WordCloud

wordcloud

파이썬 기반 데이터 시각화 중 가장 손쉽게 데이터를 시각화 해주는 워드클라우드는 텍스트의 빈도수 등을 고려해서, 글자의 크기, 색깔 등등이 강조할 수 있습니다.
주로 뉴스 기사 내용이나 SNS 글들을 수집하여 자주 사용하는 텍스트를 구분하여 빠르게 분석할 수 있는 기능 입니다.
결과물은 이미지 형태로 저장해서 웹에서 보여 줄 수 있습니다.

wordcloud

먼저 웹에서 간단하게 체험할 수 있는 사이트를 이용해 보겠습니다.
워드클라우드 생성기 : http://wordcloud.kr/

wordcloud

무료 폰트 설치 - 나눔 글꼴 설치

나눔고딕,나눔고딕,나눔손글씨 : http://hangeul.naver.com/download.nhn
고양체 : http://www.goyang.go.kr/kr/intro/sub03/09/
이순신 돋움체 : http://www.asan.go.kr/font
호국체 :http://blog.naver.com/mnd9090/221023344796
배달의민족 : http://www.woowahan.com
몬소리체 : https://brunch.co.kr/@creative/32
어비유종체 : https://noonnu.cc/font_page/198
해수체 : https://m.blog.naver.com/PostView.nhn?blogId=koreamof&logNo=120199897671
영남일보체 : http://m.yeongnam.com/jsp/view.jsp?nkey=20181009.990010819551403
부산체 : http://www.busan.go.kr/bhbusan
가비아체(봄바람,납작블록체) : https://company.gabia.com/font
다래손글씨체 : http://blog.naver.com/drstyle/30118003234
뚜꺼비체 : https://m.blog.naver.com/PostView.nhn?blogId=hp0&logNo=221206126048
포천막걸리체 : http://www.pocheon.go.kr/www/contents.do?key=5582
롯데면세점체 : http://lottedfs.voltpage.net/write
제주한라산체 : https://www.jeju.go.kr/jeju/symbol/font/infor.htm
여름물빛체 : http://blog.naver.com/PostThumbnailView.nhn?blogId=hp0&logNo=221059577653
겨울눈꽃체 : http://blog.naver.com/PostThumbnailView.nhn?blogId=hp0&logNo=221170423317
애터미체 : http://www.atomy.kr/v2/Home/About/FontsInfo
넷마블체 : http://company.netmarble.com/company/ci?tab=2
산돌떡볶이체 : http://kukde.co.kr/?page_id=627
배스킨라빈스체 : http://www.baskinrobbins.co.kr/event/view.php?flag=&seq=3722
야놀자체 : http://yanolja.in/ko/yafont/
tvn체 : http://tvn10festival.tving.com/playground/tvn10font
빙그레체 : http://www.bingfont.co.kr/bingfont.html

wordcloud

1) 워드 클라우드 설치

```
pip install wordcloud
pip3 install matplotlib
```

2) 네이버 뉴스 사이트의 기사 내용을 "news.txt" 파일로 저장 합니다.

wordcloud

```
import os
import matplotlib.pyplot as plt
from wordcloud import WordCloud
font_path = “./NanumGothicExtraBold.ttf"
wordcloud = WordCloud(font_path=font_path, background_color="white", width=400,
height=400)
try:
    f = open("news.txt", "rt", encoding='UTF8')
    text = f.read()
except FileNotFoundError:
    print("파일이 없습니다.")
finally:
    f.close()

wordcloud = wordcloud.generate(text)
fig = plt.figure(figsize=(6,6))
plt.imshow(wordcloud)
plt.axis("off")
plt.show()
fig.savefig("wordcloud.png")
```

wordcloud

갤럭시폴드는
모습
출시
방송이
4월
23일
채로
현지
연기됐다
CNBC
시각
제공
CNN
즐기는
26일
27일
화면과
경첩
현지
미국에
폴더블폰
펼친
삼성전자가
게임을
예정이었지만
삼성전자
18일
결함
영국
도입
이어
프랑스
4개국에
9월
이후
진동을

wordcloud

미션1.
다양한 기사를 넣어서 실습해 보기

미션2.
다양한 크기와 모양으로 실습해 보기

미션2.
네이버 뉴스를 크롤링해서 기사 내용을 출력하고 워드클라우드로도 출력해 보세요!

wordcloud

feedparser는 RSS feed 파싱 라이브러리
1) feedparser 설치

```
pip install feedparser
```

2) 뉴스 링크 수집

```
import feedparser

#  경향닷컴 경제뉴스 RSS
feeds = feedparser.parse('https://www.khan.co.kr/rss/rssdata/total_news.xml')
links = [entry['link'] for entry in feeds['entries']]
links
```

```
['http://news.khan.co.kr/kh_news/khan_art_view.html?artid=201203251835221&code=920100',
 'http://news.khan.co.kr/kh_news/khan_art_view.html?artid=201203251356231&code=920401',
 'http://news.khan.co.kr/kh_news/khan_art_view.html?artid=201203251356161&code=920401',
 'http://news.khan.co.kr/kh_news/khan_art_view.html?artid=201203251355591&code=920401',
 'http://news.khan.co.kr/kh_news/khan_art_view.html?artid=201203251128101&code=920100',
 'http://news.khan.co.kr/kh_news/khan_art_view.html?artid=201203231743501&code=920301',
 'http://news.khan.co.kr/kh_news/khan_art_view.html?artid=201203231411441&code=920100',
 'http://news.khan.co.kr/kh_news/khan_art_view.html?artid=201203231410391&code=920100',
```

wordcloud

newspaper는 사용자가 지정한 url에서 text를 추출해주는 모듈이다.
https://pypi.org/project/newspaper/
1) newspaper 설치 (python2와 python3에서 각각 설치 방법이 다르다.)

```
pip install newspaper3k
```

2) 뉴스 링크 수집

```
import newspaper

news_text =''
for link in links:
    article = newspaper.Article(link, language='ko')
    article.download()
    article.parse()
    news_text += article.text

news_text[:1000]
```

'·중기청장과 밤샘토론₩n₩n₩n₩n "정부가 창업 숫자를 강조하기보다 성공률을 생각해야 한다." "좋은 제품을 개발해도 벤처기업은 판로를 뚫기가 어렵다." "연구·개발 투자에만 2년이 걸리는데 창업자금은 3년 안에 갚으라고 한다." ₩n₩n₩n₩n지난 22일 경기 안산시 청년창업사관학교. 송종호 중소기업청장(56)과 청년창업자 및 사관학교 입교자 150여명이 밤샘토론을 벌였다. 창업자들은 현장에서 겪는 갖가지 어려움을 쏟아냈다. 자금만 찔끔 지원하고는 후속대책은 세워주지 않는 정부의 무책임에 대한 질타도 있었다. 22일 자정 시작된 토론회는 예정된 시간보다 1시간30분 늦은 23일 새벽 4시30분이 돼서야 마무리됐다.₩n₩n₩n₩n신발제조업체를 운영하는 권종익씨(29)는 "중기청에서 기술개발자금을 지원하지만 막상 개발을 마치고

wordcloud

수집된 기사 내용을 토대로 워드 클라우드로 만들어 보세요!

```
import matplotlib.pyplot as plt
from wordcloud import WordCloud
font_path = "./NanumGothicExtraBold.ttf"
wordcloud = WordCloud(font_path=font_path, background_color="white", width=400,
height=400)
wordcloud = wordcloud.generate(news_text)
fig = plt.figure(figsize=(6,6))
plt.imshow(wordcloud)
plt.axis("off")
plt.show()
fig.savefig("wordcloud.png")
```

matplotlib

1. matplotlib

Python에서 데이터 시각화하는 다양한 방법

데이터 분석가들은 주로 Python(또는 R, SQL)을 가지고 데이터 분석을 합니다.

- R에는 ggplot이란 시각화에 좋은 라이브러리가 있는 반면 Python에도 다양한 라이브러리들을 제공 합니다.
- 각 라이브러리들마다 특징이 있기 때문에, 데이터 특성에 맞게 사용 하면 됩니다.
- **논문 또는 레포트 작성할 때 매우 유용 합니다.**
- **plot.ly, pyecharts는 웹에서 정말 강력 합니다.**

라이브러리 종류

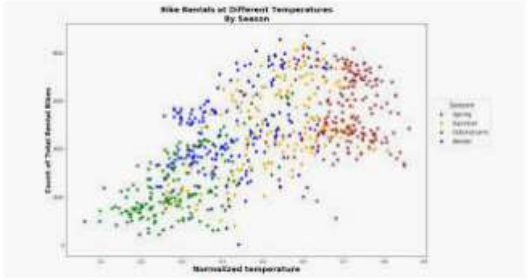
- **matplotlib**, seaborn, plotnine, folium, plot.ly, pyecharts

1. matplotlib

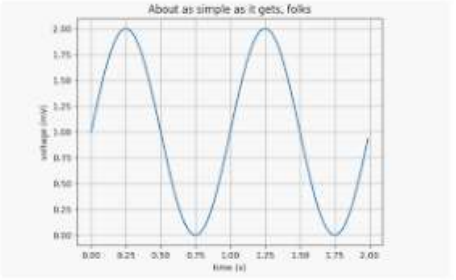
Python에서 아마 가장 많이 사용하는 시각화 라이브러리

Matplotlib는 파이썬에서 자료를 차트(chart)나 플롯(plot)으로 시각화 (visulaization) 하는 패키지이다. Matplotlib는 다음과 같은 정형화된 차트나 플롯 이외에도 저수준 api를 사용한 다양한 시각화 기능을 제공한다.

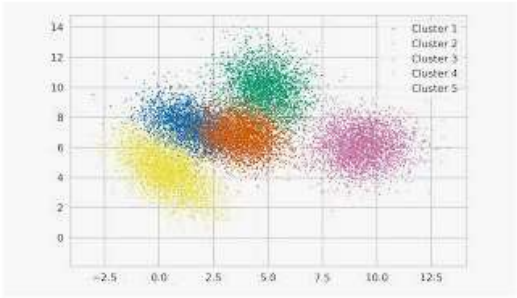
- 라인 플롯(line plot)
- 스캐터 플롯(scatter plot)
- 컨투어 플롯(contour plot)
- 서피스 플롯(surface plot)
- 바 차트(bar chart)
- 히스토그램(histogram)
- 박스 플롯(box plot)



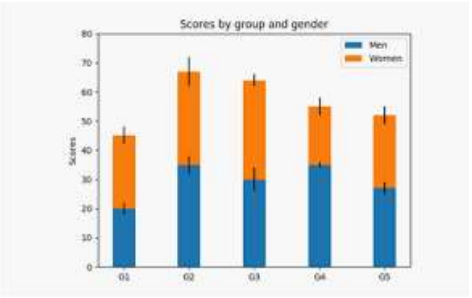
Data Visualization in Python: Matplotlib vs Seaborn
kdnuggets.com



Sample plots in Matplotlib — Matplotlib 3.1.0 d...
matplotlib.org



Resizing Matplotlib Legend Markers
intoli.com



Gallery — Matplotlib 3.1.1 documentation
matplotlib.org

1. matplotlib

Matplotlib를 사용한 시각화 예제들을 보고 싶다면 Matplotlib 갤러리 웹사이트를 방문한다.
http://matplotlib.org/gallery.html

설치

```
pip install matplotlib
```

1. matplotlib

Matplotlib 패키지에는 pylab 라는 서브패키지가 존재한다. 이 pylab 서브패키지는 matlab 이라는 수치해석 소프트웨어의 시각화 명령을 거의 그대로 사용할 수 있도록 Matplotlib 의 하위 API를 포장(wrapping)한 명령어 집합을 제공한다. 간단한 시각화 프로그램을 만드는 경우에는 pylab 서브패키지의 명령만으로도 충분하다. 다음에 설명할 명령어들도 별도의 설명이 없으면 pylab 패키지의 명령라고 생각하면 된다.

Matplotlib 패키지를 사용할 때는 보통 다음과 같이 주 패키지는 mpl 이라는 별칭(alias)으로 임포트하고 pylab 서브패키지는 plt 라는 다른 별칭으로 임포트하여 사용하는 것이 관례이므로 다음과 같이 사용한다.

```
import matplotlib as mpl
import matplotlib.pylab as plt
```

주피터 노트북을 사용하는 경우에는 다음처럼 매직(magic) 명령으로 노트북 내부에 그림을 표시하도록 지정해야 한다.

```
%matplotlib inline
```

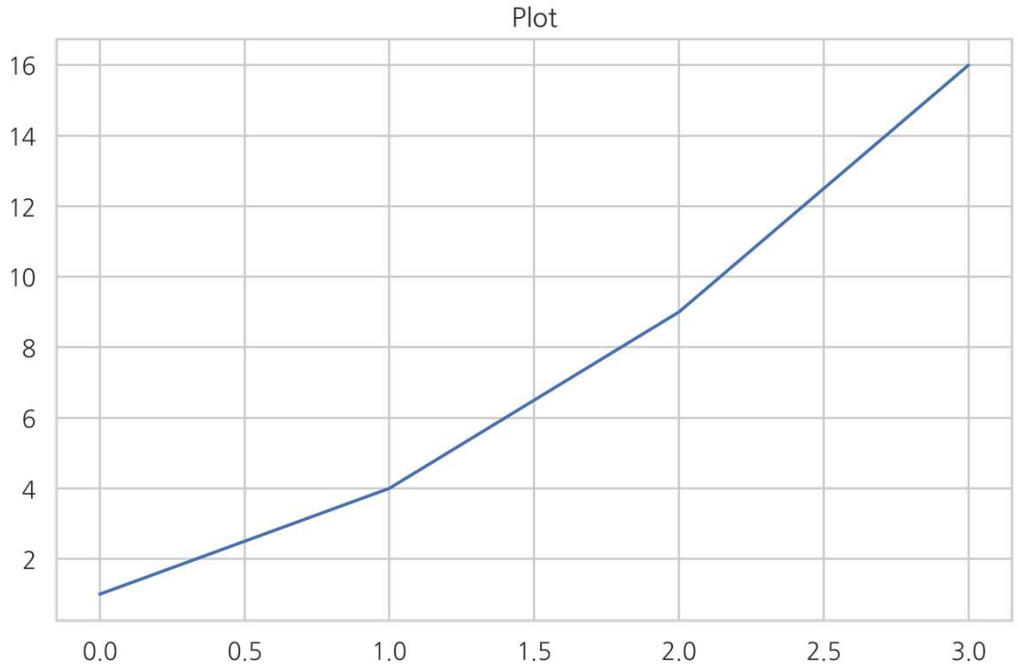
2. 라인 플롯 (line plot)

가장 간단한 플롯은 선을 그리는 라인 플롯(line plot)이다. 라인 플롯은 데이터가 시간, 순서 등에 따라 어떻게 변화하는지 보여주기 위해 사용한다.

명령은 pylab 서브패키지의 plot 명령을 사용한다.

http://Matplotlib.org/api/pyplot_api.html#Matplotlib.pyplot.plot
만약 데이터가 1, 4, 9, 16 으로 변화하였다면 다음과 같이 plot 명령에 데이터 리스트 혹은 ndarray 객체를 넘긴다.

