```
Lab. Using Matplotlib
 3
    1. Matplotlib
       1)주로 2차원의 data를 시각화를 하기 위한 third-party package이다.
 5
       2)동작하는 OS를 가리지 않는다는 점, 자세한 그리기 설정이 가능한 점, 다양한 출력 형식에 대응하고 있는 점 등 대표적인 시각화 툴로 널리 사용되고 있다.
 6
       3)2003년 version 1.0이 발표된 이후로 15년 이상의 역사를 가진 tool이다.
 7
       4)사용자가 많은 이유는 산업계, 교육계에서 널리 사용되고 있는 수치 해석 S/W로, MATLAB과 같은 그리기를 Python에서 사용할 수 있는 것이다.
 8
       5)Matplotlib History: http://jakevdp.github.io/blog/2013/03/23/matplotlib-and-the-future-of-visualization-in-python/
 9
       6)Matplotlib에서는 graph의 종류나 축, 눈금선 graph 이름 등 다양한 그림의 요소에 대해 상세한 서식(색이나 선 종류 등)을 설정할 수 있다.
10
       7)다양한 출력 형식(PNG, SVG, JPG 등)에 대응하고 있다.
11
12
13
14
    2. Information
15
       1) Version: 3.1.1
16
       2)Site: https://matplotlib.org
17
       3)Repository: <a href="https://github.com/matplotlib/matplotlib/matplotlib">https://github.com/matplotlib/matplotlib</a>
       4)PyPI: https://pypi.python.org/pypi/matplotlib/
19
       5)Gallery: https://matplotlib.org/gallery/index.html
20
       5)Installation
21
          $ pip install matplotlib
22
23
24
25
    3. Graph 그리기 기초
26
       1)Graph 그리기 준비하기
27
          -matplotlib.pyplot module을 불러온다.
28
29
            import matplotlib.pyplot as plt
30
31
          -matplotlib graph를 출력할 때는 show()를 이용한다.
32
33
            plt.show()
34
35
          -package import 및 기본 설정
36
37
            import matplotlib.pyplotas plt
38
             %matplotlib inline
             %config InlineBackend.figure_format = 'retina'
39
            print("Matplotlib 버전:", matplotlib.__version__)
40
41
42
            Matplotlib 버전: 3.1.1
43
44
          -%matplotlib inline은 notebook을 실행한 브라우저에서 바로 그림을 볼 수 있게 해 준다.
45
          -%config InlineBackend.figure_format='retina' 옵션
46
             --('png'(기본값), 'retina', 'jpeg', 'svg', 'pdf' 중 하나)은 graph를 더 높은 해상도로 그려준다.
47
48
       2)package import 및 기본 설정이 완료되면 Matplotlib로 graph를 그리기 위해서 다음 단계를 따른다.
49
         a. 데이터 준비
50
          b. graph 생성
51
          c. graph 함수로 그리기
52
         d. graph 커스터마이징
53
          e. graph 출력 및 저장
54
55
          -다음 코드는 graph를 표시하는 간단한 예이다.
56
             plt.plot([1, 2, 3, 4])
57
58
            plt.ylabel('some numbers')
59
            plt.show()
60
61
62
            import numpy as np
63
            x = np.linspace(0, 5, 11)
            y = x ** 2
64
65
            Х
66
            array([0., 0.5, 1., 1.5, 2., 2.5, 3., 3.5, 4., 4.5, 5.])
67
68
69
             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')
71
72
73
74
            plt.title('Title')
75
76
77
            plt.subplot(1,2,1)
            plt.plot(x,y,'r')
78
79
            plt.subplot(1,2,2)
80
             plt.plot(y,x,'b')
81
```

```
1)Graph 그리기에서 한글이 깨지는 문제
 86
          -Graph를 그릴 때 caption이나 label 등에 수치 이외에 문자열을 출력하는 경우 한글을 처리할 때 종종 글자가 깨지는 문제에 부딪힌다.
 87
          -다음의 예제에서 caption의 한글이 깨지는 것을 볼 수 있다.
 88
          -이것은 Matplotlib의 초기 설정에서 사용하는 font가 한글 설정에서 포함되어 있지 않기 때문에 발생하는 문제이다.
 89
          -즉 미리 font를 설정하는 것으로 처리 가능하다.
 90
           -또한, 또 하나의 graph 작성 package인 Bokeh에서는 한글 출력이 가능하다.
 91
 92
             import numpy as np
 93
             from matplotlib import pyplot as plt
 94
 95
             np.random.seed(0)
 96
 97
             x = range(5)
             y = 10 + 5 * np.random.randn(5)
 98
 99
100
             fig = plt.figure()
             ax = fig.add\_subplot(111)
101
102
             ax.set_title('한글 테스트')
103
104
             ax.bar(x, y)
105
106
             plt.show()
107
108
        2)한글 font 설치하기
109
           -우리는 'Source Han Sans'를 사용할 것이다.
110
          -이 font는 Adobe와 Google이 공동으로 개발한 한국, 중국, 일본에 사용되고 있는 문자를 이용 가능한 font family의 명칭이다.
111
          -https://github.com/adobe-fonts/source-han-sans
112
          -Downloads
113
             --https://github.com/adobe-fonts/source-han-sans/tree/release
             --[Language-specific OTFs]에서 [Korean (한국어) link click
114
115
             --SourceHanSansK.zip
116
          -Unzip
117
             --SourceHanSansK-Medium.otf, SourceHansSansK-Bold.otf, SourceHansSansK-ExtraLight.otf,
             --SourceHansSansK-Heavy.otf, SourceHansSansK-Light.otf, SourceHansSansK-Normal.otf
118
           -Double click SourceHansSansK-Regular.otf
119
120
          -Install SourceHansSansK-Regular.otf
121
122
        3)Code 수정 후 확인
123
124
             import os
125
             import numpy as np
126
             from matplotlib import pyplot as plt, font manager
127
128
             #Font cache 재구축
129
             font_manager._rebuild()
130
131
             if os.name == "nt":
                #OS가 Windows 인 경우 win32FontDirectory()를 이용할 수 있다.
132
133
               font dir = font manager.win32FontDirectory()
134
135
             #font_path = os.path.join(font_dir, "SourceHanSansK-Regular.otf") <--경로 못 찾음.
136
             font\_path = r'font\_path = r'C:\Users\Instructor\AppData\Local\Microsoft\Windows\Fonts\SourceHanSansK-Regular.otf'
137
             font = font_manager.FontProperties(fname=font_path, size=14)
138
139
             np.random.seed(0)
140
141
             x = range(5)
142
             y = 10 + 5 * np.random.randn(5)
143
             fig = plt.figure()
144
145
             ax = fig.add\_subplot(111)
146
147
             #여기서 fontproperties를 지정한다.
148
             ax.set_title('한글 테스트', fontproperties=font)
149
             ax.bar(x, y)
150
151
             plt.show()
152
153
154
155
     5. Matplotlib 한글font 사용하기
156
        -https://brunch.co.kr/@jade/203
157
        -저작권 걱정 없는 무료 한글font
158
        1)필요한 패키지를 가져오기
159
160
           # 그래프를 노트북 안에 그리기 위해 설정
161
          %matplotlib inline
162
163
          # 필요한 패키지와 라이브러리를 가져옴
          import matplotlib as mpl
164
          import matplotlib.pyplot as plt
165
166
          import matplotlib.font_manager as fm
167
168
          # 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.
170
171
           mpl.rcParams['axes.unicode_minus'] = False
172
173
174
        2)font를 설정해 주기에 앞서 설치된 matplotlib 의 버전과 위치 정보를 가져온다.
           print ('버전: ', mpl.__version__)
175
          print ('설치 위치: ', mpl.__file__)
print ('설정 위치: ', mpl.get_configdir())
print ('캐시 위치: ', mpl.get_cachedir())
176
177
178
179
180
           버전: 3.1.1
181
           설치 위치: c:\pythonhome\projectenv\lib\site-packages\matplotlib\__init__.py
           설정 위치: C:\Users\Instructor\.matplotlib
182
183
           캐시 위치: C:\Users\Instructor\.matplotlib
184
185
186
        3)matplotlib의 위치 정보를 알았으니 터미널을 이용해 해당 위치로 가보자.
187
           print ('설정 파일 위치: ', mpl.matplotlib_fname())
188
189
           설정 파일 위치: c:\pythonhome\projectenv\lib\site-packages\matplotlib\mpl-data\matplotlibrc
190
191
192
        4)System에 설치된 font 확인
193
           font_list = fm.findSystemFonts(fontpaths=None, fontext='ttf')
194
           # ttf font 전체개수
195
           print(len(font_list))
196
197
           784
198
199
           font_list_win = fm.win32InstalledFonts()
200
           print(len(font_list_win))
201
           392
202
203
204
           font_list_win
205
206
           ['C:\\Windows\\Fonts\\NanumSquareR.ttf',
207
            'C:\\Windows\\Fonts\\ELEPHNTI.TTF',
208
           'C:\\Windows\\Fonts\\Candarab.ttf',
           'C:\\Windows\\Fonts\\FRAHV.TTF',
209
210
           'C:\\Windows\\Fonts\\LCALLIG.TTF'
211
           'C:\\Windows\\Fonts\\FRAHVIT.TTF',
212
213
214
215
           for fname in font list win:
216
              print(fname[17:])
217
218
           NanumSquareR.ttf
219
           ELEPHNTI.TTF
220
           Candarab.ttf
221
           FRAHV.TTF
222
           LCALLIG.TTF
223
224
225
226
        5)나눔 고딕을 사용할 예정이기 때문에 이름에 'Nanum'이 들어간 font만 가져온다.
227
228
           [fname for fname in font_list_win if 'Nanum' in fname]
229
230
           ['C:\\Windows\\Fonts\\NanumSquareR.ttf',
231
            'C:\\Windows\\Fonts\\NanumGothic.ttf',
232
           \verb|'C:\Windows\Fonts\NanumSquareRoundEB.ttf||,
           'C:\\Windows\\Fonts\\NanumBarunGothic.ttf',
233
234
           'C:\\Windows\\Fonts\\NanumMyeongjoExtraBold.ttf',
235
           'C:\\Windows\\Fonts\\NanumSquareRoundR.ttf',
236
237
238
        6)Font를 사용하는 방법은 3가지가 있다.
239
240
           -FontProperties 를 사용하는 방법 - graph의 font가 필요한 항목마다 지정해 주어야 한다.
241
           -matplotlib.rcParams[]으로 전역 글꼴 설정 방법 - 그래프에 설정을 해주면 font가 필요한 항목에 적용된다.
242
           -바로 위의 방법을 mpl.matplotlib_fname()로 읽어지는 설정 파일에 직접 적어주는 방법, 단 모든 notebook에 적용된다.
243
              --notebook을 열 때마다 지정해 주지 않아도 돼서 편리하다.
244
245
        7)FontProperties 를 사용하는 방법
246
247
           -텍스트를 지정하는 항목에 지정해 사용할 수 있다.
248
           -지정해 준 항목에만 해당 font가 적용 된다.
