Lab. Using Matplotlib

1. Matplotlib
   1)주로 2차원의 data를 시각화를 하기 위한 third-party package이다.
   2)동작하는 OS를 가리지 않는다는 점, 자세한 그리기 설정이 가능한 점, 다양한 출력 형식에 대응하고 있는 점 등 대표적인 시각화 툴로 널리 사용되고 있다.
   3)2003년 version 1.0이 발표된 이후로 15년 이상의 역사를 가진 tool이다.
   4)사용자가 많은 이유는 산업계, 교육계에서 널리 사용되고 있는 수치 해석 S/W로, MATLAB과 같은 그리기를 Python에서 사용할 수 있는 것이다.
   5)Matplotlib History : http://jakevdp.github.io/blog/2013/03/23/matplotlib-and-the-future-of-visualization-in-python/
   6)Matplotlib에서는 graph의 종류나 축, 눈금선 graph 이름 등 다양한 그림의 요소에 대해 상세한 서식(색이나 선 종류 등)을 설정할 수 있다.
   7)다양한 출력 형식(PNG, SVG, JPG 등)에 대응하고 있다.

2. Information
   1)Version : 3.1.1
   2)Site : https://matplotlib.org
   3)Repository : https://github.com/matplotlib/matplotlib
   4)PyPI : https://pypi.python.org/pypi/matplotlib/
   5)Gallery : https://matplotlib.org/gallery/index.html
   5)Installation
      $ pip install matplotlib

3. Graph 그리기 기초
   1)Graph 그리기 준비하기
      -matplotlib.pyplot module을 불러온다.

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

      -matplotlib graph를 출력할 때는 show()를 이용한다.

```
plt.show()
```

      -package import 및 기본 설정

```
import   matplotlib.pyplotas plt
%matplotlib   inline
%config InlineBackend.figure_format    =  'retina'
print("Matplotlib 버전:",    matplotlib.__version__)
---------------------------------------------------
Matplotlib  버전: 3.1.1
```

      -%matplotlib inline은 notebook을 실행한 브라우저에서 바로 그림을 볼 수 있게 해 준다.
      -%config InlineBackend.figure_format='retina' 옵션
         --('png'(기본값), 'retina', 'jpeg', 'svg', 'pdf' 중 하나)은 graph를 더 높은 해상도로 그려준다.

   2)package import 및 기본 설정이 완료되면 Matplotlib로 graph를 그리기 위해서 다음 단계를 따른다.
      a. 데이터 준비
      b. graph 생성
      c. graph 함수로 그리기
      d. graph 커스터마이징
      e. graph 출력 및 저장

      -다음 코드는 graph를 표시하는 간단한 예이다.

```
plt.plot([1, 2, 3, 4])
plt.ylabel('some  numbers')
plt.show()
```

```
import numpy as np
x = np.linspace(0, 5, 11)
y = x ** 2
x
---------------------------------------------------
array([0. , 0.5, 1. , 1.5, 2. , 2.5, 3. , 3.5, 4. , 4.5, 5. ])
y
---------------------------------------------------
array([ 0.  ,  0.25,  1.  ,  2.25,  4.  ,  6.25,  9.  , 12.25, 16.  , 20.25, 25.  ])
plt.plot(x, y)
plt.xlabel('X Label')
plt.ylabel('Y Label')
plt.title('Title')
```

```
plt.subplot(1,2,1)
plt.plot(x,y,'r')
plt.subplot(1,2,2)
plt.plot(y,x,'b')
```

4. 한글 Font 환경의 준비

1)Graph 그리기에서 한글이 깨지는 문제
-Graph를 그릴 때 caption이나 label 등에 수치 이외에 문자열을 출력하는 경우 한글을 처리할 때 종종 글자가 깨지는 문제에 부딪힌다.
-다음의 예제에서 caption의 한글이 깨지는 것을 볼 수 있다.
-이것은 Matplotlib의 초기 설정에서 사용하는 font가 한글 설정에서 포함되어 있지 않기 때문에 발생하는 문제이다.
-즉 미리 font를 설정하는 것으로 처리 가능하다.
-또한, 또 하나의 graph 작성 package인 Bokeh에서는 한글 출력이 가능하다.

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

np.random.seed(0)

x = range(5)
y = 10 + 5 * np.random.randn(5)

fig = plt.figure()
ax = fig.add_subplot(111)

ax.set_title('한글 테스트')
ax.bar(x, y)

plt.show()
```

2)한글 font 설치하기
-우리는 'Source Han Sans'를 사용할 것이다.
-이 font는 Adobe와 Google이 공동으로 개발한 한국, 중국, 일본에 사용되고 있는 문자를 이용 가능한 font family의 명칭이다.
-https://github.com/adobe-fonts/source-han-sans
-Downloads
--https://github.com/adobe-fonts/source-han-sans/tree/release
--[Language-specific OTFs]에서 [Korean (한국어) link click
--SourceHanSansK.zip
-Unzip
--SourceHanSansK-Medium.otf, SourceHansSansK-Bold.otf, SourceHansSansK-ExtraLight.otf,
--SourceHansSansK-Heavy.otf, SourceHansSansK-Light.otf, SourceHansSansK-Normal.otf
-Double click SourceHansSansK-Regular.otf
-Install SourceHansSansK-Regular.otf

3)Code 수정 후 확인

```
import os
import numpy as np
from matplotlib import pyplot as plt, font_manager

#Font cache 재구축
font_manager._rebuild()

if os.name == "nt":
    #OS가 Windows 인 경우 win32FontDirectory()를 이용할 수 있다.
    font_dir = font_manager.win32FontDirectory()

#font_path = os.path.join(font_dir, "SourceHanSansK-Regular.otf")  <--경로 못 찾음.
font_path = r'font_path = r'C:\Users\Instructor\AppData\Local\Microsoft\Windows\Fonts\SourceHanSansK-Regular.otf'
font = font_manager.FontProperties(fname=font_path, size=14)

np.random.seed(0)

x = range(5)
y = 10 + 5 * np.random.randn(5)

fig = plt.figure()
ax = fig.add_subplot(111)

#여기서 fontproperties를 지정한다.
ax.set_title('한글 테스트', fontproperties=font)
ax.bar(x, y)

plt.show()
```

5. Matplotlib 한글font 사용하기
-https://brunch.co.kr/@jade/203
-저작권 걱정 없는 무료 한글font
1)필요한 패키지를 가져오기

```
# 그래프를 노트북 안에 그리기 위해 설정
%matplotlib inline

# 필요한 패키지와 라이브러리를 가져옴
import matplotlib as mpl
import matplotlib.pyplot as plt
import matplotlib.font_manager as fm

# 그래프에서 마이너스 font 깨지는 문제에 대한 대처
```

```
    #레이블에 '-'가 있는 경우 유니코드의 '-'문자를 그대로 출력하면 '-' 부호만 깨져 보인다.
    #이를 방지하기 위해 'axes.unicode_minus' 옵션을 False로 지정한다.
    mpl.rcParams['axes.unicode_minus'] = False


2)font를 설정해 주기에 앞서 설치된 matplotlib 의 버전과 위치 정보를 가져온다.
    print ('버전: ', mpl.__version__)
    print ('설치 위치: ', mpl.__file__)
    print ('설정 위치: ', mpl.get_configdir())
    print ('캐시 위치: ', mpl.get_cachedir())
    ------------------------------------------------
    버전:  3.1.1
    설치 위치:  c:\pythonhome\projectenv\lib\site-packages\matplotlib\__init__.py
    설정 위치:  C:\Users\Instructor\.matplotlib
    캐시 위치:  C:\Users\Instructor\.matplotlib


3)matplotlib의 위치 정보를 알았으니 터미널을 이용해 해당 위치로 가보자.
    print ('설정 파일 위치: ', mpl.matplotlib_fname())
    ------------------------------------------------------------------
    설정 파일 위치:  c:\pythonhome\projectenv\lib\site-packages\matplotlib\mpl-data\matplotlibrc


4)System에 설치된 font 확인
    font_list = fm.findSystemFonts(fontpaths=None, fontext='ttf')
    # ttf font 전체개수
    print(len(font_list))
    -------------------------
    784

    font_list_win = fm.win32InstalledFonts()
    print(len(font_list_win))
    ------------------------------------------------
    392

    font_list_win
    ------------------------------------------------
    ['C:\\Windows\\Fonts\\NanumSquareR.ttf',
     'C:\\Windows\\Fonts\\ELEPHNTI.TTF',
     'C:\\Windows\\Fonts\\Candarab.ttf',
     'C:\\Windows\\Fonts\\FRAHV.TTF',
     'C:\\Windows\\Fonts\\LCALLIG.TTF',
     'C:\\Windows\\Fonts\\FRAHVIT.TTF',
     ...
     ...

     for fname in font_list_win:
         print(fname[17:])
    ------------------------------------------------
    NanumSquareR.ttf
    ELEPHNTI.TTF
    Candarab.ttf
    FRAHV.TTF
    LCALLIG.TTF
    ...
    ...


5)나눔 고딕을 사용할 예정이기 때문에 이름에 'Nanum'이 들어간 font만 가져온다.
    [fname for fname in font_list_win if 'Nanum' in fname]
    ------------------------------------------------------------------------
    ['C:\\Windows\\Fonts\\NanumSquareR.ttf',
     'C:\\Windows\\Fonts\\NanumGothic.ttf',
     'C:\\Windows\\Fonts\\NanumSquareRoundEB.ttf',
     'C:\\Windows\\Fonts\\NanumBarunGothic.ttf',
     'C:\\Windows\\Fonts\\NanumMyeongjoExtraBold.ttf',
     'C:\\Windows\\Fonts\\NanumSquareRoundR.ttf',
     ...