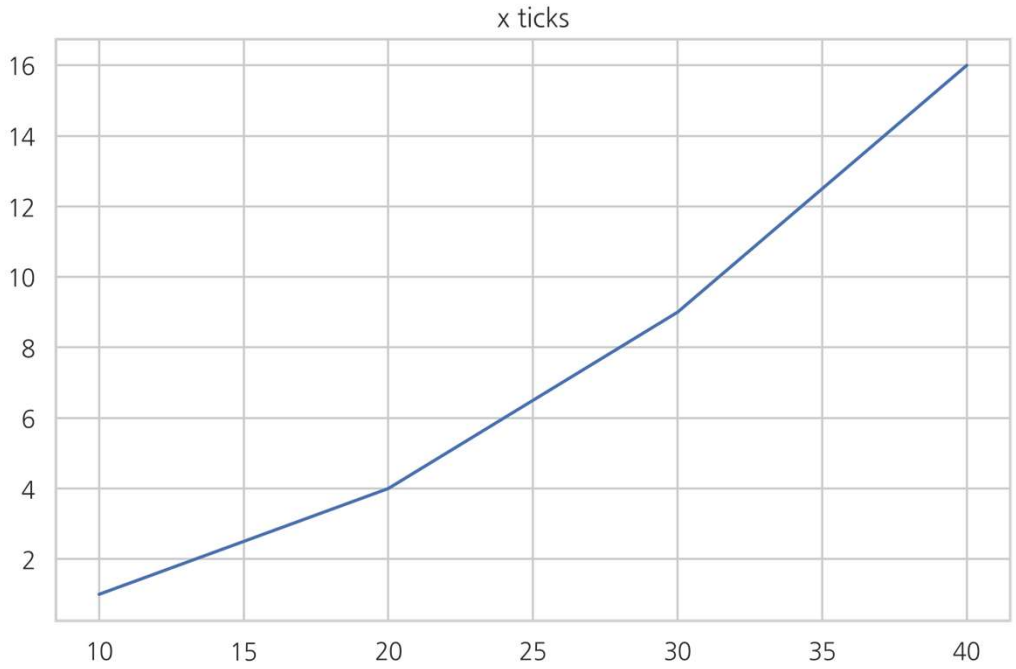
```
plt.title("Plot")
plt.plot([1, 4, 9, 16])
plt.show()
```



2. 라인 플롯 (line plot)

이 때 x 축의 자료 위치 즉, 틱(tick)은 자동으로 0, 1, 2, 3 이 된다. 만약 이 x tick 위치를 별도로 명시하고 싶다면 다음과 같이 두 개의 같은 길이의 리스트 혹은 배열 자료를 넣는다.

```
plt.title("x ticks")
plt.plot([10, 20, 30, 40], [1, 4, 9, 16])
plt.show()
```



show 명령은 시각화 명령을 실제로 차트로 렌더링(rendering)하고 마우스 움직임 등의 이벤트를 기다리라는 지시이다. 주피터 노트북에서는 셀 단위로 플롯 명령을 자동 렌더링 해주므로 show 명령이 필요없지만 일반 파이썬 인터프리터로 가동되는 경우를 대비하여 항상 마지막에 실행하도록 한다. show 명령을 주면 마지막 플롯 명령으로부터 반환된 플롯 객체의 표현도 가려주는 효과가 있다.

2. 라인 플롯 (line plot)

한글폰트 사용

설치된 폰트를 사용하는 방법은 크게 두가지 이다.

- rc parameter 설정으로 이후의 그림 전체에 적용
- 인수를 사용하여 개별 텍스트 관련 명령에만 적용

한글 문자열은 항상 유니코드를 사용해야 한다.

우선 rc parameter를 설정하여 이후의 그림 전체에 적용해 보자

```
# 폰트 설정
mpl.rc('font', family='NanumGothic')
# 유니코드에서  음수 부호설정
mpl.rc('axes', unicode_minus=False)
```

정상적으로 폰트가 설치되고 rc parameter가 제대로 설정되었다면 다음 코드를 실행하였을 때 한글이 잘 보여야 한다.

```
plt.title('한글 제목')
plt.plot([10, 20, 30, 40], [1, 4, 9, 16])
plt.xlabel("엑스축 라벨")
plt.ylabel("와이축 라벨")
plt.show()
```

2. 라인 플롯 (line plot)

한글폰트 사용

설치된 폰트를 사용하는 방법은 크게 두가지 이다.

- rc parameter 설정으로 이후의 그림 전체에 적용
- 인수를 사용하여 개별 텍스트 관련 명령에만 적용

한글 문자열은 항상 유니코드를 사용해야 한다.

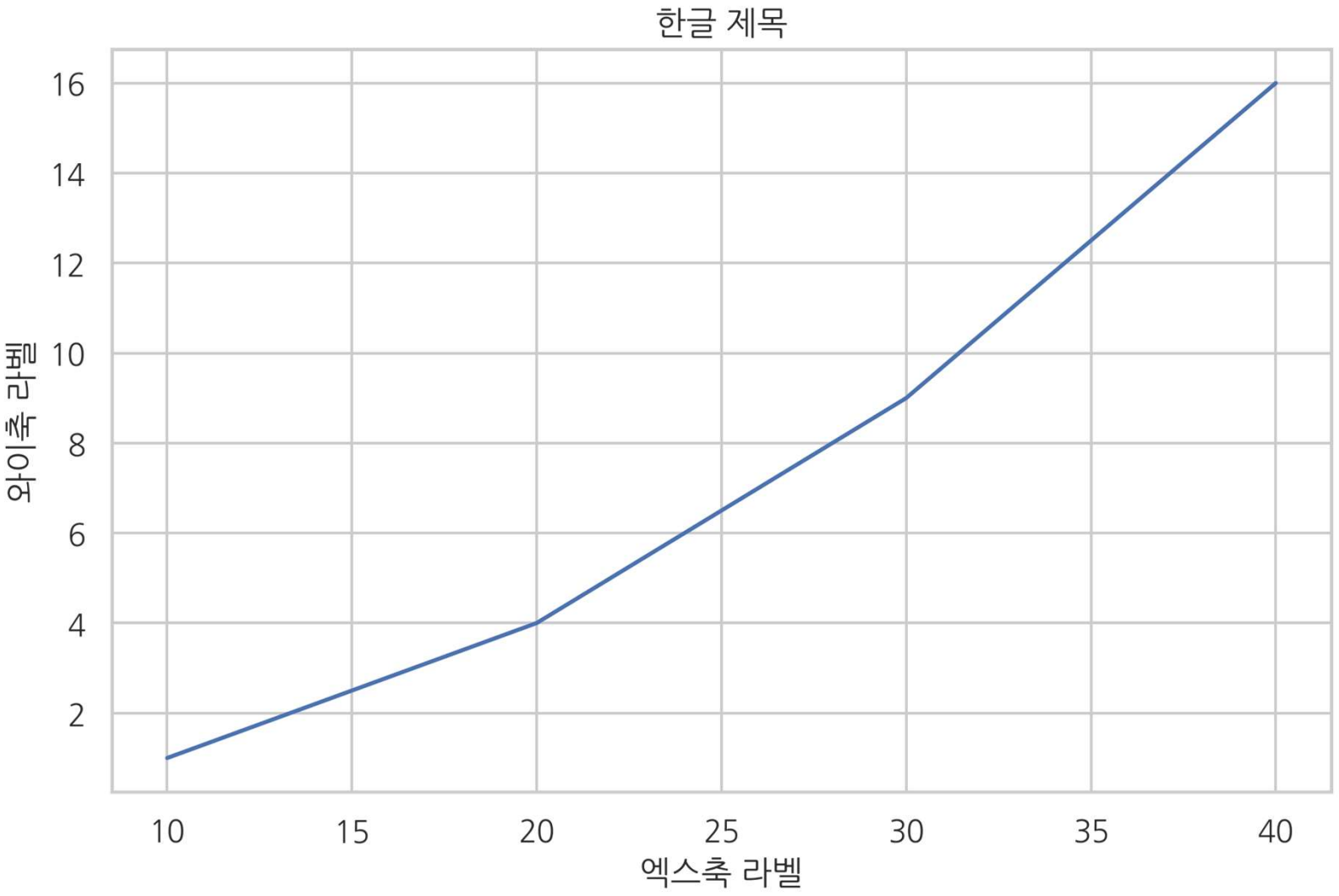
우선 rc parameter를 설정하여 이후의 그림 전체에 적용해 보자

```
# 폰트 설정
mpl.rc('font', family='NanumGothic')
# 유니코드에서  음수 부호설정
mpl.rc('axes', unicode_minus=False)
```

정상적으로 폰트가 설치되고 rc parameter가 제대로 설정되었다면 다음 코드를 실행하였을 때 한글이 잘 보여야 한다.

```
plt.title('한글 제목')
plt.plot([10, 20, 30, 40], [1, 4, 9, 16])
plt.xlabel("엑스축 라벨")
plt.ylabel("와이축 라벨")
plt.show()
```

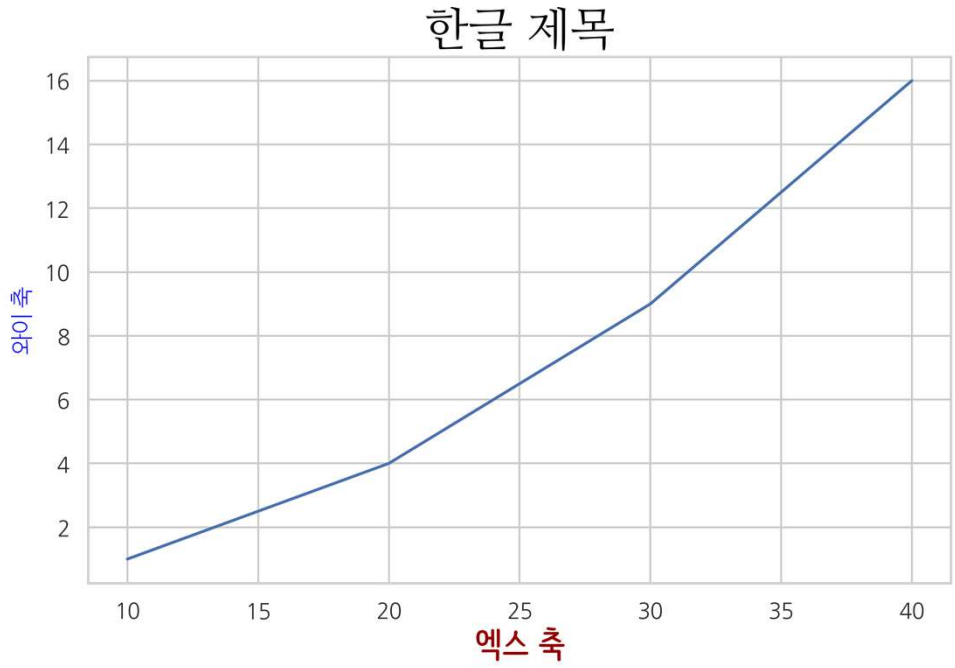
2. 라인 플롯 (line plot)



2. 라인 플롯 (line plot)

만약 객체마다 별도의 폰트를 적용하고 싶을 때는 다음과 같이 폰트 패밀리, 색상, 크기를 정하여 플롯 명령의 fontdict 인수에 넣는다.

```
font1 = {'family': 'NanumMyeongjo', 'size': 24,
'color':  'black'}
font2 = {'family': 'NanumBarunpen', 'size': 18,
'weight': 'bold', 'color':  'darkred'}
font3 = {'family': 'NanumBarunGothic', 'size':
12, 'weight': 'light', 'color':  'blue'}

plt.plot([10, 20, 30, 40], [1, 4, 9, 16])
plt.title('한글 제목', fontdict=font1)
plt.xlabel('엑스 축', fontdict=font2)
plt.ylabel('와이 축', fontdict=font3)
plt.show()
```



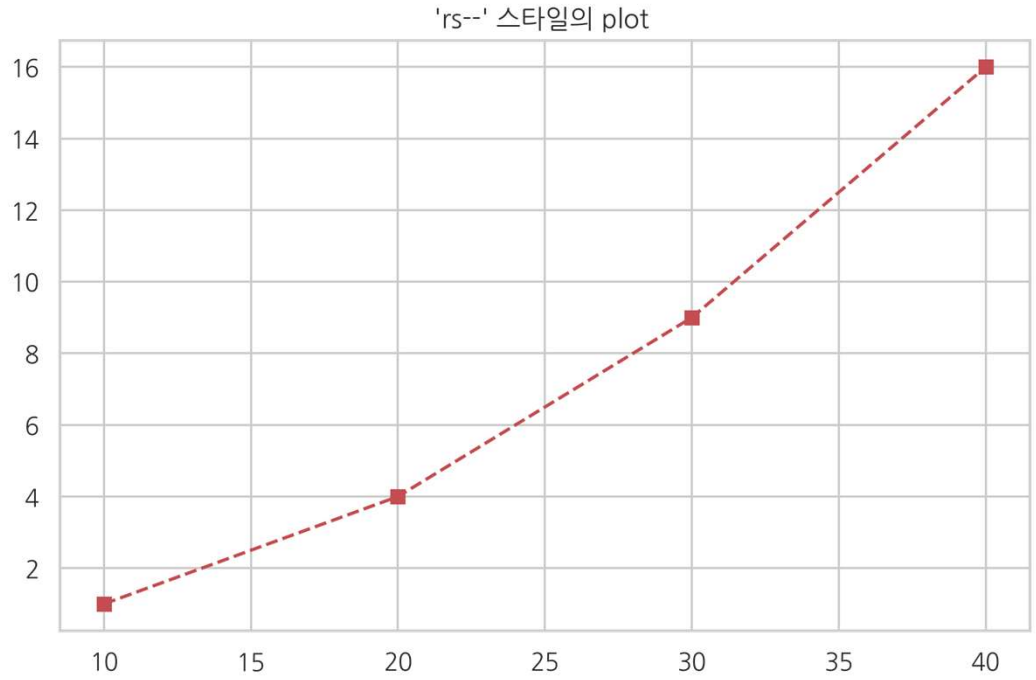
2. 라인 플롯 (line plot)

스타일 지정

플롯 명령어는 보는 사람이 그림을 더 알아보기 쉽게 하기 위해 다양한 스타일(style)을 지원한다. plot 명령어에서는 다음과 같이 추가 문자열 인수를 사용하여 스타일을 지원한다.

```
plt.title("'rs--' 스타일의 plot ")
plt.plot([10, 20, 30, 40], [1, 4, 9, 16], 'rs--')
plt.show()
```

스타일 문자열은 색깔(color), 마커(marker), 선 종류(line style)의 순서로 지정한다. 만약 이 중 일부가 생략되면 디폴트값이 적용된다.



2. 라인 플롯 (line plot)

색깔

색깔을 지정하는 방법은 색 이름 혹은 약자를 사용하거나 # 문자로 시작되는 RGB코드를 사용한다.

자주 사용되는 색깔은 한글자 약자를 사용할 수 있으며 약자는 아래 표에 정리하였다. 전체 색깔 목록은 다음 웹사이트를 참조한다.

http://Matplotlib.org/examples/color/named_colors.html

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

표 : Matplotlib에서 자주 사용하는 Color 약자

2. 라인 플롯 (line plot)

마커
데이터 위치를 나타내는 기호를 마커(marker)라고 한다. 마커의 종류는 다음과 같다.

마커 문자열	의미
.	point marker
,	pixel marker
o	circle marker
v	triangle_down marker
^	triangle_up marker
<	triangle_left marker
>	triangle_right marker
1	tri_down marker
2	tri_up marker
3	tri_left marker
4	tri_right marker
s	square marker

2. 라인 플롯 (line plot)

p	pentagon marker
*	star marker
h	hexagon1 marker
H	hexagon2 marker
+	plus marker
x	x marker
D	diamond marker
d	thin_diamond marker

표 : Matplotlib에서 자주 사용하는 Marker 종류

2. 라인 플롯 (line plot)

선 스타일
선 스타일에는 실선(solid), 대시선(dashed), 점선(dotted), 대시-점선(dash-dit) 이 있다.
지정 문자열은 다음과 같다.