249
250
             matplotlib.pyplot
251
                -title()
252
                -xlabel()
```

```
253
                -ylabel()
254
                 -legend()
2.55
                -text()
256
             matplotlib.axes
257
258
                 -set_title()
259
260
              # fname 옵션을 사용하는 방법
              path = 'C:/Windows/Fonts/NanumBarunpenR.ttf'
261
262
             font = fm.FontProperties(fname=path, size=18)
263
264
             np.random.seed(0)
265
             x = range(5)
             y = 10 + 5 * np.random.randn(5)
266
267
268
             fig = plt.figure()
269
             ax = fig.add_subplot(111)
270
              #여기서 fontproperties를 지정한다.
271
272
              ax.set_title('한글 테스트', fontproperties=font)
273
             ax.bar(x, y)
274
             plt.show()
275
276
277
        8)matplotlib.rcParams[]으로 전역 글꼴 설정
278
           # 기본 설정 읽기
279
           import matplotlib.pyplot as plt
280
281
           # size, family
           print('설정되어있는 폰트 사이즈 :', plt.rcParams['font.size'])
282
           print('설정되어있는 폰트 글꼴 :',plt.rcParams['font.family'])
283
284
285
           설정되어있는 폰트 사이즈 : 10.0
           설정되어있는 폰트 글꼴 : ['sans-serif']
286
287
288
           # serif, sans-serif, monospace
           print('serif 세리프가 있는 폰트-----')
289
290
           print (plt.rcParams['font.serif'])
291
           print('sans-serif 세리프가 없는 폰트 ------')
292
           print (plt.rcParams['font.sans-serif'])
293
           print('monospace 고정폭 글꼴-----')
294
           print (plt.rcParams['font.monospace'])
295
296
           serif 세리프가 있는 폰트-----
           ['DejaVu Serif', 'Bitstream Vera Serif', 'Computer Modern Roman', 'New Century Schoolbook', 'Century Schoolbook L',
297
           'Utopia', 'ITC Bookman', 'Bookman', 'Nimbus Roman No9 L', 'Times New Roman', 'Times', 'Palatino', 'Charter', 'serif']
298
           sans-serif 세리프가 없는 폰트 --
           ['DejaVu Sans', 'Bitstream Vera Sans', 'Computer Modern Sans Serif', 'Lucida Grande', 'Verdana', 'Geneva', 'Lucid', 'Arial',
299
           'Helvetica', 'Avant Garde', 'sans-serif']
300
           monospace 고정폭 글꼴-----
           ['DejaVu Sans Mono', 'Bitstream Vera Sans Mono', 'Computer Modern Typewriter', 'Andale Mono', 'Nimbus Mono L', 'Courier
301
           New', 'Courier', 'Fixed', 'Terminal', 'monospace']
302
           plt.rcParams["font.family"] = 'nanummyeongjo'
plt.rcParams["font.size"] = 20
303
304
305
           plt.rcParams["figure.figsize"] = (14,4)
306
307
           np.random.seed(0)
308
           x = range(5)
           y = 10 + 5 * np.random.randn(5)
309
310
           fig = plt.figure()
311
312
           ax = fig.add\_subplot(111)
313
314
           #여기서 fontproperties를 지정하지 않는다.
           ax.set_title('한글 테스트')
315
316
           ax.bar(x, y)
317
           plt.show()
318
319
           -rcParams 대신 FontProperties 와 plt.rc 를 사용하는 방법
320
           import matplotlib.font_manager as fm
321
           path = 'C:/Windows/Fonts/NanumBarunGothic.ttf'
322
           font name = fm.FontProperties(fname=path, size=18).get name()
323
           font_name
324
325
           'NanumBarunGothic'
326
327
           plt.rcParams['font.family'] = font name
328
           # or plt.rc('font', family=font_name)
329
330
           fig, ax = plt.subplots()
331
           ax.plot(range(50))
332
           ax.set_title('시간별 가격 추이')
333
           plt.ylabel('주식 가격')
```

```
335
          plt.style.use('ggplot')
336
          plt.show()
337
338
339
       9)rcParams 를 설정 파일에 직접 적어주는 방법 - 모든 notebook에 공통적용
340
          -아래의 설정 파일의 위치에 가서 matplotlibrc을 수정한다.
341
          -이곳에 폰트를 지정해 주면 Notebook을 실행할 때 바로 load되도록 설정할 수 있다.
342
343
          print ('설정 파일 위치: ', mpl.matplotlib_fname())
344
345
          설정 파일 위치: C:\ProgramData\Anaconda3\lib\site-packages\matplotlib\mpl-data\matplotlibrc
346
347
          -위의 파일을 열어서 196line의 다음을 변경한다.
                             : sans-serif <---변경 전
348
            #font.family
349
            font.family
                            : nanummyeongjo <---변경 후
350
351
          -저장 후 Jupyter Notebook를 restart 한다.
352
          -Kernel > Restart
353
354
          # 기본 설정 읽기
355
          import matplotlib.pyplot as plt
356
357
          # size, family
358
          print('설정되어있는 폰트 사이즈 :', plt.rcParams['font.size'])
          print('설정되어있는 폰트 글꼴 :',plt.rcParams['font.family'])
359
360
361
          설정되어있는 폰트 사이즈 : 10.0
362
          설정되어있는 폰트 글꼴: ['nanummyeongjo']
363
364
          # import matplotlib.pyplot as plt
365
          # import numpy as np
366
367
          fig, ax = plt.subplots()
368
          ax.plot(10*np.random.randn(100), 10*np.random.randn(100), 'o')
369
          ax.set_title('숫자 분포도 보기')
370
          plt.show()
371
372
373
374
     6. Graph 객체
375
       1)우리가 그림을 그릴 때 가장 먼저 준비해야 하는 것이 도화지와 연필일 것이다.
376
       2)Python의 Matplotlib에서 graph를 그리기 위해 필요한 객체가 Figure이다.
377
       3)이 객체는 그림이 그려지는 도화지라고 생각할 수 있다.
378
       4)이 도화지(Figure)를 plt.subplots() 함수로 분할해 각 부분에 graph를 그리는 방식으로 시각화를 한다.
379
       5)graph를 그리기 위한 Figure 객체는 figure() 함수를 이용해 생성한다.
380
381
          -다음 코드는 graphic 객체를 생성한다.
382
383
            fig = plt.figure()
384
385
          -graph 객체의 속성을 설정하는 방법은(예를 들면 그래프 영역의 크기(size)를 조절하려면)
386
            a. graph 객체의 메소드를 이용하는 방법 - 예: fig.set_size_inches(10.5, 8,5)
387
            b. graph 객체를 만들 때 설정하는 방법 - 예: fig = plt.figure(figsize=(10.5, 8,5))
388
            c. reParams를 이용하는 방법 - 예: plt.rcParams['figure.figsize'] = (10.5, 8,5)
389
390
       6)Refer to https://matplotlib.org/api/_as_gen/matplotlib.pyplot.figure.html
391
392
393
394
     7. Graph 영역 나누기
395
       1) graph 객체를 여러 영역으로 나누어 그리면 하나의 graph 객체에 여러 graph를 그릴 수 있다.
396
        2)subplot() 함수로 서브플롯 추가
397
          -subplot() 함수는 현재 figure 객체에 서브플롯을 추가한다.
398
          -Syntax
399
            matplotlib.pyplot.subplot(*args, **kwargs)
                                        **kwargs)
400
          -subplot(nrows, ncols,
                                index,
401
          -subplot(pos, **kwargs)
402
          -subplot(ax)
403
404
          -args : 서브플롯의 위치를 설명하는 3자리 정수(예: 211) 또는 정수 3개(예: 2,2,1).
405
          -nrow, ncols, index : 3개의 정수가 nrows, ncols 및 index 순서로 있는 경우 그려질 하위 그림은 nrows 행과 ncols 열이 있는 표에서 index 위치에
406
            --index는 왼쪽 상단 모서리에서 1부터 시작하여 오른쪽으로 증가한다.
407
          -pos : pos는 3 자리 정수이며 첫 번째 숫자는 행 수, 두 번째 숫자는 열 수, 세 번째 숫자는 서브 그림의 index 이다.
408
            --즉, fig.add_subplot(235)는 fig.add_subplot(2, 3, 5)와 동일하다.
409
            --pos 형식이 작동하려면 모든 정수가 10보다 작아야한다.
            --subplot() 함수는 Figure.add_subplot() 함수의 랩퍼(Wrapper)이다.
410
411
          -Returns : 이 함수는 Figure 객체와 axes.Axes 객체(또는 Axes 객체들의 배열)를 반환한다.
412
413
          -다음 코드는 graph 영역을 나누고 각각의 영역에 graph를 그리는 예이다.
414
          -아래의 코드를 작성할 때 plot을 나누는 코드(subplot)와 graph를 그리는 코드(plot)는 같은 셀에 있어야 한다.
415
416
            import numpy as np
```

plt.xlabel('시간(분)')

```
import matplotlib.pyplotas plt
418
             %matplotlib inline
419
420
             x = np.arange(0, 10, 0.01)
421
             plt.subplot(2, 1, 1)
422
             plt.plot(x, np.sin(x))
             plt.subplot(2, 2, 3)
423
424
             plt.plot(x, np.cos(x))
             plt.subplot(2, 2, 4)
425
426
             plt.plot(x, np.sin(x)*np.cos(x))
427
             plt.show()
428
429
430
       3)subplots() 함수로 서브플롯 집합 추가
431
          -subplots() 함수는 현재 figure 객체에 서브플롯 집합을 추가한다.
432
          -figure 생성과 subplot의 배치를 동시에 실행하는 함수
433
434
             #figure object 작성과 subplot 배치를 동시에 실행
435
             fig, axes = plt.subplots(2,2)
436
             print(type(axes), axes)
437
             plt.show()
438
             <class 'numpy.ndarray'> [[<matplotlib.axes._subplots.AxesSubplot object at 0x00000209BD93E248>
439
                                             <matplotlib.axes._subplots.AxesSubplot object at 0x00000209BD9263C8>]
440
441
                                             [<matplotlib.axes._subplots.AxesSubplot object at 0x00000209BD77B8C8>
                                             <matplotlib.axes._subplots.AxesSubplot object at 0x00000209BD731D08>]]
442
443
          -행렬로 subplot의 위치를 지정할 수 있다.
444
445
446
             #1행 2열째 subplot에 subplot title을 지정
447
             fig, axes = plt.subplots(2,2)
448
             axes[0,1].set_title('Subplot 0-1')
449
             plt.show()
450
451
          -다음 코드는 subplots() 함수로 그래프 영역을 나누고, index를 이용해서 지정한 위치 영역에 그래프를 그린다.
452
453
             import numpy as np
454
             import matplotlib.pyplotas plt
455
             %matplotlib inline
456
457
             x = np.arange(0, 10, 0.01)
458
            fig, axes = plt.subplots(nrows=2, ncols=2)
459
             axes[0,0].plot(x,np.sin(x))
460
             axes[0,1].plot(x,np.cos(x))
             axes[1,0].plot(x,np.tanh(x))
461
462
             axes[1,1].plot(x,np.sin(x)*np.cos(x))
463
             plt.show()
464
465
          -subplots()로 나눈 화면영역은 enumerate()함수를 이용해 index와 grapg객체를 반복 처리 할 수 있다.
466
          -다음 코드는 이전 코드와 같은 결과를 출력할 것이다.
