9. 데이터 시각화 (Visualization) #1

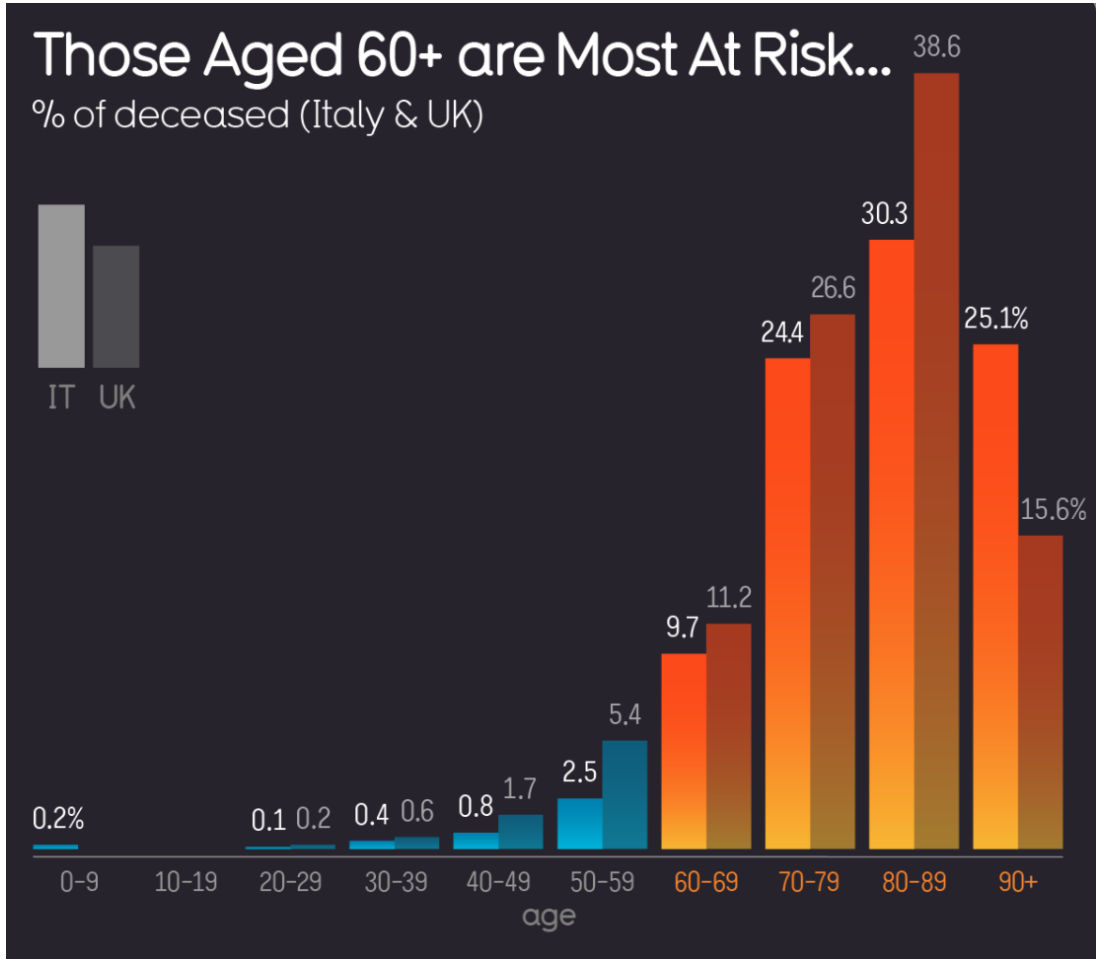
데이터 시각화

- 일반적으로 사람은 필요한 정보의 80% 가량을 시각을 통해서 받아들임
- 수치 데이터보다 시각적으로 보이는 그림이 더 직관적으로 이해할 수 있음
- 그러므로 효과적인 시각화는 데이터를 분석하고 추론하는 데 중요함