선 스타일 문자열	의미
`-`	solid line style
`--`	dashed line style
`-.`	dash-dot line style
`:`	dotted line style

표 : Matplotlib의 선 스타일(line style)

2. 라인 플롯 (line plot)

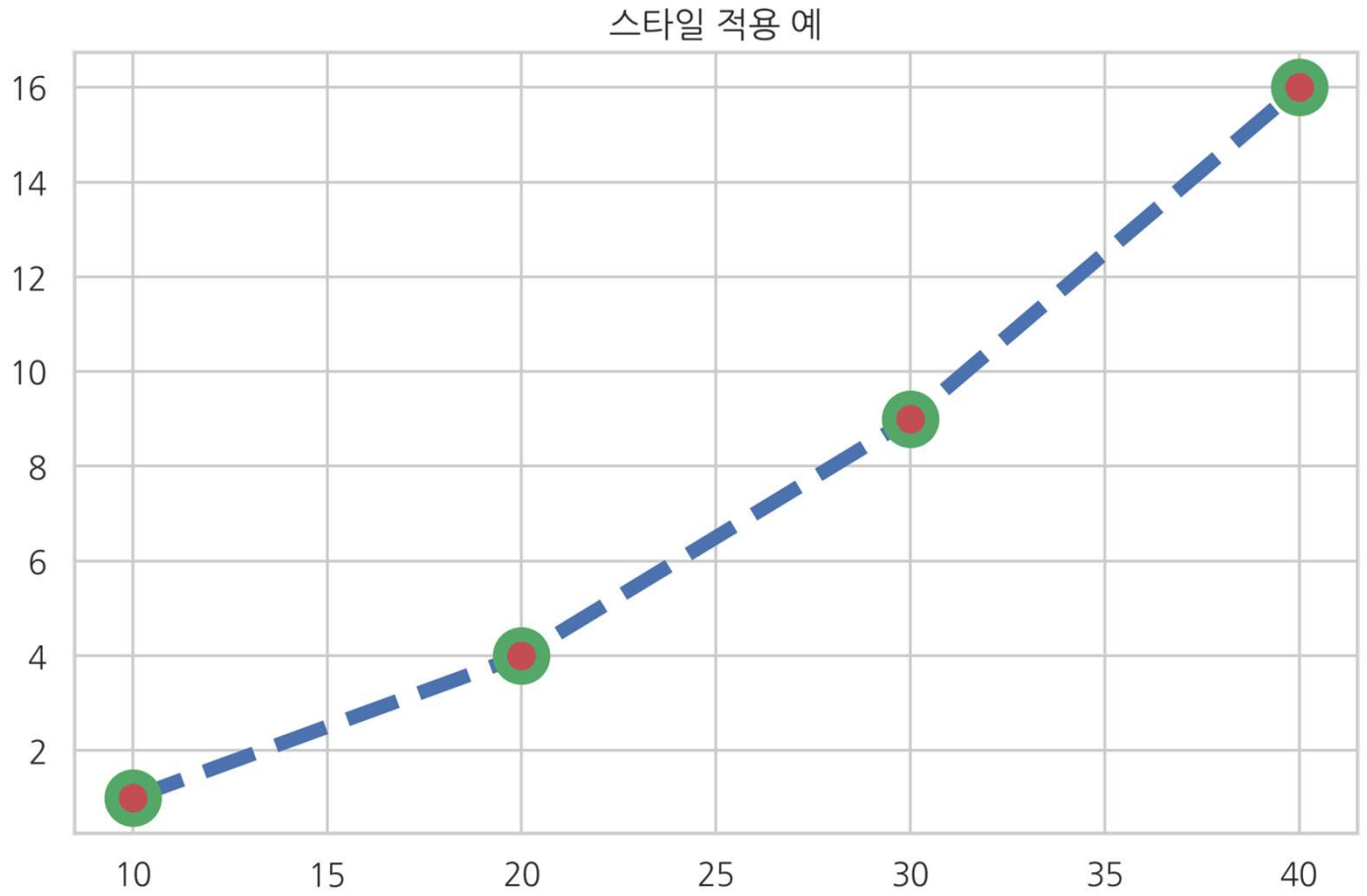
기타 스타일
라인 플롯에서는 앞서 설명한 세 가지 스타일 이외에도 여러가지 스타일을 지정할 수 있지만 이 경우에는 인수 이름을 정확하게 지정해야 한다. 사용할 수 있는 스타일 인수의 목록은 Matplotlib.lines.Line2D 클래스에 대한 다음 웹사이트를 참조한다.

http://Matplotlib.org/api/lines_api.html#Matplotlib.lines.Line2D
라인 플롯에서 자주 사용되는 기타 스타일은 다음과 같다.

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

2. 라인 플롯 (line plot)

```
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")
plt.show()
```



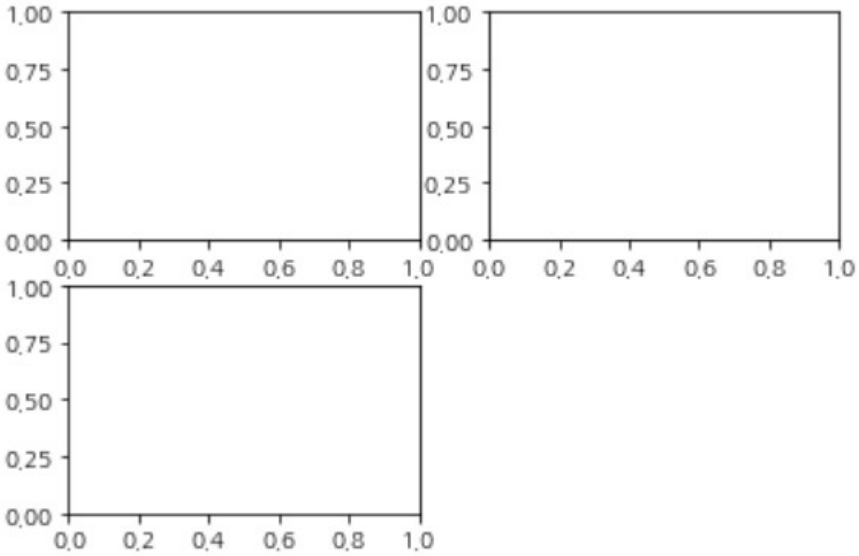
matplotlib + pandas

1. add_subplot() 메서드로 서브플롯 배치

figure() 함수로 Figure 클래스의 인스턴스를 생성한다. 그리고, add_subplot() 메서드를 서브플롯을 피켜 오브젝트의 지정된 위치에 배치한다.

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

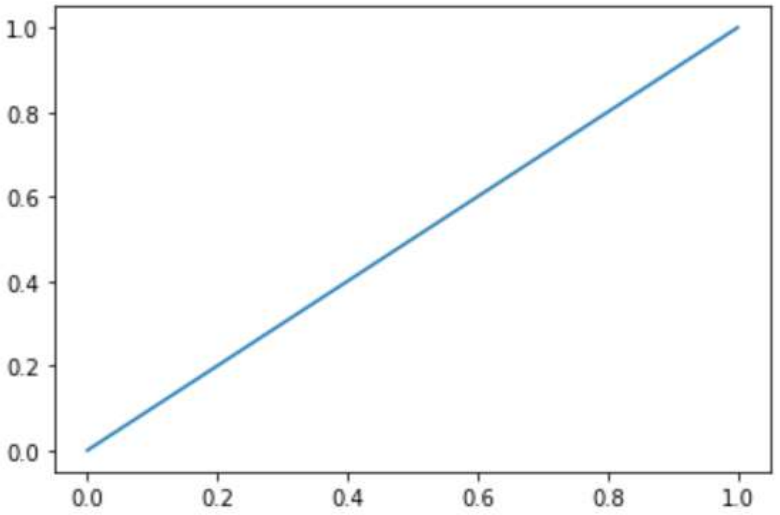
```
# 피겨 생성
fig = plt.figure()

# 피겨 안에 서브플롯을 3개 배치한다.
ax1 = fig.add_subplot(221) #2행2열 1번
ax2 = fig.add_subplot(222) #2행2열 2번
ax2 = fig.add_subplot(223) #2행2열 3번

# Jupyter Notebook에 그래프 출력
plt.show()
```



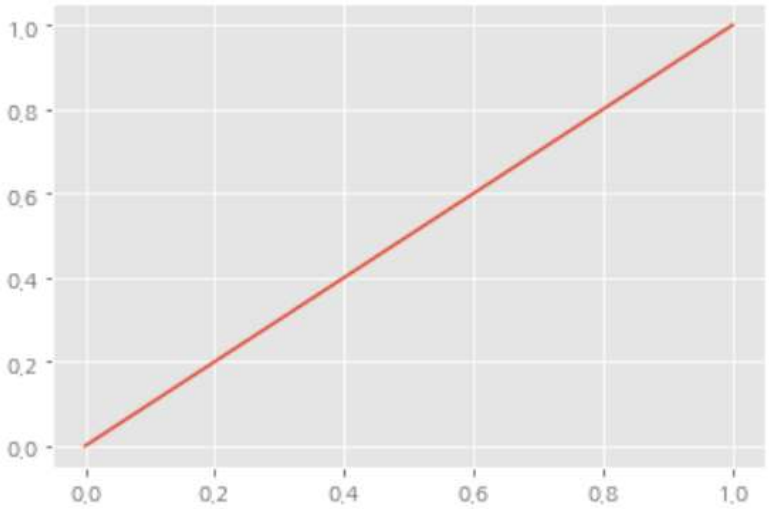
2. 스타일 적용하기

스타일이란 그래프의 선 굵기나 색 등 그래프의 "체재"에 관한 정보를 모아놓은 것이다. 스타일은 style.use() 함수를 적용할 수 있다.

```
#스타일 적용
plt.style.use('ggplot')

fig = plt.figure()
ax = fig.add_subplot(111)
dat = [0, 1]
ax.plot(dat)
plt.show()
```



적용전

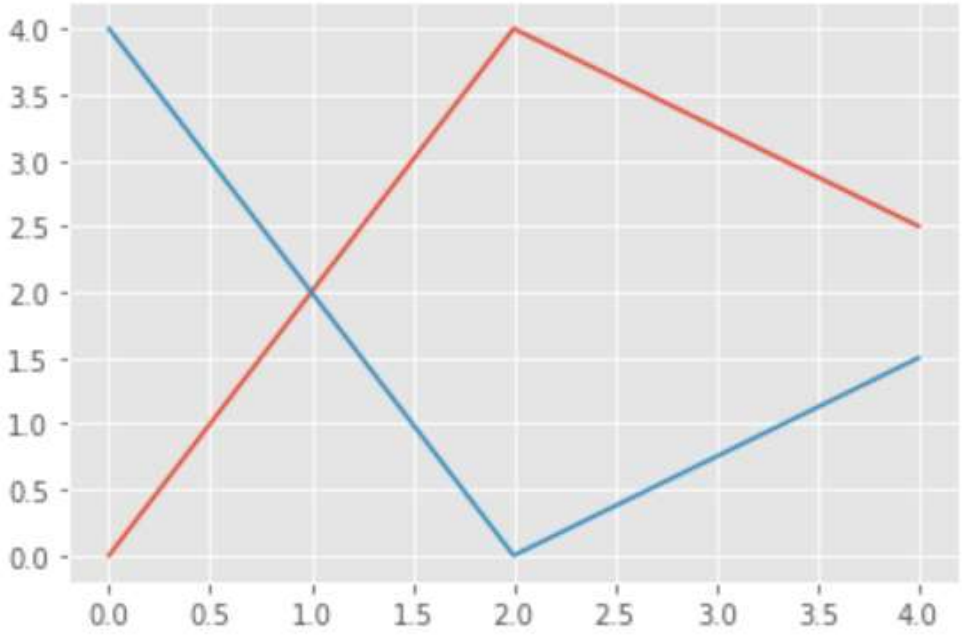


적용후

3. 여러개의 선을 그리기

여러 개의 선을 그리는 경우
plot() 메서드를 여러 번 실행하면 1개의 서브플롯에 여러 개의 그래프를 겹쳐서 그릴 수 있다.

```
import matplotlib.pyplot as plt
plt.style.use('ggplot')
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()
```



4. 꺾은선 그래프 활용하기

실제 데이터를 이용해서 그래프를 그린다. 데이터는 anime_stock_returns.csv 파일을 이용한다. anime_stock_returns.csv에는 TOEI ANIMATION 및 IG Port의 주가 등락률이 시계열(일단위)로 기록되어 있다.
데이터 불러오기에는 pandas를 이용한다.

```
import os
import pandas as pd
import matplotlib.pyplot as plt
csv = os.path.join('./sample/anime_stock_returns.csv')
df = pd.read_csv(csv, index_col=0, parse_dates=['Date']
df.head()
```

Date	TOEI ANIMATION	IG Port
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

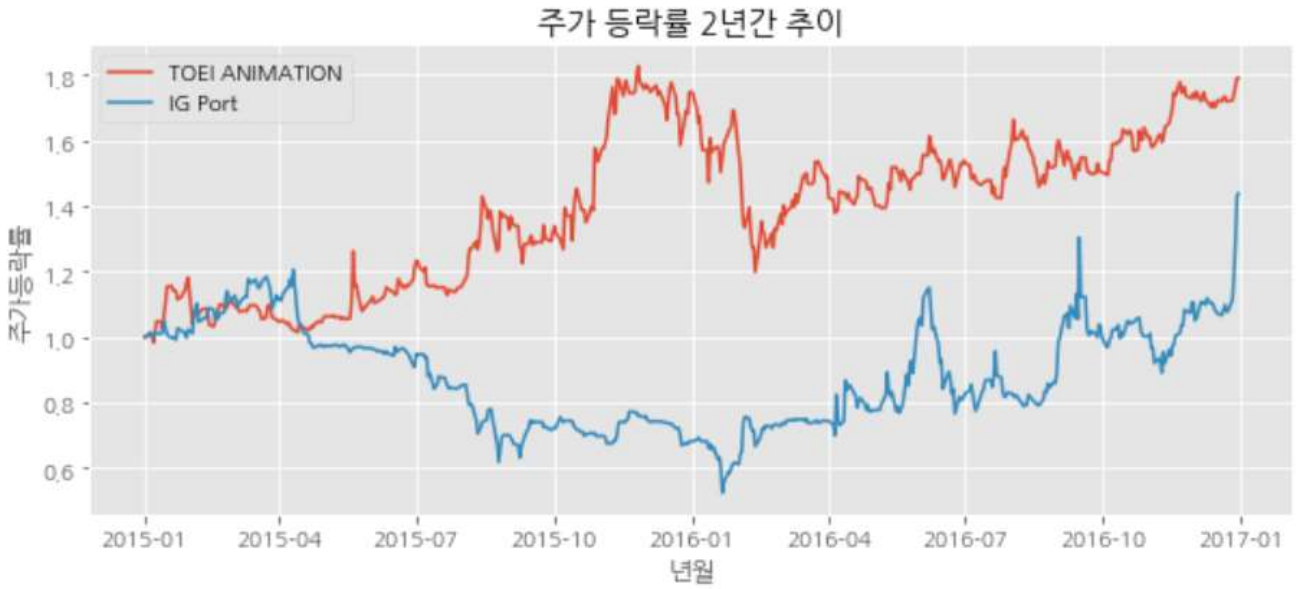
4. 꺾은선 그래프 활용하기

시계열 정보를 포함한 데이터를 표현하는 것은 꺾은선 그래프가 적당하다.
Matplotlib으로 꺾은선 그래프를 출력해 본다.

```
import os
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline

csv = os.path.join('./sample/anime_stock_returns.csv')
df = pd.read_csv(csv, index_col=0, parse_dates=['Date'])

plt.style.use('ggplot')
plt.rc('font', family='NanumGothic')
plt.rc('axes',unicode_minus=False)
```