467
468
             import numpy as np
469
            import matplotlib.pyplotas plt
470
             %matplotlib inline
471
472
             x = np.arange(0, 7, 0.01)
473
474
             def sin_cos(x):
475
               return np.sin(x)*np.cos(x)
476
               \#graph 객체를 반복 처리하기 위해 함수를 list 안에 포함시킬 함수를 정의한다.
477
478
            func_list= [np.sin, np.cos, np.tanh,sin_cos]
479
             #그리고 graph 객체에서 반복 처리하기 위한 함수를 list로 선언한다.
480
481
          -다음 코드는 subplots() 함수로 그래프 영역을 나눈다.
482
          -그리고 enumerate(axes.flat)를 이용하면 index인덱스와 graph 객체를 반복문으로 처리할 수 있다.
483
484
                  axes = plt.subplots(nrows=2,ncols=2)
485
                 i, ax in enumerate(axes.flat):
486
               ax.plot(x, func_list[i](x))
487
488
             plt.show()
489
490
          -위의 예제에서 사용한 plot() 함수는 데이터를 이용해 점/선 graph를 그려준다.
491
492
          -subplots() 함수의 인수를 어떻게 선언하느냐에 따라 graph 영역이 다양하게 나뉜다.
493
          -다음 코드는 ncols=4(4개의 열)로 그래프 영역을 나눈다.
494
495
                  axes = plt.subplots(ncols=4)
496
                  i, ax in enumerate(axes.flat):
497
               ax.plot(x, func_list[i](x))
498
499
             plt.show()
500
```

```
502
503
              fig, axes = plt.subplots(nrows=4)
504
                  i, ax in
                              enumerate(axes.flat):
505
                ax.plot(x, func_list[i](x))
506
507
              plt.show()
508
509
510
        4)add_subplot()으로 서브플롯 배치하기
511
           -add_subplot(총행수, 총열수, 서브플롯번호)
512
513
              #figure 생성
514
              fig = plt.figure()
515
516
              #figure 안에 subplot 3개 배치
              ax1 = fig.add_subplot(221) #2행 2열 1번
517
              ax2 = fig.add subplot(222) #2행 2열 2번
518
519
              ax3 = fig.add_subplot(223) #2행 2열 3번
520
521
              plt.show()
522
523
           -각 subplot의 번호를 확인해보자.
524
525
              fig = plt.figure()
526
527
              #subplot 작성
528
              ax1 = fig.add_subplot(221)
529
              ax2 = fig.add\_subplot(222)
                                           #add_subplot(2,2,2) 도 가능
530
              ax3 = fig.add\_subplot(223)
531
532
              #번호 확인
533
             for i, ax in enumerate([ax1, ax2, ax3], start = 1):
                 txt = 'ax\{0\}\n(22\{0\})'.format(i)
534
535
                 ax.text(0.2, 0.4, txt, fontsize=24)
536
537
              plt.show()
538
539
540
541
     8. Graph 그리기
542
        1)Matplotlib의 다양한 graph 함수들에 대해 알아보자.
543
        2)Refer to https://matplotlib.org/api/_as_gen/matplotlib.pyplot.html
544
545
546
           -plot() 함수는 주어진 x, y 값을 선(lines)과 점(markers)으로 표시해 준다.
547
           -Svntax
548
             matplotlib.pyplot.plot([x], y, [fmt],
                                                    data=None, **kwargs)
549
           -fmt: 색, 점, 라인의 style을 문자열로 지정.
550
              --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.
551
552
553
554
           -다음 코드는 graph 객체를 만들고 graph 영역을 2x2 분할 한 후 각각의 영역에 graph를 그린다.
555
           -마지막 graph 영역에는 graph를 두 개 그린다.
556
557
              import matplotlib
558
              import matplotlib.pyplotas plt
559
              %matplotlib inline
560
561
              fig = plt.figure()
              fig, axes = plt.subplots(2, 2, figsize=(8,5))
562
563
              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
564
565
566
                                                                           'cs-.') # cyan(c), square(s), 대시닷(-.)
567
              axes[1,0].plot(np.linspace(0,
                                            5), np.cos(np.linspace(0, 5)))
              axes[1,1].plot([3,6], [3,5],
                                            'b^:')
568
                                                            # 파랑(b), 세모(^), 점선(:)
                                            'kx--')
                                                           # 검장(k), X(x), 대시선(--)
569
              axes[1,1].plot([4,5], [5,4],
570
              plt.show()
571
572
573
574
     9. Style 적용하기
575
        1)Style이란 graph의 선 굵기나 색 등 graph의 '체재'에 관한 정보를 모아놓은 것이다.
576
        2)Style은 style.use()함수로 적용할 수 있다.
577
        3)다음 코드는 ggplot style로 그리기를 하고 있다.
578
           #style 적용
579
580
           plt.style.use('ggplot')
581
           fig = plt.figure()
582
           ax = fig.add_subplot(111)
583
584
           dat = [0, 1]
```

-nrows=4로 하면 4개 행으로 graph 영역을 나눈다.

```
586
587
           plt.show()
588
589
590
591
     10. pandas를 사용한 data의 시각화
592
        1)pandas의 Series 또는 DataFrame의 plot()를 사용하여 쉽게 시각화할 수 있다.
        2)plot()은 내부에서 Matplotlib를 사용하고 있다.
593
594
        3)Notebook에 graph 표시하기
595
           -Notebook에 graph를 표시하기 위해서는 pyplot.show()를 사용한다.
596
597
             import pandas as pd
598
             import matplotlib.pyplot as plt
599
600
             ax = pd.Series([1,2,3]).plot()
601
             ax.set_title('Line Chart')
602
             plt.show()
603
             #graph를 그릴 때는 cell을 바꾸기 않고 하나의 cell안에 모두 coding해야 한다.
604
605
             column_names = ['Hakbun', 'Name', 'Kor', 'Eng', 'Mat', 'Edp']
606
             df = pd.read_csv('pandas_data/sungjuk_utf8.csv', names = column_names)
607
             df['Total'] = df['Kor'] + df['Eng'] + df['Mat'] + df['Edp']
             df['Average'] = df['Total'] / 4
608
609
             grade_list = []
             for avg in df['Average']:
610
611
                if 90 <= avg <= 100 : grade_list.append('A')
612
                 elif 80 <= avg < 90 : grade_list.append('B')
613
                 elif 70 \le avg \le 80: grade_list.append('C')
                 elif 60 <= avg < 70 : grade_list.append('D')
614
615
                 else: grade_list.append('F')
616
             df['Grade'] = grade_list
617
618
619
             ax = df['Kor'].plot().set_title('Line Chart')
620
             plt.show()
621
622
           -Graph의 style을 변경하는 경우에는 pyplot.style.use()의 인수에 style명을 넘긴다.
623
624
             plt.style.use('ggplot') #기본은 'default'
625
626
        4)DataFrame에서 plot하기
627
628
           -DataFrame에서 plot()을 호출할 경우 기본적으로 Series와 같은 동작을 수행하는데, 열 수에 상응하는 요소가 그려진다.
629
           -Index가 X값, 이름 열의 값이 Y값이 된다.
630
631
             df = pd.DataFrame(\{'a':[1,2,3], 'b':[3,2,1]\})
632
             ax = df.plot().set_title('Line Chart')
633
             plt.show()
634
635
636
        5)Y축 범위가 다른 경우
637
           -Keyword 인수 secondary_y에 두번째 축이 되는 열 이름을 list형으로 지정한다.
638
           -Y축의 label은 set_ylabel(), right_ax.set_ylabel()의 인수에 각각의 이름을 넘겨준다.
639
640
             ser1 = df['Hakbun']
641
             ser2 = df['Kor']
642
             df2 = pd.DataFrame(ser1, columns=['Hakbun'])
643
             df2['Kor'] = ser2
644
645
             ax = df2.plot(secondary_y=['Kor'])
646
             ax.set_title('Two Line Chart')
647
             ax.set_ylabel('Hakbun')
648
             ax.right_ax.set_ylabel('Kor')
649
             plt.show()
650
651
652
        6)산포도 graph 그리기
653
           -plot.scatter()를 사용한다.
654
           -Keyword 인수 x에 X값이 되는 열 이름, keyword 인수 y에 Y 값이 되는 열 이름을 지정한다.
655
656
             ax = df2.plot.scatter(x='Hakbun', y='Kor')
657
             ax.set title('Scatter')
658
             plt.show()
659
660
661
        7)막대 Graph 작성하기
           -plot.bar()를 사용한다.
662
663
664
             import cx_Oracle
             conn = cx_Oracle.connect('scott', 'tiger', 'localhost:1521/XE')
665
666
             cursor = conn.cursor()
667
             sql = "SELECT empno, ename, job, TO CHAR(hiredate, 'YYYY'), mgr, sal, comm, deptno FROM emp"
668
             cursor.execute(sql)
```

ax.plot(dat)

```
670
             for empno, ename, job, hiredate, mgr, sal, comm, deptno in cursor:
671
                emp_list.append([empno, ename, job, hiredate, mgr, sal, comm, deptno])
              columns = ['empno', 'ename', 'job', 'hiredate', 'mgr','sal', 'comm', 'deptno']
672
673
             df = pd.DataFrame(emp_list, columns = columns)
674
675
676
             group_deptno = df.groupby('deptno')
677
             group_deptno.groups
678
679
              {10: Int64Index([6, 8, 13], dtype='int64'),
              20: Int64Index([0, 3, 7, 10, 12], dtype='int64'),
680
681
              30: Int64Index([1, 2, 4, 5, 9, 11], dtype='int64')}
682
683
             for name, group in group_deptno:
                print(str(name) + ": " + str(len(group)))
684
685
                print(group)
686
                print()
687
688
             10:3
689
                   empno
                              ename
                                        iob
                                                        hiredate
                                                                   mar
                                                                           sal
                                                                                    comm
                                                                                               deptno
             6
                                      MANAGER
                                                                   7839.0 2450.0
690
                   7782
                             CLARK
                                                      1981
                                                                                      NaN
                                                                                               10
691
             8
                   7839
                           KING
                                      PRESIDENT
                                                   1981
                                                                 NaN 5000.0
                                                                                NaN
                                                                                         10
             13
                   7934
                                                                                    NaN
692
                             MILLER CLERK
                                                   1982
                                                                 7782.0 1300.0
693
             20: 5
694
                                     job
695
                   empno ename
                                                      hiredate
                                                                   mgr
                                                                              sal
                                                                                      comm
                                                                                              deptno
696
                   7369
                             SMITH
                                        CLERK
                                                      1980
                                                                   7902.0
                                                                              800.0
                                                                                      NaN
                                                                                              20
                                                                 7839.0
                                                                            2975.0 NaN
                                                                                            20
697
             3
                   7566
                              JONES
                                     MANAGER
                                                   1981
                   7788
                                                   1987
698
                              SCOTT
                                     ANALYST
                                                                 7566.0
                                                                            3000.0 NaN
                                                                                            20
699
             10
                   7876
                              ADAMS CLERK
                                                   1987
                                                                 7788.0
                                                                            1100.0 NaN
                                                                                            20
700
                   7902
                              FORD
                                      ANALYST
                                                   1981
                                                                 7566.0
                                                                            3000.0 NaN
                                                                                            20
701
             30: 6
702
703
                   empno
                              ename
                                        job
                                                         hiredate
                                                                   mgr
                                                                                                 deptno
                                                                              sal
                                                                                       comm
                                     SALESMAN
                                                   1981
                                                                 7698.0
                                                                                      300.0
704
             1
                   7499
                              ALLEN
                                                                            1600.0
                                                                                                 30
705
             2
                   7521
                           WARD
                                  SALESMAN
                                                              7698.0
                                                                         1250.0
                                                                                    500.0
                                                                                               30
                              MARTIN
706
             4
                                        SALESMAN
                                                      1981
                                                                   7698.0
                                                                              1250.0
                                                                                         1400.0
                   7654
                                                                                                   30
             5
                                                                                                 30
707
                   7698
                              BLAKE
                                        MANAGER
                                                      1981
                                                                   7839.0
                                                                              2850.0
                                                                                        NaN
708
             9
                   7844
                              TURNER SALESMAN
                                                   1981
                                                                 7698.0
                                                                            1500.0
                                                                                      0.0
                                                                                                 30
                                                                                               30
709
             11
                   7900
                              JAMES
                                       CLERK
                                                   1981
                                                                 7698.0
                                                                            950.0
                                                                                      NaN
710
711
           names = []
           values = []
712
713
           for name, group in group_deptno:
714
             names.append(name)
715
             values.append(len(group))
716
717
           plt.figure(1, figsize=(9, 3))
718
           plt.subplot(131)
719
           plt.bar(names, values)
720
           plt.subplot(132)
721
           plt.scatter(names, values)
722
           plt.subplot(133)
723
           plt.plot(names, values)
724
           plt.suptitle('Categorical Plotting')
725
           plt.xlabel('Department Number')
726
           plt.show()
727
728
729
        8)다양한 Graph 그리기
730
731
           fig, axs = plt.subplots(2,2, figsize=(5,5))
732
           axs[0,0].hist(df['sal'])
733
           axs[1,0].scatter(df['empno'], df['sal'])
734
           axs[0,1].plot(df['empno'], df['sal'])
735
           axs[1,1].hist2d(df['empno'], df['sal'])
736
737
738
        9)scatter()
739
           -scatter() 함수는 산점도 그래프를 그려준다.
