Matplotlib

데이터분석 필수 패키지

1) Matplotlib
2) Numpy
3) Pandas

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

1. Matplotlib 개념

막대 그래프, box,선, 산점도, 히스토그램 등 기본적인 통계 그래프 작성

데이터를 지도에 매핑하는 basemap 및 cartopy

3D 그리프를 만드는 mplot3d 도구 제공

모양, 크기, 축의 범위와 단위, 눈금과 레이블, 범례 및 제목등 지정가능

1. Matplotlib 개념

막대 그래프(bar)
- 범주형 데이터 수치 표시
- 가로,세로 누적 등 그룹화된 데이터를 그래프로 사용

히스토그램(hist)
- 수치형 데이터의 분포를 표시
- 빈도수, 빈도밀도,확률,확률밀도등의 분포 표시할때 사용

선그래프(plot)
- 수치의 변화를 선으로 표시
- 보통 시간에 따른 데이터 변화 추세를 보여줌

산점도(plot)
- 키와 몸무게, 수요와 공급 등 두변수간의 관계 표현시 사용
- 두변수가 양의 상관관계이지, 음의 상관관계인지 파악 가능

상자그림(boxplot)
- 데이터에서 얻은 5가지 통계량을 가지고 만듬
- 5가지 통계량
  - 최솟값, 1사분위수, 2사분위수(중앙값),3사분위수, 최댓값
  - 상자 맨아래에는 제 1사분위수,맨위에는 3사분위수, 상자 중간에 중앙값, 상자위아래의 선은 최솟갓,최댓값을 표시.

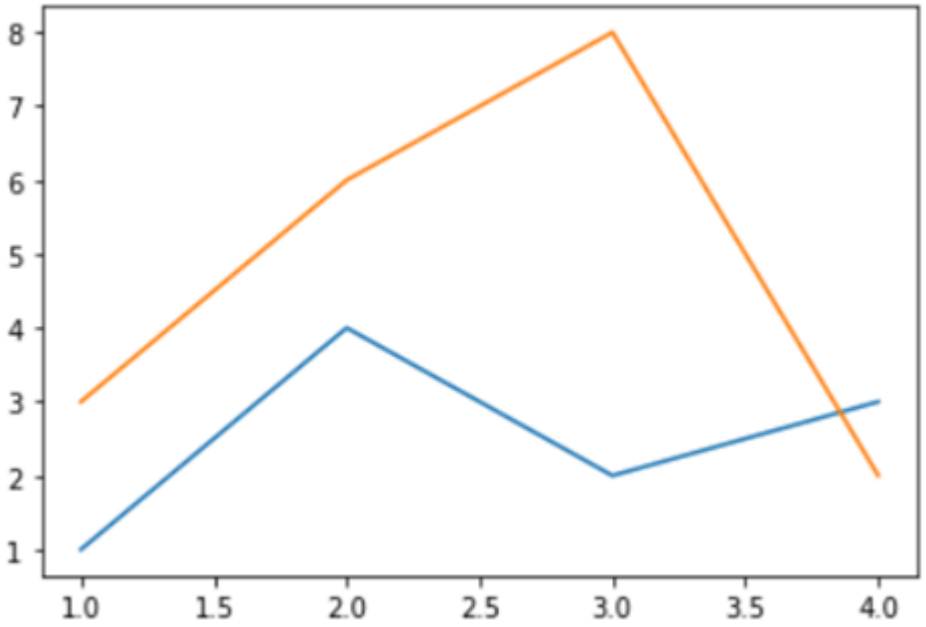
1. Matplotlib

Mapplotlib는 Figures에 데이터를 그래프로 표시

pyplot.subplots : 데이터를 그래프로 표시하는 단위

Axes.plot : 축이 있는 데이터를 그릴 수 있음.

```
import matplotlib.pyplot as plt
fig,ax =plt.subplots()
ax.plot([1,2,3,4],[1,4,2,3])
ax.plot([1,2,3,4],[3,6,8,2])
plt.show()
```