6)Font를 사용하는 방법은 3가지가 있다.
    -FontProperties 를 사용하는 방법 - graph의 font가 필요한 항목마다 지정해 주어야 한다.
    -matplotlib.rcParams[]으로 전역 글꼴 설정 방법 - 그래프에 설정을 해주면 font가 필요한 항목에 적용된다.
    -바로 위의 방법을 mpl.matplotlib_fname()로 읽어지는 설정 파일에 직접 적어주는 방법, 단 모든 notebook에 적용된다.
        --notebook을 열 때마다 지정해 주지 않아도 돼서 편리하다.


7)FontProperties 를 사용하는 방법
    -텍스트를 지정하는 항목에 지정해 사용할 수 있다.
    -지정해 준 항목에만 해당 font가 적용 된다.

        matplotlib.pyplot
            -title()
            -xlabel()
```

```
    -ylabel()
    -legend()
    -text()

matplotlib.axes
    -set_title()

# fname 옵션을 사용하는 방법
path = 'C:/Windows/Fonts/NanumBarunpenR.ttf'
font = fm.FontProperties(fname=path, size=18)

np.random.seed(0)
x = range(5)
y = 10 + 5 * np.random.randn(5)

fig = plt.figure()
ax = fig.add_subplot(111)

#여기서 fontproperties를 지정한다.
ax.set_title('한글 테스트', fontproperties=font)
ax.bar(x, y)
plt.show()


8)matplotlib.rcParams[]으로 전역 글꼴 설정
# 기본 설정 읽기
import matplotlib.pyplot as plt

# size, family
print('설정되어있는 폰트 사이즈 :', plt.rcParams['font.size'])
print('설정되어있는 폰트 글꼴 :',plt.rcParams['font.family'])
------------------------------------------------------------
설정되어있는 폰트 사이즈 : 10.0
설정되어있는 폰트 글꼴 : ['sans-serif']

# serif, sans-serif, monospace
print('serif 세리프가 있는 폰트--------')
print (plt.rcParams['font.serif'])
print('sans-serif 세리프가 없는 폰트 --------')
print (plt.rcParams['font.sans-serif'])
print('monospace 고정폭 글꼴--------')
print (plt.rcParams['font.monospace'])
------------------------------------------------------------
serif 세리프가 있는 폰트--------
['DejaVu Serif', 'Bitstream Vera Serif', 'Computer Modern Roman', 'New Century Schoolbook', 'Century Schoolbook L',
'Utopia', 'ITC Bookman', 'Bookman', 'Nimbus Roman No9 L', 'Times New Roman', 'Times', 'Palatino', 'Charter', 'serif']
sans-serif 세리프가 없는 폰트 --------
['DejaVu Sans', 'Bitstream Vera Sans', 'Computer Modern Sans Serif', 'Lucida Grande', 'Verdana', 'Geneva', 'Lucid', 'Arial',
'Helvetica', 'Avant Garde', 'sans-serif']
monospace 고정폭 글꼴--------
['DejaVu Sans Mono', 'Bitstream Vera Sans Mono', 'Computer Modern Typewriter', 'Andale Mono', 'Nimbus Mono L', 'Courier
New', 'Courier', 'Fixed', 'Terminal', 'monospace']

plt.rcParams["font.family"] = 'nanummyeongjo'
plt.rcParams["font.size"] = 20
plt.rcParams["figure.figsize"] = (14,4)

np.random.seed(0)
x = range(5)
y = 10 + 5 * np.random.randn(5)

fig = plt.figure()
ax = fig.add_subplot(111)

#여기서 fontproperties를 지정하지 않는다.
ax.set_title('한글 테스트')
ax.bar(x, y)
plt.show()

-rcParams 대신 FontProperties 와 plt.rc 를 사용하는 방법
import matplotlib.font_manager as fm
path = 'C:/Windows/Fonts/NanumBarunGothic.ttf'
font_name = fm.FontProperties(fname=path, size=18).get_name()
font_name
---------------------------------------
'NanumBarunGothic'

plt.rcParams['font.family'] = font_name
# or plt.rc('font', family=font_name)

fig, ax = plt.subplots()
ax.plot(range(50))
ax.set_title('시간별 가격 추이')
plt.ylabel('주식 가격')
```

```
plt.xlabel('시간(분)')
plt.style.use('ggplot')
plt.show()
```

9)rcParams 를 설정 파일에 직접 적어주는 방법 - 모든 notebook에 공통적용
-아래의 설정 파일의 위치에 가서 matplotlibrc을 수정한다.
-이곳에 폰트를 지정해 주면 Notebook을 실행할 때 바로 load되도록 설정할 수 있다.

```
print ('설정 파일 위치: ', mpl.matplotlib_fname())
```

```
----------------------------------------------------------
설정 파일 위치:  C:\ProgramData\Anaconda3\lib\site-packages\matplotlib\mpl-data\matplotlibrc
```

-위의 파일을 열어서 196line의 다음을 변경한다.
#font.family : sans-serif <---변경 전
font.family : nanummyeongjo <---변경 후

-저장 후 Jupyter Notebook를 restart 한다.
-Kernel > Restart

```
# 기본 설정 읽기
import matplotlib.pyplot as plt

# size, family
print('설정되어있는 폰트 사이즈 :', plt.rcParams['font.size'])
print('설정되어있는 폰트 글꼴 :',plt.rcParams['font.family'])
```

```
--------------------------------------------------------------------------------
설정되어있는 폰트 사이즈 : 10.0
설정되어있는 폰트 글꼴 : ['nanummyeongjo']
```

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

fig, ax = plt.subplots()
ax.plot(10*np.random.randn(100), 10*np.random.randn(100), 'o')
ax.set_title('숫자 분포도 보기')
plt.show()
```

6. Graph 객체

1)우리가 그림을 그릴 때 가장 먼저 준비해야 하는 것이 도화지와 연필일 것이다.
2)Python의 Matplotlib에서 graph를 그리기 위해 필요한 객체가 Figure이다.
3)이 객체는 그림이 그려지는 도화지라고 생각할 수 있다.
4)이 도화지(Figure)를 plt.subplots() 함수로 분할해 각 부분에 graph를 그리는 방식으로 시각화를 한다.
5)graph를 그리기 위한 Figure 객체는 figure() 함수를 이용해 생성한다.

-다음 코드는 graphic 객체를 생성한다.

```
fig =  plt.figure()
```

-graph 객체의 속성을 설정하는 방법은(예를 들면 그래프 영역의 크기(size)를 조절하려면)
a. graph 객체의 메소드를 이용하는 방법 – 예: fig.set_size_inches(10.5, 8,5)
b. graph 객체를 만들 때 설정하는 방법 – 예: fig = plt.figure(figsize=(10.5, 8,5))
c. reParams를 이용하는 방법 – 예: plt.rcParams[‘figure.figsize’] = (10.5, 8,5)

6)Refer to https://matplotlib.org/api/_as_gen/matplotlib.pyplot.figure.html

7. Graph 영역 나누기

1)graph 객체를 여러 영역으로 나누어 그리면 하나의 graph 객체에 여러 graph를 그릴 수 있다.
2)subplot() 함수로 서브플롯 추가
-subplot() 함수는 현재 figure 객체에 서브플롯을 추가한다.
-Syntax
matplotlib.pyplot.subplot(*args, **kwargs)
-subplot(nrows, ncols, index, **kwargs)
-subplot(pos, **kwargs)
-subplot(ax)

-args : 서브플롯의 위치를 설명하는 3자리 정수(예: 211) 또는 정수 3개(예: 2,2,1).
-nrow, ncols, index : 3개의 정수가 nrows, ncols 및 index 순서로 있는 경우 그려질 하위 그림은 nrows 행과 ncols 열이 있는 표에서 index 위치에 그려진다.
--index는 왼쪽 상단 모서리에서 1부터 시작하여 오른쪽으로 증가한다.
-pos : pos는 3 자리 정수이며 첫 번째 숫자는 행 수, 두 번째 숫자는 열 수, 세 번째 숫자는 서브 그림의 index 이다.
--즉, fig.add_subplot(235)는 fig.add_subplot(2, 3, 5)와 동일하다.
--pos 형식이 작동하려면 모든 정수가 10보다 작아야한다.
--subplot() 함수는 Figure.add_subplot() 함수의 랩퍼(Wrapper)이다.
-Returns : 이 함수는 Figure 객체와 axes.Axes 객체(또는 Axes 객체들의 배열)를 반환한다.

-다음 코드는 graph 영역을 나누고 각각의 영역에 graph를 그리는 예이다.
-아래의 코드를 작성할 때 plot을 나누는 코드(subplot)와 graph를 그리는 코드(plot)는 같은 셀에 있어야 한다.

```
import   numpy   as np
```

```
import   matplotlib.pyplotas plt
%matplotlib   inline

x  =  np.arange(0,  10,   0.01)
plt.subplot(2, 1, 1)
plt.plot(x,  np.sin(x))
plt.subplot(2, 2, 3)
plt.plot(x,  np.cos(x))
plt.subplot(2, 2, 4)
plt.plot(x,  np.sin(x)*np.cos(x))
plt.show()
```

3)subplots() 함수로 서브플롯 집합 추가

-subplots() 함수는 현재 figure 객체에 서브플롯 집합을 추가한다.

