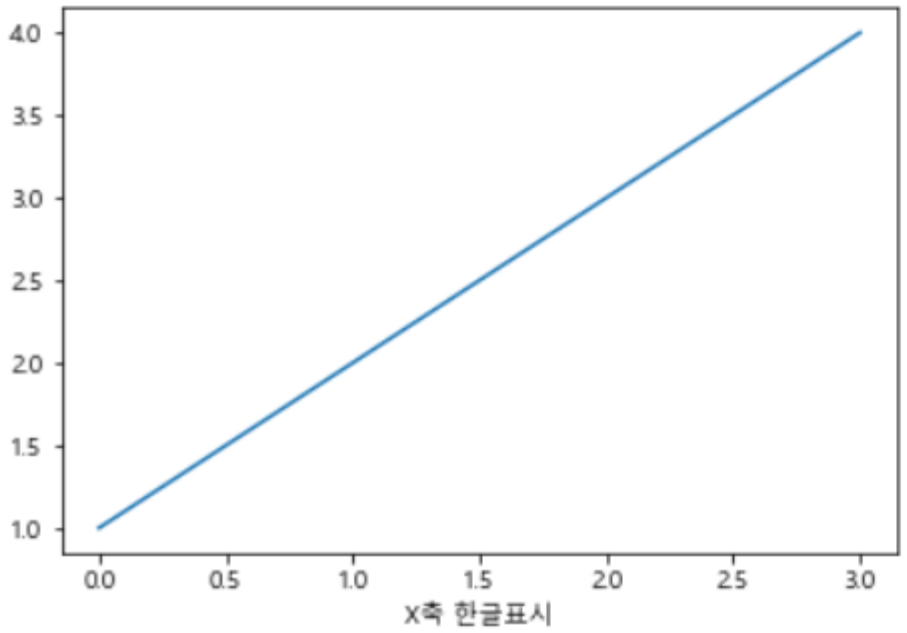
한글처리


맑은 고딕
내 PC › Windows Disk (C:) › Windows › Fonts › 맑은 고딕
맑은 고딕 검색
제어판 홈
글꼴 설정
온라인에서 글꼴 정보 가져오기
ClearType 텍스트 조정
문자 찾기
모든 언어에 대한 글꼴 다운로드
글꼴 패밀리 세부 정보
이 글꼴 패밀리의 개별 글꼴을 미리 보
구성
미리 보기
삭제
한글
맑은 고딕 가늘게
한글
맑은 고딕 굵게
malgunbd.ttf 속성
일반
보안
자세히
이전 버전
malgunbd.ttf
파일 형식: 트루타입 글꼴 파일(.ttf)
연결 프로그램: Windows 글꼴 뷰어
변경(C)...
위치: C:₩Windows₩Fonts
크기: 12.0MB (12,598,360 바이트)
디스크 할당 크기: 12.0MB (12,599,296 바이트)
만든 날짜: 2018년 4월 12일 목요일, 오전 8:34:15
수정한 날짜: 2018년 4월 12일 목요일, 오전 8:34:15
액세스한 날짜: 2018년 4월 12일 목요일, 오전 8:34:15
특성: 읽기 전용(R) 숨김(H)
고급(D)...
확인
취소
적용(A)
참고 항목
텍스트 서비스 및 입력 언어
한글
맑은 고딕 굵게 글꼴 스타일: 굵게
표시/숨기기: 표시
디자인 용도: 한국어 문자
범주: 텍스트
디자이너/제작 업체: Microsoft Corporation
글꼴 포함 가능성: 편집 가능

한글처리

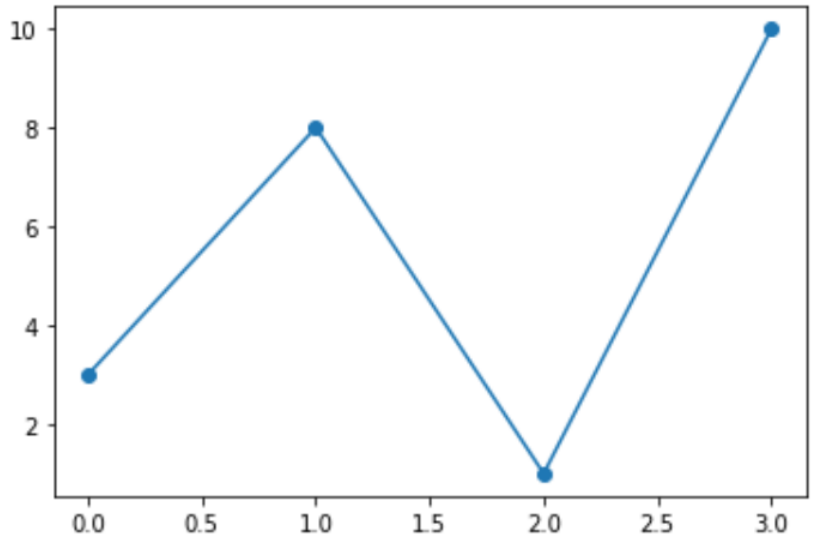
In [10]:

```
from matplotlib import font_manager, rc
import matplotlib
font_location="c:/Windows/fonts/malgunbd.ttf"
font_name = font_manager.FontProperties(fname=font_location).get_name()
matplotlib.rc('font', family=font_name)
plt.plot([1,2,3,4])
plt.xlabel("X축 한글표시")
plt.show()
```



Marker


```
[39]  1 ypoints = np.array([3, 8, 1, 10])
      2
      3 plt.plot(ypoints, marker = 'o')
      4 #plt.plot(ypoints, marker = '*')
      5 plt.show()
```

