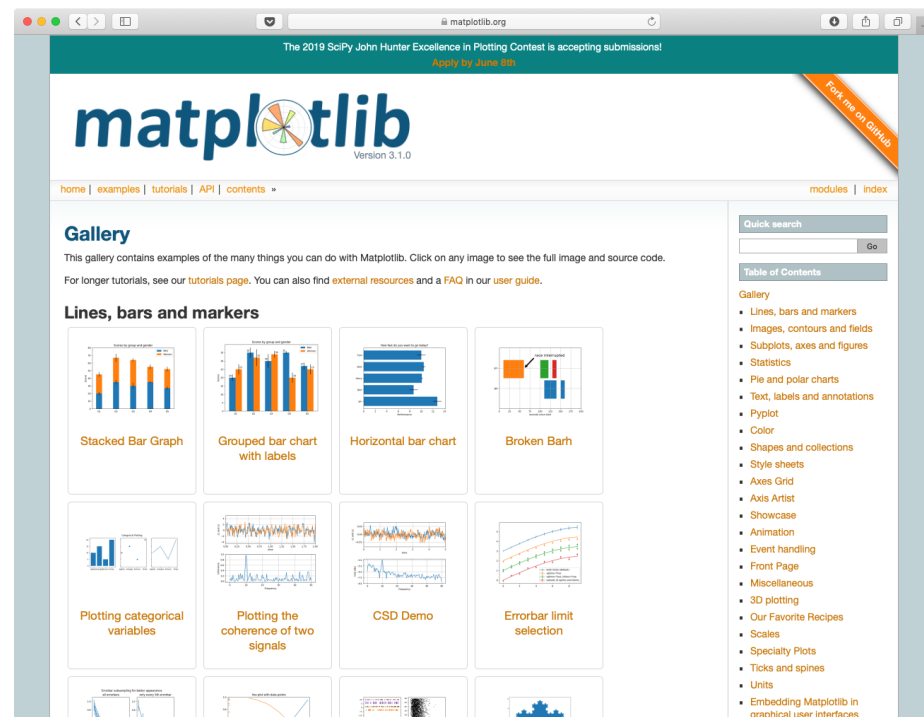
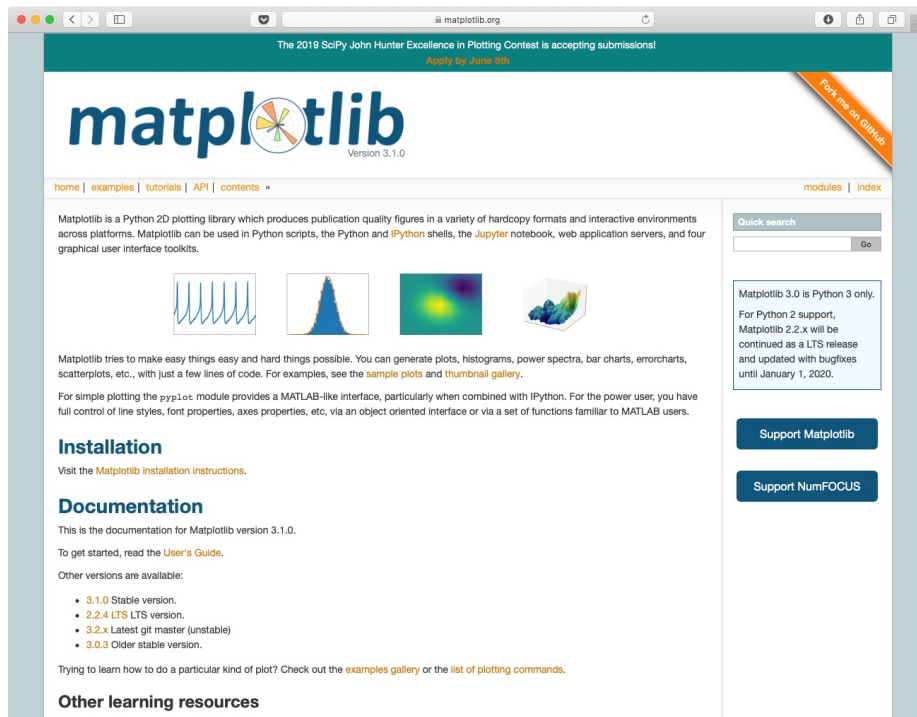