-figure 생성과 subplot의 배치를 동시에 실행하는 함수

```
#figure object 작성과 subplot 배치를 동시에 실행
fig, axes = plt.subplots(2,2)
print(type(axes), axes)
plt.show()
-------------------------------------------
<class 'numpy.ndarray'> [[<matplotlib.axes._subplots.AxesSubplot object at 0x00000209BD93E248>
                          <matplotlib.axes._subplots.AxesSubplot object at 0x00000209BD9263C8>]
                         [<matplotlib.axes._subplots.AxesSubplot object at 0x00000209BD77B8C8>
                          <matplotlib.axes._subplots.AxesSubplot object at 0x00000209BD731D08>]]
```

-행렬로 subplot의 위치를 지정할 수 있다.

```
#1행 2열째 subplot에 subplot title을 지정
fig, axes = plt.subplots(2,2)
axes[0,1].set_title('Subplot 0-1')
plt.show()
```

-다음 코드는 subplots() 함수로 그래프 영역을 나누고, index를 이용해서 지정한 위치 영역에 그래프를 그린다.

```
import   numpy   as np
import   matplotlib.pyplotas plt
%matplotlib   inline

x  =  np.arange(0,  10,   0.01)
fig,   axes  =  plt.subplots(nrows=2, ncols=2)
axes[0,0].plot(x,np.sin(x))
axes[0,1].plot(x,np.cos(x))
axes[1,0].plot(x,np.tanh(x))
axes[1,1].plot(x,np.sin(x)*np.cos(x))
plt.show()
```

-subplots()로 나눈 화면영역은 enumerate()함수를 이용해 index와 grapg객체를 반복 처리 할 수 있다.

-다음 코드는 이전 코드와 같은 결과를 출력할 것이다.

```
import   numpy   as np
import   matplotlib.pyplotas plt
%matplotlib   inline

x  =  np.arange(0,  7, 0.01)

def   sin_cos(x):
   return   np.sin(x)*np.cos(x)
   #graph 객체를 반복 처리하기 위해 함수를 list 안에 포함시킬 함수를 정의한다.

func_list=  [np.sin, np.cos,  np.tanh,sin_cos]
#그리고 graph 객체에서 반복 처리하기 위한 함수를 list로 선언한다.
```

-다음 코드는 subplots() 함수로 그래프 영역을 나눈다.

-그리고 enumerate(axes.flat)를 이용하면 index인덱스와 graph 객체를 반복문으로 처리할 수 있다.

```
fig,   axes  =  plt.subplots(nrows=2,ncols=2)
for   i,  ax in   enumerate(axes.flat):
   ax.plot(x,  func_list[i](x))

plt.show()
```

-위의 예제에서 사용한 plot() 함수는 데이터를 이용해 점/선 graph를 그려준다.

-subplots() 함수의 인수를 어떻게 선언하느냐에 따라 graph 영역이 다양하게 나뉜다.

-다음 코드는 ncols=4(4개의 열)로 그래프 영역을 나눈다.

```
fig,   axes  =  plt.subplots(ncols=4)
for   i,  ax in   enumerate(axes.flat):
   ax.plot(x,  func_list[i](x))

plt.show()
```

-nrows=4로 하면 4개 행으로 graph 영역을 나눈다.

```
fig, axes = plt.subplots(nrows=4)
for i, ax in enumerate(axes.flat):
    ax.plot(x, func_list[i](x))

plt.show()
```

4)add_subplot()으로 서브플롯 배치하기

-add_subplot(총행수, 총열수, 서브플롯번호)

```
#figure 생성
fig = plt.figure()

#figure 안에 subplot 3개 배치
ax1 = fig.add_subplot(221)  #2행 2열 1번
ax2 = fig.add_subplot(222)  #2행 2열 2번
ax3 = fig.add_subplot(223)   #2행 2열 3번

plt.show()
```

-각 subplot의 번호를 확인해보자.

```
fig = plt.figure()

#subplot 작성
ax1 = fig.add_subplot(221)
ax2 = fig.add_subplot(222)   #add_subplot(2,2,2) 도 가능
ax3 = fig.add_subplot(223)

#번호 확인
for i, ax in enumerate([ax1, ax2, ax3], start = 1):
    txt = 'ax{0}\n(22{0})'.format(i)
    ax.text(0.2, 0.4, txt, fontsize=24)

plt.show()
```

8. Graph 그리기

1)Matplotlib의 다양한 graph 함수들에 대해 알아보자.

2)Refer to https://matplotlib.org/api/_as_gen/matplotlib.pyplot.html

3)plot()

-plot() 함수는 주어진 x, y 값을 선(lines)과 점(markers)으로 표시해 준다.

-Syntax

```
matplotlib.pyplot.plot([x], y, [fmt], data=None, **kwargs)
```

-fmt : 색, 점, 라인의 style을 문자열로 지정.

--예를 들면 'ro-'는 빨간색 동그란 점을 실선으로 연결한다.

--점의 모양은 o(원), s(네모), v(역삼각형), ^(삼각형), x(x표시) 등이 있으며,

--선의 스타일은 '-'(실선), '--'(대시선), '-.'(대시닷선), ':'(점선), ''(선없음) 등이 있다.

-다음 코드는 graph 객체를 만들고 graph 영역을 2x2 분할 한 후 각각의 영역에 graph를 그린다.

-마지막 graph 영역에는 graph를 두 개 그린다.

```
import matplotlib
import matplotlib.pyplot as plt
%matplotlib inline

fig = plt.figure()
fig, axes = plt.subplots(2, 2, figsize=(8,5))
fig.suptitle('figure sample plots')
axes[0,0].plot([1,2,3,4], 'ro-') # 빨간(r), 동그라미(o), 실선(-)
axes[0,1].plot(np.random.randn(4,10), np.random.randn(4,10), #4행10열 난수
                                                    'cs-.')  # cyan(c), square(s), 대시닷(-.)
axes[1,0].plot(np.linspace(0, 5), np.cos(np.linspace(0, 5)))
axes[1,1].plot([3,6], [3,5], 'b^:')          # 파랑(b), 세모(^), 점선(:)
axes[1,1].plot([4,5], [5,4], 'kx--')         # 검정(k), X(x), 대시선(--)
plt.show()
```

9. Style 적용하기

1)Style이란 graph의 선 굵기나 색 등 graph의 '체재'에 관한 정보를 모아놓은 것이다.

2)Style은 style.use()함수로 적용할 수 있다.

3)다음 코드는 ggplot style로 그리기를 하고 있다.

```
#style 적용
plt.style.use('ggplot')
fig = plt.figure()
ax = fig.add_subplot(111)

dat = [0, 1]
```

```
    ax.plot(dat)

    plt.show()
```

10. pandas를 사용한 data의 시각화
   1)pandas의 Series 또는 DataFrame의 plot()를 사용하여 쉽게 시각화할 수 있다.
   2)plot()은 내부에서 Matplotlib를 사용하고 있다.
   3)Notebook에 graph 표시하기
      -Notebook에 graph를 표시하기 위해서는 pyplot.show()를 사용한다.

```
        import pandas as pd
        import matplotlib.pyplot as plt

        ax = pd.Series([1,2,3]).plot()
        ax.set_title('Line Chart')
        plt.show()
        #graph를 그릴 때는 cell을 바꾸기 않고 하나의 cell안에 모두 coding해야 한다.

        column_names = ['Hakbun', 'Name', 'Kor', 'Eng', 'Mat', 'Edp']
        df = pd.read_csv('pandas_data/sungjuk_utf8.csv', names = column_names)
        df['Total'] = df['Kor'] + df['Eng'] + df['Mat'] + df['Edp']
        df['Average'] = df['Total'] / 4
        grade_list = []
        for avg in df['Average'] :
            if 90 <= avg <= 100 : grade_list.append('A')
            elif 80 <= avg < 90 : grade_list.append('B')
            elif 70 <= avg < 80 : grade_list.append('C')
            elif 60 <= avg < 70 : grade_list.append('D')
            else : grade_list.append('F')
        df['Grade'] = grade_list
        df

        ax = df['Kor'].plot().set_title('Line Chart')
        plt.show()
```

      -Graph의 style을 변경하는 경우에는 pyplot.style.use()의 인수에 style명을 넘긴다.

```
        plt.style.use('ggplot')    #기본은 'default'
```

   4)DataFrame에서 plot하기
      -DataFrame에서 plot()을 호출할 경우 기본적으로 Series와 같은 동작을 수행하는데, 열 수에 상응하는 요소가 그려진다.
      -Index가 X값, 이름 열의 값이 Y값이 된다.