740
           -Refer to https://matplotlib.org/api/_as_gen/matplotlib.pyplot.scatter.html
741
742
           import numpy as np
743
           import matplotlib.pyplotas plt
744
           np.random.seed(7902)
745
           N = 50
746
           x = np.random.rand(N)
           y = np.random.rand(N)
747
748
           colors = np.random.rand(N)
749
           area = (30 * np.random.rand(N))**2
750
           plt.scatter(x, y, s=area, c=colors, alpha=0.5)
751
           plt.show()
752
```

 $emp_list = []$ 

```
753
           import matplotlib.pyplot as plt
754
           import pandas as pd
755
           from matplotlib import font_manager, rc
756
           font_name = font_manager.FontProperties(fname="c:/Windows/Fonts/malgun.ttf").get_name()
757
           rc('font', family=font_name)
758
           df = pd.read_csv('korea.csv',encoding='ms949')
           plt.figure()
759
760
           plt.scatter(x=df.index,y=df['점수'], marker='2')
761
           plt.xticks(range(0,len(df['점수']),1),df['이름'], rotation='vertical')
762
           plt.title('학생별 국어점수 산포도')
763
           plt.show()
764
765
766
        10)Histogram
767
768
           import matplotlib.pyplot as plt
769
           import pandas as pd
770
           from matplotlib import font manager, rc
           font_name = font_manager.FontProperties(fname="c:/Windows/Fonts/malgun.ttf").get_name()
771
772
           rc('font', family=font_name)
773
           df = pd.read csv('csv exam1.csv',encoding='ms949')
774
           data = pd.concat([df['국어'],df['영어'],df['수학']])
775
           plt.hist(data, bins=3)
776
           plt.xticks(range(0,100,40),['하', '중', '상'])
777
           plt.title('점수빈도')
778
           plt.show()
779
780
           import matplotlib.pyplot as plt
781
           import pandas as pd
782
           from matplotlib import font_manager, rc
783
           font_name = font_manager.FontProperties(fname="c:/Windows/Fonts/malgun.ttf").get_name()
784
           rc('font', family=font_name)
785
           df = pd.read_csv('csv_exam1.csv',encoding='ms949')
786
           plt.hist((df['국어'],df['영어'],df['수학']), bins=10, label=('국어','영어','수학'))
787
           plt.title('점수빈도')
788
           plt.legend()
789
           plt.show()
790
791
792
        11)bar
793
794
           import matplotlib.pyplot as plt
795
           import pandas as pd
796
           from matplotlib import font_manager, rc
797
           font_name = font_manager.FontProperties(fname="c:/Windows/Fonts/malgun.ttf").get_name()
798
           rc('font', family=font_name)
799
           df = pd.read csv('korea.csv',encoding='ms949')
800
           print(df)
801
           plt.figure()
           plt.bar(df.index, df['점수'],width=1.0, color='r')
802
803
           plt.xticks(range(0,len(df.index),1),df['이름'], rotation='vertical')
804
           plt.title('학생별 국어 점수')
805
           plt.show()
806
807
           import matplotlib.pyplot as plt
808
           import pandas as pd
809
           from matplotlib import font manager, rc
810
           font name = \
811
               font manager.FontProperties(fname="c:/Windows/Fonts/malgun.ttf").get name()
812
           rc('font', family=font_name)
813
           df = pd.read_csv('csv_exam1.csv',encoding='ms949')
814
           print(df)
815
           plt.figure()
           plt.barh(df.index, df['국어'], color='r', label='국어')
plt.barh(df.index, -df['영어'], color='g', label='영어')
816
817
818
           plt.title('학생별 국어,영어 점수')
           plt.yticks(range(0,len(df.index),1),df['이름'], rotation='horizontal')
819
820
           plt.xticks([-100,-50,0,50,100],(100,50,0,50,100))
821
           plt.legend()
822
           plt.show()
823
824
825
        12)line
826
827
           import matplotlib.pyplot as plt
828
           from pandas import Series, DataFrame
829
           s = Series([84900, 818000, 1756,292000])
           # 객체생성
830
831
           plt.figure()
832
           # 출력
833
           plt.plot(s)
834
           plt.show()
835
836
           import matplotlib.pyplot as plt
```

```
837
           from pandas import Series
           s1 = Series([84900, 81800, 71756, 92000]) #Series
s2 = Series([80500, 82000, 71736, 90000]) #Series
838
839
840
           plt.figure(figsize=(10,4))
           plt.plot(s1, label='04-10')
plt.plot(s2, label='04-11')
841
842
843
           plt.grid()
844
           plt.xlabel('index')
845
           plt.ylabel('stock')
846
           plt.title('plot graph')
847
           plt.legend()
848
           plt.show()
849
850
851
        13)box
852
853
           import matplotlib.pyplot as plt
854
           import pandas as pd
855
856
           from matplotlib import font_manager, rc
857
           font_name = \
858
               font_manager.FontProperties(fname="c:/Windows/Fonts/malgun.ttf").get_name()
859
           rc('font', family=font_name)
860
           df = pd.read_csv('csv_exam1.csv',encoding='ms949')
861
           print(df)
           plt.boxplot((df['국어'],df['영어'],df['수학']), labels=('국어','영어','수학'))
862
           print(df['수학'].min())
print(df['수학'].mean())
863
864
865
           print(df['수학'].median())
           plt.title('점수분포')
866
867
           plt.show()
868
869
870
871
     11. 꺾은선 그래프
872
        1)꺾은선 그래프는 plot된 점과 점을 직선으로 연결한 그래프이다.
873
        2)꺾은선 그래프 작성하기
874
           -Axes.plot()으로 그린다.
875
           -plot()의 인수가 하나뿐인 경우, 부여된 인수는 Y값으로 설정되어 X값은 자동적으로 '최소값=0' '최대값=list의 요소수-1'의 정수열이 지정된다.
876
877
              fig = plt.figure()
878
              ax = fig.add\_subplot(111)
879
880
              ax.plot([1,3])
881
              plt.show()
882
883
           -위의 코드에서는 그리기 대상 data로 list형 데이터가 넘겨졌지만, plot()에서는 다음과 같은 data형이 사용된다.
884
              a. list
885
              b. tuple
886
              c. numpy.ndarray
887
              d. pandas. Series
888
889
           -다음 코드는 전형적인 꺾은선 그래프이다.
890
891
              fig = plt.figure()
892
              ax = fig.add\_subplot(111)
893
894
              x = [0,2,4]
895
              y = [0,4,2]
896
              ax.plot(x, y)
897
              plt.show()
898
899
           -여러 개의 선을 그리는 경우
900
           -plot()을 여러 번 실행하면 1개의 subplot에 여러 개의 graph를 겹쳐서 그릴 수 있다.
901
902
              fig = plt.figure()
              ax = fig.add_subplot(111)
903
904
905
              x = [0,2,4]
906
              y1 = [0, 4, 2.5]
907
              y2 = [4,0, 1.5]
908
909
              #2개의 선 그리기
              ax.plot(x, y1)
910
911
              ax.plot(x, y2)
912
              plt.show()
913
914
915
        3)꺾은선 그래프 활용하기
916
           -실제 데이터를 이용해서 그래프를 그려보자.
917
           -데이터는 anime_stock_returns.csv 파일이다.
918
           -이 파일에는 TOEI ANIMATION 및 IG Port의 주가 등락률이 시계열(일단위)로 기록되어 있다.
919
920
              import os
```

```
922
              base url = 'https://raw.githubusercontent.com/practical-jupyter/sample-data/master/anime/'
 923
              anime_stock_returns_csv = os.path.join(base_url, 'anime_stock_returns.csv')
 924
 925
              df = pd.read_csv(anime_stock_returns_csv, index_col = 0, parse_dates = ['Date'])
 926
 927
 928
                                  TOEI ANIMATION
                                                       IG Port
 929
              Date
              2015-01-01
 930
                                1.000000
                                                       1.000000
 931
              2015-01-02
                                1.000000
                                                       1.000000
 932
              2015-01-05
                                1.011695
                                                       1.014082
 933
              2015-01-06
                                1.001463
                                                       1.000000
 934
              2015-01-07
                                0.982457
                                                       1.000824
 935
 936
           -시계열 정보를 포함한 데이터를 표현하는 것은 꺾은선 그래프가 적당하다.
 937
 938
              from matplotlib import font manager, rc
 939
              font_name = font_manager.FontProperties(fname="c:/Windows/Fonts/malgun.ttf").get_name()
 940
              rc('font', family=font_name)
 941
 942
              fig = plt.figure(figsize=(10,4))
 943
              ax = fig.add\_subplot(111)
 944
 945
              #data와 범례 지정
              ax.plot(df.index, df['TOEI ANIMATION'], label='TOEI ANIMATION')
 946
 947
              ax.plot(df.index, df['IG Port'], label='IG Port')
 948
 949
              #title, 축레이블 지정
 950
              ax.set_title('주가등락률 2년간 추이')
 951
              ax.set_ylabel('주가등락률')
 952
              ax.set_xlabel('년월')
 953
 954
              #범례 유효화
 955
              ax.legend()
 956
              plt.show()
 957
 958
 959
         4)2개의 축을 가진 graph 그리기
 960
           -Matplotlib에서 X축을 공유해서 2개의 Y축을 가진 그림을 작성하는 경우에는 Axes.twinx() 함수를 사용한다.