데이터 시각화: matplotlib 라이브러리

공공데이터

matplotlib 라이브러리

- 파이썬에서 2D 형태의 그래프, 이미지 등을 그릴 때 널리 사용
- matplotlib 홈페이지
 - <https://matplotlib.org>
- pyplot 모듈을 주로 사용



pyplot 모듈 사용

- pyplot
 - 공학용 도구로 널리 알려진 Matlab과 사용법이 유사함
- 사용 방법

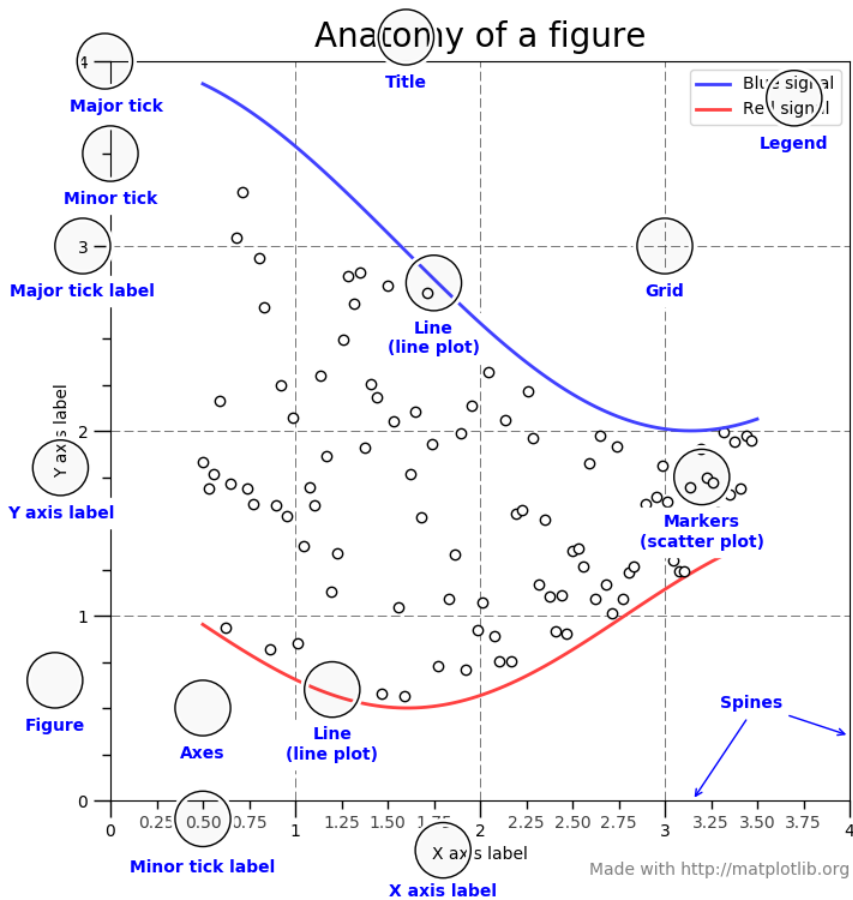
```
import matplotlib.pyplot
```

또는

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

- 기본 그래프 그리기
 - plot() 함수: 직선 또는 꺾은선 형태의 그래프를 그릴 때 사용함
 - plt.plot(): 데이터 입력 및 그래프 옵션 설정
 - plt.show(): 그래프 보여주기
 - plt.savefig(“파일이름”, dpi=200)
 - 그래프를 파일로 저장 및 그림 파일 해상도(dpi) 설정

그래프 구성



실제 사례:

Impact of CoAP and MQTT on NB-IoT System Performance

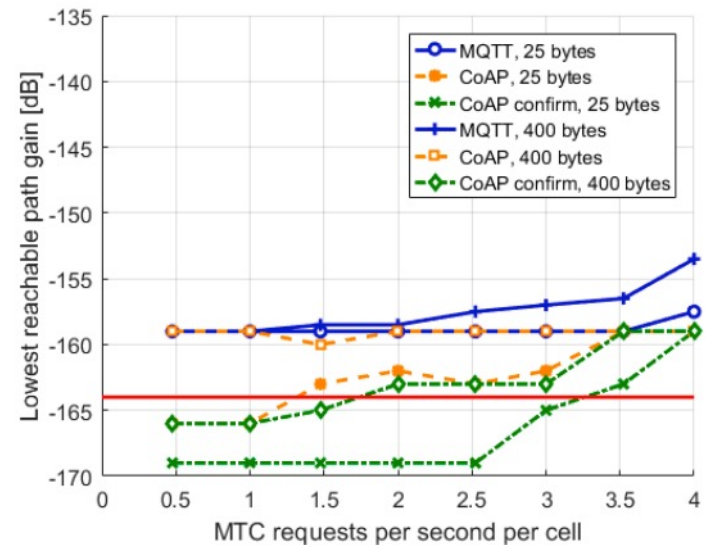


Figure 4. Lowest reachable path gain.