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

   5)Y축 범위가 다른 경우
      -Keyword 인수 secondary_y에 두번째 축이 되는 열 이름을 list형으로 지정한다.
      -Y축의 label은 set_ylabel(), right_ax.set_ylabel()의 인수에 각각의 이름을 넘겨준다.

```
        ser1 = df['Hakbun']
        ser2 = df['Kor']
        df2 = pd.DataFrame(ser1, columns=['Hakbun'])
        df2['Kor'] = ser2

        ax = df2.plot(secondary_y=['Kor'])
        ax.set_title('Two Line Chart')
        ax.set_ylabel('Hakbun')
        ax.right_ax.set_ylabel('Kor')
        plt.show()
```

   6)산포도 graph 그리기
      -plot.scatter()를 사용한다.
      -Keyword 인수 x에 X값이 되는 열 이름, keyword 인수 y에 Y 값이 되는 열 이름을 지정한다.

```
        ax = df2.plot.scatter(x='Hakbun', y='Kor')
        ax.set_title('Scatter')
        plt.show()
```

   7)막대 Graph 작성하기
      -plot.bar()를 사용한다.

```
        import cx_Oracle
        conn = cx_Oracle.connect('scott', 'tiger', 'localhost:1521/XE')
        cursor = conn.cursor()
        sql = "SELECT empno, ename, job, TO_CHAR(hiredate, 'YYYY'), mgr, sal, comm, deptno FROM emp"
        cursor.execute(sql)
```

```
emp_list = []
for empno, ename, job, hiredate, mgr, sal, comm, deptno in cursor:
    emp_list.append([empno, ename, job, hiredate, mgr, sal, comm, deptno])
columns = ['empno', 'ename', 'job', 'hiredate', 'mgr','sal', 'comm', 'deptno']
df = pd.DataFrame(emp_list, columns = columns)
df

group_deptno = df.groupby('deptno')
group_deptno.groups
--------------------------------------------------------------------------
{10: Int64Index([6, 8, 13], dtype='int64'),
 20: Int64Index([0, 3, 7, 10, 12], dtype='int64'),
 30: Int64Index([1, 2, 4, 5, 9, 11], dtype='int64')}

for name, group in group_deptno:
    print(str(name) + ": " + str(len(group)))
    print(group)
    print()
-----------------------------------------------------------------------------------------
10: 3
     empno     ename      job              hiredate    mgr     sal       comm      deptno
6    7782       CLARK    MANAGER       1981          7839.0  2450.0     NaN      10
8    7839    KING       PRESIDENT    1981          NaN  5000.0     NaN      10
13   7934       MILLER  CLERK         1982          7782.0  1300.0     NaN      10

20: 5
     empno  ename      job              hiredate       mgr        sal       comm   deptno
0    7369       SMITH      CLERK         1980          7902.0     800.0   NaN      20
3    7566       JONES   MANAGER     1981          7839.0     2975.0  NaN      20
7    7788       SCOTT   ANALYST      1987          7566.0     3000.0  NaN      20
10   7876       ADAMS  CLERK         1987          7788.0     1100.0  NaN      20
12   7902       FORD    ANALYST      1981          7566.0     3000.0  NaN      20

30: 6
     empno      ename      job              hiredate    mgr        sal       comm       deptno
1    7499       ALLEN   SALESMAN    1981          7698.0     1600.0     300.0       30
2    7521    WARD    SALESMAN    1981          7698.0     1250.0     500.0       30
4    7654       MARTIN    SALESMAN    1981          7698.0     1250.0     1400.0    30
5    7698       BLAKE      MANAGER     1981          7839.0     2850.0    NaN      30
9    7844       TURNER  SALESMAN   1981          7698.0     1500.0    0.0          30
11   7900       JAMES     CLERK       1981          7698.0     950.0      NaN      30

names = []
values = []
for name, group in group_deptno:
    names.append(name)
    values.append(len(group))

plt.figure(1, 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.xlabel('Department Number')
plt.show()


8)다양한 Graph 그리기

fig, axs = plt.subplots(2,2, figsize=(5,5))
axs[0,0].hist(df['sal'])
axs[1,0].scatter(df['empno'], df['sal'])
axs[0,1].plot(df['empno'], df['sal'])
axs[1,1].hist2d(df['empno'], df['sal'])


9)scatter()
    -scatter() 함수는 산점도 그래프를 그려준다.
    -Refer to https://matplotlib.org/api/_as_gen/matplotlib.pyplot.scatter.html

    import   numpy   as np
    import   matplotlib.pyplotas plt
    np.random.seed(7902)
    N  =  50
    x  =  np.random.rand(N)
    y  =  np.random.rand(N)
    colors    =  np.random.rand(N)
    area  =  (30   *  np.random.rand(N))**2
    plt.scatter(x,  y, s=area, c=colors,   alpha=0.5)
    plt.show()
```

```
    import matplotlib.pyplot as plt
    import pandas as pd
    from matplotlib import font_manager, rc
    font_name = font_manager.FontProperties(fname="c:/Windows/Fonts/malgun.ttf").get_name()
    rc('font', family=font_name)
    df = pd.read_csv('korea.csv',encoding='ms949')
    plt.figure()
    plt.scatter(x=df.index,y=df['점수'], marker='2')
    plt.xticks(range(0,len(df['점수']),1),df['이름'], rotation='vertical')
    plt.title('학생별 국어점수 산포도')
    plt.show()
```

10)Histogram

```
    import matplotlib.pyplot as plt
    import pandas as pd
    from matplotlib import font_manager, rc
    font_name = font_manager.FontProperties(fname="c:/Windows/Fonts/malgun.ttf").get_name()
    rc('font', family=font_name)
    df = pd.read_csv('csv_exam1.csv',encoding='ms949')
    data = pd.concat([df['국어'],df['영어'],df['수학']])
    plt.hist(data, bins=3)
    plt.xticks(range(0,100,40),['하', '중', '상'])
    plt.title('점수빈도')
    plt.show()

    import matplotlib.pyplot as plt
    import pandas as pd
    from matplotlib import font_manager, rc
    font_name = font_manager.FontProperties(fname="c:/Windows/Fonts/malgun.ttf").get_name()
    rc('font', family=font_name)
    df = pd.read_csv('csv_exam1.csv',encoding='ms949')
    plt.hist((df['국어'],df['영어'],df['수학']), bins=10, label=('국어','영어','수학'))
    plt.title('점수빈도')
    plt.legend()
    plt.show()
```

11)bar

```
    import matplotlib.pyplot as plt
    import pandas as pd
    from matplotlib import font_manager, rc
    font_name = font_manager.FontProperties(fname="c:/Windows/Fonts/malgun.ttf").get_name()
    rc('font', family=font_name)
    df = pd.read_csv('korea.csv',encoding='ms949')
    print(df)
    plt.figure()
    plt.bar(df.index, df['점수'],width=1.0, color='r')
    plt.xticks(range(0,len(df.index),1),df['이름'], rotation='vertical')
    plt.title('학생별 국어 점수')
    plt.show()

    import matplotlib.pyplot as plt
    import pandas as pd
    from matplotlib import font_manager, rc
    font_name = \
        font_manager.FontProperties(fname="c:/Windows/Fonts/malgun.ttf").get_name()
    rc('font', family=font_name)
    df = pd.read_csv('csv_exam1.csv',encoding='ms949')
    print(df)
    plt.figure()
    plt.barh(df.index, df['국어'], color='r', label='국어')
    plt.barh(df.index, -df['영어'], color='g', label='영어')
    plt.title('학생별 국어,영어 점수')
    plt.yticks(range(0,len(df.index),1),df['이름'], rotation='horizontal')
    plt.xticks([-100,-50,0,50,100],(100,50,0,50,100))
    plt.legend()
    plt.show()
```

12)line

```
    import matplotlib.pyplot as plt
    from pandas import Series, DataFrame
    s = Series([84900, 818000, 1756,292000])
    # 객체생성
    plt.figure()
    # 출력
    plt.plot(s)
    plt.show()

    import matplotlib.pyplot as plt
```

```
from pandas import Series
s1 = Series([84900, 81800, 71756, 92000]) #Series
s2 = Series([80500, 82000, 71736, 90000]) #Series
plt.figure(figsize=(10,4))
plt.plot(s1, label='04-10')
plt.plot(s2, label='04-11')
plt.grid()
plt.xlabel('index')
plt.ylabel('stock')
plt.title('plot graph')
plt.legend()
plt.show()
```

13)box

```
import matplotlib.pyplot as plt
import pandas as pd

from matplotlib import font_manager, rc
font_name = \
    font_manager.FontProperties(fname="c:/Windows/Fonts/malgun.ttf").get_name()
rc('font', family=font_name)
df = pd.read_csv('csv_exam1.csv',encoding='ms949')
print(df)
plt.boxplot((df['국어'],df['영어'],df['수학']), labels=('국어','영어','수학'))
print(df['수학'].min())
print(df['수학'].mean())
print(df['수학'].median())
plt.title('점수분포')
plt.show()
```

11. 꺾은선 그래프

1)꺾은선 그래프는 plot된 점과 점을 직선으로 연결한 그래프이다.

2)꺾은선 그래프 작성하기

-Axes.plot()으로 그린다.

-plot()의 인수가 하나뿐인 경우, 부여된 인수는 Y값으로 설정되어 X값은 자동적으로 '최소값=0' '최대값=list의 요소수-1'의 정수열이 지정된다.

```
fig = plt.figure()
ax = fig.add_subplot(111)

ax.plot([1,3])
plt.show()
```

-위의 코드에서는 그리기 대상 data로 list형 데이터가 넘겨졌지만, plot()에서는 다음과 같은 data형이 사용된다.

a. list
b. tuple
c. numpy.ndarray
d. pandas.Series

-다음 코드는 전형적인 꺾은선 그래프이다.

```
fig = plt.figure()
ax = fig.add_subplot(111)

x = [0,2,4]
y = [0,4,2]
ax.plot(x, y)
plt.show()
```

-여러 개의 선을 그리는 경우

-plot()을 여러 번 실행하면 1개의 subplot에 여러 개의 graph를 겹쳐서 그릴 수 있다.

```
fig = plt.figure()
ax = fig.add_subplot(111)

x = [0,2,4]
y1 = [0, 4, 2.5]
y2 = [4,0, 1.5]

#2개의 선 그리기
ax.plot(x, y1)
ax.plot(x, y2)
plt.show()
```

3)꺾은선 그래프 활용하기

-실제 데이터를 이용해서 그래프를 그려보자.

-데이터는 anime_stock_returns.csv 파일이다.

-이 파일에는 TOEI ANIMATION 및 IG Port의 주가 등락률이 시계열(일단위)로 기록되어 있다.

```
import os
```

```
import pandas as pd
base_url = 'https://raw.githubusercontent.com/practical-jupyter/sample-data/master/anime/'
anime_stock_returns_csv = os.path.join(base_url, 'anime_stock_returns.csv')

df = pd.read_csv(anime_stock_returns_csv, index_col = 0, parse_dates = ['Date'])
df.head()
------------------------------------------------------------
                TOEI ANIMATION      IG Port
Date
2015-01-01      1.000000            1.000000
2015-01-02      1.000000            1.000000
2015-01-05      1.011695            1.014082
2015-01-06      1.001463            1.000000
2015-01-07      0.982457            1.000824
```

-시계열 정보를 포함한 데이터를 표현하는 것은 꺾은선 그래프가 적당하다.