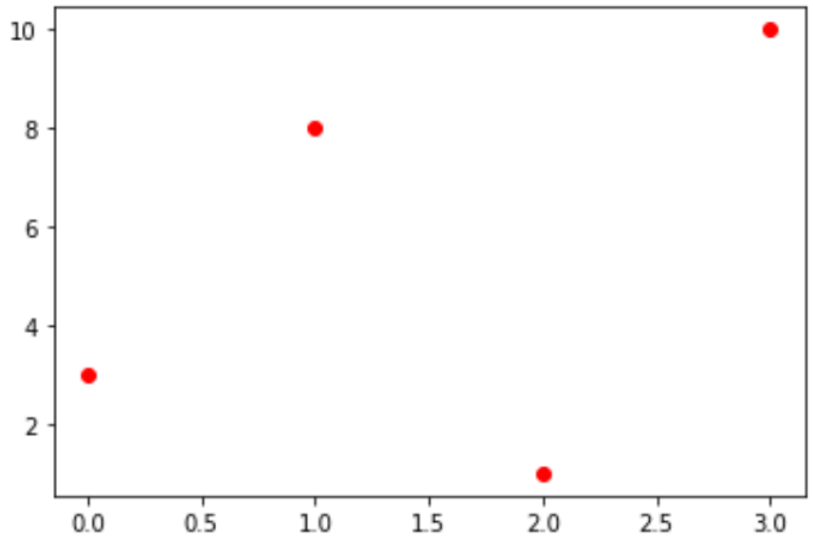


Marker

Marker	Description	Marker	Description
'o'	Circle	'H'	Hexagon
'*'	Star	'h'	Hexagon
'.'	Point	'v'	Triangle Down
','	Pixel	'^'	Triangle Up
'x'	X	'<'	Triangle Left
'X'	X (filled)	'>'	Triangle Right
'+'	Plus	'1'	Tri Down
'P'	Plus (filled)	'2'	Tri Up
's'	Square	'3'	Tri Left
'D'	Diamond	'4'	Tri Right
'd'	Diamond (thin)	'|'	Vline
'p'	Pentagon	'_'	Hline

Marker

[45]
```
1 # 기본값 : 파란색(b) 라인(-)
2 ypoints = np.array([3, 8, 1, 10])
3
4 plt.plot(ypoints, 'or')
5 #plt.plot(ypoints, 'o:r')
6 plt.show()
```



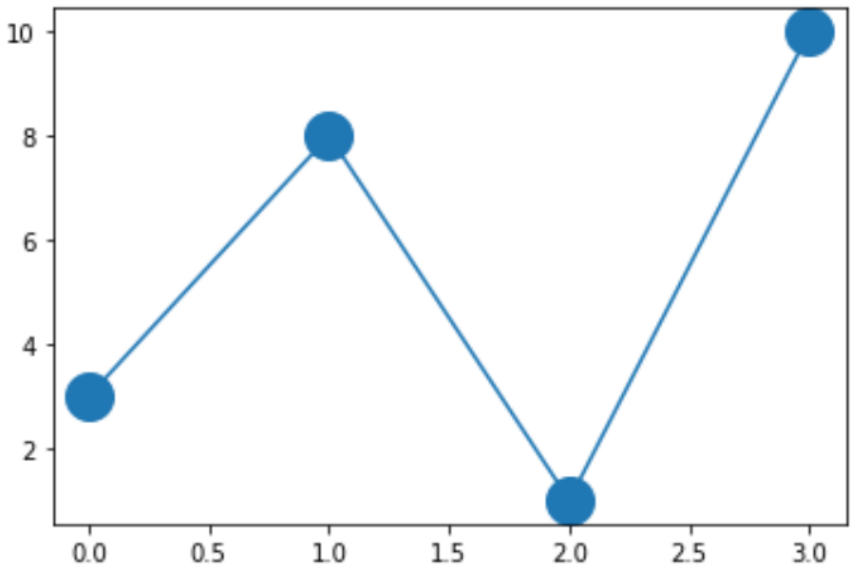
Marker

Line Syntax	Description
'-'	Solid line
':'	Dotted line
'--'	Dashed line
'-.'	Dashed/dotted line

Color Syntax	Description
'r'	Red
'g'	Green
'b'	Blue
'c'	Cyan
'm'	Magenta
'y'	Yellow
'k'	Black
'w'	White

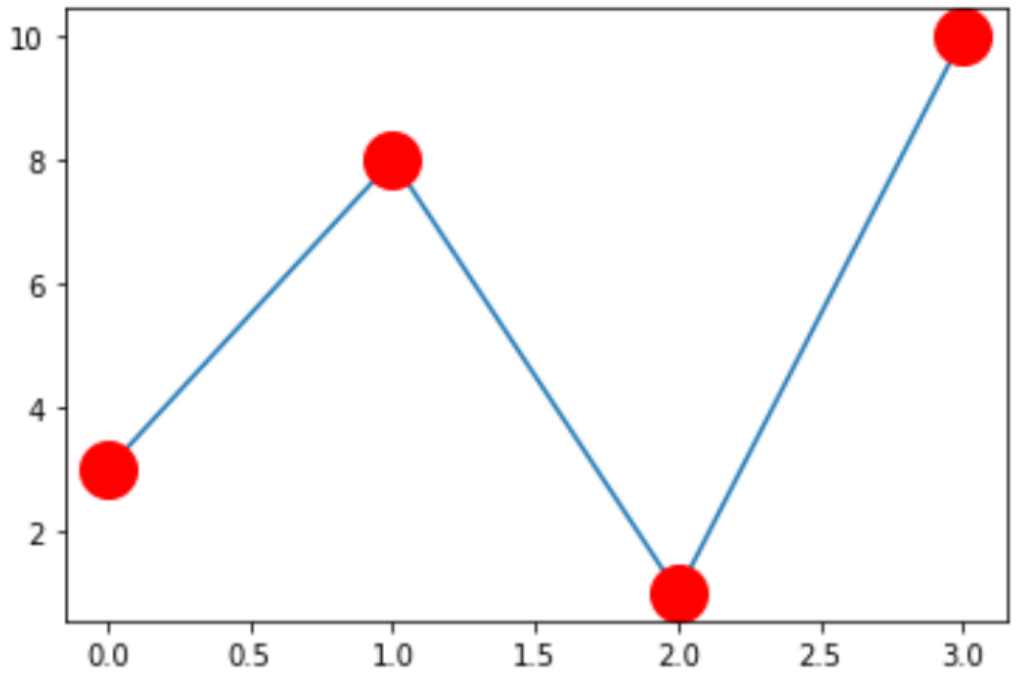
Marker


```
[46]  1 ypoints = np.array([3, 8, 1, 10])
      2
      3 plt.plot(ypoints, marker = 'o', ms = 20) # marker size
      4 #plt.plot(ypoints, marker = 'o', ms = 20, mec = 'r') # marker edge color
      5 #plt.plot(ypoints, marker = 'o', ms = 20, mfc = 'r') # marker face color
      6 plt.show()
```