4. 꺽은선 그래프 활용하기

```
fig = plt.figure(figsize=(10, 4))
ax = fig.add_subplot(111)
ax.plot(df.index, df['TOEI ANIMATION'], label='TOEI ANIMATION')
ax.plot(df.index, df['IG Port'], label='IG Port')
ax.set_title('주가 등락률 2년간 추이')
ax.set_ylabel('주가등락률')
ax.set_xlabel('년월')
ax.legend()
plt.show()
```



5. 두 개의 축을 가진 그래프 그리기

Matplotlib에서 X축을 공유해 2개의 Y축을 가진 그램을 작성하는 경우에는 Axes.twinx() 메서드를 사용한다. Y축을 공유해서 2개의 X축을 가진 그림을 그리는 경우에는 twiny() 메서드를 사용한다. twinx() 메서드를 사용해서 마감가(Close)와 거래량(Volume)을 하나의 그래프로 나타내고 있다. 마감가는 꺾은선 그래프로, 거래량은 막대 그래프로 나타내는 것이 일반적이다.

```
csv = os.path.join('./sample/4816.csv')
df = pd.read_csv(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)
```

5. 두 개의 축을 가진 그래프 그리기

```
# 축과 축레이블 설정
ax1.set_xlabel('년월')
ax1.set_yticks([i * 2000 for i in range(5)])
ax1.set_ylabel('주가')
ax2.set_yticks([i * 5000 for i in range(5)])
ax2.set_ylabel('거래총액')

# 그래프 타이틀 설정
ax1.set_title('주가와 거래총액')

# 범래설정
ax1.legend(loc=1)
ax2.legend(loc=2)

plt.show()
```

5. 두 개의 축을 가진 그래프 그리기

첫 번째 그래프에 왼쪽 축(Y축)과 X축을 그린 후, 두 번째 그래프 오른쪽 축(Y축)을 그리고 있다. twinx() 메서드로 ax1과 ax2의 X축을 공유하는 설정이 되어 있기 때문에 두 번째 그래프는 첫 번째의 서브플롯에 겹쳐져 그려진다.
그래프에서 알기 쉽게 두 축의 눈금을 설정해서 축이름과 범례를 표시하고 있다.

matplotlib - 산포도그래프

1. 산포도 그래프

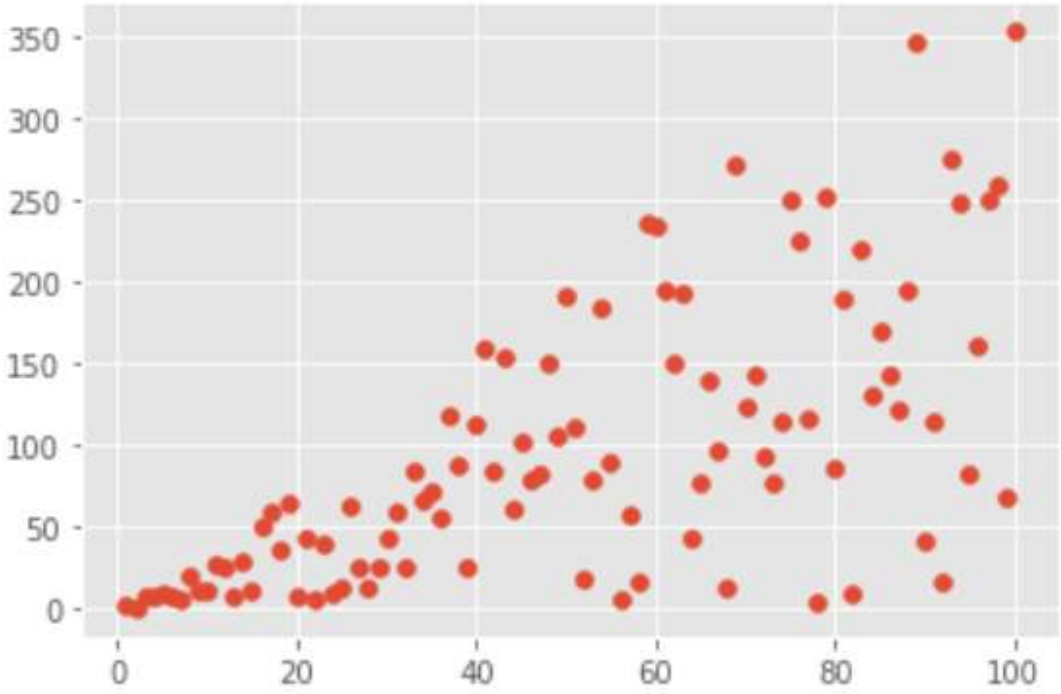
산포도 그래프는 X축과 Y축에 수량이나 크기 등을 대응시켜서 적합한 점에 데이터를 플롯한 그래프이다. 산포도 그래프는 X축과 Y축에 취한 두 개의 값(Z축이 있는 경우에는 3개의 값)에 함수가 있는지 없는지 보는 것에 유용하다. 또한, 데이터의 분포 상황을 확인할 때에도 활용할 수 있다.

산포도 그래프는 Axes.scatter() 메서드를 사용해서 그린다. scatter() 메서드의 제1, 제2인수에 각각 x값을 부여한다. x에 1부터 100까지의 정수 값을 100개, y에 0부터 1의 값을 취한 난수에 x와 4를 공합 것을 부여해서 각각 X값, Y값으로 하고 있다. 데이터로서 리스트형/오브젝트를 이용할 수 있다.

note: numpy.random.seed() 메서드는 난수의 seed를 고정하는 것에 따라, 항상 같은 난수를 발생시키는 것이 가능하다. 여기에서는 독자가 같은 그림을 재현할 수 있도록 해당 메서드를 실행하고 있다.

1. 산포도 그래프

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

plt.style.use('ggplot')

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

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



2. 산포도 그래프 활용하기

실제 데이터를 이용해서 그래프를 작성한다. 데이터는 anime_master.csv 파일을 이용한다. 데이터를 불러오면 애니메이션의 제목이나 장르, 에피소드 수나 평점 데이터가 포함되어 있는 것을 알 수 있다.

```
import os
import pandas as pd
csv = os.path.join('./sample/anime_master.csv')
df = pd.read_csv(csv)
df.head()
```

	anime_id	name	genre	type	episodes	rating	members
0	32281	Kimi no Na wa.	Drama, Romance, School, Supernatural	Movie	1	9.37	200630
1	5114	Fullmetal Alchemist: Brotherhood	Action, Adventure, Drama, Fantasy, Magic, Mili...	TV	64	9.26	793665
2	28977	Gintama°	Action, Comedy, Historical, Parody, Samurai, S...	TV	51	9.25	114262
3	9253	Steins;Gate	Sci-Fi, Thriller	TV	24	9.17	673572
4	9969	Gintama'	Action, Comedy, Historical, Parody, Samurai, S...	TV	51	9.16	151266

2. 산포도 그래프 활용하기

자동적으로 0부터 시작하는 정수값이 인덱스로서 부여되어 있지만, anime_id를 인수 index_col로 지정해서 anime_idf를 인덱스에 설정할 수 있다.

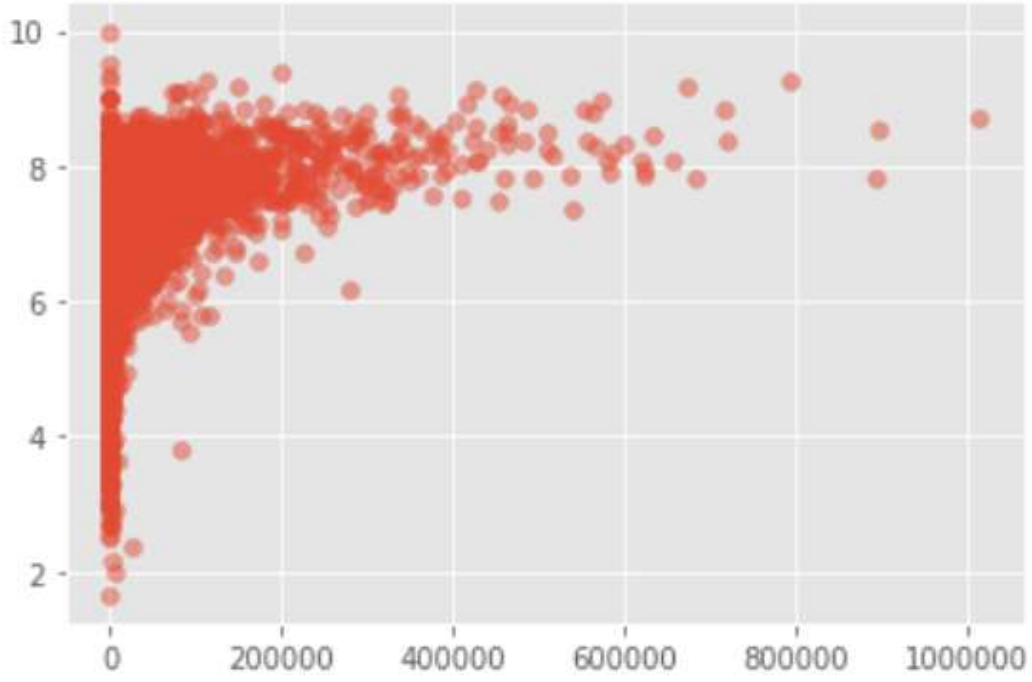
```
df = pd.read_csv(csv, index_col = 'anime_id')
df.head()
```

anime_id	name	genre	type	episodes	rating	members
32281	Kimi no Na wa.	Drama, Romance, School, Supernatural	Movie	1	9.37	200630
5114	Fullmetal Alchemist: Brotherhood	Action, Adventure, Drama, Fantasy, Magic, Mili...	TV	64	9.26	793665
28977	Gintama°	Action, Comedy, Historical, Parody, Samurai, S...	TV	51	9.25	114262
9253	Steins;Gate	Sci-Fi, Thriller	TV	24	9.17	673572
9969	Gintama'	Action, Comedy, Historical, Parody, Samurai, S...	TV	51	9.16	151266

2. 산포도 그래프 활용하기

X값으로 "member"를, Y값으로 "rating"을 지정하는 것에 따라 산포도 그래프를 작성할 수 있다. 그려진 기호를 반투명으로 하는 값(alpha=0.5)도 설정되어 있다.

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



멤버 수(members)와 평점(rating) 사이에 명확한 상관관계는 없지만 수의 증가에 따른 평점이 8 부근에 위치되는 경향을 파악할 수 있다.

2. 산포도 그래프 활용하기

멤버 수 80만 이상 작품
그림에는 멤버 수 80만 이상의 위치에 명 개의 애니메이션이 플롯되어 있다.
데이터를 추출한 결과, 멤버 수가 많은 "Death Note", "Shingeki no Kyojin", "Sword Art Online" 세 작품인 것을 확인할 수 있다.

```
# members 값을 데이터를 추출한다.
df.loc[df['members'] >= 800000, ['name', 'members']]
```

anime_id	name	members
1535	Death Note	1013917
16498	Shingeki no Kyojin	896229
11757	Sword Art Online	893100

2. 산포도 그래프 활용하기

멤버 수 60만 이상이며 평점이 8.5이상의 작품
다음으로, 멤버 수의 범위를 넓히는 동시에 복수의 조건으로 애니메이션을 검색한다. 데이터를 추출한 결과, "Fullmetal Alchemist: Brotherhood", "Steins; Gate", "Code Geass: Hangyaku no Lelouch"등을 확인할 수 있다.

```
# members와 rating 값으로 데이터를 추출한다.
df.loc[(df['members'] >= 600000) & (df['rating'] >= 8.5), ['name', 'rating']]
```

anime_id	name	rating
5114	Fullmetal Alchemist: Brotherhood	9.26
9253	Steins;Gate	9.17
1575	Code Geass: Hangyaku no Lelouch	8.83
1535	Death Note	8.71
16498	Shingeki no Kyojin	8.54

3. 그룹화된 산포도 그래프 작성하기

산포도 그래프 그리기에 이용된 DataFrame을 계속해서 이용한다. 이 데이터는 type이라는 열을 가지고 있다. type은 애니메이션 작품의 배급 종별을 의미한다. 먼저, type 중복 없는 리스트를 작성한다. 실행하면 리스트 데이터를 얻을 수 있다.

```
types = df['type'].unique()
types
```

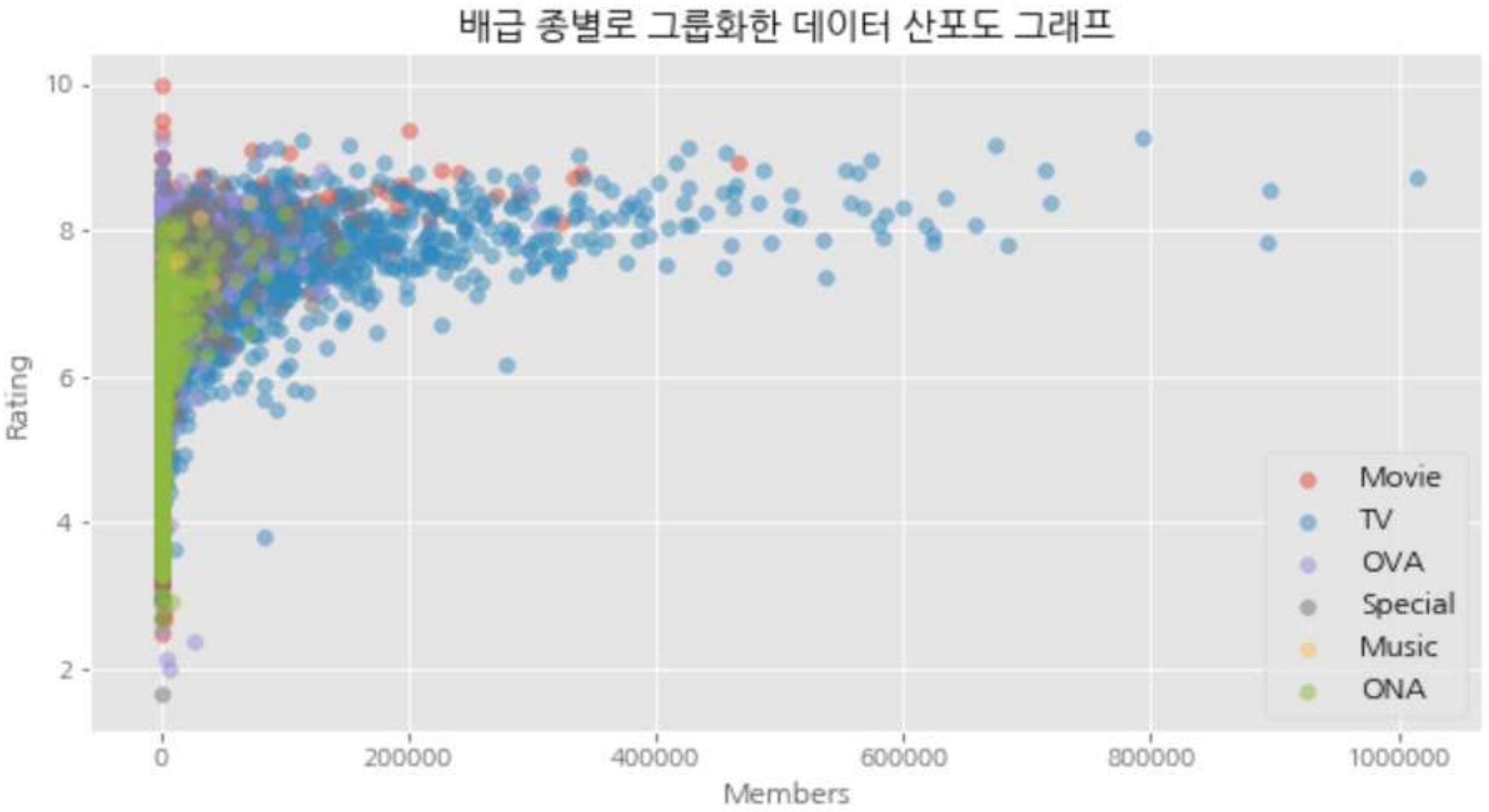
```
array(['Movie', 'TV', 'OVA', 'Special', 'Music', 'ONA'], dtype=object)
```