1. Matplotlib

플로팅 함수에 대한 입력 유형

- 모든 플로팅 함수는 예상 numpy.array 하거나, numpy.masked_array 입력으로 사용 됨
- Pandas 데이터 객체와 같이 배열과 유사하고 numpy.matrix 로 작동 가능 할 수도 있고, 아닐 수도 있는 numpy.array로 플로팅 하기 전에 이를 객체로 변환하는 것이 좋음

```
import pandas
a = pandas.DataFrame(np.random.rand(4,5),columns=list(“abcde’))
a_asarray = a.values

b = np.matrix([[1,2],[3,4]])
b_as array = np.asarray(b)
```

1. Matplotlib

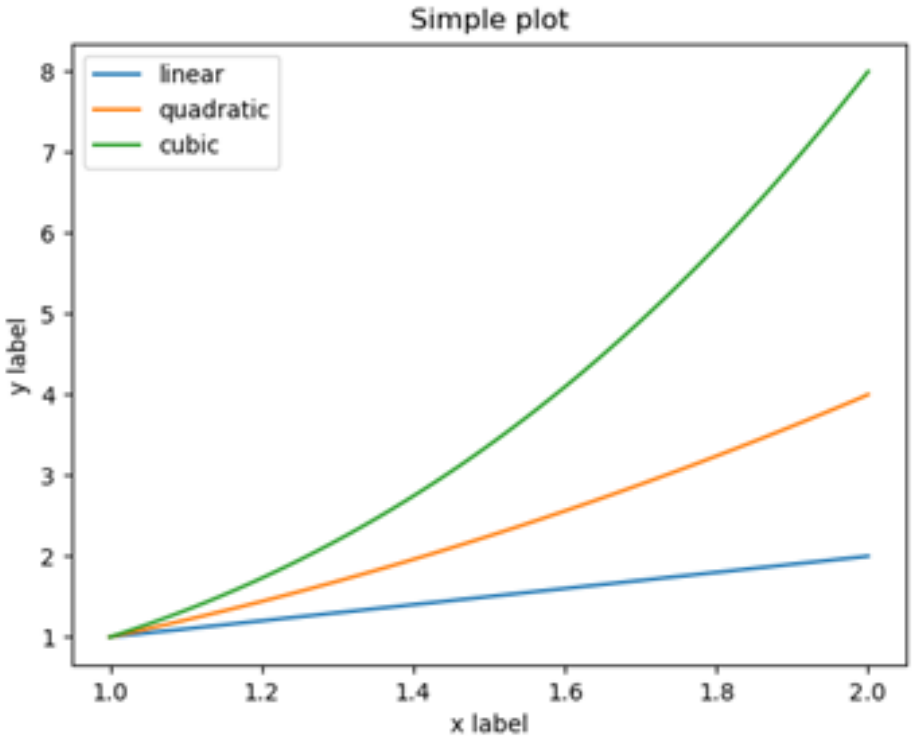
기본적으로 Matplotlib를 사용법

1. 객체지향 : 그림과 축을 명시적으로 만들고 그에 대한 메서드 호출
2. pyplot : 그림과 축을 자동으로 생성 및 관리하고 플로팅에 pyplot 함수를 사용