Marker

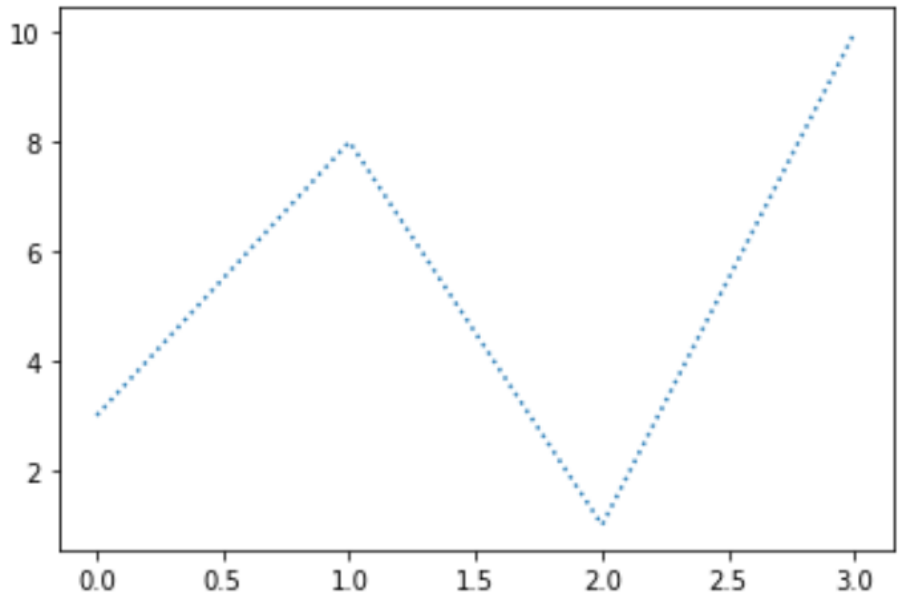

```
[52]  1 ypoints = np.array([3, 8, 1, 10])
      2
      3 plt.plot(ypoints, marker = 'o', ms = 20, mec = 'r', mfc = 'r')
      4 #plt.plot(ypoints, marker = 'o', ms = 20, mec = '#4CAF50', mfc = '#4CAF50')
      5 #plt.plot(ypoints, marker = 'o', ms = 20, mec = 'hotpink', mfc = 'hotpink')
      6 plt.show()
```



Line

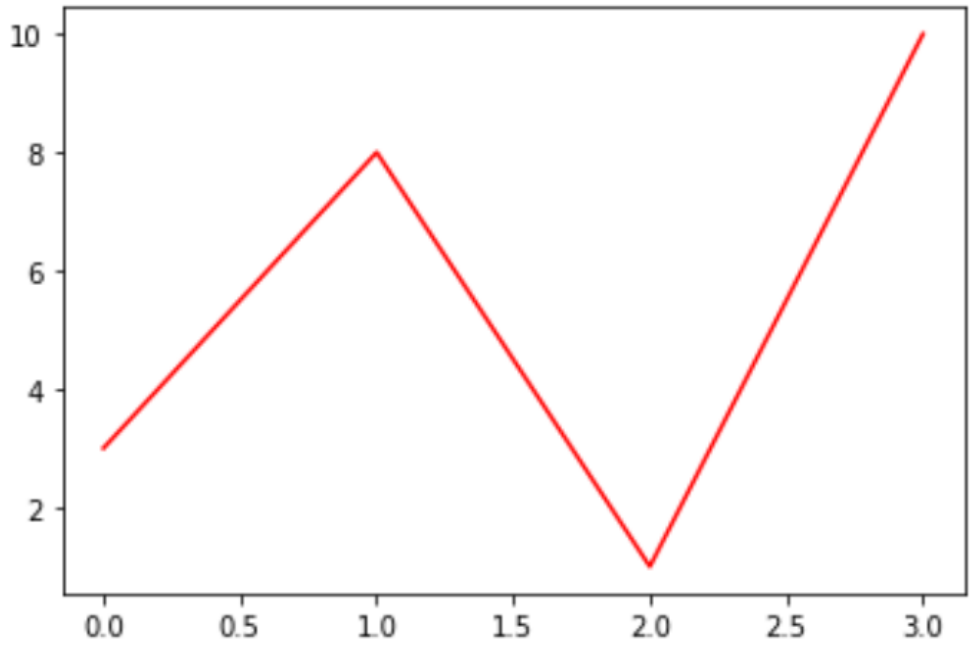

```
[5]  1 import matplotlib.pyplot as plt
     2 import numpy as np
     3
     4 ypoints = np.array([3, 8, 1, 10])
     5
     6 plt.plot(ypoints, linestyle = 'dotted')
     7 #plt.plot(ypoints, linestyle = 'dashed')
     8 plt.show()
```

Style	Or
'solid' (default)	'-'
'dotted'	':'
'dashed'	'--'
'dashdot'	'-.'
'None'	'' or ' '