데이터 시각화 예시1

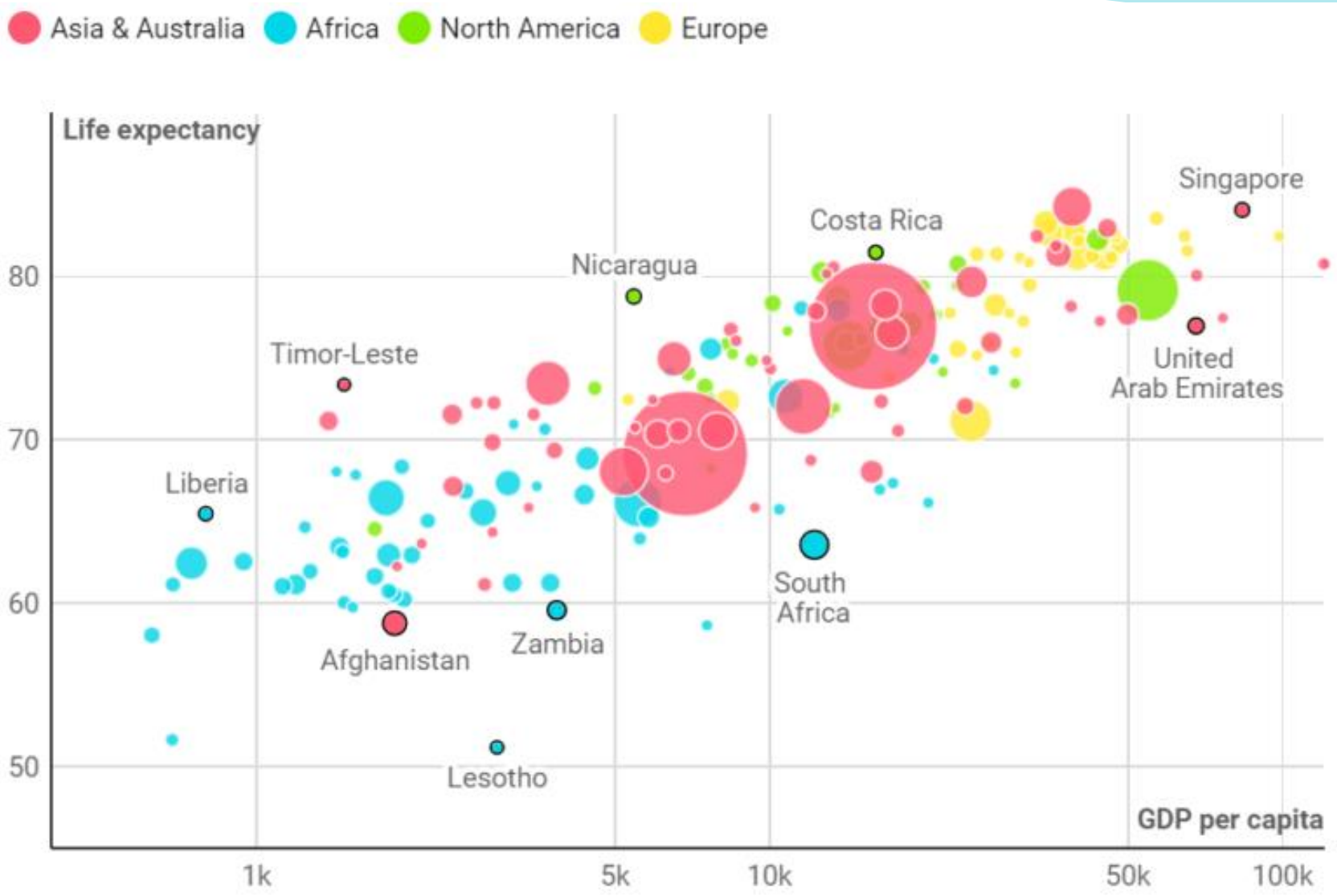
- COVID-19 감염질환은 나이가 어린 감염자는 치명률이 낮지만 고령층 감염자의 치명률은 높은 것을 보여줌.

- 시각화를 통해서 그 의미를 잘 전달할 수 있음.



source: informationisbeautiful.net

데이터 시각화 예시2


Asia & Australia
Africa
North America
Europe
Life expectancy
80
70
60
50
Singapore
Costa Rica
Nicaragua
Timor-Leste
United
Arab Emirates
Liberia
South
Africa
Afghanistan
Zambia
Lesotho
GDP per capita
1k
5k
10k
50k
100k