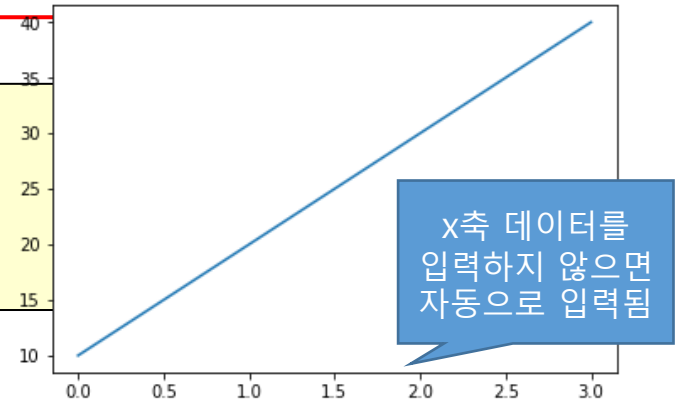
<https://pbpython.com/effective-matplotlib.html>

<https://www.ncbi.nlm.nih.gov/pmc/articles/PMC6338939/>

기본 그래프 그리기

```
import matplotlib.pyplot as plt

plt.plot([10, 20, 30, 40]) #y축의 값
plt.show()
```



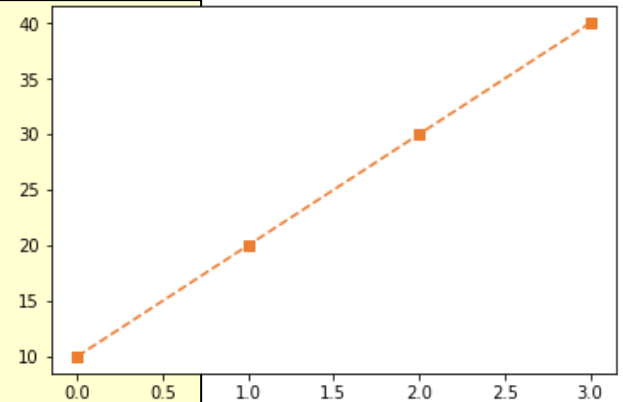
Marker 및 색상 변경

```
import matplotlib.pyplot as plt

COLOR_ORANGE = '#ED7D31'
COLOR_BLUE = '#5B9BD5'

SQUARE_MARKER = 's'           # Square
SOLID_LINE = '-'              # ':'
DASHED_LINE = '--'            # '--'

plt.plot([10, 20, 30, 40], color=COLOR_ORANGE,
         marker=SQUARE_MARKER, linestyle=DASHED_LINE,
         markersize = 6)
plt.show()
```



그래프 옵션

■ 색상 정보

- 6자리의 hexa값을 활용함
- https://www.rapidtables.com/web/color/RGB_Color.html 참고
- COLOR_RED = #FF0000
- COLOR_GREEN = #00FF00
- COLOR_BLUE = #0000FF
- 기본적인 색에 대해서는 이름으로 사용 가능함 ('red', 'green', 'blue', 'black', 'yellow' 등)

■ 선 종류

선 종류	값
dotted line	'.'
dash dot line	'-.'
dashed line	'--'
solid line	'_'

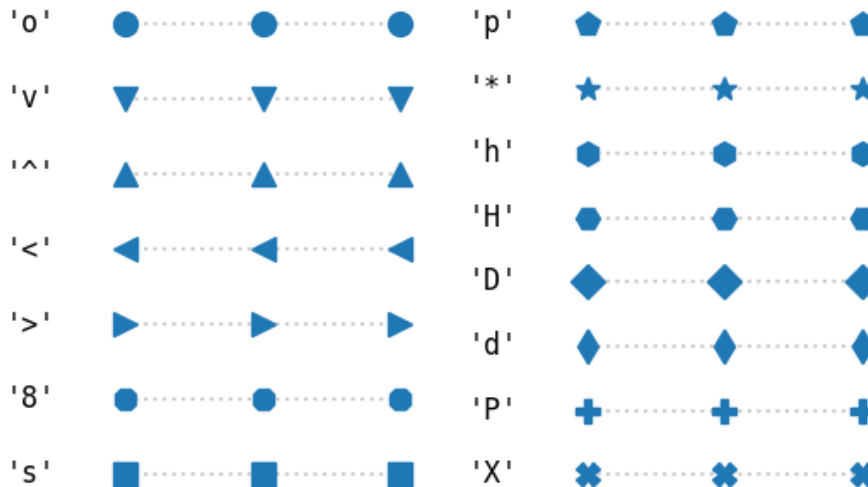
line styles



그래프 옵션 (Marker)

■ Marker 옵션

```
DIAMOND_MARKER = 'D'      # Big diamond  
CIRCLE_MARKER = 'o'       # circle  
TRIANGLE_MARKER = '^'     # triangle  
SQUARE_MARKER = 's'       # Square  
HEXA_MARKER = 'h'         # Hexagon  
CROSS_MARKER = 'x'        # X
```