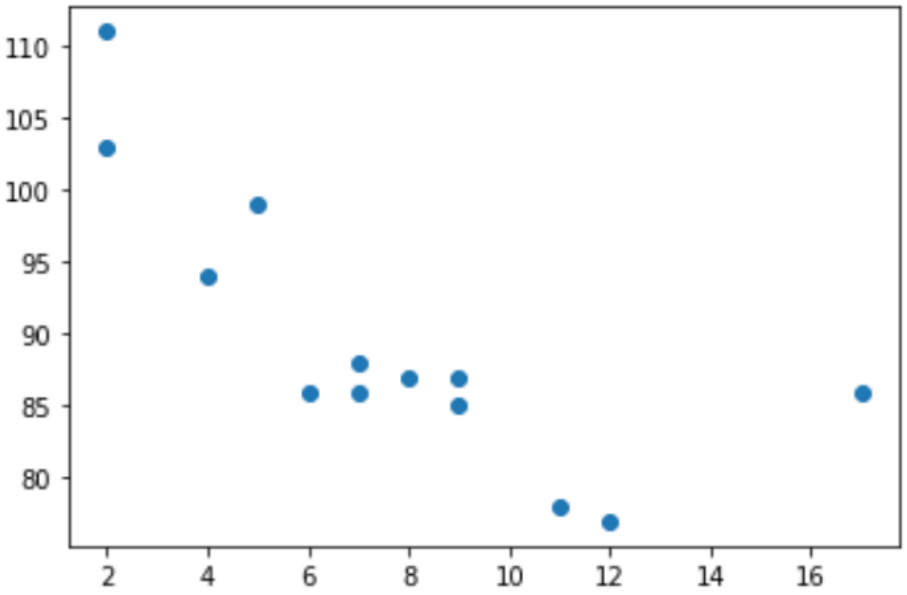
Line Color

[53]
```
1 ypoints = np.array([3, 8, 1, 10])
2
3 plt.plot(ypoints, color = 'r') # line color
4 # plt.plot(ypoints, c = '#4CAF50') # line color
5 # plt.plot(ypoints, c = 'hotpink') # line color
6 # plt.plot(ypoints, linewidth = '20.5') # line width
7 plt.show()
```



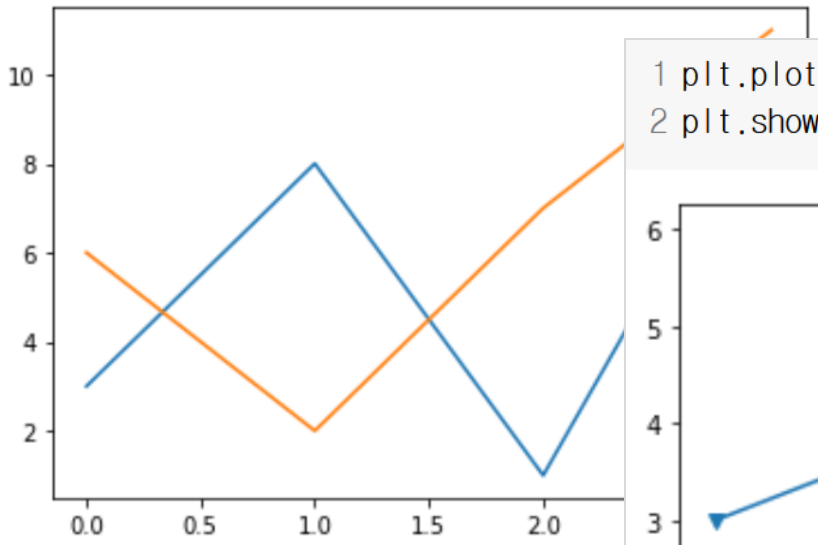
scatter


```
[6]  1 x = np.array([5,7,8,7,2,17,2,9,4,11,12,9,6])
     2 y = np.array([99,86,87,88,111,86,103,87,94,78,77,85,86])
     3
     4 plt.scatter(x, y)
     5 plt.show()
```

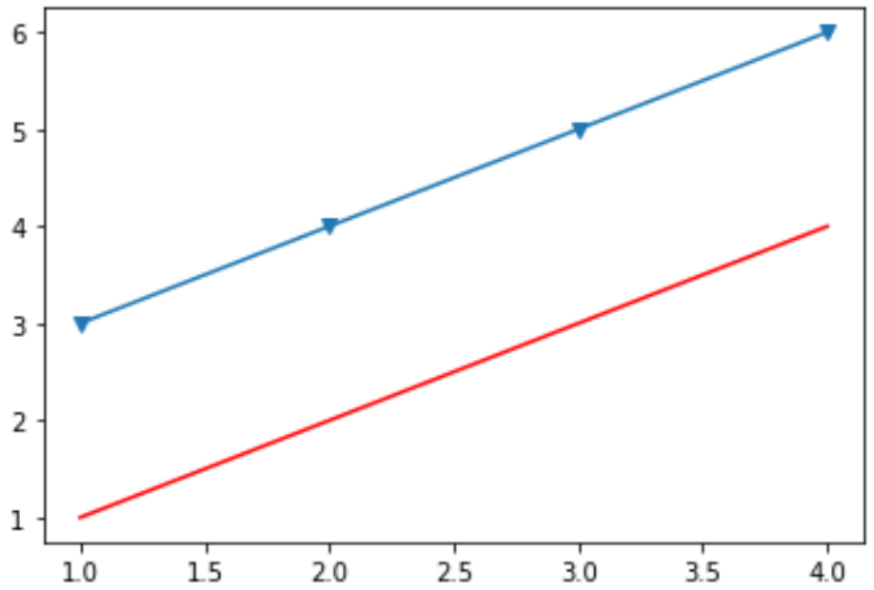


Multiple Lines


```
1 x1 = np.array([0, 1, 2, 3])
2 y1 = np.array([3, 8, 1, 10])
3 x2 = np.array([0, 1, 2, 3])
4 y2 = np.array([6, 2, 7, 11])
5
6 plt.plot(x1, y1, x2, y2)
7 plt.show()
```