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

fig = plt.figure(figsize=(10,4))
ax = fig.add_subplot(111)

#data와 범례 지정
ax.plot(df.index, df['TOEI ANIMATION'], label='TOEI ANIMATION')
ax.plot(df.index, df['IG Port'], label='IG Port')

#title, 축레이블 지정
ax.set_title('주가등락률 2년간 추이')
ax.set_ylabel('주가등락률')
ax.set_xlabel('년월')

#범례 유효화
ax.legend()
plt.show()
```

4)2개의 축을 가진 graph 그리기

-Matplotlib에서 X축을 공유해서 2개의 Y축을 가진 그림을 작성하는 경우에는 Axes.twinx() 함수를 사용한다.

-Y축을 공유해서 2개의 X축을 가진 그림을 그리는 경웨는 twiny() 함수를 사용한다.

-다음 코드는 twinx()를 사용해서 마감가(Close)와 거래량(Volume)을 하나의 graph로 나타내고 있다.

-마감가는 꺾은선 graph로, 거래량은 막대 graph로 나타내는 것이 일반적이다.

```
t4816_csv = os.path.join(base_url, "4816.csv")
df = pd.read_csv(t4816_csv, index_col=0, parse_dates=["Date"])

fig = plt.figure(figsize=(10, 4))
ax1 = fig.add_subplot(111)

ax1.plot(df.index, df["Close"], color="b", label="주가")

#X축을 공유해서 Y축을 2개 사용하는 설정
ax2 = ax1.twinx()
ax2.bar(df.index, df["Volume"], color="g", label="거래총액", width=2)

# 축과 축레이블 설정
ax1.set_yticks([i * 2000 for i in range(5)])
ax1.set_ylabel("주가")
ax2.set_yticks([i * 50000 for i in range(5)])
ax2.set_ylabel("거래총액")
ax1.set_xlabel("년월")

# Graph 타이틀 설정
ax1.set_title("주가와 거래총액")

#범례설정
ax1.legend(loc=1)
ax2.legend(loc=2)
plt.show()
```

12. 산포도 Graph

1)산포도 Graph는 X축과 Y축에 수량이나 크기 등을 대응시켜서 적합한 점에 데이터를 플롯한 graph이다.

2)산포도 Graph는 X축과 Y축에 취한 2개의 값(Z축이 있는 경우에는 3개의 값)에 함수가 있는지 없는지 보는 것에 유용하다.

3)또한 데이터의 분포 상황을 확인할 때에도 활용할 수 있다.

4)산포도 Graph 작성하기

-Axes.scatter() 함수를 사용해서 그린다.

-제1,제2인수에 각각 X값과 Y값을 부여한다.

```
plt.style.use("ggplot")

#입력값 생성
```

```
np.random.seed(2)
x = np.arange(1, 101)
y = 4 * x * np.random.rand(100)

# 산포도 graph 그리기
fig = plt.figure()
ax = fig.add_subplot(111)
ax.scatter(x, y)
plt.show()
```

5)산포도 Graph 활용하기

-실제 데이터를 이용해서 그래프를 그려보자.

-데이터는 anime_master.csv 파일이다.

-데이터를 불러오면 애니메이션의 제목이나 장르, 에피소드 수나 평점 데이터가 포함되어 있다.

```
import os
import pandas as pd

base_url = "https://raw.githubusercontent.com/practical-jupyter/sample-data/master/anime/"
anime_master_csv = os.path.join(base_url, "anime_master.csv")
df = pd.read_csv(anime_master_csv)
df.head()
--------------------------------------------------
```

-anime_id를 인수 index_col로 지정해서 anime_id를 index에 설정할 수 있다.

```
df = pd.read_csv(anime_master_csv, index_col="anime_id")
df.head()
------------------------------------------------
```

-X값으로 members를, Y값으로 rating을 지정하는 것에 따라 산포도 그래프를 작성할 수 있다.

-그려진 기호를 반투명으로 하는 값(alpha=0.5)도 설정한다.

```
fig = plt.figure()
ax = fig.add_subplot(111)
ax.scatter(df["members"], df["rating"], alpha=0.5)
plt.show()
```

6)그룹화된 산포도 graph 작성하기

-위의 데이터는 type이라는 열을 가지고 있다.

-type은 애니메이션 작품의 배급 종별을 의미한다.

-먼저 type 중복 없는 list를 작성한다.

```
types = df['type'].unique()
types
--------------------------------------------
array(['Movie', 'TV', 'OVA', 'Special', 'Music', 'ONA'], dtype=object)
```

-하나의 subplot에 겹쳐서 산포도 graph를 그린다.

-다음 코드는 배급 종별(type)마다 일치하는 데이터를 추출해서 그리고 있다.

-배급 종별은 6종류가 있기 때문에 6개의 데이터 세트가 플롯되어 있다.

```
fig = plt.figure(figsize=(10, 5))
ax = fig.add_subplot(111)
for t in types:
    x = df.loc[df["type"] == t, "members"]
    y = df.loc[df["type"] == t, "rating"]
    ax.scatter(x, y, alpha=0.5, label=t)
ax.set_title("배급 종별로 그룹화한 데이터 산포도 그래프")
ax.set_xlabel("Members")
ax.set_ylabel("Rating")
ax.legend(loc="lower right", fontsize=12)
plt.show()
```

13. 막대 그래프

1)막대그래프는 수량을 막대의 길이로 나타낸 그래프이다.

2)작성하기

-막대그래프는 Axes.bar() 함수를 사용해서 그린다.

-제1인수, 제2인수에 각각 X값과 Y값을 부여한다.

-데이터로서 list형, object를 이용할 수 있다.

```
plt.style.use("ggplot")
fig = plt.figure()
ax = fig.add_subplot(111)
x = [1, 2]
y = [1, 3]
ax.bar(x, y)
plt.show()
```

3)눈금레이블을 붙일 경우

-인수 tick_label에 눈금레이블을 설정해서 작성한다.
-label은 list나 tuple로 부여한다.

```
fig = plt.figure()
ax = fig.add_subplot(111)
labels = ["apple", "orange"]
ax.bar(x, y, tick_label=labels)
plt.show()
```

-graph를 그린 후 Axes.set_xticks()로 X축 눈금을 설정하고 Axes.set_xtickabels()로 눈금 레이블을 설정한다.

```
#그리기
fig = plt.figure()
ax = fig.add_subplot(111)
ax.bar(x, y)

#X축의 축눈금과 축눈금 레이블
ax.set_xticks(x)
ax.set_xticklabels(labels)
plt.show()
```

4)수평 막대그래프를 작성하는 경우
-수평 막대그래프는 Axes.barh()를 이용해서 그린다.
-barh()의 인수는 기본적으로 bar()와 같다.

```
fig = plt.figure()
ax = fig.add_subplot(111)
ax.barh(x, y, tick_label=labels)
plt.show()
```

5)막대 그래프 활용하기
-실제 데이터를 이용해서 그래프를 그려보자.
-데이터는 anime_master.csv 파일이다.

```
import os
import pandas as pd

base_url = "https://raw.githubusercontent.com/practical-jupyter/sample-data/master/anime/"
anime_master_csv = os.path.join(base_url, "anime_master.csv")
dfac = pd.read_csv(anime_master_csv)
dfac.head()
```

-막대 그래프는 수량의 대소를 시각화할 때 적당하다.
-여기서는 작품의 배급 종별마다 멤버수의 합계 값을 추출해서 막대그래프로 그린다.
-이처럼 데이터를 시각화하여 배급 종별에서는 텔레비젼 작품의 멤버 수가 돌출되는 것을 확인할 수 있다.

```
fig = plt.figure()
ax = fig.add_subplot(111)
y = dfac.groupby("type").sum()["members"]
x = range(len(y))
xlabels = y.index
ax.bar(x, y, tick_label=xlabels)
ax.set_ylabel("합계 멤버수")
plt.show()
```

6)여러가지 그룹에 대한 막대그래프 작성하기
-여러 번 bar()를 실행하면 최초에 그려진 오브젝트가 뒤에 그려진 오브젝트에 의해 덮어씌워진다.

```
import numpy as np

# 데이터 세트 작성
x = [1, 2]
y1, y2, y3 = [1, 2], [2, 4], [3, 6]

# 복수 그룹의 막대 그래프
fig = plt.figure()
ax = fig.add_subplot(111)

w = 0.2
ax.bar(x, y1, label="y1")
ax.bar(x, y2, label="y2")
ax.bar(x, y3, label="y3")
ax.legend()
plt.show()
```

-다음 코드는 같은 X값을 가진 오브젝트가 겹쳐진다.
-이것을 피하기 위해서는 X값을 막대의 가로 폭만큼 비켜서 그릴 필요가 있다.
-다음 코드에서는 막대그래프의 가로 폭 w를 0.2로 설정하고 X값을 0.2씩 비켜서 그리고 있다.

```
fig = plt.figure()
ax = fig.add_subplot(111)
```

```
    w = 0.2
    ax.bar(x, y1, width=w, label="y1")
    ax.bar(np.array(x) + w, y2, width=w, label="y2")
    ax.bar(np.array(x) + w * 2, y3, width=w, label="y3")
    ax.legend()
    plt.show()
```

7)여러 그룹의 막대그래프 활용하기

-실제 데이터를 이용해서 그래프를 그려보자.

-데이터는 anime_genre_top10_pivoted.csv파일이다.