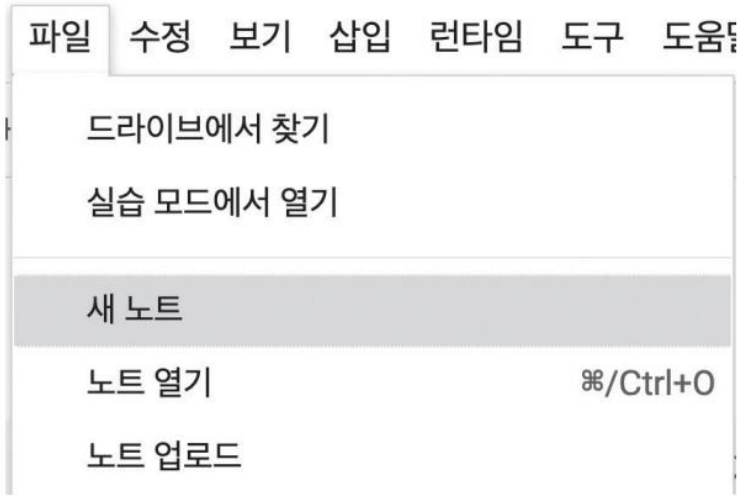
파이썬 데이터 시각화 도구

- matplotlib
  - 파이썬에서 널리 사용되는 시각화 도구로 간단한 막대, 선, 산점도 그래프를 생성
  - pip install matplotlib

    https://matplotlib.org/

구글 코랩 (Colab)

- Colab 은 클라우드 기반 개발 환경
- 구글 계정으로 로그인하면 사용. https://colab.research.google.com/
- 파일 메뉴의 새 노트를 선택하여 시작

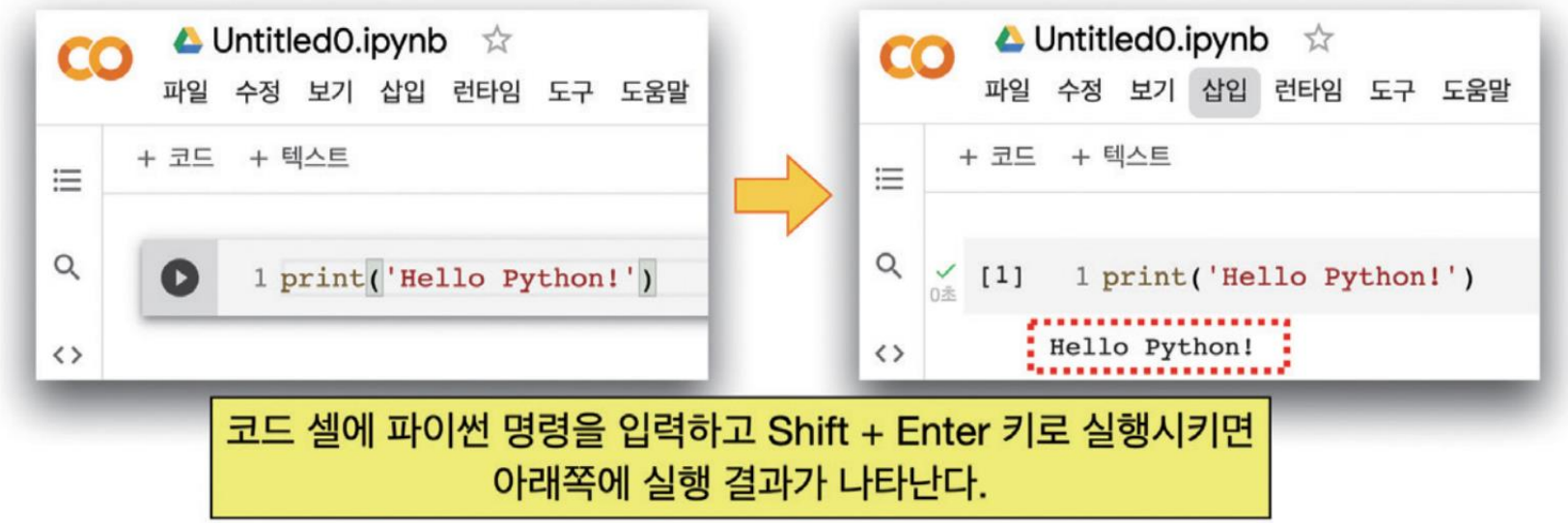


* 구름 에서는 그래프 출력 기능을 지원하지 않아, 데이터 시각화 실습을 위해선 코랩을 사용.