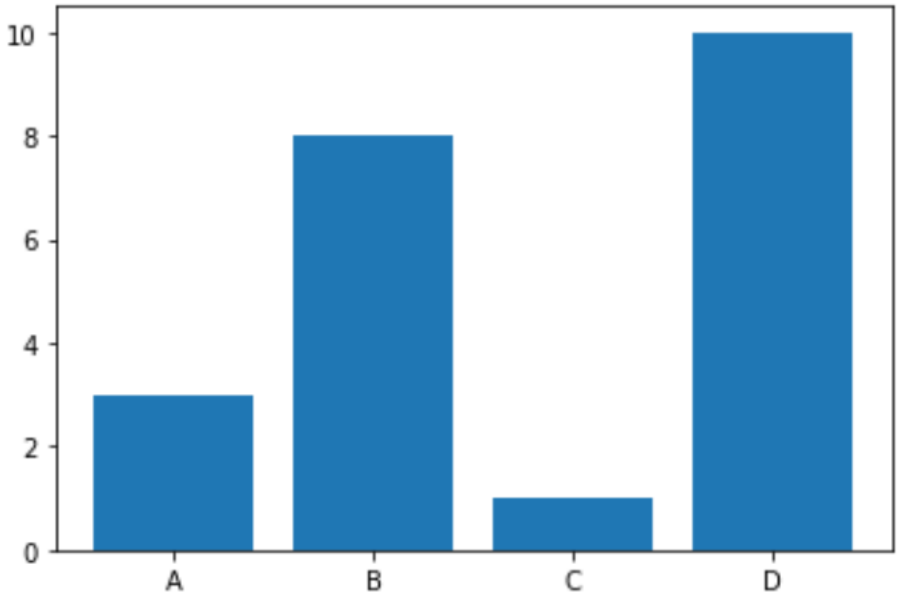



```
1 plt.plot([1,2,3,4],[1,2,3,4],'r-',[1,2,3,4],[3,4,5,6], 'v-')
2 plt.show()
```



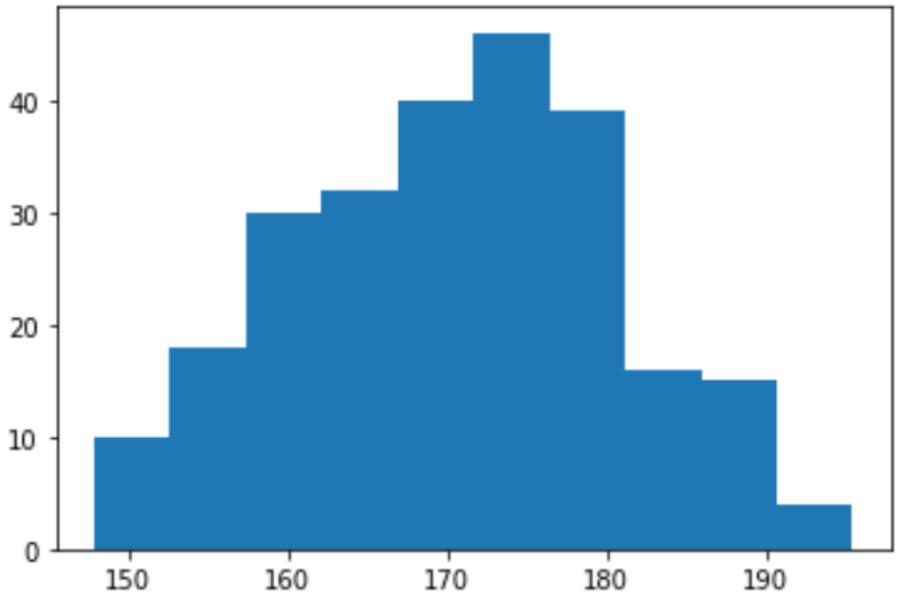
bar


```
[7]  1 x = np.array(["A", "B", "C", "D"])
     2 y = np.array([3, 8, 1, 10])
     3
     4 plt.bar(x,y)
     5 plt.show()
```



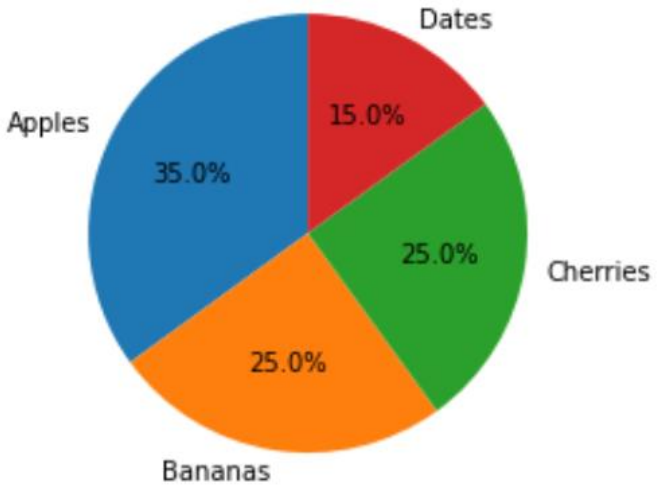
hisograms

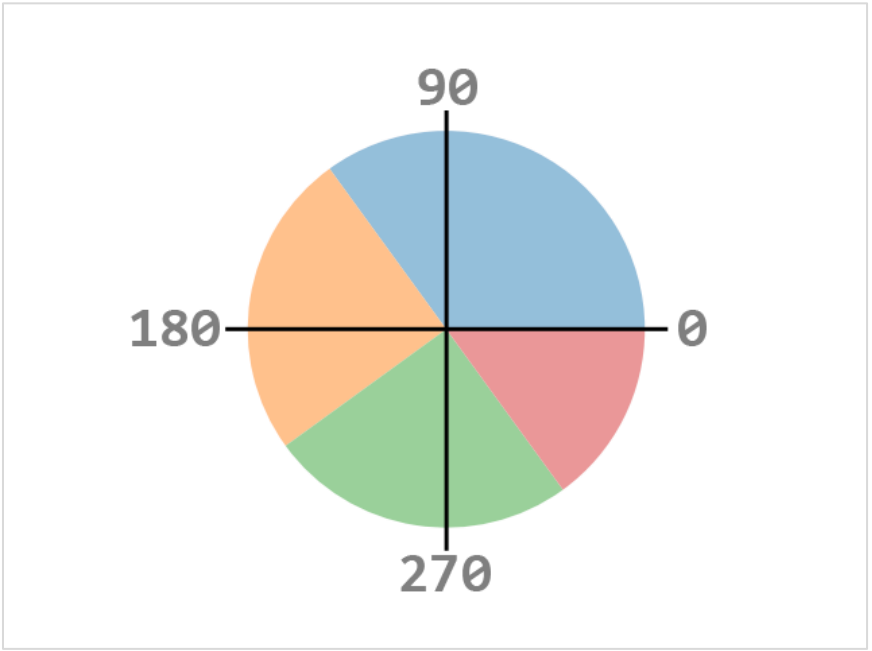

```
[9]  1 x = np.random.normal(170, 10, 250)
     2
     3 #print(x)
     4 plt.hist(x)
     5 plt.show()
```