3. 그룹화된 산포도 그래프 작성하기

꺽은선 그래프와 마찬가지로, 하나의 서브플롯에 겹쳐서 산포도 그래프를 그린다.

```
plt.rc('font', family='NanumGothic')
plt.rc('axes',unicode_minus=False)

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()
```

3. 그룹화된 산포도 그래프 작성하기



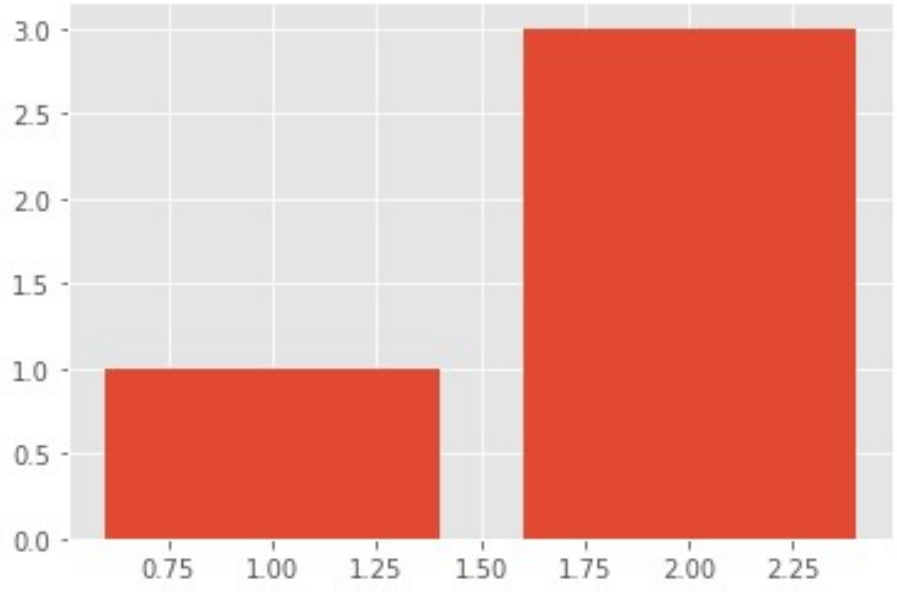
배급 종별마다 일치하는 데이터를 추출해서 그리고 있다.
배급 종별은 6종류가 있기 때문에 6개의 데이터 세트가 플롯되어 있다.

matplotlib - 막대그래프

1. 막대 그래프

- 막대 그래프는 수량을 막대의 길이로 나타낸 그래프이다.
- Axes.bar() 메서드를 사용해서 그린다. bar(인수1, 인수2) 인수1에 X값과 인수2에 Y값을 부여한다.
- 데이터로서 리스트형, 오브젝트를 이용할 수 있다.

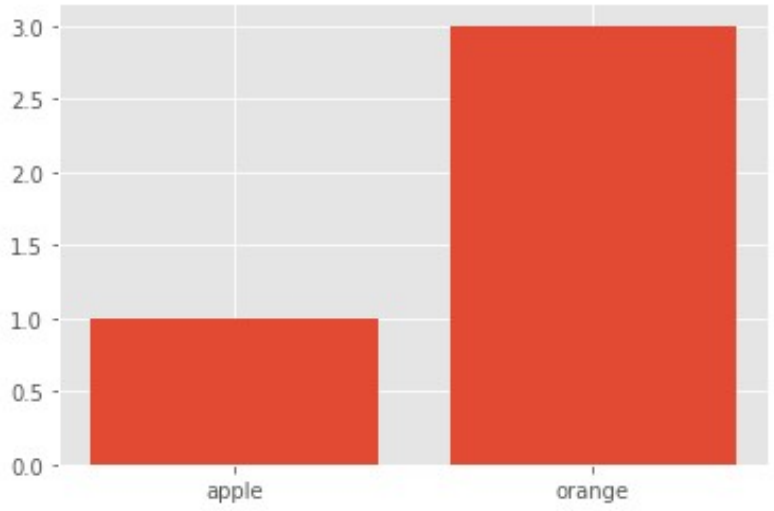
```
import matplotlib.pyplot as plt
%matplotlib inline

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

x = [1, 2]
y = [1, 3]

ax.bar(x, y)

plt.show()
```



1. 막대 그래프

눈금 레이블
X값을 부여해서 인수 tick_label에 눈금을 설정해서 작성한다.

```
# 횡축의 눈금에 레이블 설정
fig = plt.figure()
ax = fig.add_subplot(111)

labels = ['apple', 'orange']
ax.bar(x, y, tick_label=labels)

plt.show()
```



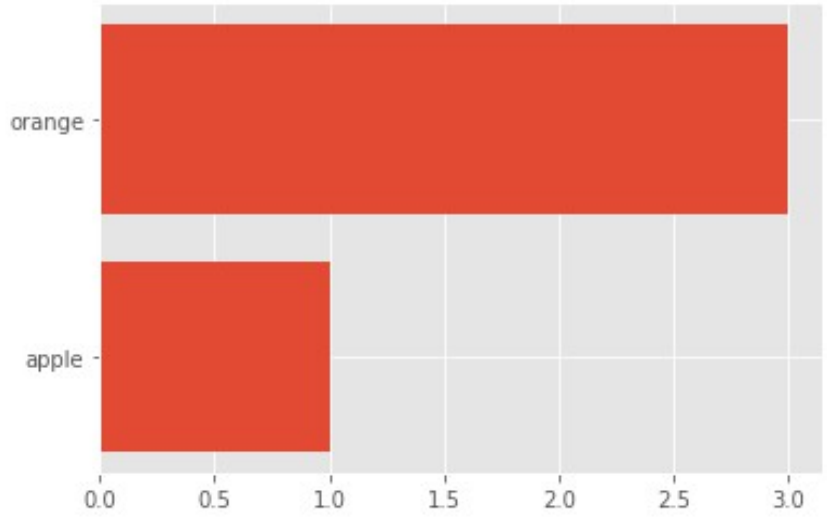
레이블은 리스트나 튜플로 부여한다.
레이블을 부여한 후에 눈금을 설정하는 방법도 있다.

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

ax.bar(x, y)

# X축의 축눈금과 축눈금 레이블
labels = ['apple', 'orange']
ax.set_xticks(x)
ax.set_xticklabels(labels)

plt.show()
```

1. 막대 그래프

수평 막대 그래프 작성
수평 막대 그래프는 Axes.barh() 메서드를 이용해서 그린다. 메서드의 인수는 기본적으로 bar()와 같다.

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

ax.barh(x, y, tick_label=labels)

plt.show()
```



2. 막대 그래프 활용

실제 데이터를 이용해서 그래프를 작성한다. 데이터는 anime_master.csv 파일을 이용한다.

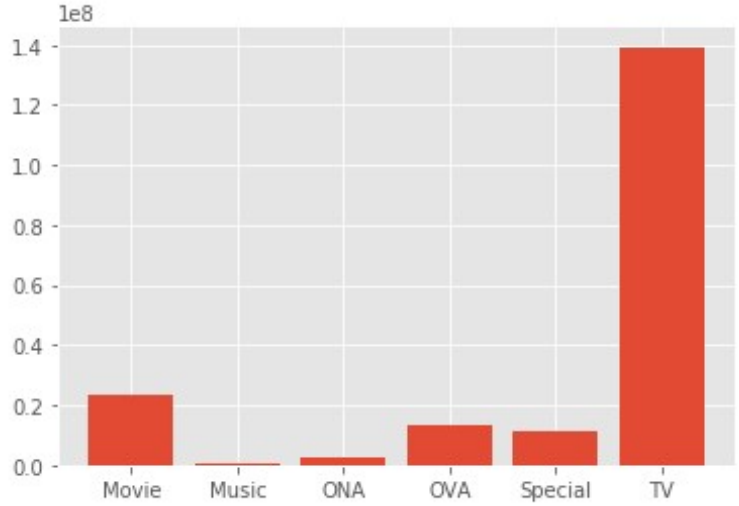
```
import os
import pandas as pd

csv = os.path.join('./sample/anime_master.csv')
dfac = pd.read_csv(csv)

dfac.head()
```

	anime_id	name	genre	type	episodes	rating	members
0	32281	Kimi no Na wa.	Drama, Romance, School, Supernatural	Movie	1	9.37	200630
1	5114	Fullmetal Alchemist: Brotherhood	Action, Adventure, Drama, Fantasy, Magic, Mili...	TV	64	9.26	793665
2	28977	Gintama°	Action, Comedy, Historical, Parody, Samurai, S...	TV	51	9.25	114262
3	9253	Steins;Gate	Sci-Fi, Thriller	TV	24	9.17	673572
4	9969	Gintama'	Action, Comedy, Historical, Parody, Samurai, S...	TV	51	9.16	151266

2. 막대 그래프 활용

막대그래프는 수량의 대소를 시각화할 때 적당하다. 여기에서는 작품의 배급 종별마다 멤버수의 합계 값을 추출해서 막대 그래프를 그린다.
Matplotlib은 pandas의 DataFrame이나 Series를 이용해서 그래프를 그릴 수 있다.
X값에 배급 종별, Y값에 배급 종별마다의 합계 멤버수를 부여해서 그려보자.

```
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)

plt.show()
```

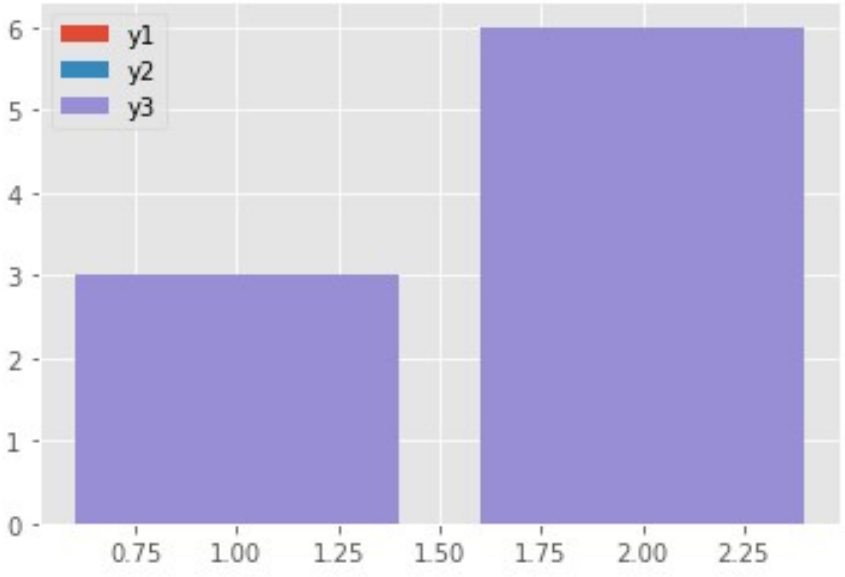


x에 대해서는 len() 함수로 DataFrame의 데이터 수를 계수하고 range() 함수로 그 범위의 정수 열을 생성하고 있다. 또한, X축 레이블에는 Series의 인덱스를 부여한다.
이처럼, 데이터를 시각화하여 배급 종별에서는 텔레비전 작품의 수가 돌출 되는 것을 확인할 수 있다.

3. 그룹 막대그래프

여러 그룹의 막대그래프를 그릴 때는 bar()를 여러 번 실행 했을 때 최초에 그려진 오브젝트가 뒤에 그려진 오브젝트에 의해 덮어 씌어 지는 원리를 활용한다.

```
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()
```

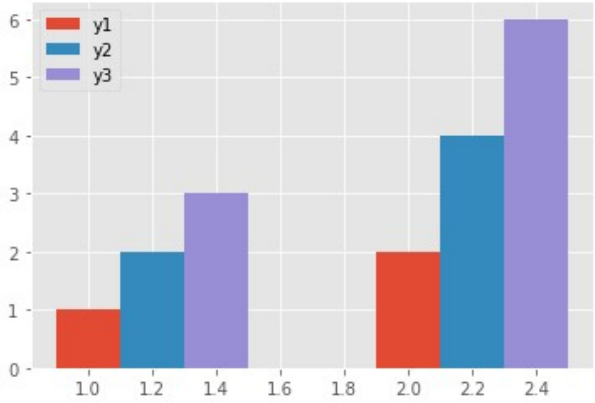


이와같이 x값을 가진 오브젝트가 겹쳐진다.

3. 그룹 막대그래프

겹쳐지는 것을 피하기 위해서는 x값을 막대의 가로 폭만큼 비켜서 그릴 필요가 있다. 막대그래프의 가로 폭 w를 0.2로 설정하고 x값을 0.2씩 비켜서 그린다.

```
#복수 그룹의 막대 그래프
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()
```



4. 그룹 막대그래프 활용

실제의 데이터를 이용해서 그룹 막대 그래프를 그려보자. 데이터는 anime_genre_top10_pivoted.csv 파일을 이용한다.

```
csv = os.path.join('./sample/anime_genre_top10_pivoted.csv')
dfag = pd.read_csv(csv, index_col='genre')
dfag
```

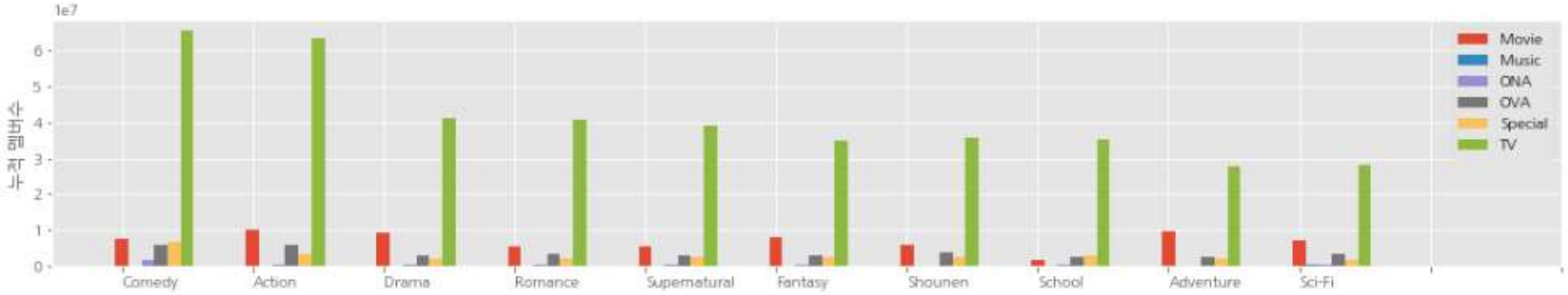
genre	Movie	Music	ONA	OVA	Special	TV
Comedy	7293127.0	20860.0	1477266.0	5614758.0	6659293.0	65420862.0
Action	10224960.0	77054.0	524907.0	5793680.0	3412689.0	63364032.0
Drama	9034099.0	100734.0	188427.0	3043374.0	1915578.0	41011557.0
Romance	5245386.0	42811.0	411331.0	3143167.0	2015820.0	40703388.0
Supernatural	5452779.0	9189.0	192989.0	2696715.0	2336723.0	38956520.0
Fantasy	8019406.0	43962.0	188937.0	2754224.0	2504131.0	34932563.0
Shounen	5698808.0	NaN	97833.0	3861296.0	2591988.0	35532847.0
School	1512533.0	5496.0	523223.0	2417660.0	2661425.0	35489099.0
Adventure	9485223.0	42829.0	70431.0	2373765.0	2052024.0	27529975.0
Sci-Fi	6967146.0	154801.0	415311.0	3358525.0	1616450.0	28072322.0

4. 그룹 막대그래프 활용

불러온 데이터(dfag)를 시각화 그룹 막대그래프로 시각화 한다. x값을 0.1씩 증가 시키면서 열별로 그려보자.

```
plt.rc('font', family='NanumGothic')
plt.rc('axes',unicode_minus=False)
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()
```

4. 그룹 막대그래프 활용

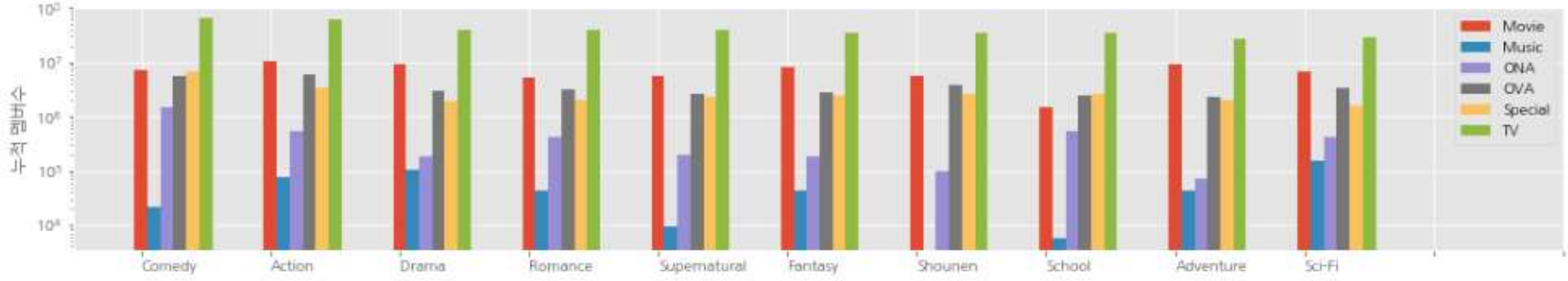


이 결과에서도 TV 합계 멤버 수가 돌출되어 있고 다음에 Movie 멤버 수가 많은 것을 확인할 수 있다.