```
    base_url = "https://raw.githubusercontent.com/practical-jupyter/sample-data/master/anime/"
    anime_genre_top10_pivoted_csv = os.path.join(base_url, "anime_genre_top10_pivoted.csv")
    dfag = pd.read_csv(anime_genre_top10_pivoted_csv, index_col="genre")
    dfag
    ------------------------------------------------
```

-다음으로 불러온 데이터(dfag)를 시각화한다.

-X값을 0.1씩 증가시키면서 열별로 그리고 있다.

-이 결과에서도 TV 합계 멤버 수가 돌출되어 있고 다음에 Movie 멤버 수가 많은 것을 확인할 수 있다.

```
    fig = plt.figure(figsize=(18, 3))
    ax = fig.add_subplot(111)
    wt = np.array(range(len(dfag)))
    w = 0.1

    for i in dfag.columns:
        ax.bar(wt, dfag[i], width=w, label=i)
        wt = wt + w

    ax.set_xticks(np.array(range(len(dfag) + 2)))
    ax.set_xticklabels(dfag.index, ha="left")
    ax.set_ylabel("누적 멤버수")
    ax.legend()
    plt.show()
```

-결과에서 보듯이, Music이나 ONA 값이 상대적으로 작기 때문에 눈으로 확인하는 것이 어렵다.

-이럴 때는 로그 축을 이용하면 가독성이 좋아진다.

-Y축을 로그축에 설정하는 경우에는 set_yscale()에 log를 지정한다.

-다음 코드는 작은 값의 그룹도 눈으로 확인할 수 있게 되었다.

```
    fig = plt.figure(figsize=(18, 3))
    ax = fig.add_subplot(111)

    wt = np.array(range(len(dfag)))
    w = 0.1

    for i in dfag.columns:
        ax.bar(wt, dfag[i], width=w, label=i)
        wt = wt + w

    ax.set_xticks(np.array(range(len(dfag) + 2)))
    ax.set_xticklabels(dfag.index, ha="left")
    ax.set_ylabel("누적 멤버수")
    ax.set_yscale("log")
    ax.legend()
    plt.show()
```

8)누적 막대그래프 작성하기

-누적 막대그래프를 그릴 때에도 여러 그룹의 막대그래프와 같이 작성시 요령이 필요하다.

-다음 코드는 y1, y2, y3의 3개의 값을 누적한 경우의 그리기 순서이다.

a. y1과 y2와 y3의 합을 그린다.

b. a에 y2와 y3의 합을 겹쳐서 그린다.

c. b에 y1을 겹쳐서 그린다.

-다시 말하면, 같은 X값을 부여해서 그리면 뒤에 그린 막대에 겹쳐지기 때문에 수동으로 값의 합계를 내서 합계가 많은 쪽부터 순서대로 그리는 작업을 한다.

```
    x = np.arange(5)
    np.random.seed(0)
    y = np.random.rand(15).reshape((3, 5))
    y1, y2, y3 = y

    y1b = np.array(y1)
    y2b = y1b + np.array(y2)
    y3b = y2b + np.array(y3)

    # 누적 막대 그래프 그리기
    fig = plt.figure(figsize=(10, 3))
    ax = fig.add_subplot(111)
    ax.bar(x, y3b, label="y3")
    ax.bar(x, y2b, label="y2")
```

```
ax.bar(x, y1b, label="y1")
ax.legend()
plt.show()
```

9)bottom 옵션으로 누적 설정하기
-누적 막대그래프 작성시 옵션으로 bottom 옵션이 있다.
-하단에 오는 list형.오브젝트를 인수 bottom에 설정하는 것에 의해 누적 표시가 이루어진다.
-2개 그룹의 누적까지는 bottom 옵션이 유효하지만, 그 이상을 누적할 때에는 위의 방법으로 해야 한다.

```
figure = plt.figure(figsize=(10, 3))
ax = figure.add_subplot(111)
ax.bar(x, y3, bottom=y2b, label="y3")
ax.bar(x, y2, bottom=y1, label="y2")
ax.bar(x, y1, label="y1")
ax.legend()
plt.show()
```

10)누적 막대그래프 활용하기
-데이터는 앞에서 이용한 anime_genre_top10_pivoted.csv파일이다.
-데이터는 dfag에 DataFrame으로 저장되어 있다.

```
fig = plt.figure(figsize=(15, 3))
ax = fig.add_subplot(111)
rows, cols = len(dfag), len(dfag.columns)
x = range(rows)
for i, t in enumerate(dfag.columns):
    # i열부터 마지막까지 합을 계산
    y = dfag.iloc[:, i:cols].sum(axis=1)
    ax.bar(x, y, label=t)
ax.set_xticks(range(rows + 2))
ax.set_xticklabels(dfag.index)
ax.set_ylabel("누적 멤버수")
ax.legend()
plt.show()
```

14. 히스토그램
1)히스토그램은 세로축에 회수(값의 출현빈도), 가로축에 계급(값의 상한값 ~ 하한값)을 취급하는 그래프로, 데이터의 분포 형상을 시각적으로 인식하기 위해 이용한다.
2)데이터의 분포 현상(분포형)은 통계학적으로 매우 중요한 의미를 가지고 있다.
3)히스토그램 작성하기
-Axes.hist()를 사용해서 작성한다.
-이 함수에 넘기는 데이터는 list형 오브젝트를 시용할 수 있다.
-다음 코드는 평균값 100, 표준편자 10의 정규분포에 따라 만 개의 데이터 히스토그램을 그린다.

```
plt.style.use("ggplot")

#데이터 세트 작성
mu = 100  # 평균값
sigma = 10  # 표준편차
np.random.seed(0)
x = np.random.normal(mu, sigma, 10000)

#히스토그램 그리기
fig = plt.figure()
ax = fig.add_subplot(111)
ax.hist(x)
plt.show()
```

4)막대의 폭과 수를 변경하는 경우
-hist()에는 대이터 외에 히스토그램 그림에 관한 인수를 부여할 수 있다.
-rwidth로 막대의 폭을, bins로 막대의 갯수를 지정할 수 있다.

```
fig = plt.figure()
ax = fig.add_subplot(111)
ax.hist(x, rwidth=0.9, bins=16)
plt.show()
```

5)히스토그램 활용하기
-실제 데이터를 이용해서 그래프를 그려보자.
-anime_master.csv 파일을 이용한다.

```
import os
import pandas as pd

base_url = "https://raw.githubusercontent.com/practical-jupyter/sample-data/master/anime/"
anime_master_csv = os.path.join(base_url, "anime_master.csv")
df = pd.read_csv(anime_master_csv, index_col="anime_id")
df.head()
-----------------------------------------------
```

-평점 분포에 대해 matplotib으로 시각화를 실행해 보자.
-pandas의 Series를 hist()의 인수에 넘겨서 출력한다.
-평점이 0 ~ 10의 범위에서 실행되고 있기 때문에 값의 범위를 range 0 ~ 10으로 지정한다.

```
fig = plt.figure()
ax = fig.add_subplot(111)
ax.hist(df["rating"], range=(0, 10), rwidth=0.9)
ax.set_title("Rating")
plt.show()
```

-에피소드 수도 히스토그램으로 그려본다.
-다음 코드를 실행하면 왼쪽으로 크게 치우쳐진 히스토그램이 된다.

```
fig = plt.figure()
ax = fig.add_subplot(111)
df_tv = df[df["type"] == "TV"]
ax.hist(df_tv["episodes"], rwidth=0.9)
ax.set_title("Episodes")
plt.show()
```

-그래서 이번에는 히스토그램의 범위를 지정해 보자.

```
fig = plt.figure()
ax = fig.add_subplot(111)

# range의 값을 (0, 100)으로 지정.
ax.hist(df_tv["episodes"], rwidth=0.9, range=(0, 100))
ax.set_title("Episodes(0-100)")
plt.show()
```

15. 다양한 히스토그램 작성하기

1)수평 히스토그램
-인수 orientation에 horizontal(초기 설정은 vertical)을 지정하면 된다.

```
np.random.seed(0)
x = np.random.normal(100, 10, 10000)
fig = plt.figure()
ax = fig.add_subplot(111)

# orientation을 horizontal에 지정
ax.hist(x, rwidth=0.9, bins=16, orientation="horizontal")
plt.show()
```

2)상대도수 히스토그램
-데이터 수가 다른 그룹의 히스토그램을 비교하는 경우에는 상대도수를 이용해서 히스토그램화하면 비교가 용이하다.
-상대도수 히스토그램을 그리는 경우에는 인수 normed에 True를 지정한다.
-상대도수 히스토그램에서는 상대도수의 합계가 1이 된다.

```
fig = plt.figure()
ax = fig.add_subplot(111)

# normed을 True로 지정
ax.hist(df["rating"], normed=True, rwidth=0.9)
plt.show()
```

3)누적 히스토그램(누적도수 그림)
-누적도수를 확인하는 경우에는 누적 히스토그램을 이용한다.
-누적 히스토그램을 그리는 경우 인수 cumulative에 True를 지정한다.

```
fig = plt.figure()
ax = fig.add_subplot(111)

# cumulative를 True로 지정
ax.hist(df["rating"], normed=True, cumulative=True, rwidth=0.9)
plt.show()
```

4)계급 폭 지정
-bins 옵션에 list형 수열을 부여하는 것에 따라 계급 폭을 지정할 수 있다.
-계급 폭은 같은 간격이 아니어도 상관없다.