Colab 실행 예시

- 상단의 "+코드", "+텍스트"는 코드 셀이나 텍스트 셀을 생성
- 코드 셀에 파이썬 코드 작성 후 **Shift + Enter 키를 입력하여 셀을 실행**

  (혹은 왼쪽 화살표를 클릭)



Colab 에서 matplotlib 라이브러리 불러오기

- 아래와 같이 import 명령을 통해 라이브러리 로드
- 한글 지원을 위해 아래 코드 작성

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

!sudo apt-get install -y fonts-nanum
!sudo fc-cache -fv
!rm ~/.cache/matplotlib -rf
plt.rc('font', family='NanumBarunGothic')
mpl.rcParams['axes.unicode_minus'] = False
```

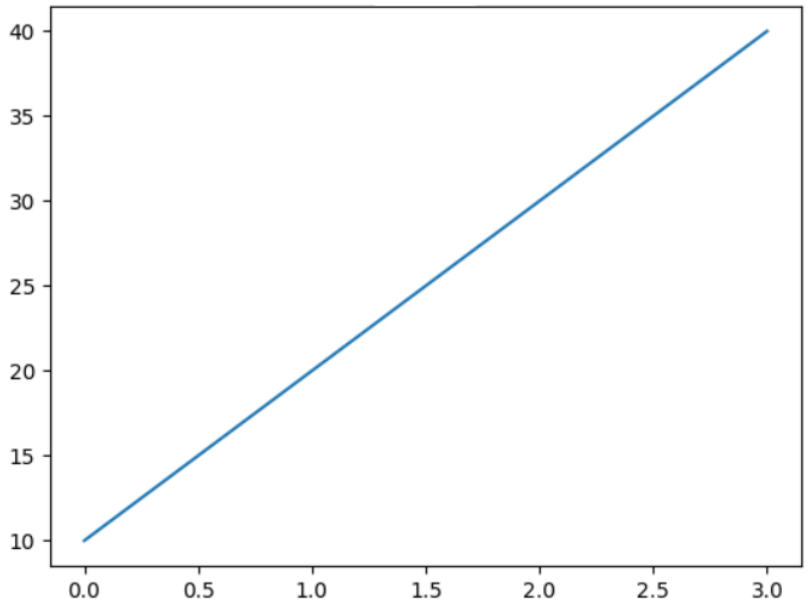
한글이 정상적으로 나오지 않을 땐,

메뉴 "런타임 -> 런타임 다시 시작" 후 재실행

matplotlib: pyplot 서브패키지

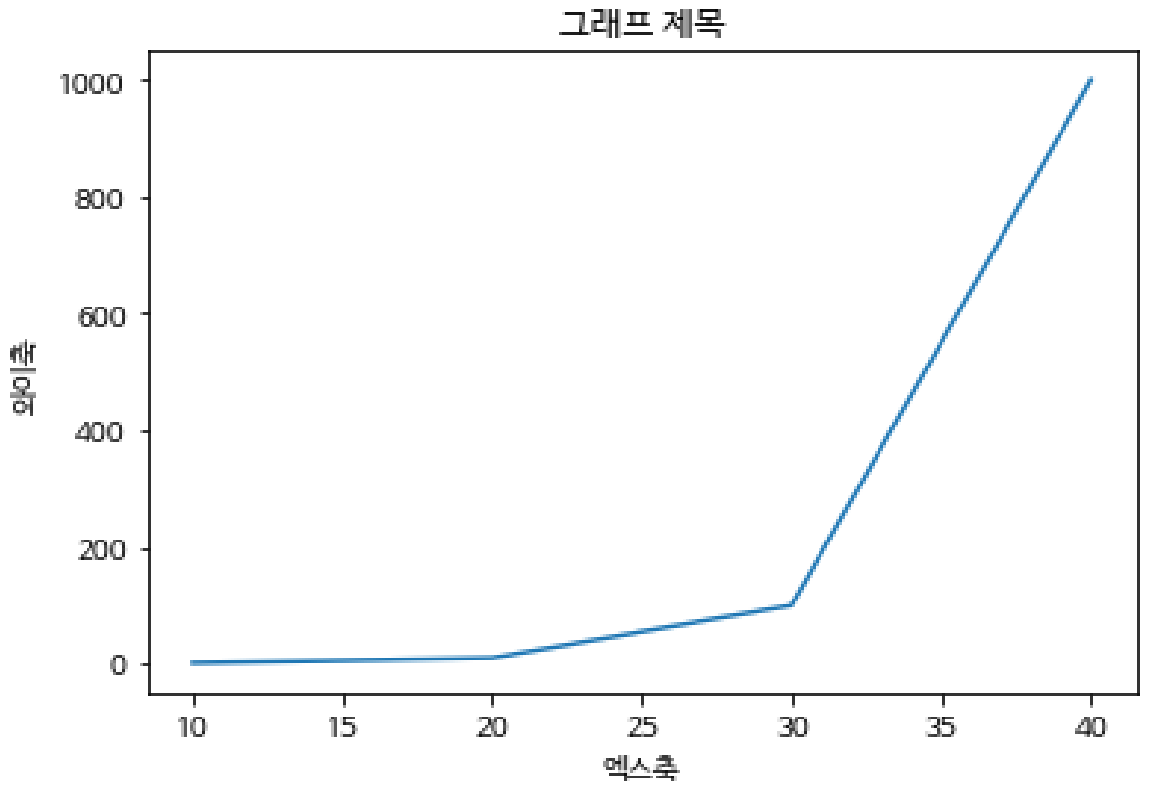
- pyplot 은 매트랩(matlab) 이라는 수치해석 소프트웨어와 비슷하게 동작하는 함수 모음. 간단한 시각화를 위해 pyplot 을 주로 사용.

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