4. 그룹 막대그래프 활용

Music이나 ONA 값이 상대적으로 작기 때문에 눈으로 확인하는 것이 어렵다.
이 같은 경우에 로그 축을 이용하면 가독성이 좋아진다. Y축을 로그축에 설정하는 경우에는 set_yscale() 메서드에 log를 지정한다. 작은 값의 그룹도 눈으로 확인할 수 있게 된다.

```
... 생략 ...
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()
```

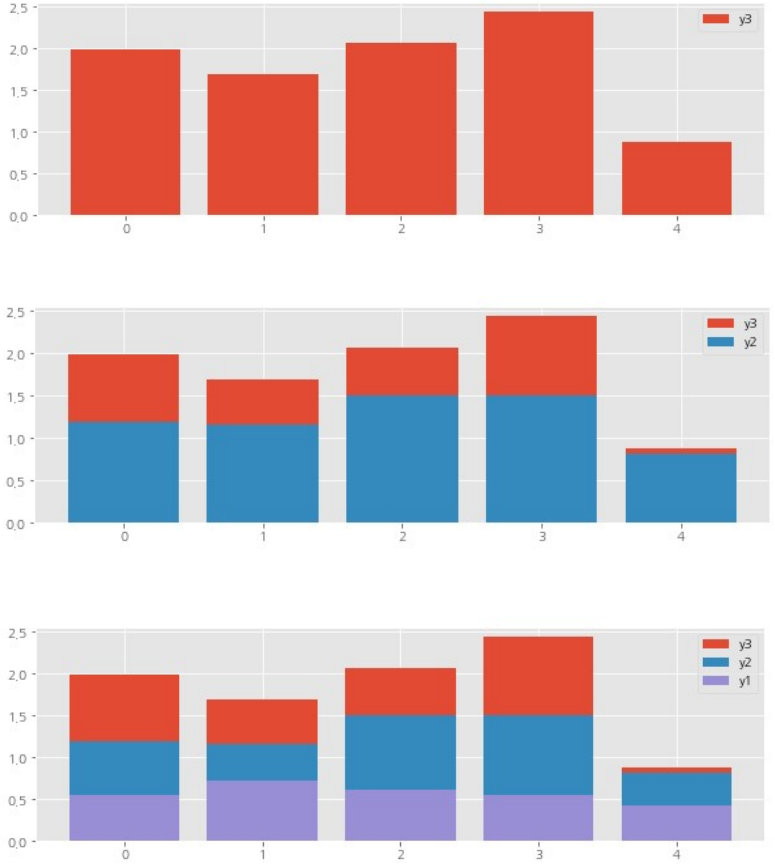


5. 누적 막대그래프

누적 막대그래프를 그릴 때에도 여러 그룹의 막대그래프와 같이 작성시 요령이 필요하다. y1, y2, y3이라는 세 개의 값을 누적한 경우의 그래프를 그려보자.
실제 그리기 순서는 다음과 같다.

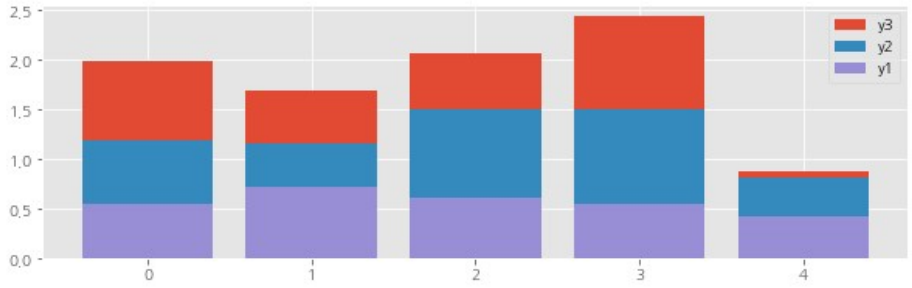
① y1과 y2와 y3의 합을 그린다.
② y2와 y3의 합을 겹쳐서 그린다.
③ y1을 겹쳐서 그린다.

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



5. 누적 막대그래프

```
# 데이터 세트 작성
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()
```

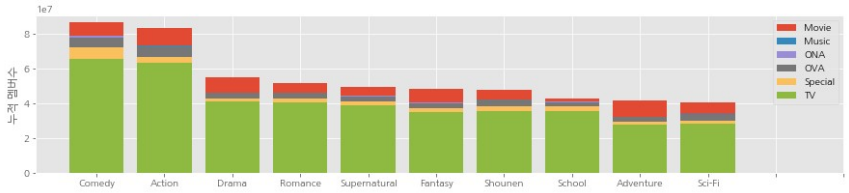


Note:
배열의 가산을 실행할 때,
numpy.ndarray를 사용하는 이유는
데이터의 요소끼리 가산하기 위해서이다.
Python 리스트끼리 가산하는 것과
동작이 다른 것에 주의한다.

6. 누적 막대그래프 활용

실제 데이터를 이용해 누적 막대그래프를 그려보자. 데이터는 앞에서 이용한 anime_genre_top10_pivoted.csv 파일을 이용한다. 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 마지막까지의 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()
```



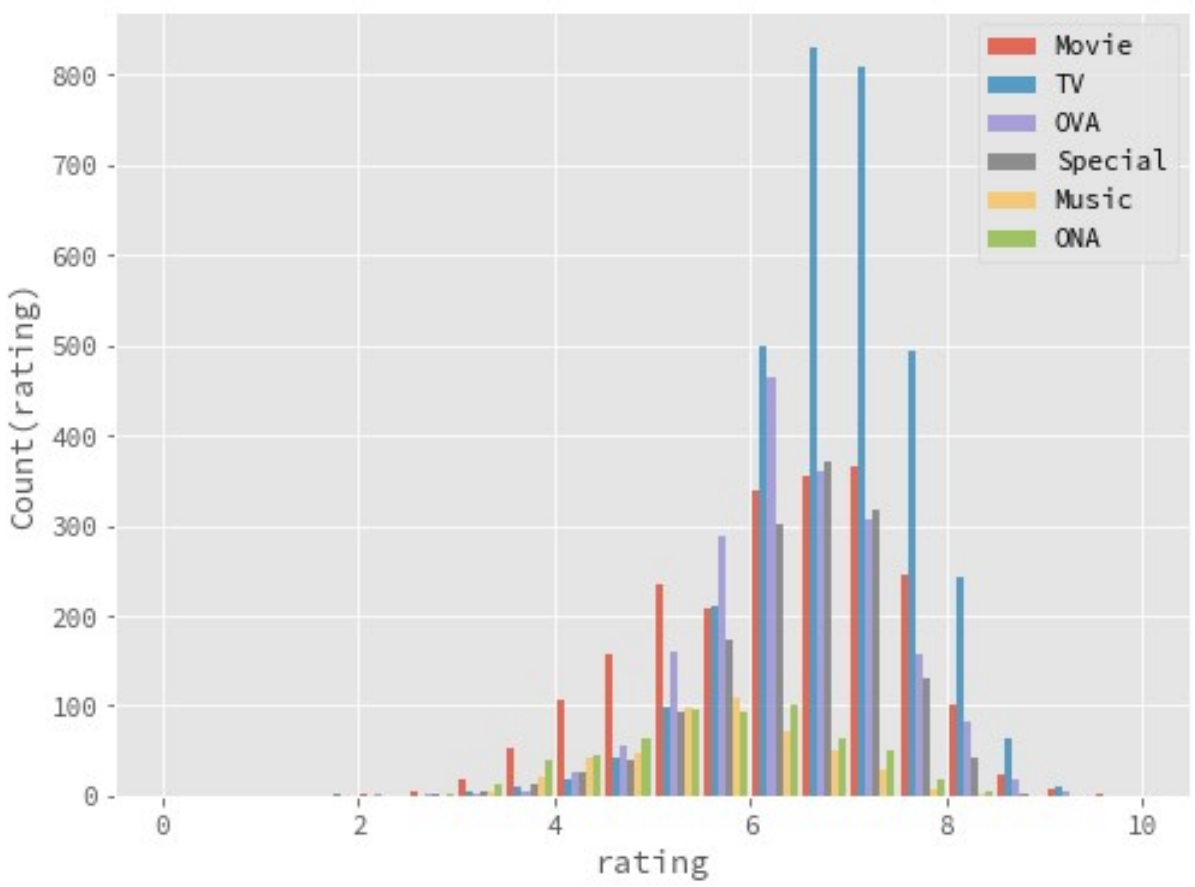
각 열에서 마지막 열까지의 합계 값을 y에 저장하고 Y값으로 그리고 있다.

배급 타입별에서는 어느 장르도 TV의 비율이 많은 것과 더불어 장르별로는 Comedy와 Action의 총멤버 수가 특히 많은 것을 확인할 수 있다.

matplotlib - 히스토그램

1. 히스토그램

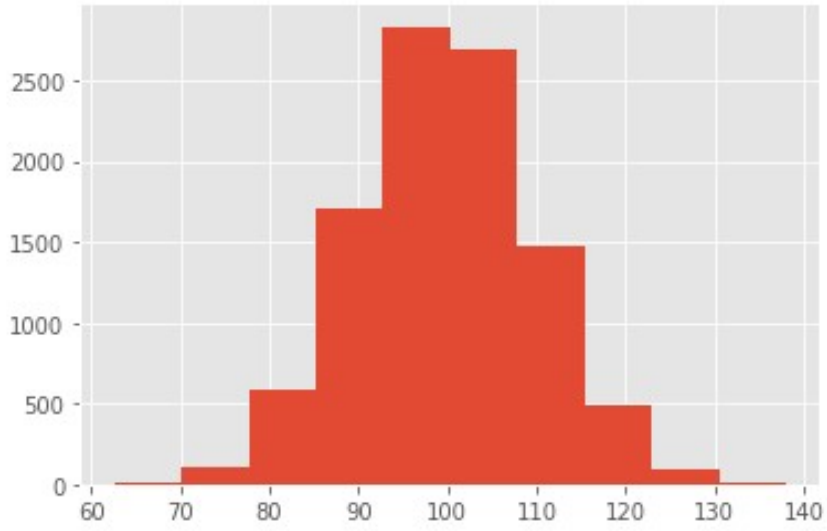
히스토그램은 세로축에 횟수(값의 출현빈도), 가로축에 계급(값의 상한값), 가로축에 계급(값의 출현빈도), 가로축에 계급(값의 상한 값 ~ 하한 값의 폭)을 취급하는 그래프로, 데이터의 분포 형상을 시각적으로 인식하기 위해 이용된다. 데이터의 분포 형상(분포형)은 통계학적으로 중요한 의미를 가지고 있다.



1. 히스토그램

히스토그램은 Axes.hist() 메서드를 사용해서 작성한다. 메서드에 넘기는 데이터에는 리스트형, 오브젝트를 이용할 수 있다.
리스트 평균값 100, 표준편차 10의 정규분포에 따라 만 개의 데이터 히스토그램을 그려보자.

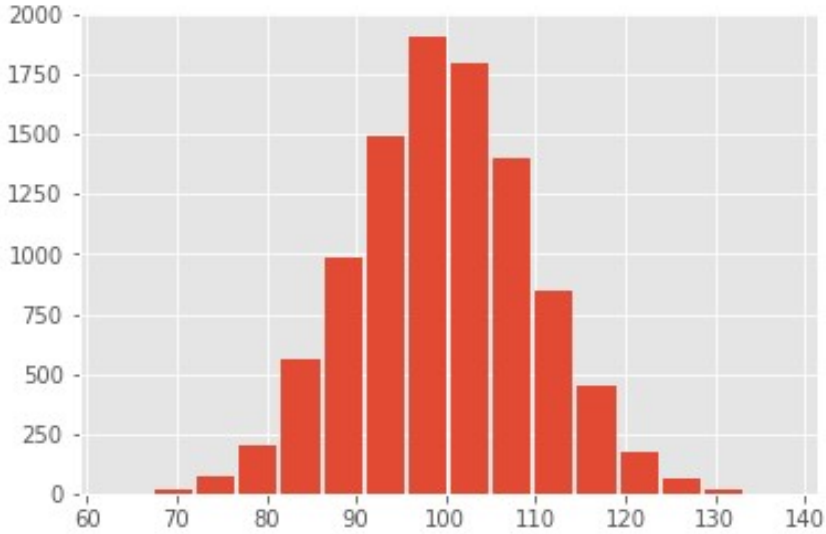
```
import numpy as np
import matplotlib.pyplot as plt