Named Colors

Base Colors

 b	 c	 k
 g	 m	 w
 r	 y	

CSS Colors

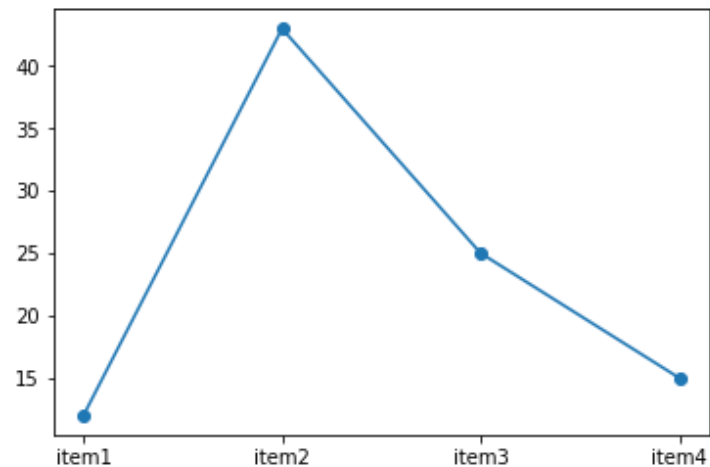
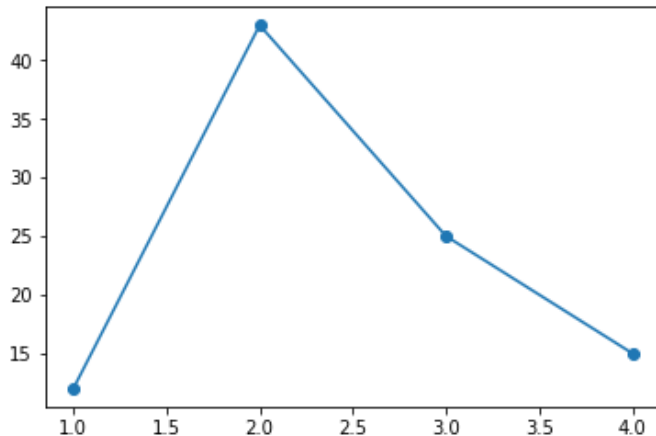
 black	 bisque	 forestgreen	 slategrey
 dimgray	 darkorange	 limegreen	 lightsteelblue
 dimgrey	 burlywood	 darkgreen	 cornflowerblue
 gray	 antiquewhite	 green	 royalblue
 grey	 tan	 lime	 ghostwhite
 darkgray	 navajowhite	 seagreen	 lavender
 darkgrey	 blanchedalmond	 mediumseagreen	 midnightblue
 silver	 papayawhip	 springgreen	 navy
 lightgray	 moccasin	 mintcream	 darkblue
 lightgrey	 orange	 mediumspringgreen	 mediumblue
 gainsboro	 wheat	 mediumaquamarine	 blue
 whitesmoke	 oldlace	 aquamarine	 slateblue
 white	 floralwhite	 turquoise	 darkslateblue
 snow	 darkgoldenrod	 lightseagreen	 mediumslateblue
 rosybrown	 goldenrod	 mediumturquoise	 mediumpurple
 lightcoral	 cornsilk	 azure	 rebeccapurple
 indianred	 gold	 lightcyan	 blueviolet
 brown	 lemonchiffon	 paleturquoise	 indigo
 firebrick	 khaki	 darkslategray	 darkorchid
 maroon	 palegoldenrod	 darkslategrey	 darkviolet
 darkred	 darkkhaki	 teal	 mediumorchid
 red	 ivory	 darkcyan	 thistle
 mistyrose	 beige	 aqua	 plum
 salmon	 lightyellow	 cyan	 violet
 tomato	 lightgoldenrodyellow	 darkturquoise	 purple
 darksalmon	 olive	 cadetblue	 darkmagenta
 coral	 yellow	 powderblue	 fuchsia
 orangered	 olivedrab	 lightblue	 magenta
 lightsalmon	 yellowgreen	 deepskyblue	 orchid
 sienna	 darkolivegreen	 skyblue	 mediumvioletred
 seashell	 greenyellow	 lightskyblue	 deeppink
chocolate	chartreuse	steelblue	hotpink
saddlebrown	lawngreen	aliceblue	lavenderblush
sandybrown	honeydew	dodgerblue	palevioletred
peachpuff	darkseagreen	lightslategray	crimson
peru	palegreen	lightslategrey	pink
linen	lightgreen	slategray	lightpink

https://matplotlib.org/gallery/color/named_colors.html#sphx-glr-gallery-color-named-colors-py

그래프 그리기 (x축, y축 데이터)

- `plt.plot([x축 데이터], [y축 데이터])`
 - x축 데이터에 문자열 가능함
 - `['item1', 'item2', 'item3', 'item4']`

```
import matplotlib.pyplot as plt
CIRCLE_MARKER = 'o'
plt.plot([1, 2, 3, 4], [12, 43, 25, 15],
         marker=CIRCLE_MARKER, linestyle = '-')
plt.show()
```



그래프 옵션 추가

- 그래프에 제목 넣기:
 - `title('제목 문자열')`
- 그래프에 범례(legend) 넣기
 - 범례는 두 개 이상의 데이터를 표시할 때 각 그래프를 구분하기 위해 서로 다른 색상이나 marker를 사용하여 각 데이터를 구분함
 - 각 그래프에서 `label="범례 이름"`을 이용하여 데이터를 구분함
 - 레전드 위치 설정
 - `plt.legend(loc = 5)`