1. Matplotlib

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

x = np.linspace(1,2,100)
fig,ax = plt.subplots()
ax.plot(x,x,label='linear')
ax.plot(x,x**2,label='quadratic')
ax.plot(x,x**3,label='cubic')
ax.set_xlabel('x label')
ax.set_ylabel('y label')
ax.set_title('Simple plot')
ax.legend()
```



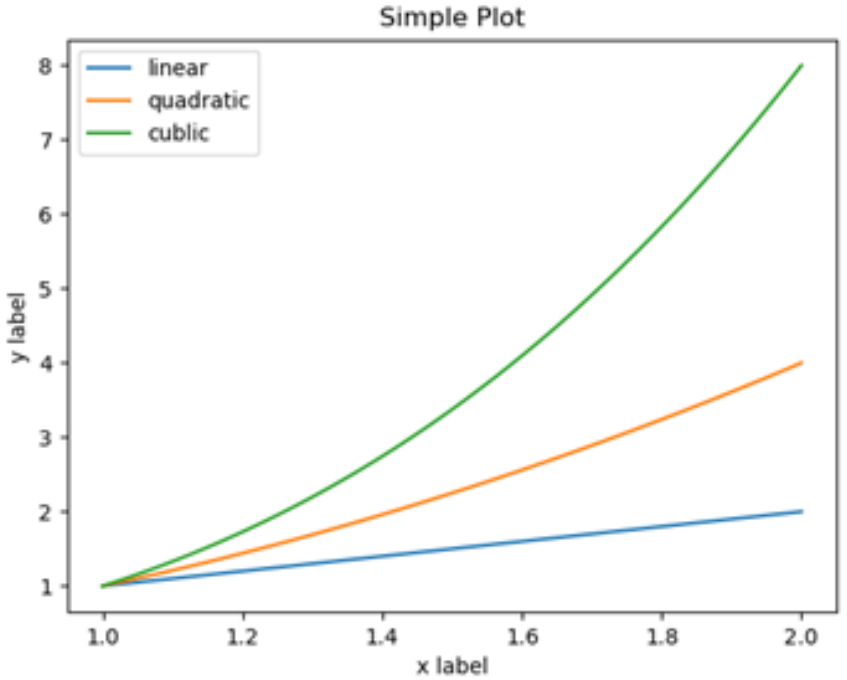
1. Matplotlib

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

x = np.linspace(1,2,100)

plt.plot(x,x,label="linear")
plt.plot(x,x**2,label="quadratic")
plt.plot(x,x**3,label="cublic")

plt.xlabel('x label')
plt.ylabel('y label')
plt.title('Simple Plot')
plt.legend()
```

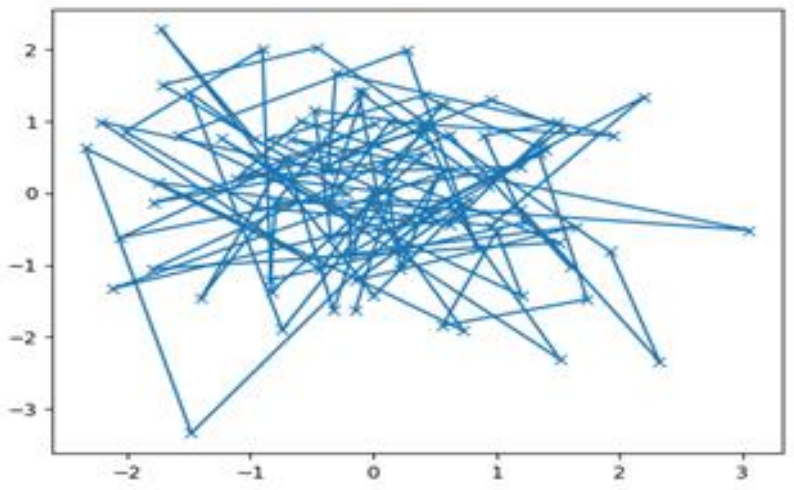


함수로 정의 하기

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

def my_plotter(ax,data1,data2,param_dict):
    out = ax.plot(data1,data2, **param_dict)
    return out