 961
           -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.
 962
 963
           -마감가는 꺾은선 graph로, 거래량은 막대 graph로 나타내는 것이 일반적이다.
 964
 965
              t4816_csv = os.path.join(base_url, "4816.csv")
 966
              df = pd.read_csv(t4816_csv, index_col=0, parse_dates=["Date"])
 967
 968
              fig = plt.figure(figsize=(10, 4))
 969
              ax1 = fig.add_subplot(111)
 970
 971
              ax1.plot(df.index, df["Close"], color="b", label="주가")
 972
 973
              \#X축을 공유해서 Y축을 2개 사용하는 설정
 974
              ax2 = ax1.twinx()
 975
              ax2.bar(df.index, df["Volume"], color="g", label="거래총액", width=2)
 976
 977
              # 축과 축레이블 설정
              ax1.set_yticks([i * 2000 for i in range(5)])
 978
 979
              ax1.set ylabel("주가")
              ax2.set_yticks([i * 50000 for i in range(5)])
 980
              ax2.set_ylabel("거래총액")
ax1.set_xlabel("년월")
 981
 982
 983
 984
              # Graph 타이틀 설정
 985
              ax1.set_title("주가와 거래총액")
 986
 987
              #범례설정
 988
              ax1.legend(loc=1)
 989
              ax2.legend(loc=2)
 990
              plt.show()
 991
 992
 993
 994
      12. 산포도 Graph
 995
         1)산포도 Graph는 X축과 Y축에 수량이나 크기 등을 대응시켜서 적합한 점에 데이터를 플롯한 graph이다.
 996
         2)산포도 Graph는 X축과 Y축에 취한 2개의 값(Z축이 있는 경우에는 3개의 값)에 함수가 있는지 없는지 보는 것에 유용하다.
 997
         3)또한 데이터의 분포 상황을 확인할 때에도 활용할 수 있다.
 998
         4)산포도 Graph 작성하기
 999
           -Axes.scatter() 함수를 사용해서 그린다.
1000
           -제1,제2인수에 각각 X값과 Y값을 부여한다.
1001
1002
              plt.style.use("ggplot")
1003
1004
              #입력값 생성
```

import pandas as pd

```
1006
                           x = np.arange(1, 101)
                           y = 4 * x * np.random.rand(100)
1007
1008
1009
                           # 산포도 graph 그리기
1010
                           fig = plt.figure()
1011
                           ax = fig.add\_subplot(111)
1012
                           ax.scatter(x, y)
1013
                           plt.show()
1014
1015
                 5)산포도 Graph 활용하기
1016
                      -실제 데이터를 이용해서 그래프를 그려보자.
1017
                      -데이터는 anime_master.csv 파일이다.
1018
                      -데이터를 불러오면 애니메이션의 제목이나 장르, 에피소드 수나 평점 데이터가 포함되어 있다.
1019
1020
                           import os
1021
                           import pandas as pd
1022
                           base\_url = "https://raw.githubusercontent.com/practical-jupyter/sample-data/master/anime/" and the property of the property 
1023
1024
                           anime_master_csv = os.path.join(base_url, "anime_master.csv")
1025
                           df = pd.read_csv(anime_master_csv)
1026
                           df.head()
1027
1028
1029
                      -anime_id를 인수 index_col로 지정해서 anime_id를 index에 설정할 수 있다.
1030
1031
                           df = pd.read_csv(anime_master_csv, index_col="anime_id")
1032
                           df.head()
1033
1034
1035
                      -X값으로 members를, Y값으로 rating을 지정하는 것에 따라 산포도 그래프를 작성할 수 있다.
1036
                      -그려진 기호를 반투명으로 하는 값(alpha=0.5)도 설정한다.
1037
1038
                           fig = plt.figure()
1039
                           ax = fig.add_subplot(111)
                           ax.scatter(df["members"], df["rating"], alpha=0.5)
1040
1041
1042
1043
1044
                 6)그룹화된 산포도 graph 작성하기
1045
                      -위의 데이터는 type이라는 열을 가지고 있다.
1046
                      -type은 애니메이션 작품의 배급 종별을 의미한다.
1047
                      -먼저 type 중복 없는 list를 작성한다.
1048
                           types = df['type'].unique()
1049
1050
1051
                           array(['Movie', 'TV', 'OVA', 'Special', 'Music', 'ONA'], dtype=object)
1052
1053
1054
                      -하나의 subplot에 겹쳐서 산포도 graph를 그린다.
1055
                      -다음 코드는 배급 종별(type)마다 일치하는 데이터를 추출해서 그리고 있다.
1056
                      -배급 종별은 6종류가 있기 때문에 6개의 데이터 세트가 플롯되어 있다.
1057
1058
                           fig = plt.figure(figsize=(10, 5))
                           ax = fig.add\_subplot(111)
1059
1060
                           for t in types:
                                 x = df.loc[df["type"] == t, "members"]
y = df.loc[df["type"] == t, "rating"]
1061
1062
                                 ax.scatter(x, y, alpha=0.5, label=t)
1063
                           ax.set_title("배급 종별로 그룹화한 데이터 산포도 그래프")
1064
                           ax.set_xlabel("Members")
1065
                           ax.set_ylabel("Rating")
1066
1067
                           ax.legend(loc="lower right", fontsize=12)
1068
                           plt.show()
1069
1070
1071
1072
            13. 막대 그래프
1073
                 1)막대그래프는 수량을 막대의 길이로 나타낸 그래프이다.
1074
1075
                      -막대그래프는 Axes.bar() 함수를 사용해서 그린다.
1076
                      -제1인수, 제2인수에 각각 X값과 Y값을 부여한다.
1077
                      -데이터로서 list형, object를 이용할 수 있다.
1078
1079
                           plt.style.use("ggplot")
1080
                           fig = plt.figure()
1081
                           ax = fig.add\_subplot(111)
                           x = [1, 2]
1082
1083
                           y = [1, 3]
1084
                           ax.bar(x, y)
1085
                           plt.show()
1086
1087
1088
                 3)눈금레이블을 붙일 경우
```

np.random.seed(2)

```
-인수 tick_label에 눈금레이블을 설정해서 작성한다.
           -label은 list나 tuple로 부여한다.
1090
1091
1092
              fig = plt.figure()
1093
              ax = fig.add_subplot(111)
1094
              labels = ["apple", "orange"]
1095
              ax.bar(x, y, tick_label=labels)
1096
1097
1098
           -graph를 그린 후 Axes.set_xticks()로 X축 눈금을 설정하고 Axes.set_xtickabels()로 눈금 레이블을 설정한다.
1099
1100
              #그리기
1101
              fig = plt.figure()
1102
              ax = fig.add\_subplot(111)
1103
              ax.bar(x, y)
1104
1105
              #X축의 축눈금과 축눈금 레이블
1106
              ax.set xticks(x)
1107
              ax.set_xticklabels(labels)
1108
              plt.show()
1109
1110
        4)수평 막대그래프를 작성하는 경우
1111
1112
           -수평 막대그래프는 Axes.barh()를 이용해서 그린다.
1113
           -barh()의 인수는 기본적으로 bar()와 같다.
1114
1115
              fig = plt.figure()
1116
              ax = fig.add_subplot(111)
1117
              ax.barh(x, y, tick_label=labels)
1118
              plt.show()
1119
1120
1121
         5)막대 그래프 활용하기
1122
           -실제 데이터를 이용해서 그래프를 그려보자.
1123
           -데이터는 anime_master.csv 파일이다.
1124
1125
              import os
1126
             import pandas as pd
1127
1128
              base_url = "https://raw.githubusercontent.com/practical-jupyter/sample-data/master/anime/"
1129
              anime_master_csv = os.path.join(base_url, "anime_master.csv")
1130
              dfac = pd.read_csv(anime_master_csv)
1131
              dfac.head()
1132
1133
           -막대 그래프는 수량의 대소를 시각화할 때 적당하다.
1134
           -여기서는 작품의 배급 종별마다 멤버수의 합계 값을 추출해서 막대그래프로 그린다.
1135
           -이처럼 데이터를 시각화하여 배급 종별에서는 텔레비젼 작품의 멤버 수가 돌출되는 것을 확인할 수 있다.
1136
1137
              fig = plt.figure()
1138
              ax = fig.add_subplot(111)
1139
              y = dfac.groupby("type").sum()["members"]
1140
             x = range(len(y))
1141
              xlabels = y.index
1142
              ax.bar(x, y, tick_label=xlabels)
1143
              ax.set_ylabel("합계 멤버수")
1144
              plt.show()
1145
1146
1147
        6)여러가지 그룹에 대한 막대그래프 작성하기
1148
           -여러 번 bar()를 실행하면 최초에 그려진 오브젝트가 뒤에 그려진 오브젝트에 의해 덮어씌워진다.
1149
1150
             import numpy as np
1151
1152
              # 데이터 세트 작성
1153
             x = [1, 2]
1154
             y1, y2, y3 = [1, 2], [2, 4], [3, 6]
1155
1156
              # 복수 그룹의 막대 그래프
1157
             fig = plt.figure()
1158
              ax = fig.add\_subplot(111)
1159
1160
             w = 0.2
1161
             ax.bar(x, y1, label="y1")
              ax.bar(x, y2, label="y2")
1162
1163
              ax.bar(x, y3, label="y3")
1164
              ax.legend()
1165
              plt.show()
1166
1167
           -다음 코드는 같은 X값을 가진 오브젝트가 겹쳐진다.
           -이것을 피하기 위해서는 X값을 막대의 가로 폭만큼 비켜서 그릴 필요가 있다.
1168
1169
           -다음 코드에서는 막대그래프의 가로 폭 W를 0.2로 설정하고 X값을 0.2씩 비켜서 그리고 있다.
1170
1171
              fig = plt.figure()
1172
              ax = fig.add\_subplot(111)
```

```
1174
                         w = 0.2
1175
                         ax.bar(x, y1, width=w, label="y1")
1176
                         ax.bar(np.array(x) + w, y2, width=w, label="y2")
1177
                         ax.bar(np.array(x) + w * 2, y3, width=w, label="y3")
1178
                         ax.legend()
1179
                         plt.show()
1180
1181
1182
                7)여러 그룹의 막대그래프 활용하기
1183
                     -실제 데이터를 이용해서 그래프를 그려보자.
1184
                     -데이터는 anime_genre_top10_pivoted.csv파일이다.
1185
                         base\_url = "https://raw.githubusercontent.com/practical-jupyter/sample-data/master/anime/" and the state of the state of
1186
1187
                         anime_genre_top10_pivoted_csv = os.path.join(base_url, "anime_genre_top10_pivoted.csv")
1188
                         dfag = pd.read_csv(anime_genre_top10_pivoted_csv, index_col="genre")
1189
                         dfaa
1190
1191
1192
                     -다음으로 불러온 데이터(dfag)를 시각화한다.
                     -X값을 0.1씩 증가시키면서 열별로 그리고 있다.
1193
1194
                     -이 결과에서도 TV 합계 멤버 수가 돌출되어 있고 다음에 Movie 멤버 수가 많은 것을 확인할 수 있다.