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



pyplot : plot (선 그래프)

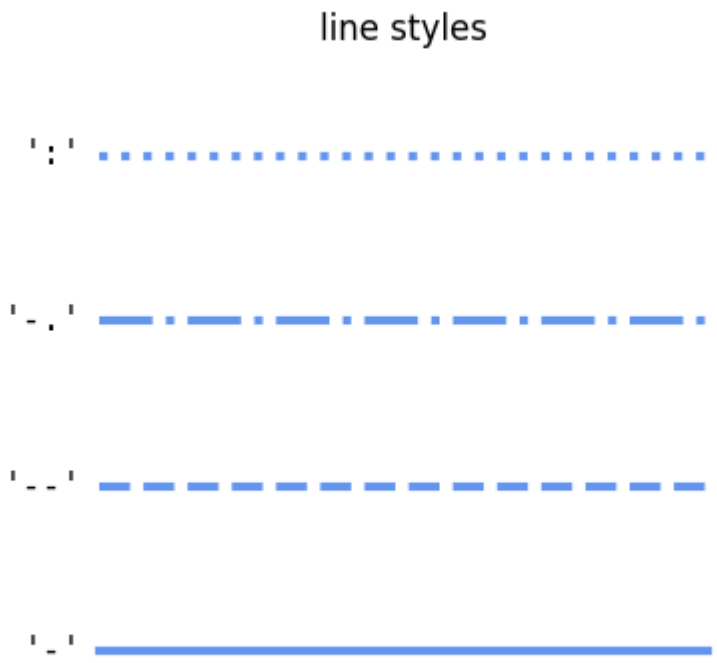
```
#그래프 제목 설정
plt.title("그래프 제목")
# 그래프 축 이름 설정
plt.xlabel("엑스축")
plt.ylabel("와이축")
# 그래프 x축, y축 값 지정
plt.plot([10,20,30,40], [1,10,100,1000])
plt.show()
```



plot 스타일 지정 : 색상

https://matplotlib.org/2.0.2/examples/color/named_colors.html

선 종류, 마커 모양 지정

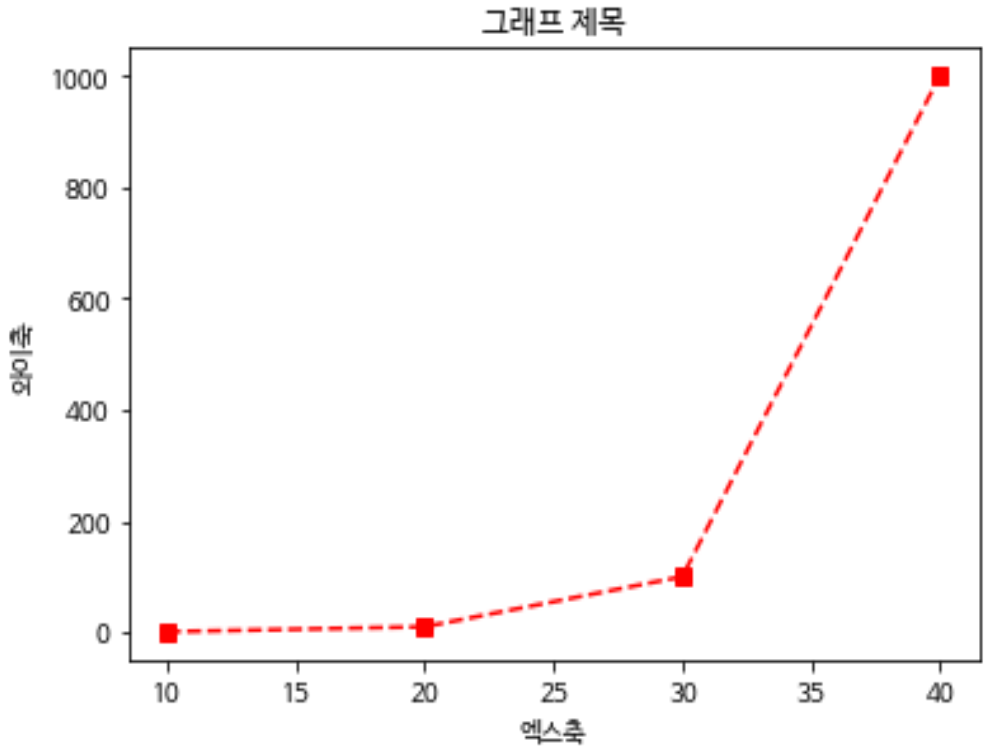


https://matplotlib.org

marker	symbol	description
"."	●	point
","	·	pixel
"o"	●	circle
"v"	▼	triangle_down
"^"	▲	triangle_up
"<"	◀	triangle_left
">"	▶	triangle_right
"1"	Y	tri_down
"2"	⅄	tri_up
"3"	≺	tri_left
"4"	≻	tri_right
"8"	●	octagon
"s"	■	square
"p"	⬟	pentagon
"P"	✚	plus (filled)
"*"	★	star

plot 스타일 지정 예시