data1,data2,data3,data4 = np.random.randn(4,100)
fig,ax = plt.subplots(1,1)
my_plotter(ax,data1,data2,{"marker":"x"})
```

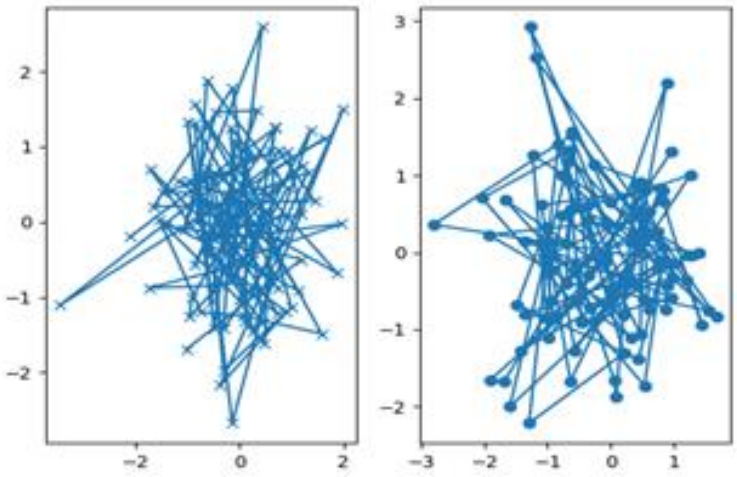


2개의 서브플롯

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

def my_plotter(ax,data1,data2,param_dict):
    out = ax.plot(data1,data2, **param_dict)
    return out

data1,data2,data3,data4 = np.random.randn(4,100)
fig,(ax1,ax2) = plt.subplots(1,2)
my_plotter(ax1,data1,data2,{"marker":"x"})
my_plotter(ax2,data3,data4,{"marker":"o"})
```

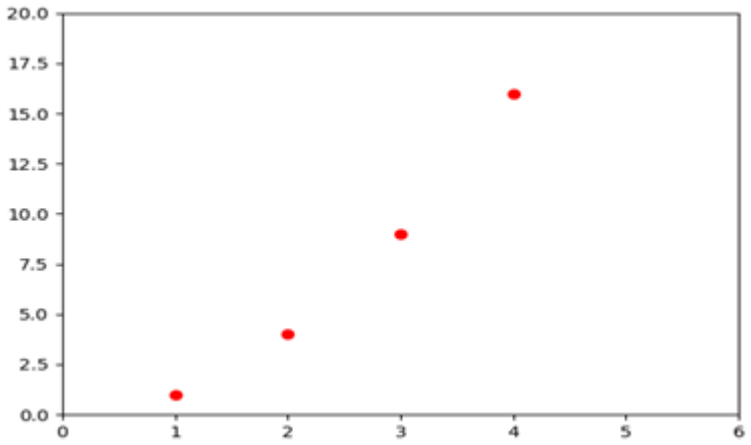


플롯 스타일 서식 지정

모든 x,y,쌍의 인수에 대해 플롯의 색상와 선 유형을 나타내는 형식 문자열인 세번째 인수로 설정.
형식 문자열의 문자와 기호는 matlab에서 가져왔으면, 색상 문자열을 선 스타일 문자열과 연결함.
기본 형식 문자열은 파란색 실선인 b-
예를 들어 빨간색 원으로 그리기위한 예제는 다음과 같다.

```
import matplotlib.pyplot as plt

plt.plot([1,2,3,4],[1,4,9,16],'ro')
plt.axis([0,6,0,20])
plt.show()
```



플롯 스타일 서식 지정

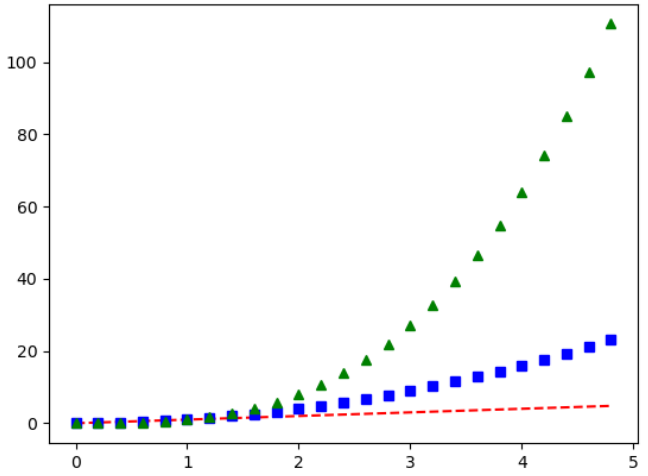
일반적으로 matplotlib는 자료 numpy 배열을 사용함.