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()
```



1. 히스토그램

막대의 폭과 수를 변경하는 경우
hist() 메서드에는 데이터 외에 히스토그램 그림에 관한 인수를 부여할 수 있다. rwidth로 막대의 폭을, bins로 막대의 개수를 지정하고 있다. 이와 같이 인수를 지정하면 그림처럼 막대의 폭이나 개수가 변경된 그래프가 출력된다.

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



2. 히스토그램 활용

실제 데이터를 이용해서 그래프를 그려보자. 데이터는 anime_master.csv 파일을 이용한다.

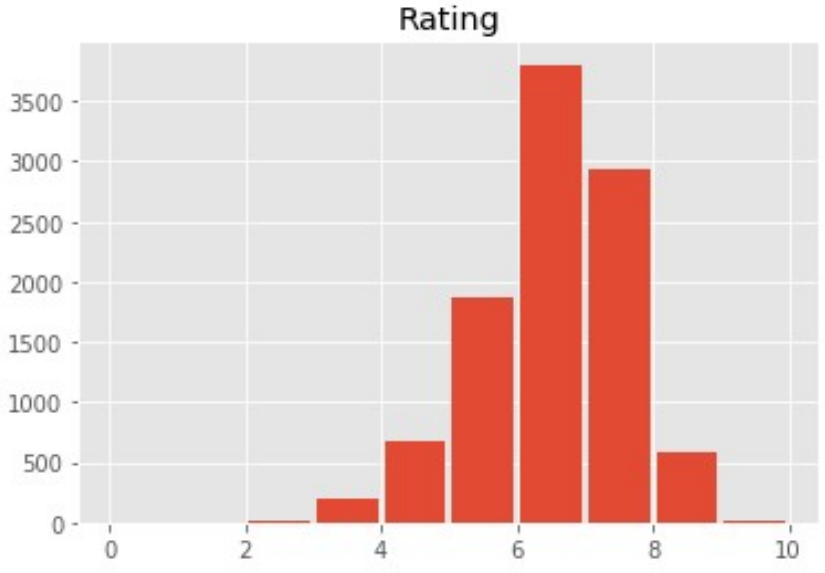
```
import os
import pandas as pd

csv = os.path.join("./sample/anime_master.csv")
df = pd.read_csv(csv, index_col="anime_id")
df.head()
```

anime_id	name	genre	type	episodes	rating	members
32281	Kimi no Na wa.	Drama, Romance, School, Supernatural	Movie	1	9.37	200630
5114	Fullmetal Alchemist: Brotherhood	Action, Adventure, Drama, Fantasy, Magic, Mili...	TV	64	9.26	793665
28977	Gintama°	Action, Comedy, Historical, Parody, Samurai, S...	TV	51	9.25	114262
9253	Steins;Gate	Sci-Fi, Thriller	TV	24	9.17	673572
9969	Gintama'	Action, Comedy, Historical, Parody, Samurai, S...	TV	51	9.16	151266

2. 히스토그램 활용

평점 분포에 대해 Matplotlib으로 시각화 실행한다. pandas의 Series을 hist() 메서드의 인수에 넘겨서 출력 한다. 평점이 0 ~ 10의 범위에서 실행되고 있기 때문에 값의 범위를 range에서 0 ~ 10으로 지정한다. 6 ~ 7 사이의 데이터가 가장 많고 0 ~ 3이나 9 ~ 10의 데이터는 극단적으로 적은 것을 확인할 수 있다.

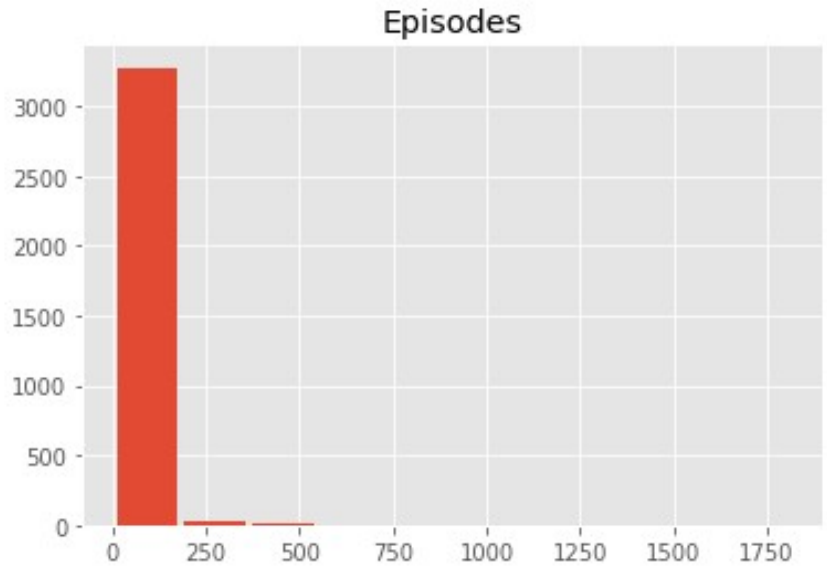
```
fig = plt.figure()
ax = fig.add_subplot(111)
ax.hist(df["rating"], range=(0, 10), rwidth=0.9)
ax.set_title("Rating")
plt.show()
```



2. 히스토그램 활용

에피소드 수도 히스토그램을 이용해서 시각화한다. 에피소드 수는 텔레비전 애니메이션의 ㅇ ㅇ화라는 의미로, 영화의 경우에는 기본적으로 갑이 1이 된다. 여기에서는 에피소드 수의 불균형이 크다고 예상되는 TV에 한정해서 시각화한다. 실행했더니 왼쪽으로 크게 치우쳐진 히스토그램이 되었다.

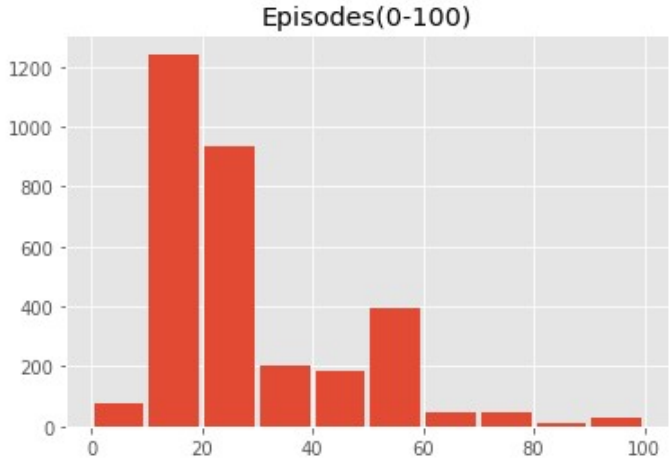
```
# 에피소드 수의 시각화
fig = plt.figure()
ax = fig.add_subplot(111)
df_tv = df[df["type"] == "TV"]
ax.hist(df_tv["episodes"], rwidth=0.9)
ax.set_title("Episodes")
plt.show()
```



2. 히스토그램 활용

대부분의 데이터가 처음 계급으로 뭉쳐 있다. 극히 일부 극단적으로 이야기 수가 많은 작품이 있기 때문에 이 같은 히스토그램이 되었다고 본다. 그 같은 경우, 히스토그램 범위 지정이 유효화다. 다음과 같이 실행 했더니 데이터 상황이 잘 보이게 되었다.

```
# 히스토그램 범위 지정
fig = plt.figure()
ax = fig.add_subplot(111)
# range의 값을 0 , 100) 으로 지정한다
ax.hist(df_tv["episodes"], rwidth=0.9, range=(0, 100))
ax.set_title("Episodes(0-100)")
plt.show()
```



TV 애니메이션은 텔리비전 방영 구조상, 1작품 분량(10 ~ 13화 정도) 또는 2작품 분량(23 ~ 25화 정도)의 작품이 많다. 그것을 뒷받침하는 분포를 확인할 수 있다.

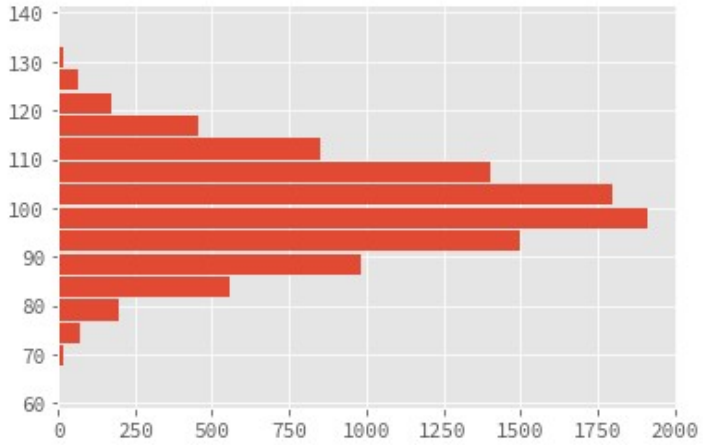
3. 다양한 히스토그램

여기부터는 다양한 패턴의 히스토그램을 작성하는 방법을 설명한다.

수평 히스토그램

인수 orientatio에 "horizontal" (초기설정은 "vertical")을 지정하면 수평 히스토그램을 그릴 수 있다.

```
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()
```

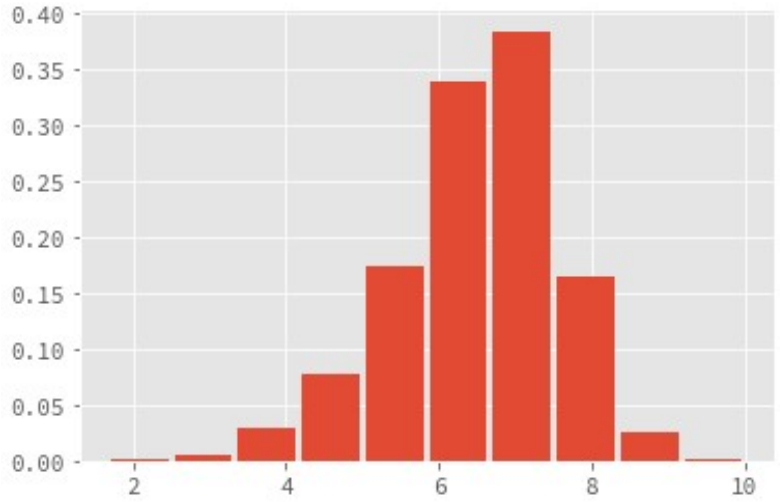


3. 다양한 히스토그램

상대도수 히스토그램

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

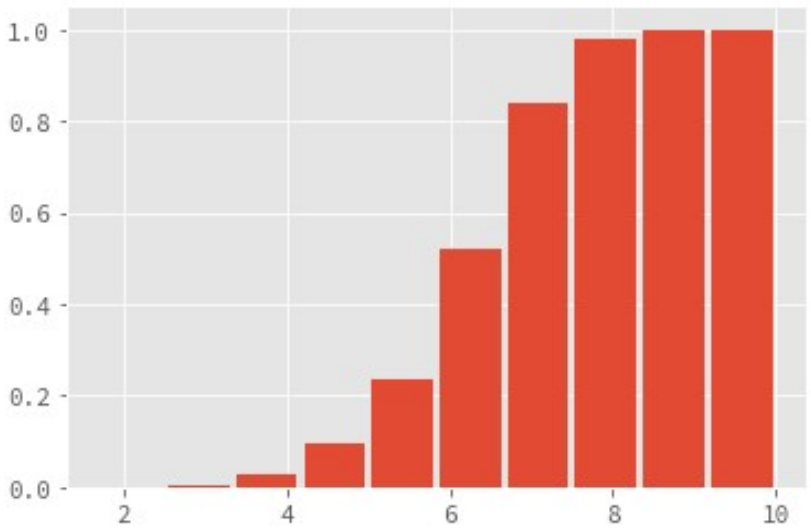
```
fig = plt.figure()
ax = fig.add_subplot(111)
# normed을 True로 지정
ax.hist(df["rating"], normed=True, rwidth=0.9)
plt.show()
```



3. 다양한 히스토그램

누적 히스토그램(누적도수 그림)
누적도수를 확인하는 경우에는 누적 히스토그램을 이용한다. 누적 히스토그램을 그리는 경우 cumulative에 True를 지정한다. 데이터를 사용해서 상대도수의 누적 히스토그램을 그리고 있다. 상대도수를 누적하면 합계가 1이 되는 것을 확인할 수 있다.

```
fig = plt.figure()
ax = fig.add_subplot(111)
# cumulative을 True로 지정
ax.hist(df["rating"], normed=True, cumulative=True, rwidth=0.9)
plt.show()
```

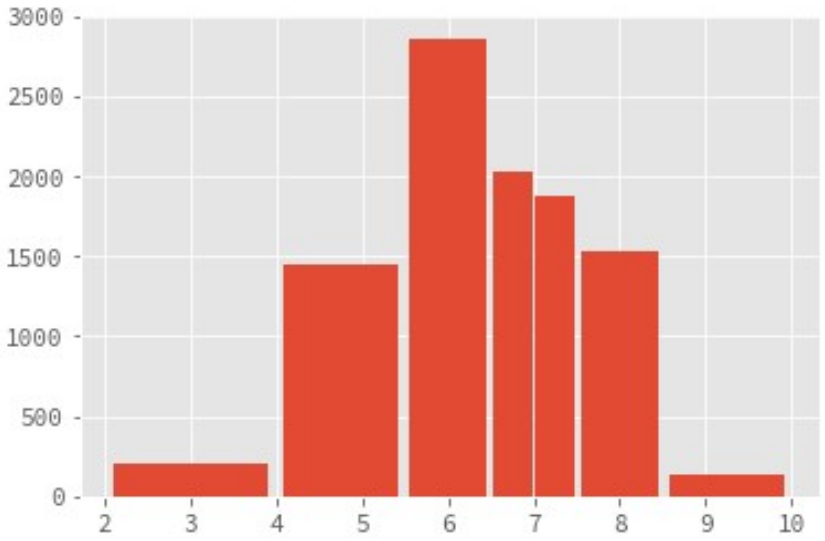


3. 다양한 히스토그램

계급 폭 지정

bins 옵션에 리스트형 수열을 부여하는 것에 따라 계급 폭을 지정할 수 있다. 계급 같은 간격이 아니어도 상관없다. "2", "1.5", "1", "0.5", "0.5", "1", "1.5"로 계급 폭을 변화시키고 있다.

```
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()
```

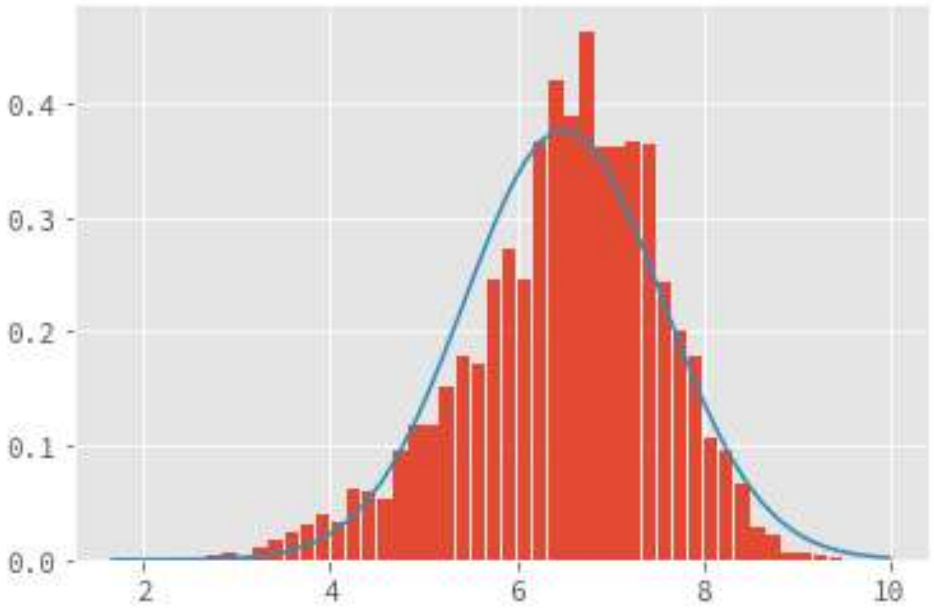


3. 다양한 히스토그램

근사 곡선 추가
근사 곡선은 히스토그램을 그린 후 에 꺾은선 그래프로 그린다. 여기에서도 정규분포를 이용해서 근사한다.