```
plt.title("그래프 제목")
# 스타일: red 색상, 사각형, 점선
plt.plot([10,20,30,40],
         [1,10,100,1000], 'rs--')
plt.xlabel("엑스축")
plt.ylabel("와이축")
plt.show()
```



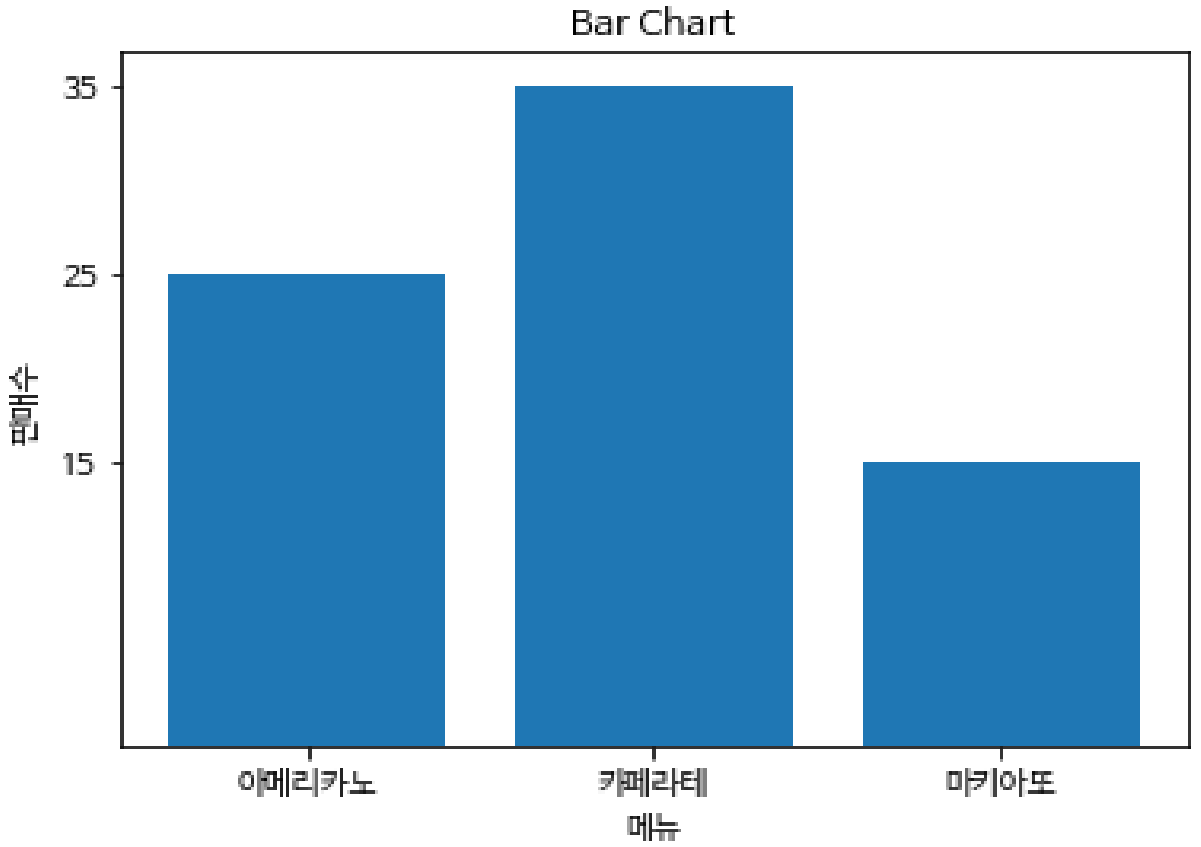
다른 종류의 그래프

- Matplotlib는 기본적인 라인 그래프 이외에도 다양한 그래프/차트 유형을 지원한다.

막대 그래프 (bar)