```
fig = plt.figure()
ax = fig.add_subplot(111)
ax.hist(df["rating"], bins=[2, 4, 5.5, 6.5, 7, 7.5, 8.5, 10], rwidth=0.9)
plt.show()
```

5)근사 곡선 추가
-근사 곡선은 히스토그램을 그린 후에 꺾은선 그래프로 그린다.

-근사 곡선은 다음 단계로 그린다.
a. df['rating'] 데이터 세트의 평균값과 표준편차 구하기
b. numpy.linespace로 각 막대 단락 값(막대의 상한값과 하한값) 구하기
c. 구해진 평균값, 표준편차, 단락값으로부터 정규분포의 확률밀도함수에 따라 Y값 산출하기
d. 구해진 X값과 Y값으로 근사 곡선 그리기

```
bins = 50  # 막대수
dfmin = np.min(df["rating"])  # 데이터 최소값
dfmax = np.max(df["rating"])  # 데이터 최대값

#히스토그램 그리기
fig = plt.figure()
ax = fig.add_subplot(111)
ax.hist(df["rating"], bins=bins, range=(dfmin, dfmax), normed=True, rwidth=0.9)

# 평균과 표준편차
mu, sigma = df["rating"].mean(), df["rating"].std()

#X값
x = np.linspace(dfmin, dfmax, bins)  # 막대 단락 값

#근사적 확률밀도함수를 사용해 Y값 생성
y = 1 / (sigma * np.sqrt(2 * np.pi)) * np.exp(-(x - mu) ** 2 / (2 * sigma ** 2))

# 근사 곡선 그리기
ax.plot(x, y)
plt.show()
```

6)여러 그룹을 겹쳐서 그리기
-같은 subplot에 histgram을 반복해서 그리면 여러 그룹의 histgram을 겹쳐서 그리는 것이 가능하다.
-평균이 0부터 10의 범위에서 실행되기 때문에 range()함수를 사용해서 b_num에 0.5씩 0부터 10 사이의 수치를 저장하고 있다.
-또한 히스토그램이 겹쳐져 그려지기 때문에 alpha 옵션으로 불투명도를 낮추고 있다.
-투명도는 0인경우 완전 투명, 1이면 완전히 불투명이 된다.

```
types = df["type"].unique()
labels = types.tolist()
fig = plt.figure(figsize=(8, 6))
ax = fig.add_subplot(111)
b_num = np.arange(0, 10.5, 0.5)

for t in types:
    ax.hist(df.loc[df["type"] == t, "rating"], bins=b_num, rwidth=0.9, alpha=0.5, label=t)

ax.legend()
ax.set_xlabel("rating")
ax.set_ylabel("Count(rating)")
plt.show()
```

7)여러 그룹을 나열하여 그리기
-여러 그룹의 히스토그램을 겹쳐 그려서 시인성이 떨어지는 경우에는 그룹을 나열하는 방법이 있다.
-중첩 list를 작성한 후 그리면 여러 그룹을 옆으로 나열한 히스토그램을 그릴 수 있다.

```
dataset = [df.loc[df["type"] == t, "rating"] for t in types]
fig = plt.figure(figsize=(8, 6))
ax = fig.add_subplot(111)
ax.hist(dataset, bins=np.arange(0, 10.5, 0.5), rwidth=0.9, alpha=0.8, label=labels)
ax.legend()
ax.set_xlabel("rating")
ax.set_ylabel("Count(rating)")
plt.show()
```

8)여러 그룹을 누적해서 그리기
-여러 그룹의 히스토그램을 그려서 전체의 분포와 그 내역을 확인하는 경우에는 누적 히스토그램이 유효하다.
-여러 그룹을 나열해서 그린 방법과 같이 데이터 세트를 작성한 후 인수 stacked에 True를 지정해서 그리면 누적 히스토그램이 된다.

```
fig = plt.figure(figsize=(8, 6))
ax = fig.add_subplot(111)
ax.hist(dataset,
        bins=np.arange(0, 10.5, 0.5),
        rwidth=0.9,
        alpha=0.7,
        label=labels,
        stacked=True)
ax.legend()
ax.set_xlabel("rating")
ax.set_ylabel("Count(rating)")
plt.show()
```

16. 상자수염 그래프

1)상자수염 그래프는 데이터의 불균형을 알기 쉽게 표현하는 그래프이다.
2)상자수염 그래프 요소
-제3사분위점:모든 데이터의 하위부터 3/4로 나눈 값(=Q3), 상자의 상부 끝
-중앙값 : 모든 데이터의 하위부터 1/2로 나눈 값(=제2사분위점)
-제1사분위점 : 모든 데이터의 하위부터 1/4로 나누 값(=Q1), 상자의 하부 끝
-수염 상부 끝 : Q3 + 1.5 x IQR
-수염 하부 끝 : Q1-1.5 x IQR
-IQR : 사분위 범위(=Q3Q1)
-벗어난 값 : 수염의 하부 끝 ~ 상부 끝의 범위 외에 있는 데이터

3)상자수염 그래프 작성하기
-Axes.boxplot()를 사용해서 그린다.

```
plt.style.use("ggplot")
x = [1, 2, 3, 3, 11, 20]
fig = plt.figure()
ax = fig.add_subplot(111)
ax.boxplot(x)
plt.show()
```

4)여러 개의 상자수염 그래프를 그리는 경우
-복수의 list를 부여하면 여러 개의 상자수염 그래프를 그릴 수 있다.

```
x = [[1, 2, 3, 3, 11, 20], [1, 2, 9, 10, 15, 16]]
labels = ["A", "B"]
fig = plt.figure()
ax = fig.add_subplot(111)

#데이터와 레이블 지정
ax.boxplot(x, labels=labels)
plt.show()
```

5)상자수염 그래프 활용하기
-실제 데이터를 이용해서 그린다.
-데이터는 히스토그램에서 사용했던 anime_master.csv 파일을 이용한다.

```
import os
import pandas as pd

base_url = "https://raw.githubusercontent.com/practical-jupyter/sample-data/master/anime/"
anime_master_csv = os.path.join(base_url, "anime_master.csv")
df = pd.read_csv(anime_master_csv, index_col="anime_id")
df.head(3)
-------------------------------------------------------
```

-배급 종별마다 에피소드 수의 상자수염 그래프를 작성한다.
-배급 종별은 6종류이기 때문에 6개의 상자수염 그래프가 출력된다.

```
labels = []
types_list = []

#배급 종별마다 에피소드 수 정보를 list화
for label, df_per_type in df.groupby("type"):
    labels.append(label)
    types_list.append(df_per_type["episodes"].tolist())

fig = plt.figure()
ax = fig.add_subplot(111)
ax.boxplot(types_list, labels=labels)
plt.show()
```

-위 코드의 결과를 보면, 텔레비젼 애니메이션을 의미하는 TV에만 큰 값이 포함되어 있는 것을 확인할 수 있다.
-그래서 상자수염 그래프 그리기 범위를 지정하면 에피소드수 0부터 100까지의 값에 한정된다.

```
fig = plt.figure(figsize=(8, 6))
ax = fig.add_subplot(111)
ax.boxplot(types_list, labels=labels)

# Y축 그리기 범위를 0부터 100까지 한정
ax.set_ylim(0, 100)
plt.show()
```

6)상자수염 그래프의 서식 일괄 설정하기
-상자수염 그래프의 서식은 각 요소의 서식을 사전 형식으로 부여하여 일괄로 설정할 수 있다.
-요소에 따라 설정 가능한 항목이 다르지만 상자 부분은 patches.PathPatch 클래스로, 그 이외의 요소는 lines.Line2D 클래스의 인스턴스로 그려진다.
-상자수염 그림의 주요 서식 설정 항목
a. color : 색
b. facecolor : 채움색
c. linestyle : 선 종류
d. linewidth : 선 굵기
e. maker : 마커

f. makerfacecolor : 마커 채움색
g. makeredgecolor : 마커 테두리선 색
h. makersize : 마커 크기

-서식 일괄 순서
a. 데이터 세트 작성하기

```
import numpy as np

np.random.seed(3)
dataset = [np.random.normal(20 + mu, 5, 1000) for mu in range(1, 5)]
```

b. 서식 사전 만들기
-상자에 서식을 설정하기 위해 사전을 만든다.
-상자의 요소, '벗어난 값', '상자', '수염', '수염끝단', '중앙값', '평균값'의 서식을 설정할 수 있다.