pie


```
[36]  1 y = np.array([35, 25, 25, 15])
      2 labels = ['Apples', 'Bananas', 'Cherries', 'Dates']
      3
      4 plt.pie(y, labels = mylabels, autopct='%.1f%%', startangle = 90)
      5 plt.show()
```

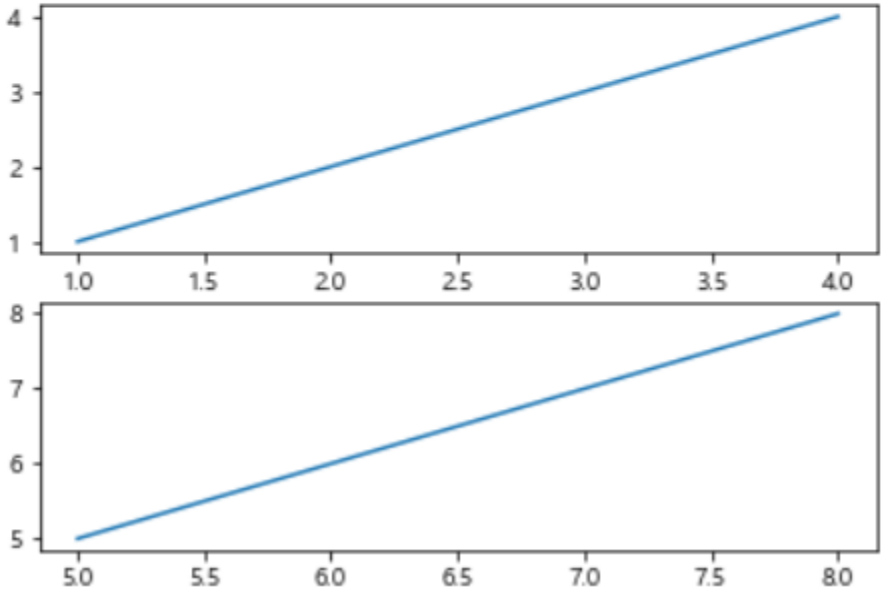




subplot

In [15]:

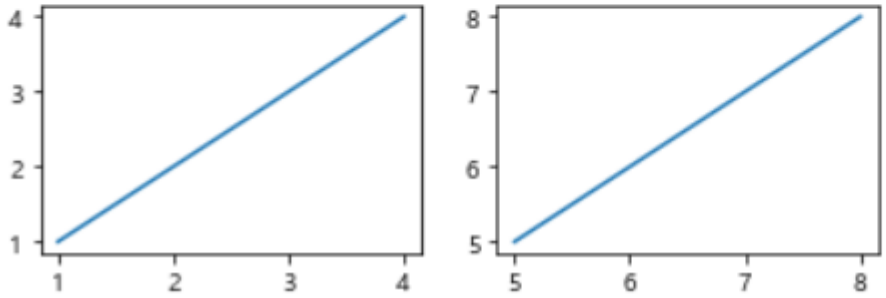
```
plt.figure()  # 하나의 캔버스를 생성
# subplot(m,n,idx)
# 매트릭스 형태로 행2 열1개인 창을 의미. idx는 mn형태의 idx번째
plt.subplot(2,1,1)
plt.plot([1,2,3,4],[1,2,3,4])
plt.subplot(2,1,2)
plt.plot([5,6,7,8],[5,6,7,8])
plt.show()
```



subplot

In [18]:

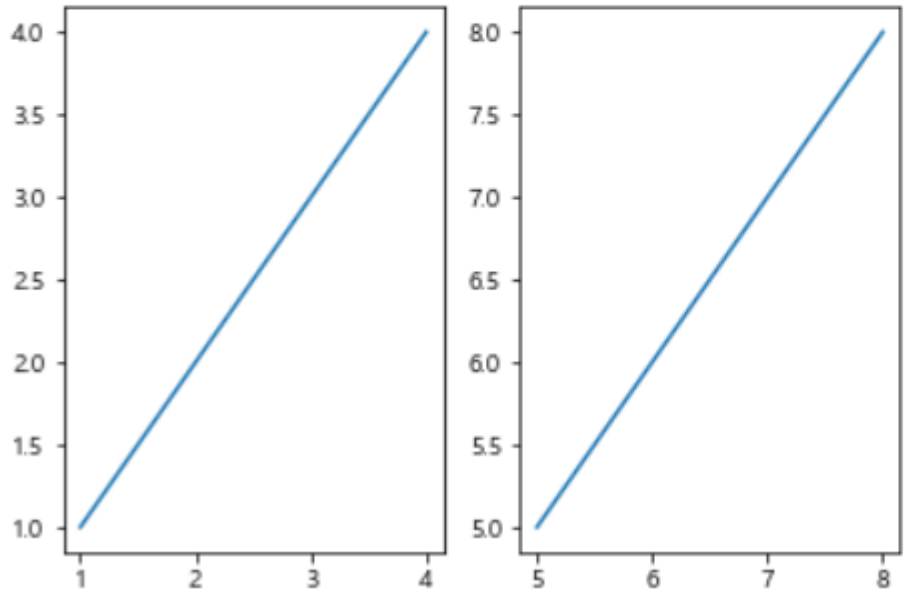
```
1 plt.figure()
2 plt.subplot(2,2,1)
3 plt.plot([1,2,3,4],[1,2,3,4])
4 plt.subplot(2,2,2)
5 plt.plot([5,6,7,8],[5,6,7,8])
6 plt.show()
```