1195
1196
                         fig = plt.figure(figsize=(18, 3))
1197
                         ax = fig.add_subplot(111)
1198
                         wt = np.array(range(len(dfag)))
1199
                         w = 0.1
1200
1201
                         for i in dfag.columns:
1202
                               ax.bar(wt, dfag[i], width=w, label=i)
1203
                               wt = wt + w
1204
1205
                         ax.set_xticks(np.array(range(len(dfag) + 2)))
1206
                         ax.set_xticklabels(dfag.index, ha="left")
1207
                         ax.set_ylabel("누적 멤버수")
1208
                         ax.legend()
1209
                         plt.show()
1210
1211
                     -결과에서 보듯이, Music이나 ONA 값이 상대적으로 작기 때문에 눈으로 확인하는 것이 어렵다.
1212
                     -이럴 때는 로그 축을 이용하면 가독성이 좋아진다.
1213
                     -Y축을 로그축에 설정하는 경우에는 set_yscale()에 log를 지정한다.
1214
                     -다음 코드는 작은 값의 그룹도 눈으로 확인할 수 있게 되었다.
1215
1216
                         fig = plt.figure(figsize=(18, 3))
1217
                         ax = fig.add\_subplot(111)
1218
1219
                         wt = np.array(range(len(dfag)))
1220
                         w = 0.1
1221
                         for \ i \ in \ dfag.columns:
1222
1223
                                ax.bar(wt, dfag[i], width=w, label=i)
1224
                               wt = wt + w
1225
                         ax.set_xticks(np.array(range(len(dfag) + 2)))
1226
1227
                         ax.set_xticklabels(dfag.index, ha="left")
1228
                         ax.set_ylabel("누적 멤버수")
                         ax.set_yscale("log")
1229
1230
                         ax.legend()
1231
                         plt.show()
1232
1233
1234
                8)누적 막대그래프 작성하기
1235
                     -누적 막대그래프를 그릴 때에도 여러 그룹의 막대그래프와 같이 작성시 요령이 필요하다.
1236
                     -다음 코드는 y1, y2, y3의 3개의 값을 누적한 경우의 그리기 순서이다.
1237
                         a. y1과 y2와 y3의 합을 그린다.
1238
                         b. a에 y2와 y3의 합을 겹쳐서 그린다.
1239
                         c. b에 y1을 겹쳐서 그린다.
1240
1241
                     -다시 말하면, 같은 X값을 부여해서 그리면 뒤에 그린 막대에 겹쳐지기 때문에 수동으로 값의 합계를 내서 합계가 많은 쪽부터 순서대로 그리는 작업을 한다.
1242
1243
                         x = np.arange(5)
1244
                         np.random.seed(0)
1245
                         y = np.random.rand(15).reshape((3, 5))
1246
                         y1, y2, y3 = y
1247
1248
                         y1b = np.array(y1)
1249
                         y2b = y1b + np.array(y2)
                         y3b = y2b + np.array(y3)
1250
1251
1252
                         # 누적 막대 그래프 그리기
                         fig = plt.figure(figsize=(10, 3))
1253
1254
                         ax = fig.add_subplot(111)
                         ax.bar(x, y3b, label="y3")
1255
1256
                         ax.bar(x, y2b, label="y2")
```

```
ax.bar(x, y1b, label="y1")
1258
             ax.legend()
1259
             plt.show()
1260
1261
1262
        9)bottom 옵션으로 누적 설정하기
1263
           -누적 막대그래프 작성시 옵션으로 bottom 옵션이 있다.
1264
           -하단에 오는 list형.오브젝트를 인수 bottom에 설정하는 것에 의해 누적 표시가 이루어진다.
1265
           -2개 그룹의 누적까지는 bottom 옵션이 유효하지만, 그 이상을 누적할 때에는 위의 방법으로 해야 한다.
1266
1267
             figure = plt.figure(figsize=(10, 3))
1268
             ax = figure.add_subplot(111)
1269
             ax.bar(x, y3, bottom=y2b, label="y3")
             ax.bar(x, y2, bottom=y1, label="y2")
1270
1271
             ax.bar(x, y1, label="y1")
1272
             ax.legend()
1273
             plt.show()
1274
1275
1276
        10)누적 막대그래프 활용하기
           -데이터는 앞에서 이용한 anime_genre_top10_pivoted.csv파일이다.
1277
1278
           -데이터는 dfag에 DataFrame으로 저장되어 있다.
1279
1280
             fig = plt.figure(figsize=(15, 3))
1281
             ax = fig.add_subplot(111)
1282
             rows, cols = len(dfag), len(dfag.columns)
1283
             x = range(rows)
1284
             for i, t in enumerate(dfag.columns):
1285
                 # i열부터 마지막까지 합을 계산
1286
                 y = dfag.iloc[:, i:cols].sum(axis=1)
1287
                 ax.bar(x, y, label=t)
1288
             ax.set_xticks(range(rows + 2))
1289
             ax.set_xticklabels(dfag.index)
1290
             ax.set_ylabel("누적 멤버수")
1291
             ax.legend()
1292
             plt.show()
1293
1294
1295
1296
      14. 히스토그램
1297
        1)히스토그램은 세로축에 회수(값의 출현빈도), 가로축에 계급(값의 상한값 ~ 하한값)을 취급하는 그래프로, 데이터의 분포 형상을 시각적으로 인식하기 위해 이용한다.
1298
        2)데이터의 분포 현상(분포형)은 통계학적으로 매우 중요한 의미를 가지고 있다.
1299
        3)히스토그램 작성하기
1300
           -Axes.hist()를 사용해서 작성한다.
1301
           -이 함수에 넘기는 데이터는 list형 오브젝트를 시용할 수 있다.
1302
           -다음 코드는 평균값 100, 표준편자 10의 정규분포에 따라 만 개의 데이터 히스토그램을 그린다.
1303
1304
             plt.style.use("ggplot")
1305
1306
             #데이터 세트 작성
1307
             mu = 100 # 평균값
1308
             sigma = 10 # 표준편차
1309
             np.random.seed(0)
1310
             x = np.random.normal(mu, sigma, 10000)
1311
1312
             #히스토그램 그리기
1313
             fig = plt.figure()
1314
             ax = fig.add\_subplot(111)
1315
             ax.hist(x)
1316
             plt.show()
1317
1318
1319
        4)막대의 폭과 수를 변경하는 경우
1320
           -hist()에는 대이터 외에 히스토그램 그림에 관한 인수를 부여할 수 있다.
1321
           -rwidth로 막대의 폭을, bins로 막대의 갯수를 지정할 수 있다.
1322
1323
             fig = plt.figure()
1324
             ax = fig.add_subplot(111)
1325
             ax.hist(x, rwidth=0.9, bins=16)
1326
             plt.show()
1327
1328
1329
        5)히스토그램 활용하기
1330
           -실제 데이터를 이용해서 그래프를 그려보자.
1331
           -anime_master.csv 파일을 이용한다.
1332
1333
             import os
1334
             import pandas as pd
1335
1336
             base_url = "https://raw.githubusercontent.com/practical-jupyter/sample-data/master/anime/"
             anime_master_csv = os.path.join(base_url, "anime_master.csv")
1337
1338
             df = pd.read_csv(anime_master_csv, index_col="anime_id")
1339
             df.head()
1340
                         -----
```

```
1342
           -평점 분포에 대해 matplotib으로 시각화를 실행해 보자.
1343
           -pandas의 Series를 hist()의 인수에 넘겨서 출력한다.
1344
           -평점이 0\sim 10의 범위에서 실행되고 있기 때문에 값의 범위를 range 0\sim 10으로 지정한다.
1345
1346
              fig = plt.figure()
1347
              ax = fig.add\_subplot(111)
1348
              ax.hist(df["rating"], range=(0, 10), rwidth=0.9)
1349
              ax.set_title("Rating")
1350
              plt.show()
1351
1352
           -에피소드 수도 히스토그램으로 그려본다.
1353
           -다음 코드를 실행하면 왼쪽으로 크게 치우쳐진 히스토그램이 된다.
1354
1355
              fig = plt.figure()
1356
              ax = fig.add\_subplot(111)
1357
              df_tv = df[df["type"] == "TV"]
1358
              ax.hist(df tv["episodes"], rwidth=0.9)
1359
              ax.set_title("Episodes")
1360
              plt.show()
1361
1362
           -그래서 이번에는 히스토그램의 범위를 지정해 보자.
1363
1364
              fig = plt.figure()
1365
              ax = fig.add_subplot(111)
1366
1367
              # range의 값을 (0, 100)으로 지정.
              ax.hist(df_tv["episodes"], rwidth=0.9, range=(0, 100))
1368
1369
              ax.set_title("Episodes(0-100)")
1370
              plt.show()
1371
1372
1373
1374
      15. 다양한 히스토그램 작성하기
1375
         \mathbf{1})수평 히스토그램
1376
           -인수 orientation에 horizontal(초기 설정은 vertical)을 지정하면 된다.
1377
1378
              np.random.seed(0)
1379
              x = np.random.normal(100, 10, 10000)
              fig = plt.figure()
1380
1381
              ax = fig.add\_subplot(111)
1382
1383
              # orientation을 horizontal에 지정
1384
              ax.hist(x, rwidth=0.9, bins=16, orientation="horizontal")
1385
              plt.show()
1386
1387
1388
         2)상대도수 히스토그램
1389
           -데이터 수가 다른 그룹의 히스토그램을 비교하는 경우에는 상대도수를 이용해서 히스토그램화하면 비교가 용이하다.
1390
           -상대도수 히스토그램을 그리는 경우에는 인수 normed에 True를 지정한다.
1391
           -상대도수 히스토그램에서는 상대도수의 합계가 1이 된다.
1392
1393
              fig = plt.figure()
1394
              ax = fig.add\_subplot(111)
1395
1396
              # normed을 True로 지정
1397
              ax.hist(df["rating"], normed=True, rwidth=0.9)
1398
              plt.show()
1399
1400
1401
         3)누적 히스토그램(누적도수 그림)
1402
           -누적도수를 확인하는 경우에는 누적 히스토그램을 이용한다.
1403
           -누적 히스토그램을 그리는 경우 인수 cumulative에 True를 지정한다.
1404
1405
              fig = plt.figure()
1406
              ax = fig.add_subplot(111)
1407
1408
              # cumulative를 True로 지정
1409
              ax.hist(df["rating"], normed=True, cumulative=True, rwidth=0.9)
1410
              plt.show()
1411
1412
1413
         4)계급 폭 지정
1414
           -bins 옵션에 list형 수열을 부여하는 것에 따라 계급 폭을 지정할 수 있다.
1415
           -계급 폭은 같은 간격이 아니어도 상관없다.
1416
1417
              fig = plt.figure()
1418
              ax = fig.add subplot(111)
1419
              ax.hist(df["rating"], bins=[2, 4, 5.5, 6.5, 7, 7.5, 8.5, 10], rwidth=0.9)
1420
              plt.show()
1421
1422
1423
         5)근사 곡선 추가
1424
           -근사 곡선은 히스토그램을 그린 후에 꺾은선 그래프로 그린다.
```

```
-근사 곡선은 다음 단계로 그린다.