실제로 모든 시퀀스는 내부적으로 numpy 배열로 변환

아래 예제는 배열을 사용하여 하나의 함수 호출에서 서로다른 형식 스타일로 여러 줄을 그린다.

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

t = np.arange(0.,5.,0.2)
plt.plot(t,t,"r--",t,t**2,"bs", t,t**3, "g^")
plt.show()
```

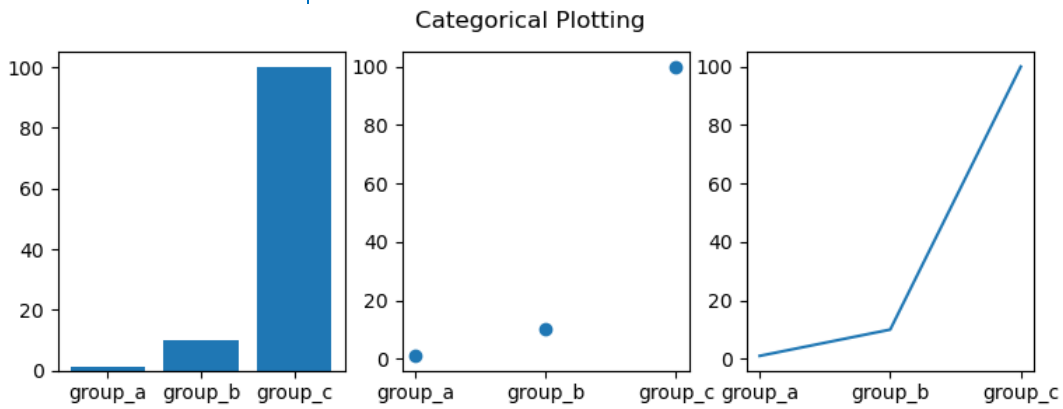


플롯 스타일 서식 지정

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

names=["group_a","group_b","group_c"]
values = [1,10,100]
plt.figure(figsize=(9,3))

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()
```