subplot

In [19]:

```
1  plt.figure()
2  plt.subplot(1,2,1) # 1행의 첫 번째 컬럼
3  plt.plot([1,2,3,4],[1,2,3,4])
4  plt.subplot(1,2,2) # 1행의 두 번째 컬럼
5  plt.plot([5,6,7,8],[5,6,7,8])
6  plt.show()
```



subplot


```
x = np.array([0, 1, 2, 3])
y = np.array([3, 8, 1, 10])

plt.subplot(2, 3, 1)
plt.plot(x,y)

x = np.array([0, 1, 2, 3])
y = np.array([10, 20, 30, 40])

plt.subplot(2, 3, 2)
plt.plot(x,y)

x = np.array([0, 1, 2, 3])
y = np.array([3, 8, 1, 10])

plt.subplot(2, 3, 3)
plt.plot(x,y)

x = np.array([0, 1, 2, 3])
y = np.array([10, 20, 30, 40])

plt.subplot(2, 3, 4)
plt.plot(x,y)
```

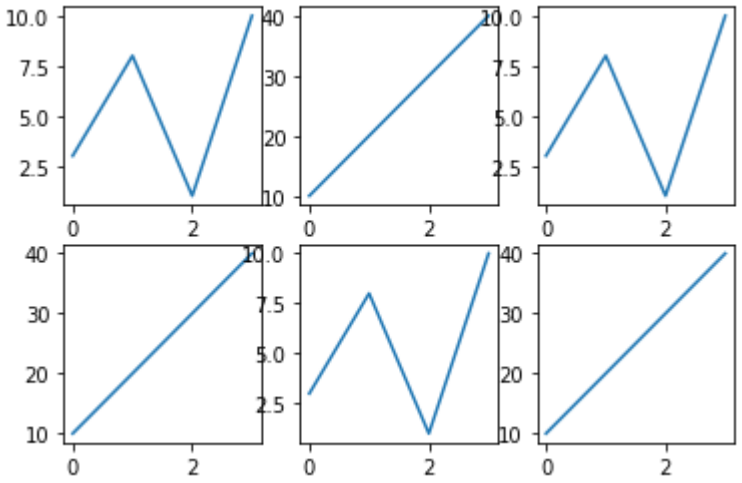
```
x = np.array([0, 1, 2, 3])
y = np.array([3, 8, 1, 10])

plt.subplot(2, 3, 5)
plt.plot(x,y)

x = np.array([0, 1, 2, 3])
y = np.array([10, 20, 30, 40])

plt.subplot(2, 3, 6)
plt.plot(x,y)

plt.show()
```




Numpy
Matplotlib
Pandas

Pandas

- Powerful Python data analysis toolkit
- BSD-licensed library providing high-performance, easy-to-use data structures and data analysis tools for the Python programming language.
- The two primary data structures of pandas, Series (1-dimensional) and DataFrame (2-dimensional), handle the vast majority of typical use cases in finance, statistics, social science, and many areas of engineering.
- Ref : https://pandas.pydata.org

Series(1-dimensional)

- 인덱스(레이블)를 가지는 동일한 데이터형의 1차원 데이터
- 레이블 또는 데이터의 위치를 지정한 추출가능.
- 인덱스에 대한 슬라이스가 가능
- 산술 연산이 가능. 통계량을 산출하는 장점을 가지고 있음

Index	Data
1	'A'
2	'B'
3	'C'
4	'D'
5	'E'

Series


```
import numpy as np
import pandas as pd
```

```
s = pd.Series(np.random.randn(5))
s
```

```
0   -0.086872
1    0.260547
2    0.012375
3   -1.185436
4   -0.985884
dtype: float64
```

```
s = pd.Series(np.random.randn(5), index=['A','B','C','D','E'])
s
```

```
A    1.172776
B    0.295672
C   -1.450593
D    0.394875
E    1.411907
dtype: float64
```

Series


```
# dictionary
d = {'a' : 0., 'b' : 1., 'c' : 2.}
pd.Series(d)
```

```
a    0.0
b    1.0
c    2.0
dtype: float64
```

```
pd.Series(d, index=['a', 'b', 'B', 'c'])
```

```
a    0.0
b    1.0
B    NaN
c    2.0
dtype: float64
```

```
# 스칼라값
pd.Series(7, index=['a', 'b', 'c', 'd', 'e'])
```

```
a    7
b    7
c    7
d    7
e    7
dtype: int64
```

Series


```
s = pd.Series([1,2,3,4,5], index=['a', 'b', 'c', 'd', 'e'])
s[0]
```

```
1
```

```
s[:3]
```

```
a    1
b    2
c    3
dtype: int64
```

```
s[[4,1]]
```

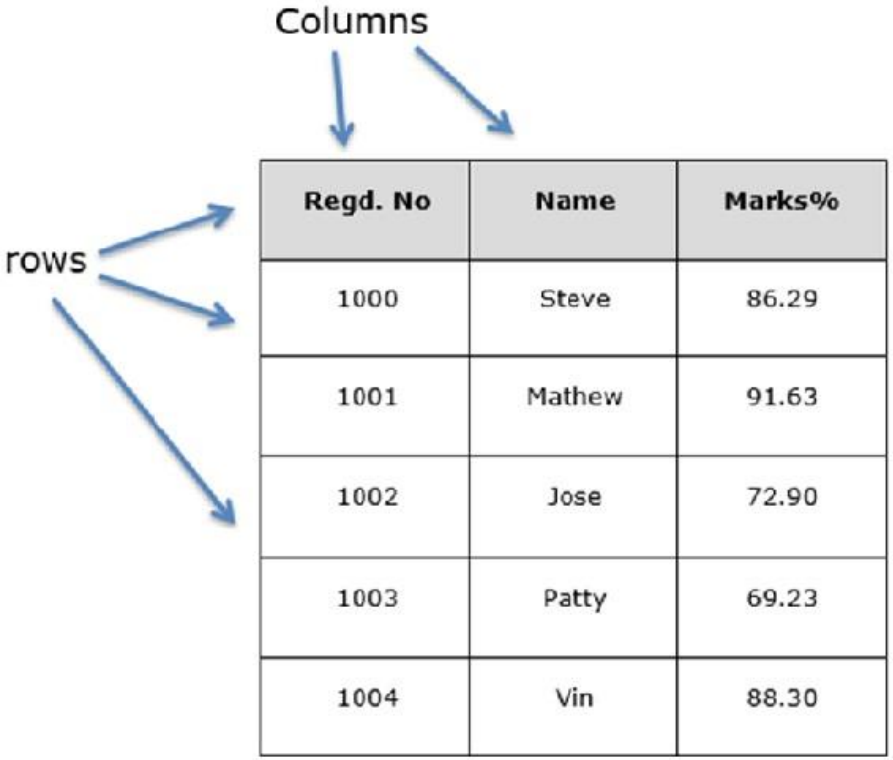
```
e    5
b    2
dtype: int64
```

```
np.power(s, 2)
```

```
a     1
b     4
c     9
d    16
e    25
dtype: int64
```