1426
              a. df['rating'] 데이터 세트의 평균값과 표준편차 구하기
1427
              b. numpy.linespace로 각 막대 단락 값(막대의 상한값과 하한값) 구하기
1428
              C. 구해진 평균값, 표준편차, 단락값으로부터 정규분포의 확률밀도함수에 따라 Y값 산출하기
1429
              d. 구해진 X값과 Y값으로 근사 곡선 그리기
1430
1431
              bins = 50 # 막대수
1432
              dfmin = np.min(df["rating"]) # 데이터 최소값
              dfmax = np.max(df["rating"]) # 데이터 최대값
1433
1434
1435
              #히스토그램 그리기
1436
             fig = plt.figure()
1437
              ax = fig.add\_subplot(111)
1438
              ax.hist(df["rating"], bins=bins, range=(dfmin, dfmax), normed=True, rwidth=0.9)
1439
1440
              # 평균과 표준편차
             mu, sigma = df["rating"].mean(), df["rating"].std()
1441
1442
1443
              #X값
1444
              x = np.linspace(dfmin, dfmax, bins) # 막대 단락 값
1445
1446
              \#근사적 확률밀도함수를 사용해 Y값 생성
             y = 1 / (sigma * np.sqrt(2 * np.pi)) * np.exp(-(x - mu) ** 2 / (2 * sigma ** 2))
1447
1448
1449
              # 근사 곡선 그리기
1450
              ax.plot(x, y)
1451
              plt.show()
1452
1453
1454
         6)여러 그룹을 겹쳐서 그리기
1455
           -같은 subplot에 histgram을 반복해서 그리면 여러 그룹의 histgram을 겹쳐서 그리는 것이 가능하다.
1456
           -평균이 0부터 10의 범위에서 실행되기 때문에 range()함수를 사용해서 b_num에 0.5씩 0부터 10 사이의 수치를 저장하고 있다.
1457
           -또한 히스토그램이 겹쳐져 그려지기 때문에 alpha 옵션으로 불투명도를 낮추고 있다.
1458
           -투명도는 0인경우 완전 투명, 1이면 완전히 불투명이 된다.
1459
              types = df["type"].unique()
1460
1461
              labels = types.tolist()
1462
             fig = plt.figure(figsize=(8, 6))
1463
              ax = fig.add_subplot(111)
1464
             b_num = np.arange(0, 10.5, 0.5)
1465
1466
              for t in types:
                 ax.hist(df.loc[df["type"] == t, "rating"], bins=b_num, rwidth=0.9, alpha=0.5, label=t)
1467
1468
1469
              ax.legend()
1470
              ax.set_xlabel("rating")
1471
              ax.set_ylabel("Count(rating)")
1472
              plt.show()
1473
1474
1475
         7)여러 그룹을 나열하여 그리기
1476
           -여러 그룹의 히스토그램을 겹쳐 그려서 시인성이 떨어지는 경우에는 그룹을 나열하는 방법이 있다.
1477
           -중첩 list를 작성한 후 그리면 여러 그룹을 옆으로 나열한 히스토그램을 그릴 수 있다.
1478
1479
              dataset = [df.loc[df["type"] == t, "rating"] for t in types]
1480
              fig = plt.figure(figsize=(8, 6))
              ax = fig.add\_subplot(111)
1481
1482
              ax.hist(dataset, bins=np.arange(0, 10.5, 0.5), rwidth=0.9, alpha=0.8, label=labels)
1483
              ax.legend()
1484
              ax.set_xlabel("rating")
1485
              ax.set_ylabel("Count(rating)")
1486
             plt.show()
1487
1488
1489
         8)여러 그룹을 누적해서 그리기
           -여러 그룹의 히스토그램을 그려서 전체의 분포와 그 내역을 확인하는 경우에는 누적 히스토그램이 유효하다.
1490
1491
           -여러 그룹을 나열해서 그린 방법과 같이 데이터 세트를 작성한 후 인수 stacked에 True를 지정해서 그리면 누적 히스토그램이 된다.
1492
1493
              fig = plt.figure(figsize=(8, 6))
1494
              ax = fig.add_subplot(111)
1495
              ax.hist(dataset,
1496
                    bins=np.arange(0, 10.5, 0.5),
1497
                    rwidth=0.9,
1498
                    alpha=0.7
1499
                    label=labels,
1500
                    stacked=True)
1501
              ax.legend()
              ax.set_xlabel("rating")
1502
              ax.set_ylabel("Count(rating)")
1503
1504
              plt.show()
1505
1506
1507
```

```
1509
         1)상자수염 그래프는 데이터의 불균형을 알기 쉽게 표현하는 그래프이다.
1510
         2)상자수엽 그래프 요소
1511
           -제3사분위점:모든 데이터의 하위부터 3/4로 나눈 값(=Q3), 상자의 상부 끝
1512
           -중앙값 : 모든 데이터의 하위부터 1/2로 나눈 값(=M2사분위점)
           -제1사분위점 : 모든 데이터의 하위부터 1/4로 나누 값(=Q1), 상자의 하부 끝
1513
1514
           -수염 상부 끝 : Q3 + 1.5 x IQR
           -수염 하부 끝 : Q1-1.5 x IQR
1515
1516
           -IQR: 사분위 범위(=Q3Q1)
1517
           -벗어난 값 : 수염의 하부 끝 ~ 상부 끝의 범위 외에 있는 데이터
1518
1519
         3)상자수엽 그래프 작성하기
1520
           -Axes.boxplot()를 사용해서 그린다.
1521
1522
              plt.style.use("ggplot")
              x = [1, 2, 3, 3, 11, 20]
1523
1524
              fig = plt.figure()
1525
             ax = fig.add\_subplot(111)
1526
              ax.boxplot(x)
1527
              plt.show()
1528
1529
1530
         4)여러 개의 상자수염 그래프를 그리는 경우
1531
           -복수의 list를 부여하면 여러 개의 상자수염 그래프를 그릴 수 있다.
1532
             x = [[1, 2, 3, 3, 11, 20], [1, 2, 9, 10, 15, 16]]
labels = ["A", "B"]
1533
1534
1535
              fig = plt.figure()
1536
              ax = fig.add\_subplot(111)
1537
1538
              #데이터와 레이블 지정
1539
              ax.boxplot(x, labels=labels)
1540
              plt.show()
1541
1542
1543
         5)상자수염 그래프 활용하기
1544
           -실제 데이터를 이용해서 그린다.
1545
           -데이터는 히스토그램에서 사용했던 anime_master.csv 파일을 이용한다.
1546
1547
              import os
1548
              import pandas as pd
1549
1550
              base url = "https://raw.githubusercontent.com/practical-jupyter/sample-data/master/anime/"
              anime_master_csv = os.path.join(base_url, "anime_master.csv")
1551
1552
              df = pd.read_csv(anime_master_csv, index_col="anime_id")
1553
             df.head(3)
1554
1555
1556
           -배급 종별마다 에피소드 수의 상자수염 그래프를 작성한다.
1557
           -배급 종별은 6종류이기 때문에 6개의 상자수염 그래프가 출력된다.
1558
1559
              labels = []
1560
              types_list = []
1561
1562
             #배급 종별마다 에피소드 수 정보를 list화 for label, df_per_type in df.groupby("type"):
1563
1564
                 labels.append(label)
1565
                 types_list.append(df_per_type["episodes"].tolist())
1566
1567
              fig = plt.figure()
1568
              ax = fig.add_subplot(111)
1569
              ax.boxplot(types_list, labels=labels)
1570
             plt.show()
1571
1572
           -위 코드의 결과를 보면, 텔레비젼 애니메이션을 의미하는 \mathsf{TV}에만 큰 값이 포함되어 있는 것을 확인할 수 있다.
1573
           -그래서 상자수염 그래프 그리기 범위를 지정하면 에피소드수 0부터 100까지의 값에 한정된다.
1574
1575
              fig = plt.figure(figsize=(8, 6))
1576
              ax = fig.add_subplot(111)
1577
              ax.boxplot(types_list, labels=labels)
1578
1579
              # Y축 그리기 범위를 0부터 100까지 한정
1580
              ax.set_ylim(0, 100)
1581
              plt.show()
1582
1583
1584
         6)상자수염 그래프의 서식 일괄 설정하기
1585
           -상자수염 그래프의 서식은 각 요소의 서식을 사전 형식으로 부여하여 일괄로 설정할 수 있다.
1586
           -요소에 따라 설정 가능한 항목이 다르지만 상자 부분은 patches.PathPatch 클래스로, 그 이외의 요소는 lines.Line2D 클래스의 인스턴스로 그려진다.
1587
           -상자수염 그림의 주요 서식 설정 항목
1588
              a. color : 색
1589
              b. facecolor : 채움색
1590
              c. linestyle: 선 종류
             d. linewidth : 선 굵기
1591
1592
              e. maker : 마커
```

```
1593
              f. makerfacecolor: 마커 채움색
              g. makeredgecolor : 마커 테두리선 색
1594
1595
              h. makersize : 마커 크기
1596
1597
           -서식 일괄 순서
1598
              a. 데이터 세트 작성하기
1599
1600
                 import numpy as np
1601
1602
                 np.random.seed(3)
1603
                 dataset = [np.random.normal(20 + mu, 5, 1000) for mu in range(1, 5)]
1604
1605
              b. 서식 사전 만들기
1606
                 -상자에 서식을 설정하기 위해 사전을 만든다.
1607
                 -상자의 요소, '벗어난 값', '상자', '수염', '수염끝단', '중앙값', '평균값'의 서식을 설정할 수 있다.
1608
1609
                    #벗어난 값의 서식 사전
                    flierprop = {"color": "#EC407A",
"marker": "o",
1610
1611
                               "markerfacecolor": "#2196F3",
1612
                               "markeredgecolor": "white",
1613
                               "markersize": 5,
1614
                               "linestyle": "None",
1615
                               "linewidth": 0.1}
1616
1617
1618
                    # 상자의 서식 사전
                    boxprop = {"color": "#2196F3"
1619
                             "facecolor": "#BBDEFB",
1620
                             "linewidth": 1,
1621
1622
                             "linestyle": "-"}
1623
1624
                    # 수염의 서식 사전
                    whiskerprop = {"color": "#2196F3",
"linewidth": 1,
"linestyle": "--"}
1625
1626
1627
1628
1629
                    # 수염 끝단 서식 사전
                    capprop = {"color": "#2196F3",
1630
                             "linewidth": 1,
1631
1632
                             "linestyle": ":"}
1633
1634
                    # 중앙값 서식 사전
                    medianprop = {"color": "#2196F3",
"linewidth": 2,
1635
1636
                                "linestyle": "-"}
1637
1638
1639
                    # 평균값 서식 사전
                    meanprop = {"color": "#2196F3",
"marker": "^",
1640
1641
                               "markerfacecolor": "#2196F3",
"markeredgecolor": "white",
1642
1643
1644
                               "markersize": 10,
                               "linewidth": 1,
1645
1646
                               "linestyle": ""}
1647
1648
                 C. 그리기
1649
1650
                    fig = plt.figure()
1651
                    ax = fig.add subplot(111)
1652
                    ax.boxplot(
1653
                       dataset,
                       patch_artist="Patch", # 서식을 설정하는 경우「Patch」 를 선택 labels=["A", "B", "C", "D"], # 항목 레이블
1654
1655
                       showmeans=True, # 평균값 그리기
1656
1657
                       flierprops=flierprop, # 벗어난 값 서식 설정
                       boxprops=boxprop, # 상자 서식 설정
1658
                       whiskerprops=whiskerprop, # 수염 서식 설정
1659
1660
                       capprops=capprop, # 수염 끝단 서식 설정
1661
                       medianprops=medianprop, # 중앙값 서식 설정
1662
                       meanprops=meanprop, # 평균값 서식 설정
1663
1664
                    plt.show()
1665
1666
1667
         7)상자마다 서식 설정하기
1668
            -서식을 개별로 설정하는 것도 가능하다.