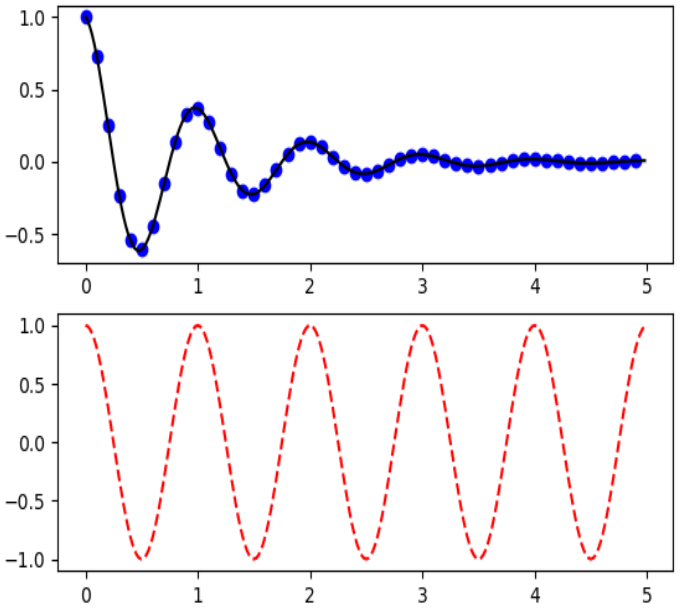


여러 도형 및 축작업

MATLAB 및 pyplot 에는 현재 Figure와 현재 좌표축 적용 . 플로팅 기능은 현재 좌표축에 적용됨

두개의 서브 플롯을 만드는 스크립트

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



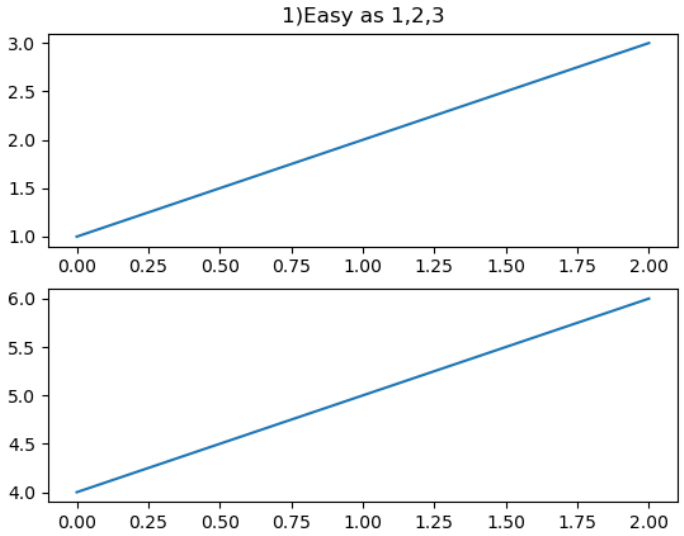
2개의 figure

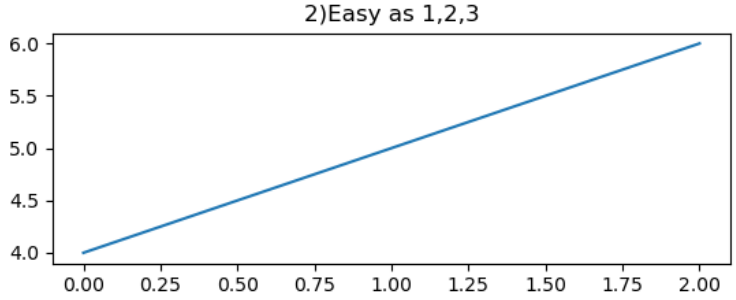
```
import numpy as np
import matplotlib.pyplot as plt


plt.figure(1)
plt.subplot(211)
plt.title("1)Easy as 1,2,3")
plt.plot([1,2,3])

plt.subplot(212)
plt.plot([4,5,6])

plt.figure(2)
plt.subplot(211)
plt.title("2)Easy as 1,2,3")
plt.plot([4,5,6])
```