```
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 확률밀도 함수를 사용해 Y값 생성
y = 1 / (sigma * np.sqrt(2 * np.pi)) * np.exp(-(x - mu) ** 2 / (2 * sigma ** 2))
ax.plot(x, y) # 근사곡선그리기
plt.show()
```

3. 다양한 히스토그램



근사 곡선은 다음 단계로 그린다.

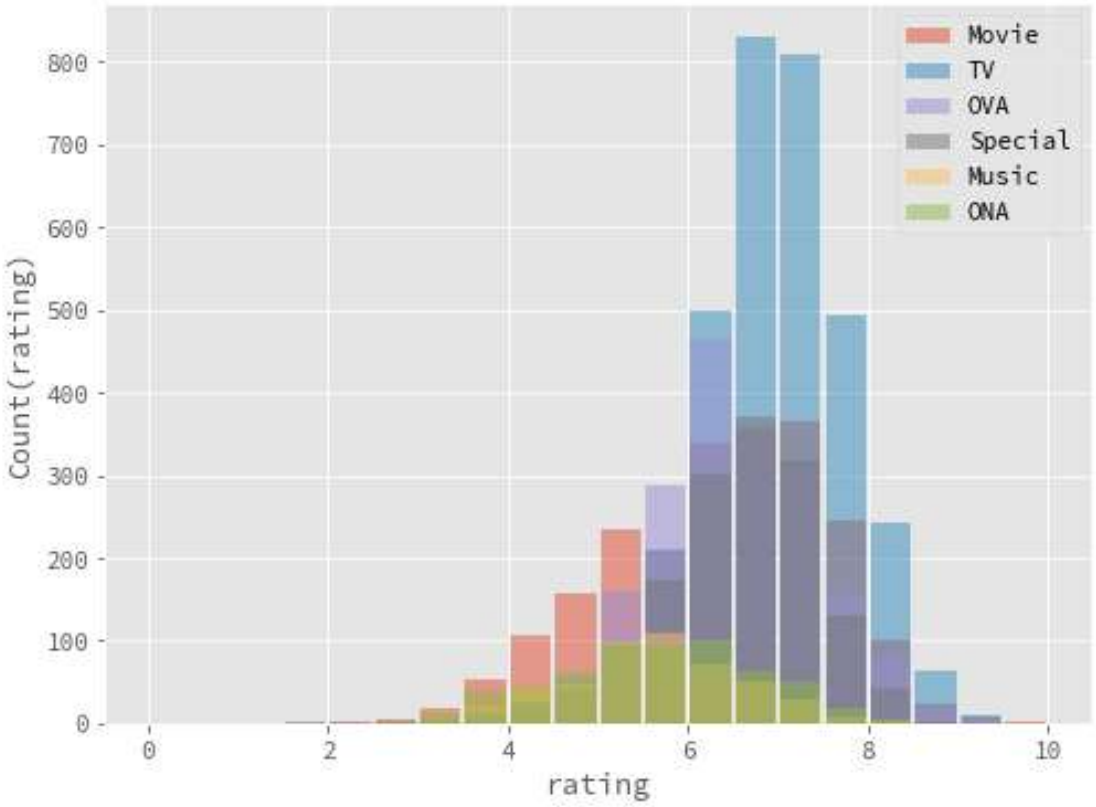
① df['rating'] 데이터 세트의 평균값과 표준편차 구하기
② numpy.linespace로 각 막대 단락 값(막대의 상한값과 하한값) 구하기
③ 구해진 평균값, 표준편차, 단락값으로부터 정규분포의 확률밀도함수에 따라 Y값 산출하기
④ 구해진 X값과 Y값으로 근사 곡선 그리기

3. 다양한 히스토그램

여러 그룹을 겹쳐서 그리기
같은 서브플롯에 히스토그램을 반복해서 그리면 여러 그룹의 히스토그램을 겹쳐서 그리는 것이 가능하다.

```
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()
```

3. 다양한 히스토그램



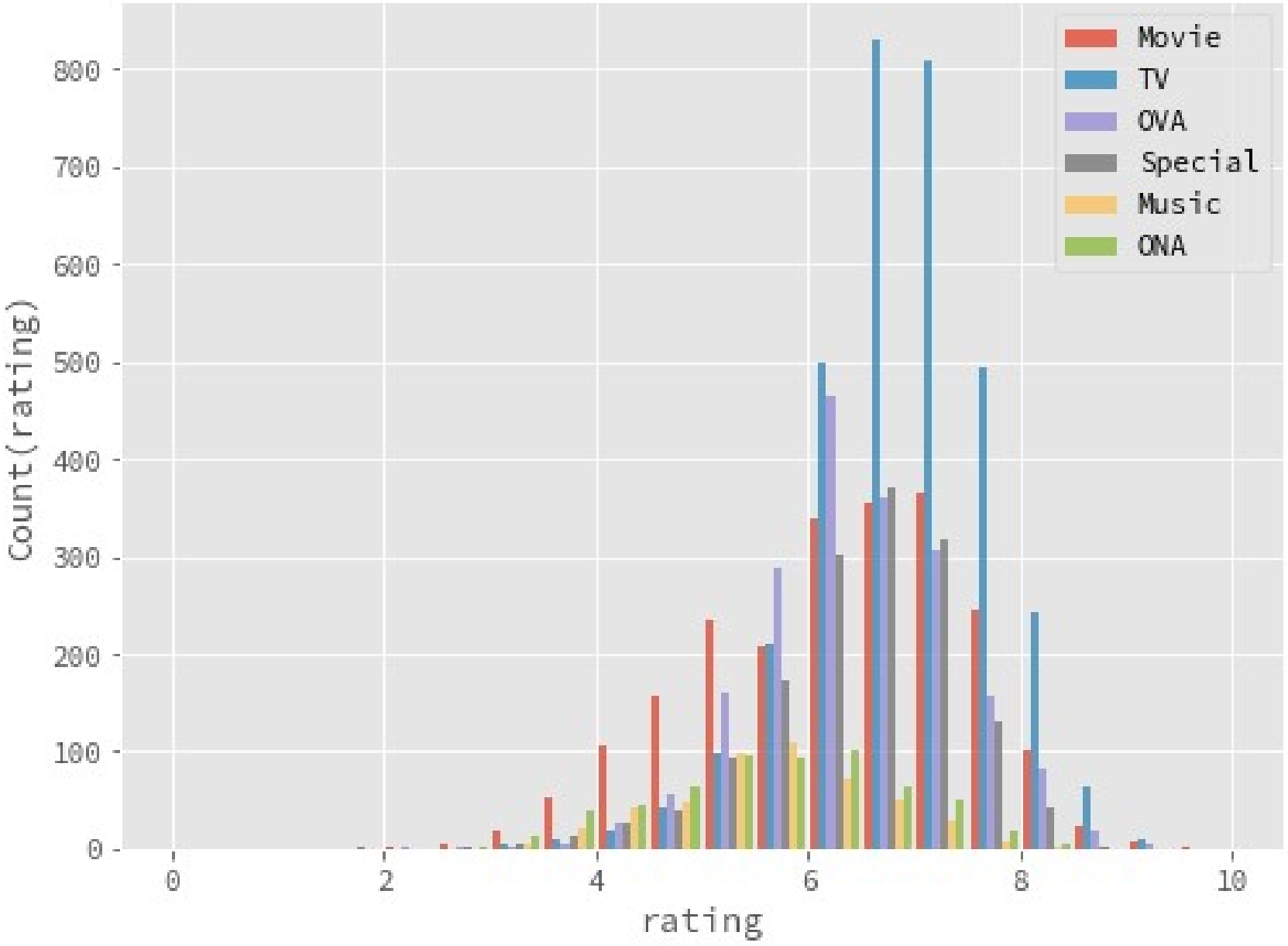
평균이 0부터 10의 범위에서 실행되기 때문에 range() 함수를 사용해서 b_num에 0.5씩 0부터 10 사이의 수치를 저장하고 있다. 또한 히스토그램이 겹쳐져 그려지기 때문에 alpha 옵션으로 불투명도를 낮추고 있다. alpha는 0부터 1의 실수로 지정한다. 0인 경우 완전히 투명, 1인 경우 완전히 불투명이 된다.

3. 다양한 히스토그램

여러그룹을 나열하여 그리기
여러 그룹의 히스토그램을 겹쳐 그려서 시인성이 떨어지는 경우에는 그룹을 나열하는 방법이 있다. 중첩 리스트를 작성한 후 그리면 여러 그룹을 옆으로 나열한 히스토그램을 그릴 수 있다.

```
# 데이터세트 작성
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()
```

3. 다양한 히스토그램

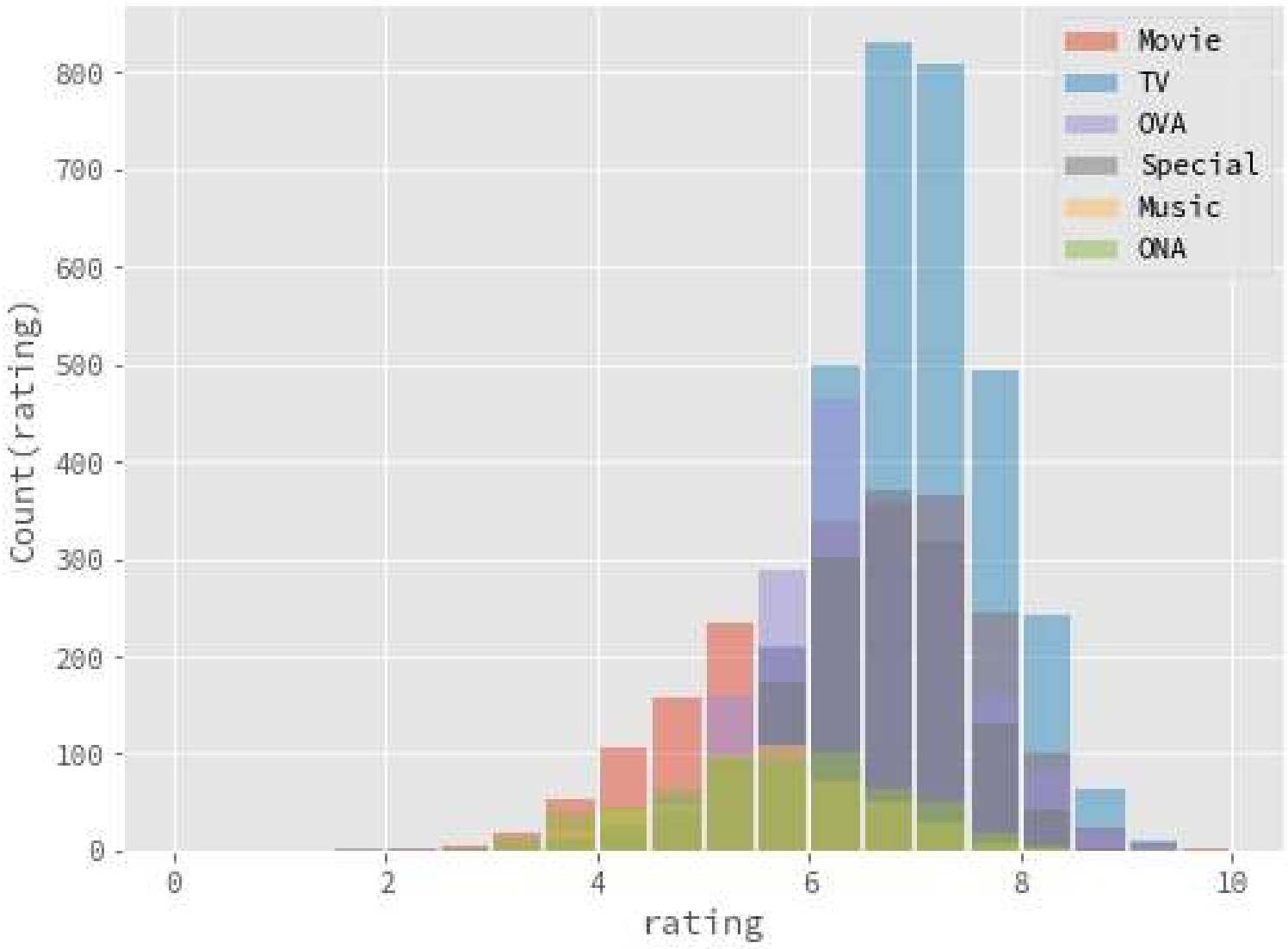

Movie
TV
OVA
Special
Music
ONA
Count(rating)
800
700
600
500
400
300
200
100
0
0
2
4
6
8
10
rating

3. 다양한 히스토그램

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 그려진다.

```
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()
```

3. 다양한 히스토그램

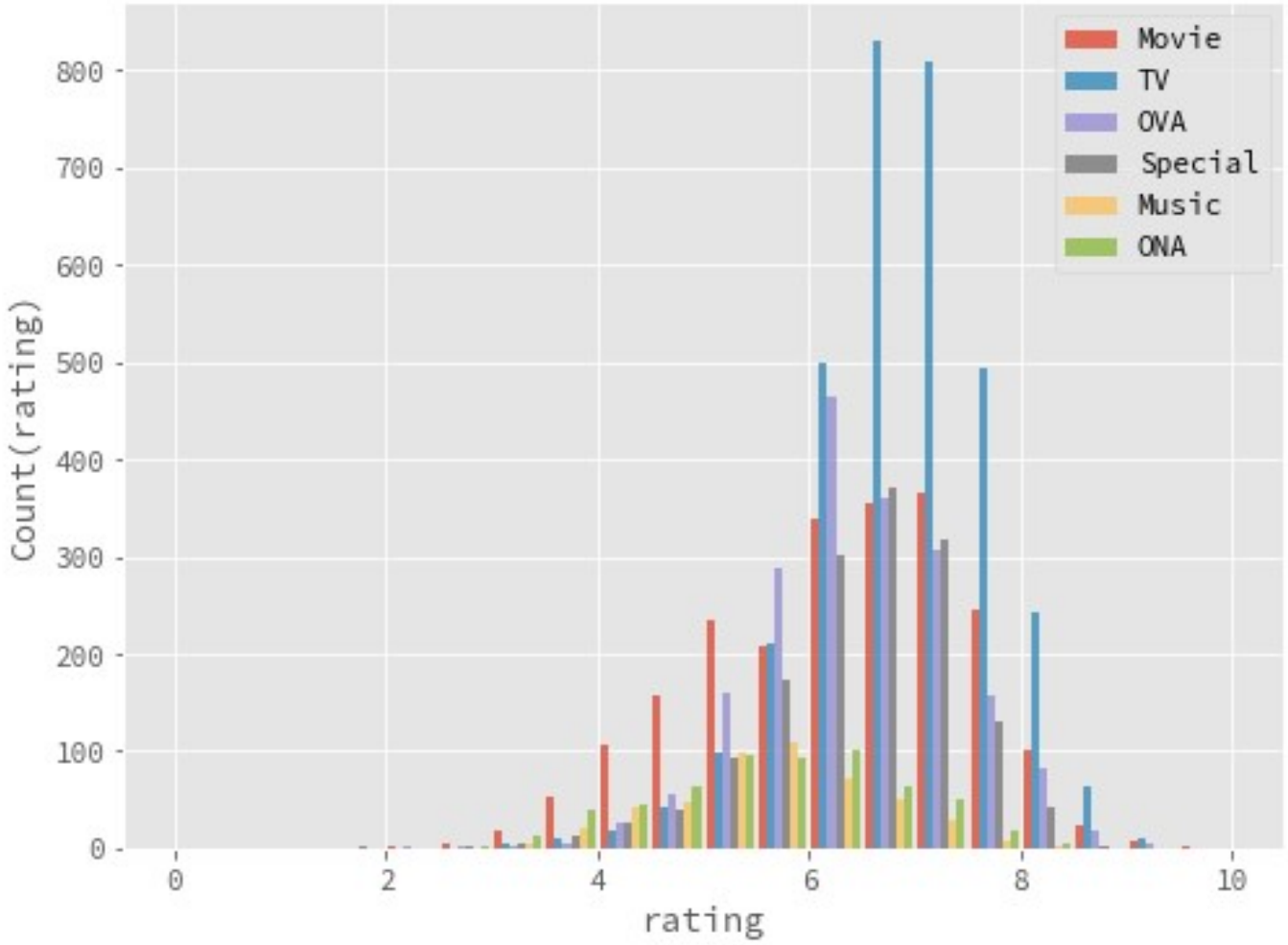

Movie
TV
OVA
Special
Music
ONA
Count(rating)
800
700
600
500
400
300
200
100
0
0
2
4
6
8
10
rating

3. 다양한 히스토그램

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 그려진다.

```
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()
```

3. 다양한 히스토그램



히스토그램은 그밖에도 상세한 그리기 설정이 가능하다.