그래프에서 Legend 위치

2	9	1
6	10	5, 7
3	8	4

그래프 옵션: 범례(legend) 추가

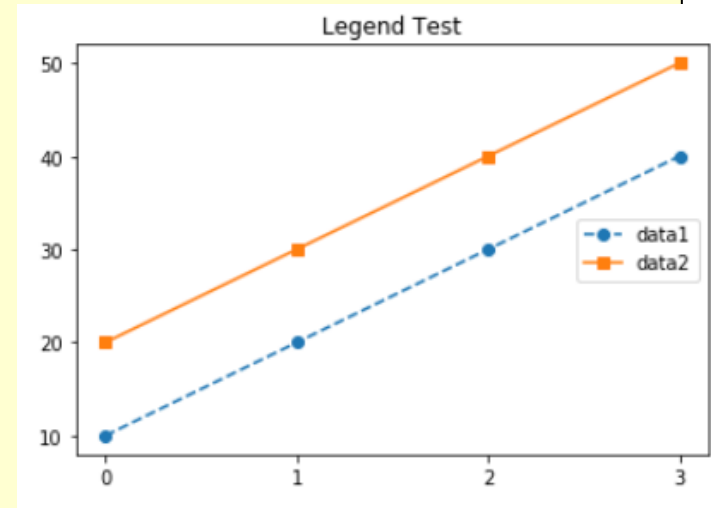
```
import matplotlib.pyplot as plt
CIRCLE_MARKER = 'o'
SQUARE_MARKER = 's'
DASHED_LINE = '--'

x_axis = [0, 1, 2, 3] # x축 tick
y_axis = [10, 20, 30, 40, 50] # y축 tick

data1 = [10, 20, 30, 40]
data2 = [20, 30, 40, 50]

plt.title('Legend Test')
plt.plot(x_axis, data1, marker = CIRCLE_MARKER, label='data1',
         linestyle=DASHED_LINE)
plt.plot(x_axis, data2, marker = SQUARE_MARKER, label='data2')

plt.legend(loc=5)
plt.xticks(x_axis)
plt.yticks(y_axis)
plt.show()
```



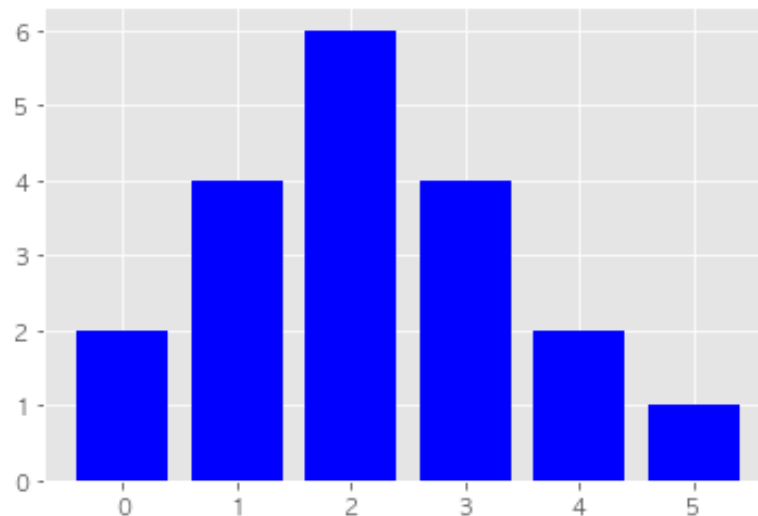
label:
legend에서
사용할 이름

막대 그래프 그리기: bar()

■ bar() 함수

- 막대그래프를 표현하는 명령어
- 막대그래프에서 막대의 길이는 각 데이터의 크기를 의미함
- `bar(레이블값, 막대의 높이(y축 값))`

```
import matplotlib.pyplot as plt
# bar([x축 데이터], [y축 데이터])
plt.bar([0, 1, 2, 3, 4, 5], [2, 4, 6, 4, 2, 1], color='b')
#plt.bar(range(6), [2, 4, 6, 4, 2, 1], color='b')
plt.show()
```

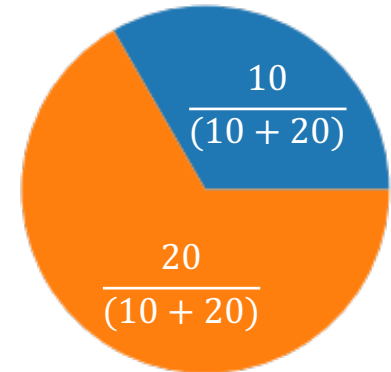


파이 차트로 표시

- pie() 함수
 - 전체 데이터에서 특정 데이터의 비율을 보기 쉽게 표현
 - 100명을 대상으로 혈액형을 조사하여 A형이 몇 명인지 그 비율을 파악
 - pie([x, y])
 - 각 데이터의 비율은 $x/(x+y)$, $y/(x+y)$ 의 형태로 표시됨

```
import matplotlib.pyplot as plt

plt.pie([10, 20])
plt.show()
```