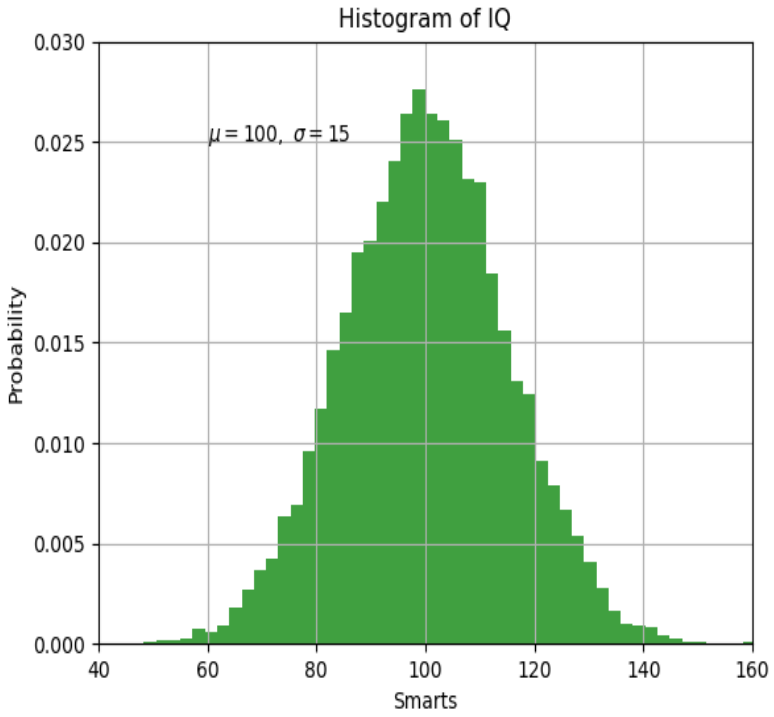
텍스트 작업

Text 임의의 위치에 텍스트를 추가 할 수 있고, xlabel, ylabel 그리고 title 표시 된 위치에 텍스트를 추가

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

mu,sigma = 100,15
x = mu + sigma * np.random.randn(10000)

n,bins, patches = plt.hist(x,50,density=1,
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()
```



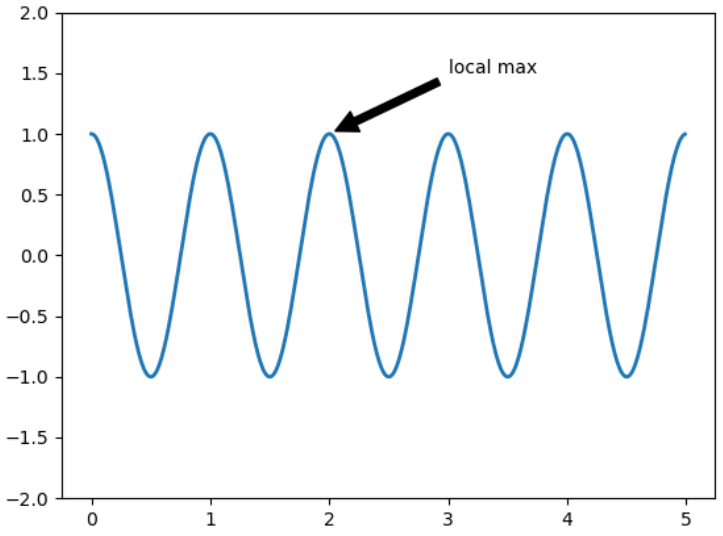
텍스트 주석 달기

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

ax = plt.subplot(111)
t = np.arange(0.0,5.0,0.01)
s = np.cos(2*np.pi*t)

line = plt.plot(t,s,lw=2)

plt.annotate("local max",xy=(2,1),xytext=(3,1.5),
        arrowprops=dict(facecolor="black",shrink
plt.ylim(-2,2)
plt.show()
```



로그 및 기타 비선형 축

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

np.random.seed(19680801)

y = np.random.normal(loc=0.5, scale=0.4, size=1000)
y=y[(y > 0) & (y < 1)]
y.sort()
x=np.arange(len(y))
plt.figure()