```
#벗어난 값의 서식 사전
flierprop = {"color": "#EC407A",
             "marker": "o",
             "markerfacecolor": "#2196F3",
             "markeredgecolor": "white",
             "markersize": 5,
             "linestyle": "None",
             "linewidth": 0.1}

# 상자의 서식 사전
boxprop = {"color": "#2196F3",
           "facecolor": "#BBDEFB",
           "linewidth": 1,
           "linestyle": "-"}

# 수염의 서식 사전
whiskerprop = {"color": "#2196F3",
               "linewidth": 1,
               "linestyle": "--"}

# 수염 끝단 서식 사전
capprop = {"color": "#2196F3",
           "linewidth": 1,
           "linestyle": ":"}

# 중앙값 서식 사전
medianprop = {"color": "#2196F3",
              "linewidth": 2,
              "linestyle": "-"}

# 평균값 서식 사전
meanprop = {"color": "#2196F3",
            "marker": "^",
            "markerfacecolor": "#2196F3",
            "markeredgecolor": "white",
            "markersize": 10,
            "linewidth": 1,
            "linestyle": ""}
```

c. 그리기

```
fig = plt.figure()
ax = fig.add_subplot(111)
ax.boxplot(
    dataset,
    patch_artist="Patch",  # 서식을 설정하는 경우「Patch」 를 선택
    labels=["A", "B", "C", "D"],  # 항목 레이블
    showmeans=True,  # 평균값 그리기
    flierprops=flierprop,  # 벗어난 값 서식 설정
    boxprops=boxprop,  # 상자 서식 설정
    whiskerprops=whiskerprop,  # 수염 서식 설정
    capprops=capprop,  # 수염 끝단 서식 설정
    medianprops=medianprop,  # 중앙값 서식 설정
    meanprops=meanprop,  # 평균값 서식 설정
)
plt.show()
```

7)상자마다 서식 설정하기
-서식을 개별로 설정하는 것도 가능하다.
-상자의 서식을 요소마다 설정하는 경우에는 각 항목에 접두사 set을 붙여서 이용한다.
-다음 순서로 서식을 설정해서 그린다.

a. 그림 그리기
b. 상자 요소 수와 같은 요소 수의 색 세트(컬러 세트)(colors1과 colors2)를 작성하기
c. 위쪽과 아래쪽이 나눠져 있는 요소의 서식 설정용에 수열 list n을 작성하기
d. 상자와 벗어난 값, 중앙값(요소가 상하로 나워져 있지 않은 것 또한 상하 같은 색을 부여한 것)의 서식 설정하기
e. 수염과 수염의 끝단(요소가 상하로 나워져 있는 것 상하 다른 색을 설정할 수 있는 것)의 서식 설정하기

f. 평균값의 서식 설정하기

```
# 그림 그리기
fig = plt.figure()
ax = fig.add_subplot(111)

bp = ax.boxplot(
    dataset,
    patch_artist="Patch",
    labels=["A", "B", "C", "D"],
    meanline=True,
    showmeans=True,
)

# 컬러 세트
colors1 = ["#2196F3", "#43A047", "#FBC02D", "#FB8C00"]
colors2 = ["#BBDEFB", "#C8E6C9", "#FFF9C4", "#FFE0B2"]

# 위 아래오 나위어진 요소에 설정하기 위해 용도의 수열
n = [0, 0, 1, 1, 2, 2, 3, 3]

# 서식 설정
# 상자와 벗어난 값, 중앙값의 서식 설정
for params in zip(bp["boxes"], bp["fliers"], bp["medians"], colors1, colors2):
    bpb, bpf, med, color1, color2 = params

    # 상자 서식 설정
    bpb.set_color(color1)
    bpb.set_facecolor(color2)
    bpb.set_linewidth(2)

    # 벗어난 값 서식 설정
    bpf.set(marker="^", color=color2)
    bpf.set_markeredgecolor("white")
    bpf.set_markerfacecolor(color1)

    # 중앙값 서식 설정
    med.set_color(color1)
    med.set_linewidth(2)

#수염과 수염 끝단 서식 설정
for bpc, bpw, m in zip(bp["caps"], bp["whiskers"], n):
    bpc.set_color(colors1[m])
    bpc.set_linewidth(2)
    bpw.set_color(colors1[m])
    bpw.set_linewidth(2)

# 평균값 서식 설정
for mean, color2 in zip(bp["means"], colors2):
    mean.set_color("grey")
    mean.set_linewidth(2)
    mean.set_linestyle("--")

plt.show()
```

17. 원 그래프

1)원 그래프는 전체에 대한 각 요소의 비율을 추출하고, 추출한 비율에 따라 원형을 부채꼴로 분할한 그래프이다.
2)원 그래프는 각 요소의 비율을 비교할 때 유용하다.
3)원 그래프 그리기
-Axes.pie()를 사용한다.
-제1인수에 요소의 값을 부여해서 그린다.

```
plt.style.use("ggplot")
labels = ["자전거", "버스" ,"차"]
sizes = [25, 40, 35]
fig = plt.figure(figsize=(3, 3))
ax = fig.add_subplot(111)

ax.pie(sizes, labels=labels)
plt.show()
```

-초기 설정에서는 도수법으로 0도의 위치(시계 세 시의 위치)에서 반시계 방향으로 요소를 그려간다.
-중심 좌표는 (0,0), 반경은 1이다.
-다음 코드에서는 radius에 0.9를, 인수 frame에 True를 설정해서 축과 함께 그래프를 그린다.

```
fig = plt.figure(figsize=(3, 3))
ax = fig.add_subplot(111)
ax.pie(sizes, labels=labels, radius=0.9, frame=True)
ax.text(-0.3, 0, "(0, 0)", fontsize=9)
plt.show()
```

-위의 코드에서 원의 중심 좌표는 (0,0), 처음 요소 자전거가 좌표(0.9, 0)에서 시작해서 부채꼴로 그려져 있는 것을 확인할 수 있다.

-계속해서 두 번째 요소 '버스'가 좌표(0, 0.9)에서 시작해서 부채꼴로 그려져 있다.

4)원 그래프 서식 설정
-explode : 각 요소를 분리하여 표시하는 경우에 설정. list형 또는 tuple형 지정. 예를 들어 요소가 4개 있고 3번째 요소를 분리하고 싶은 경우에는 -(0,0,0.5,0)]과 같이 지정.
-labels : 레이블 표시. list형 또는 tuple형으로 지정.
-colors : 각 요소의 색을 설정. list형 또는 tuple형으로 지정.
-autopct : 수치 레이블 서식 설정. 표시 형식은문자열로 지정.
-pctdistance : 수치 레이블의 위치 지정. 수치열로. 수치는 각 요소의 중심부터의 거리가 되고 explode를 설정하고 있는 경우에도 이 거리도 가산됨.
-shadow : 배경의 표시/비표시 설정. 논리값
-labeldistance : 레이블의 위치 설정. 수치열로. 수치는 각 요소의 중심부터의 거리가 되고 explode를 설정하고 있는 경우에도 이 거리도 가산됨.
-startangle : 시작 각도 설정. 단위는 도수법으로 수치형으로
-radius : 반경을 설정. 수치열(초기 설정 1)
-counterclock : 표시순서 설정. 논리값. True(반시계방향), False(시계방향)
-wedgeprops : 각 요소의 서식 설정. 서식을 등록한 dict.
-textprops : 텍스트의 서식 설정. 서식을 등록한 dict.
-center : 원 그래프의 중심 좌표 설정. tuple형
-frame : 축/테두리선의 유무 설정. 논리값.

5)원 그래프의 서식 설정을 하는 경우

```
fig = plt.figure(figsize=(3, 3))
ax = fig.add_subplot(111)

#부채꼴 서식 설정용 사전
wprops = {"edgecolor": "black", "linewidth": 2}

# 텍스트 서식 설정용 사전
tprops = {"fontsize": 18}
ax.pie(
    sizes,
    explode=(0.0, 0.05, 0),
    labels=labels,
    autopct="%1.0f%%",
    pctdistance=0.5,
    shadow=False,
    labeldistance=1.35,
    startangle=90,
    radius=0.3,
    counterclock=False,
    wedgeprops=wprops,
    textprops=tprops,
    center=(0.5, 0.5),
    frame=True,
)
plt.show()
```

6)원 그래프 활용하기
-실제의 데이터를 이용해서 그려보자.
-데이터는 anime_genre_top10_pivoted.csv 파일이다.

```
import os
import pandas as pd

base_url = "https://raw.githubusercontent.com/practical-jupyter/sample-data/master/anime/"
anime_genre_top10_pivoted_csv = os.path.join(base_url, "anime_genre_top10_pivoted.csv")
df = pd.read_csv(anime_genre_top10_pivoted_csv, index_col="genre")
df
------------------------------------------------------------
```

-원 그래프는 데이터의 비율을 비교할 때 유용한 그래프이다.
-여기서는 Movie와 TV의 총 멤버수 내역을 원그래프로 그려서 장르의 내역 비율을 비교한다.
-원 그래프는 90도 위치에서 시계 방향으로 내림차순으로 요소를 나열하는 것이 일반적이다.
-먼저 TV와 Movie의 데이터를 각각 내림차순으로 정렬한 Series를 작성한다.

```
df_tv = df.sort_values(by="TV", ascending=False)["TV"]

df_movie = df.sort_values(by="Movie", ascending=False)["Movie"]
df_tv
------------------------------------------------
```

-멤버 수가 많은 Comedy에서 내림차순으로 데이터가 내열되어 있다.
-다음 코드는 소트한 Movie의 데이터를 사용해서 그래프를 그린다.

```
fig = plt.figure(figsize=(9, 4))
ax1 = fig.add_subplot(121)
ax2 = fig.add_subplot(122)

#컬러 세트
colors1 = (
    "gold",
```

```
            "coral",
            "plum",
            "orchid",
            "lightseagreen",
            "yellowgreen",
            "lightskyblue",
            "pink",
            "cornflowerblue",
            "orangered",
        )
        colors2 = (
            "coral",
            "orangered",
            "plum",
            "pink",
            "gold",
            "cornflowerblue",
            "yellowgreen",
            "lightseagreen",
            "orchid",
            "lightskyblue",
        )

        # TV 원 그래프
        ax1.pie(
            df_tv,
            explode=(0, 0, 0, 0, 0, 0, 0.15, 0, 0, 0.15),
            labels=df_tv.index,
            autopct="%1.0f%%",
            colors=colors1,
            startangle=90,
            counterclock=False,
        )

        # Movie 원 그래프
        ax2.pie(
            df_movie,
            explode=(0, 0.15, 0, 0, 0, 0, 0, 0, 0, 0.15),
            labels=df_movie.index,
            autopct="%1.0f%%",
            colors=colors2,
            startangle=90,
            counterclock=False,
        )
        ax1.set_title("TV")
        ax2.set_title("Movie")
        plt.subplots_adjust(wspace=0.3)  # 서브블록 사이의 공간 조정
        plt.show()
```