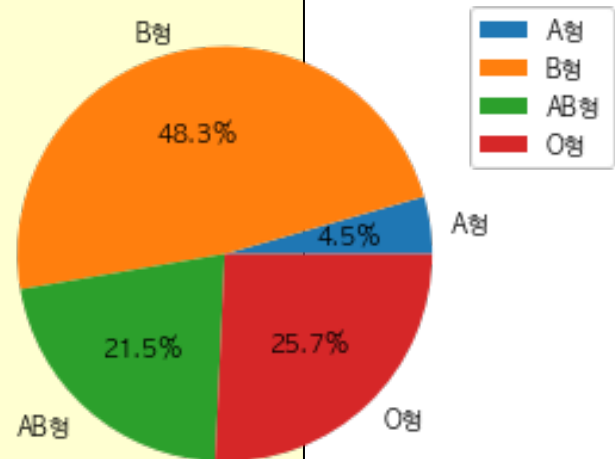
레이블, 비율 및 범례 표시

- label = ['A형', 'B형', 'AB형', 'O형']
 - 데이터의 개수만큼 레이블 항목 추가:
 - plt.pie()에서 labels 속성 사용
- 비율: autopct 속성(auto percentage)
 - 각 항목의 비율을 자동으로 계산해서 표시
 - 표시할 자리수 지정: plt.pie()에서 autopct='%.1f%%'
- 범례: legend()

```
import matplotlib.pyplot as plt
import platform

if platform.system() == 'Windows': # MacOS: 'Darwin'
    plt.rc('font', family='Malgun Gothic')
else:
    plt.rc('font', family='AppleGothic')

numbers = [214, 2312, 1031, 1233]
blood_type = ['A형', 'B형', 'AB형', 'O형']
plt.axis('equal') # 파이 차트를 원형으로 그려줌
plt.pie(numbers, labels=blood_type, autopct='%.1f%%')
plt.legend()
plt.show()
```



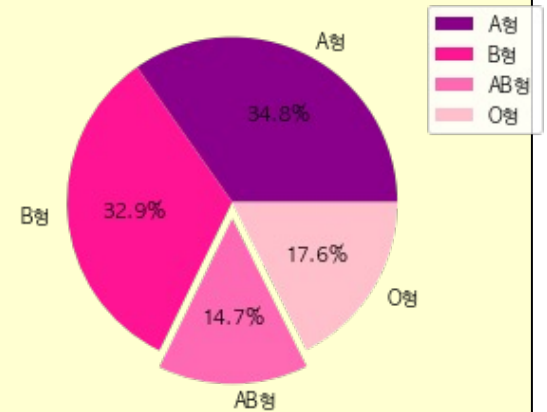
파이 차트 돌출 효과

- 색상 정보: colors 속성 사용
- 돌출 효과
 - pie() 메소드의 explode 속성으로 설정(0: 돌출되지 않음, 1: 돌출)
 - ex) `explode=(0, 0, 0.1, 0)`

```
import matplotlib.pyplot as plt
import platform

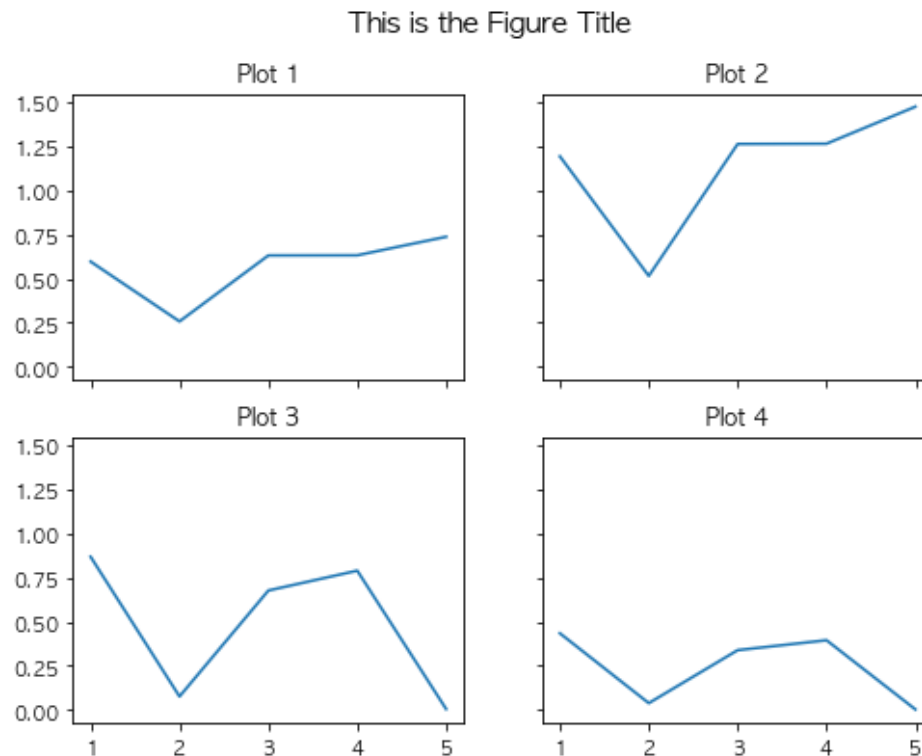
if platform.system() == 'Windows':
    plt.rc('font', family='Malgun Gothic')
else:
    plt.rc('font', family='AppleGothic')

color=['darkmagenta', 'deeppink', 'hotpink', 'pink']
size = [2441, 2312, 1031, 1233]
blood_type = ['A형', 'B형', 'AB형', 'O형']
plt.axis('equal')
plt.pie(size, labels=blood_type, autopct='%.1f%', colors = color,
        explode=(0, 0, 0.1, 0))
plt.legend()
plt.show()
```