1669
            -상자의 서식을 요소마다 설정하는 경우에는 각 항목에 접두사 Set을 붙여서 이용한다.
1670
            -다음 순서로 서식을 설정해서 그린다.
1671
1672
              a. 그림 그리기
1673
              b. 상자 요소 수와 같은 요소 수의 색 세트(컬러 세트)(colors1과 colors2)를 작성하기
1674
              C. 위쪽과 아래쪽이 나눠져 있는 요소의 서식 설정용에 수열 list n을 작성하기
1675
              d. 상자와 벗어난 값, 중앙값(a소가 상하로 나워져 있지 않은 것 또한 상하 같은 색을 부여한 것)의 서식 설정하기
1676
              e. 수염과 수염의 끝단(요소가 상하로 나워져 있는 것 상하 다른 색을 설정할 수 있는 것)의 서식 설정하기
```

```
1678
1679
                 # 그림 그리기
1680
                 fig = plt.figure()
1681
                 ax = fig.add\_subplot(111)
1682
1683
                 bp = ax.boxplot(
1684
                     dataset,
1685
                     patch artist="Patch",
                     labels=["A", "B", "C", "D"],
1686
1687
                     meanline=True,
1688
                    showmeans=True,
1689
                 )
1690
1691
                 # 컬러 세트
                 colors1 = ["#2196F3", "#43A047", "#FBC02D", "#FB8C00"]
colors2 = ["#BBDEFB", "#C8E6C9", "#FFF9C4", "#FFE0B2"]
1692
1693
1694
1695
                 # 위 아래오 나위어진 요소에 설정하기 위해 용도의 수열
1696
                 n = [0, 0, 1, 1, 2, 2, 3, 3]
1697
1698
                 # 서식 설정
                 # 상자와 벗어난 값, 중앙값의 서식 설정 for params in zip(bp["boxes"], bp["fliers"], bp["medians"], colors1, colors2):
1699
1700
1701
                     bpb, bpf, med, color1, color2 = params
1702
1703
                     # 상자 서식 설정
1704
                     bpb.set color(color1)
1705
                     bpb.set_facecolor(color2)
1706
                     bpb.set_linewidth(2)
1707
                     # 벗어난 값 서식 설정
1708
1709
                     bpf.set(marker="^", color=color2)
1710
                     bpf.set_markeredgecolor("white")
1711
                     bpf.set_markerfacecolor(color1)
1712
1713
                     # 중앙값 서식 설정
1714
                    med.set_color(color1)
1715
                     med.set_linewidth(2)
1716
1717
                 #수염과 수염 끝단 서식 설정
1718
                 for bpc, bpw, m in zip(bp["caps"], bp["whiskers"], n):
                     bpc.set_color(colors1[m])
1719
1720
                     bpc.set_linewidth(2)
1721
                     bpw.set_color(colors1[m])
1722
                     bpw.set_linewidth(2)
1723
1724
                 # 평균값 서식 설정
1725
                 for mean, color2 in zip(bp["means"], colors2):
                     mean.set_color("grey")
1726
1727
                     mean.set_linewidth(2)
1728
                    mean.set_linestyle("--'")
1729
1730
                 plt.show()
1731
1732
1733
1734
      17. 원 그래프
1735
         1)원 그래프는 전체에 대한 각 요소의 비율을 추출하고, 추출한 비율에 따라 원형을 부채꼴로 분할한 그래프이다.
1736
         2)원 그래프는 각 요소의 비율을 비교할 때 유용하다.
1737
         3)원 그래프 그리기
1738
            -Axes.pie()를 사용한다.
1739
            -제\mathbf{1}인수에 요소의 값을 부여해서 그린다.
1740
1741
               plt.style.use("ggplot")
1742
              labels = ["자전거", "버스" ,"차"]
              sizes = [25, 40, 35]
1743
1744
              fig = plt.figure(figsize=(3, 3))
1745
              ax = fig.add_subplot(111)
1746
1747
              ax.pie(sizes, labels=labels)
1748
              plt.show()
1749
1750
            -초기 설정에서는 도수법으로 0도의 위치(시계 세 시의 위치)에서 반시계 방향으로 요소를 그려간다.
1751
            -중심 좌표는 (0,0), 반경은 1이다.
1752
            -다음 코드에서는 radius에 0.9를, 인수 frame에 True를 설정해서 축과 함께 그래프를 그린다.
1753
              fig = plt.figure(figsize=(3, 3))
1754
1755
              ax = fig.add\_subplot(111)
1756
              ax.pie(sizes, labels=labels, radius=0.9, frame=True)
1757
               ax.text(-0.3, 0, "(0, 0)", fontsize=9)
1758
1759
1760
            -위의 코드에서 원의 중심 좌표는 (0,0), 처음 요소 자전거가 좌표(0.9,0)에서 시작해서 부채꼴로 그려져 있는 것을 확인할 수 있다.
```

f. 평균값의 서식 설정하기

```
1762
1763
1764
        4)원 그래프 서식 설정
1765
           -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.
1766
           -labels : 레이블 표시. list형 또는 tuple형으로 지정.
1767
           -colors : 각 요소의 색을 설정. list형 또는 tuple형으로 지정.
1768
           -autopct : 수치 레이블 서식 설정. 표시 형식은문자열로 지정.
1769
           -pctdistance : 수치 레이블의 위치 지정. 수치열로. 수치는 각 요소의 중심부터의 거리가 되고 explode를 설정하고 있는 경우에도 이 거리도 가산됨.
1770
           -shadow : 배경의 표시/비표시 설정. 논리값
1771
           -labeldistance : 레이블의 위치 설정. 수치열로. 수치는 각 요소의 중심부터의 거리가 되고 explode를 설정하고 있는 경우에도 이 거리도 가산됨.
1772
           -startangle : 시작 각도 설정. 단위는 도수법으로 수치형으로
1773
           -radius : 반경을 설정. 수치열(초기 설정 1)
1774
           -counterclock : 표시순서 설정. 논리값. True(반시계방향), False(시계방향)
1775
           -wedgeprops : 각 요소의 서식 설정. 서식을 등록한 dict.
1776
           -textprops : 텍스트의 서식 설정. 서식을 등록한 dict.
1777
           -center: 워 그래프의 중심 좌표 설정. tuple형
1778
           -frame : 축/테두리선의 유무 설정. 논리값.
1779
1780
1781
        5)원 그래프의 서식 설정을 하는 경우
1782
1783
           fig = plt.figure(figsize=(3, 3))
1784
           ax = fig.add_subplot(111)
1785
1786
           #부채꼴 서식 설정용 사전
1787
           wprops = {"edgecolor": "black", "linewidth": 2}
1788
1789
           # 텍스트 서식 설정용 사전
           tprops = {"fontsize": 18}
1790
           ax.pie(
1791
1792
              sizes
1793
              explode=(0.0, 0.05, 0),
1794
              labels=labels,
              autopct="%1.0f%%",
1795
1796
              pctdistance=0.5,
1797
              shadow=False,
1798
              labeldistance=1.35,
1799
              startangle=90,
1800
              radius=0.3,
1801
              counterclock=False,
1802
              wedgeprops=wprops,
1803
              textprops=tprops,
              center=(0.5, 0.5),
1804
1805
              frame=True,
1806
1807
           plt.show()
1808
1809
1810
        6)원 그래프 활용하기
1811
           -실제의 데이터를 이용해서 그려보자.
1812
           -데이터는 anime_genre_top10_pivoted.csv 파일이다.
1813
1814
             import os
1815
             import pandas as pd
1816
1817
             base url = "https://raw.githubusercontent.com/practical-jupyter/sample-data/master/anime/"
1818
             anime genre top10 pivoted csv = os.path.ioin(base url, "anime genre top10 pivoted.csv")
             df = pd.read_csv(anime_genre_top10_pivoted_csv, index_col="genre")
1819
1820
1821
1822
1823
           -원 그래프는 데이터의 비율을 비교할 때 유용한 그래프이다.
1824
           -여기서는 Movie와 TV의 총 멤버수 내역을 원그래프로 그려서 장르의 내역 비율을 비교한다.
1825
           -원 그래프는 90도 위치에서 시계 방향으로 내림차순으로 요소를 나열하는 것이 일반적이다.
1826
           -먼저 TV와 Movie의 데이터를 각각 내림차순으로 정렬한 Series를 작성한다.
1827
1828
             df_tv = df.sort_values(by="TV", ascending=False)["TV"]
1829
1830
             df_movie = df.sort_values(by="Movie", ascending=False)["Movie"]
1831
             df_tv
1832
1833
1834
           -멤버 수가 많은 Comedy에서 내림차순으로 데이터가 내열되어 있다.
1835
           -다음 코드는 소트한 Movie의 데이터를 사용해서 그래프를 그린다.
1836
1837
             fig = plt.figure(figsize=(9, 4))
             ax1 = fig.add\_subplot(121)
1838
1839
             ax2 = fig.add_subplot(122)
1840
1841
              #컬러 세트
1842
             colors1 = (
                 "gold",
1843
```

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

```
1844
                   "coral",
                   "plum"
1845
                   "orchid",
1846
1847
                   "lightseagreen",
                   "yellowgreen",
"lightskyblue",
1848
1849
1850
                   "pink",
1851
                   "cornflowerblue",
1852
                   "orangered",
1853
1854
               colors2 = (
1855
                   "coral",
                   "orangered",
1856
1857
                   "plum",
                   "pink",
"gold",
"cornflowerblue",
1858
1859
1860
1861
                   "yellowgreen",
1862
                   "lightseagreen",
                   "orchid",
1863
                   "lightskyblue",
1864
1865
               )
1866
1867
               # TV 원 그래프
1868
               ax1.pie(
1869
                   df_tv,
1870
                   explode=(0, 0, 0, 0, 0, 0, 0.15, 0, 0, 0.15),
1871
                   labels=df_tv.index,
                   autopct="%1.0f%%",
1872
1873
                   colors=colors1,
1874
                   startangle=90,
                   counterclock=False,
1875
1876
               )
1877
1878
               # Movie 원 그래프
1879
               ax2.pie(
1880
                   df_movie,
                   explode=(0, 0.15, 0, 0, 0, 0, 0, 0, 0, 0.15),
1881
                   labels=df_movie.index,
1882
1883
                   autopct="%1.0f%%",
1884
                   colors=colors2,
1885
                   startangle=90,
                   counterclock=False,
1886
1887
1888
               ax1.set_title("TV")
               ax2.set_title("Movie")
1889
1890
               plt.subplots_adjust(wspace=0.3) # 서브블록 사이의 공간 조정
1891
               plt.show()
```