DataFrame(2-dimensional)

- 행과 열에 레이블을 가진 2차원 데이터
- 열마다 다른 형태를 가질 수 있음
- 테이블형 데이터에 대해 불러오기, 데이터 쓰기가 가능
- DataFrame끼리 여러 가지 조건을 사용한 결합 처리가 가능
- 크로스 집계가 가능



DataFrame


```
# Series/Dict 데이터의 활용
d = {'one' : pd.Series([1., 2., 3.], index=['a', 'b', 'c']),
     'two' : pd.Series([1., 2., 3., 4.], index=['a', 'b', 'c', 'd'])}
d
```

```
{'one': a    1.0
 b    2.0
 c    3.0
 dtype: float64, 'two': a    1.0
 b    2.0
 c    3.0
 d    4.0
 dtype: float64}
```

```
pd.DataFrame(d)
```

	one	two
a	1.0	1.0
b	2.0	2.0
c	3.0	3.0
d	NaN	4.0

```
# 인덱스 값을 부여하지 않으면 자동으로 0부터 두개의 데이터 중 큰 배열의 길이 - 1 만큼이 부여
d = {'one' : pd.Series([1., 2., 3.]), 'two' : pd.Series([1., 2., 3., 4.])}
pd.DataFrame(d)
```

	one	two
0	1.0	1.0
1	2.0	2.0
2	3.0	3.0
3	NaN	4.0

DataFrame


```
d = {'one' : [1., 2., 3., 4.], 'two' : [4., 3., 2., 1.]}
pd.DataFrame(d)
```

	one	two
0	1.0	4.0
1	2.0	3.0
2	3.0	2.0
3	4.0	1.0

```
# Dict 리스트 데이터의 활용
data2 = [{'a': 1, 'b': 2}, {'a': 5, 'b': 10, 'c': 20}]
pd.DataFrame(data2)
```

	a	b	c
0	1	2	NaN
1	5	10	20.0

```
pd.DataFrame(data2, index=['first', 'second'])
```

	a	b	c
first	1	2	NaN
second	5	10	20.0

DataFrame


```
pd.DataFrame(data2, columns=['a', 'b'])
```

	a	b
0	1	2
1	5	10

```
df = pd.DataFrame(data2, columns=['a', 'b'])
df.rename(columns={'a':'COL1'})
```

	COL1	b
0	1	2
1	5	10

```
df.set_index('b')
```

	a
b	
2	1
10	5

DataFrame


```
# 데이터 추가 및 합치기(merge)
data1 = [{'name':'Mark'},{'name':'Eric'},{'name':'Jennifer'}]
df = pd.DataFrame(data1)
df
```

	name
0	Mark
1	Eric
2	Jennifer

```
df['age'] = [10, 11, 12]
pd.DataFrame(data1)
df
```

	name	age
0	Mark	10
1	Eric	11
2	Jennifer	12

DataFrame


```
data2 = [{'sido':'서울'}, {'sido':'경기'}, {'sido':'인천'}]
df2 = pd.DataFrame(data2)
df2
```

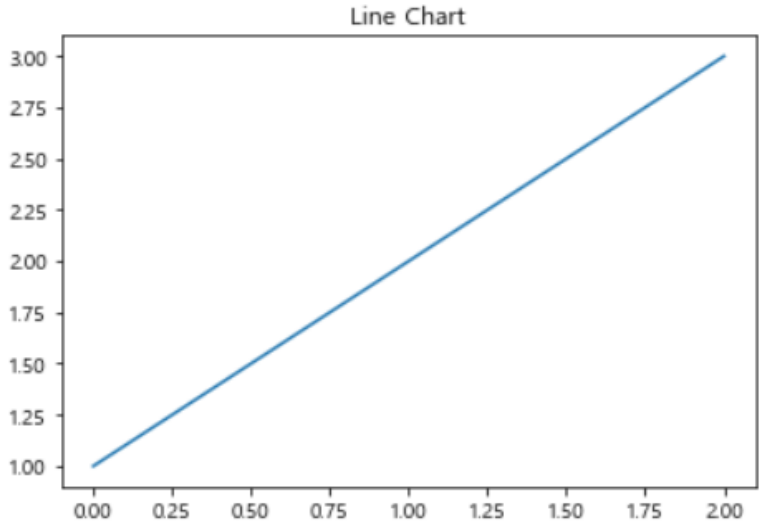
	sido
0	서울
1	경기
2	인천

```
pd.merge(df, df2, left_index=True, right_index=True)
```

	name	age	sido
0	Mark	10	서울
1	Eric	11	경기
2	Jennifer	12	인천

데이터 시각화


```python
# Series에서 시각화
s = pd.Series([1,2,3])
ax = s.plot()
ax.set_title('Line Chart')
plt.show()
```




```python
# DataFrame에서 시각화
df = pd.DataFrame({'a':[1,2,3], 'b':[3,2,1]})
ax = df.plot()
ax.set_title('Line Chart')
plt.show()
```