여러 그래프를 한번에 그리기

- **subplots**(행의 수, 열의 수, figsize=(x, y))
 - 전체 subplot의 개수를 설정함
 - figsize: 각 subplot들의 크기
- **subplot**(행, 열, index)
 - index는 1부터 시작함




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

```
x = [1,2,3,4,5]
```

```
y1 = [0.59705847, 0.25786401, 0.63213726, 0.63287317, 0.73791151]
```

```
y2 = [1.19411694, 0.51572803, 1.26427451, 1.26574635, 1.47582302]
```

```
y3 = [0.86793828, 0.07563408, 0.67670068, 0.78932712, 0.0043694]
```

```
# 5 more random values
```

```
y4 = [0.43396914, 0.03781704, 0.33835034, 0.39466356, 0.0021847]
```

```
fig, axes = plt.subplots(2, 2, figsize=(8, 6), sharex=True, sharey=True)
```

```
fig.suptitle('This is the Figure Title', fontsize=15)
```

```
# Top Left Subplot
```

```
axes[0,0].plot(x, y1)
```

```
axes[0,0].set_title("Plot 1")
```

```
# Top Right Subplot
```

```
axes[0,1].plot(x, y2)
```

```
axes[0,1].set_title("Plot 2")
```

```
# Bottom Left Subplot
```

```
axes[1,0].plot(x, y3)
```

```
axes[1,0].set_title("Plot 3")
```

```
# Bottom Right Subplot
```

```
axes[1,1].plot(x, y4)
```

```
axes[1,1].set_title("Plot 4")
```

```
plt.show()
```

```
plt.subplot(2, 2, 1)
```

```
plt.plot(x, y1)
```

```
plt.title('Plot 1')
```

```
plt.subplot(2, 2, 2)
```

```
plt.plot(x, y2)
```

```
plt.title('Plot 2')
```

```
plt.subplot(2, 2, 3)
```

```
plt.plot(x, y3)
```

```
plt.title('Plot 3')
```

```
plt.subplot(2, 2, 4)
```

```
plt.plot(x, y4)
```

```
plt.title('Plot 4')
```

```
plt.show()
```

index는 1부터
시작

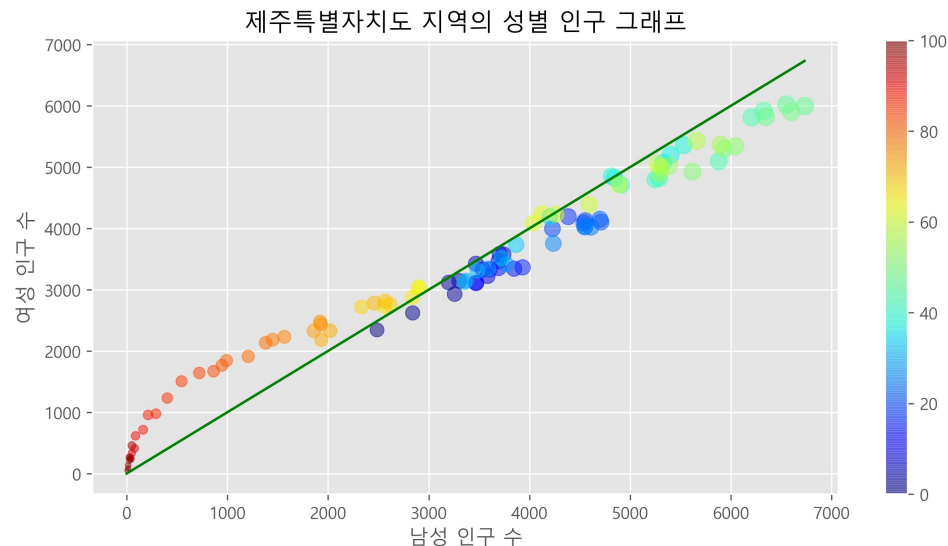
방법 #2

방법 #1

산점도(scatter)로 표현하기

■ 산점도

- 가로축과 세로축을 기준으로 두 요소가 서로 어떤 관계를 맺고 있는지를 파악하기 쉽게 나타낸 그래프
- x축, y축 상관 관계 표시
- 각 점들은 오른쪽 color bar를 참고하여 색깔 별로 나이를 표시함
- 버블의 위치: 남녀 비율
- 버블의 크기: 연령대별 인구수를 표현