plt.subplot(221)
plt.plot(x,y)
plt.yscale("linear")
plt.title("linear")
plt.grid(True)
```

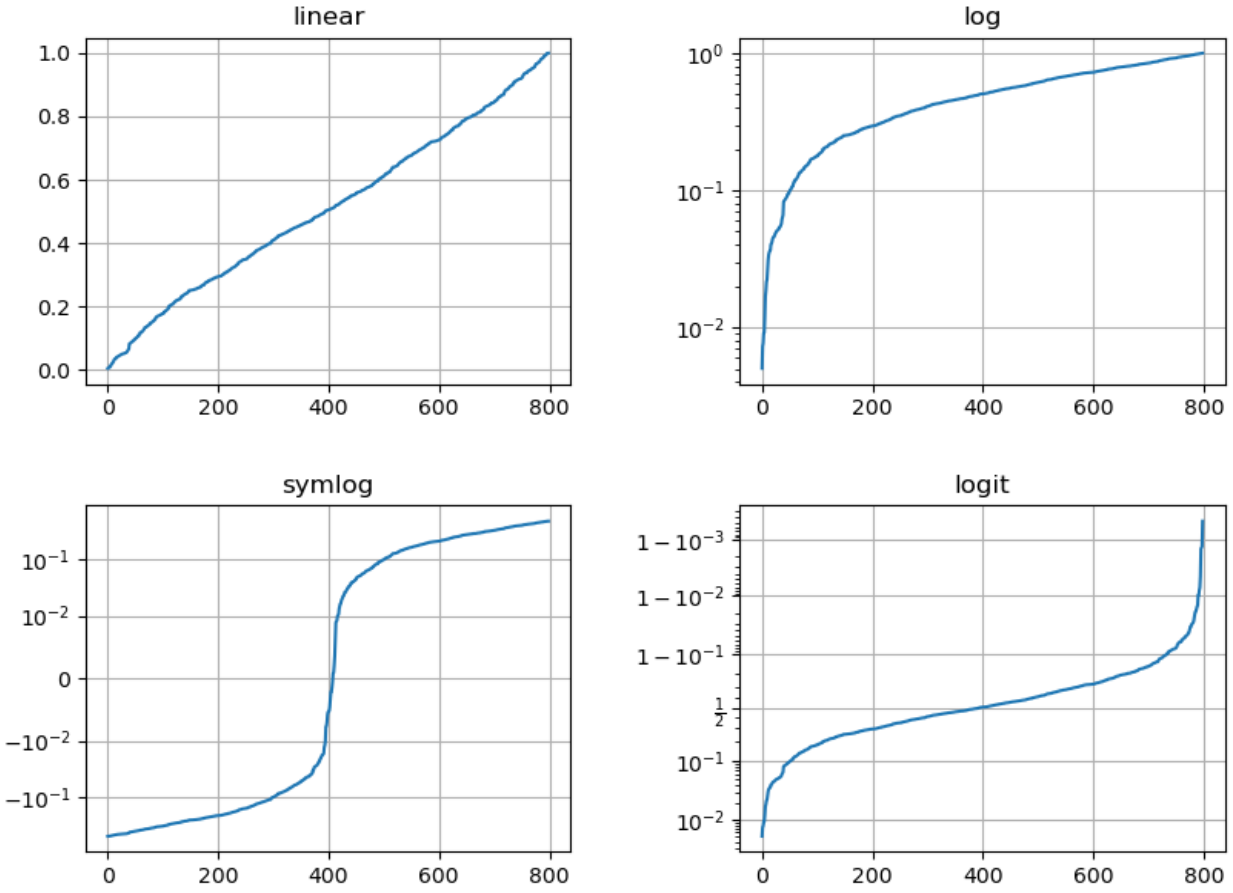
로그 및 기타 비선형 축

```
plt.subplot(222)
plt.plot(x,y)
plt.yscale("log")
plt.title("log")
plt.grid(True)

plt.subplot(223)
plt.plot(x,y-y.mean())
plt.yscale("symlog",linthresh=0.01)
plt.title("symlog")
plt.grid(True)

plt.subplot(224)
plt.plot(x,y)
plt.yscale("logit")
plt.title("logit")
plt.grid(True)
```

```
plt.subplots_adjust(top=0.92,bottom=0.08,left
=0.10,right=0.96,
                    hspace=0.35,wspace=0.35)
plt.show()
```


linear
log
symlog
logit

2. Seaborn

파이썬에서 통계그래프와 그림을 만드는 과정을 단순화 해주는 패키지.

Matplotlib 기반으로 만들어져 numpy,pandas의 자료구조 지원

Scipy 및 statsmodels 같은 통계 분석 패키지의 기능과도 통합 가능함.

히스토그램, 밀도그래프, 막대그래프,상자그래프,산점도 등 기본적인 통계 그래프 작성 기능 제공

이변량 관계, 선형 및 비선형 회귀 모형, 추정값의 불확실도등의 시각화 가능

변수를 조정해 가면서 변수 사이의 관계를 확인하고 복잡한 관계를 드러내는 격자 그래프 작성 가능

Mapplotlib 기반이므로 matplotlib 의 명령어를 사용하여 그래프 커스터마이즈 가능