산점도(산포도) 표현

▪ scatter() 함수 사용

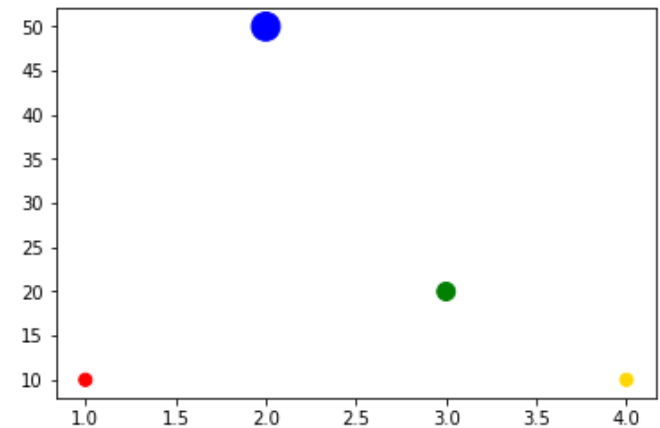
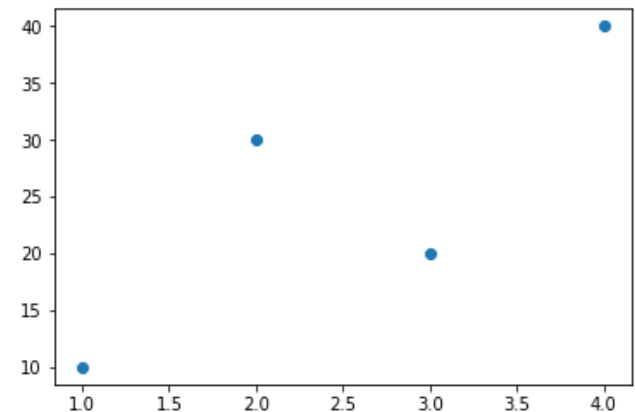
- x축에 해당하는 데이터와 y축에 해당하는 데이터를 넣으면 산점도가 완성됨

- 간단한 산점도

```
import matplotlib.pyplot as plt
plt.scatter([1, 2, 3, 4], [10, 30, 20, 40])
plt.show()
```

- 버블의 크기 표시: s 속성
- 버블의 색상 변경: c 속성

```
import matplotlib.pyplot as plt
y_value = [10, 50, 20, 10]
x_value = [1, 2, 3, 4]
size = []
for item in y_value:
    size.append(item*5)
# s(size): 버블의 크기
plt.scatter(x_value, y_value, s=size,
            c=['red', 'blue', 'green', 'gold'])
plt.show()
```

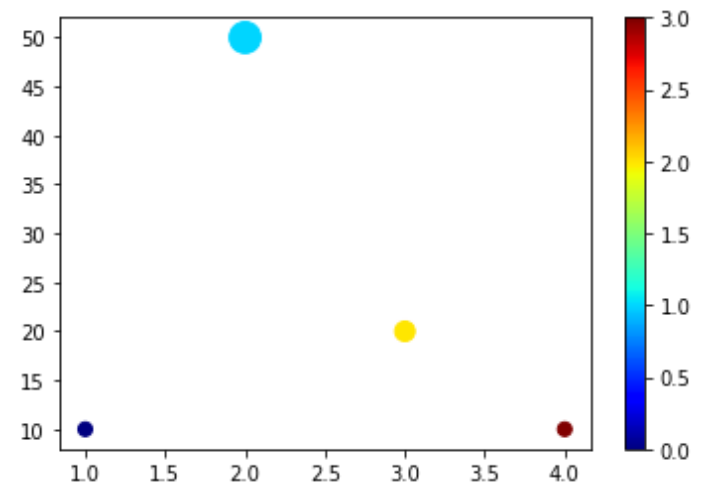


산점도에 color bar 추가

- `colorbar()` 함수
 - 그래프 우측에 color bar를 추가함
- `scatter()` 함수 속성 추가
 - `c=range(색상 개수)`
 - 각 데이터에 해당하는 color bar의 색으로 정해짐
 - `cmap`: 컬러맵 속성 사용 (`cmap='jet'`) – 무지개색
 - <https://matplotlib.org/tutorials/colors/colormaps.html?highlight=colormap>

```
import matplotlib.pyplot as plt

y_value = [10, 50, 20, 10]
x_value = [1, 2, 3, 4]
size = []
for item in y_value:
    size.append(item*5)
plt.scatter(x_value, y_value, s=size,
            c=range(4), cmap='jet')
plt.colorbar()
plt.show()
```





Questions?