- bar( ) 함수로 막대 그래프를 시각화

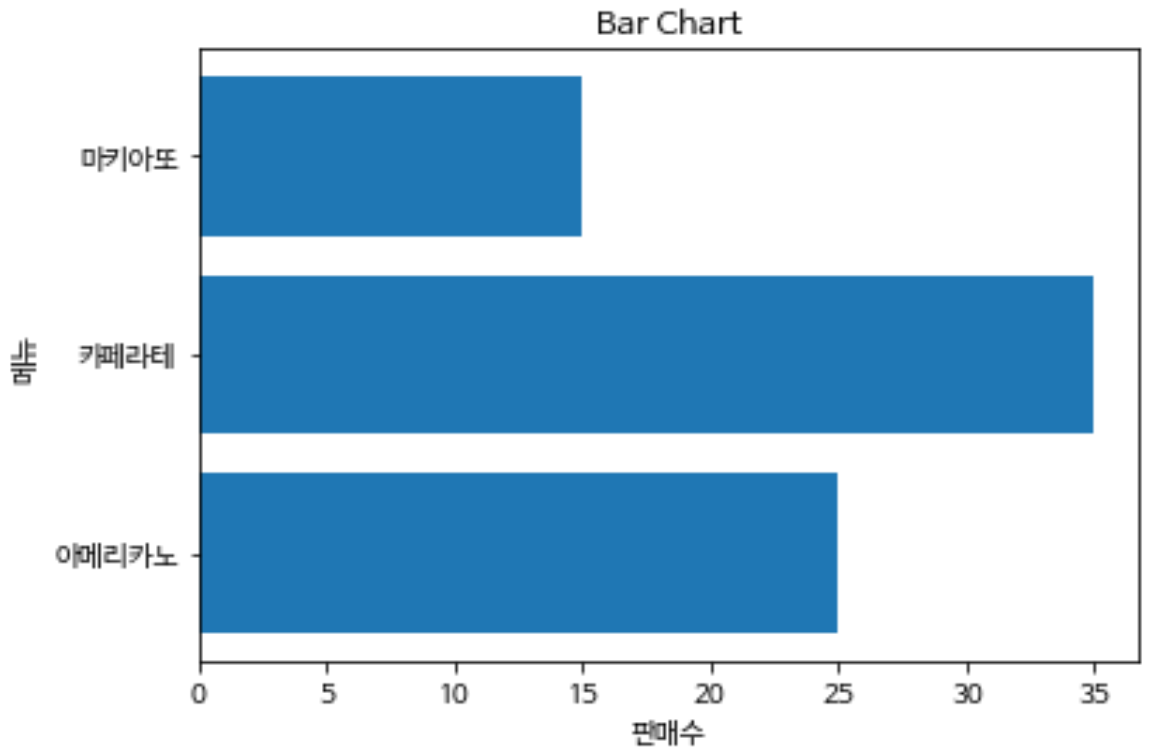
```
sales = [25, 35, 15]
menu = ['아메리카노', '카페라테', '마키아또']
plt.title("Bar Chart")
plt.xlabel("메뉴")
plt.ylabel("판매수")
# 막대 그래프 : x축은 메뉴, y축은 판매수
plt.bar(menu, sales)
plt.show()
```



수평 막대 그래프 (Barh)

- barh( ) 함수로 수평 막대 그래프 시각화

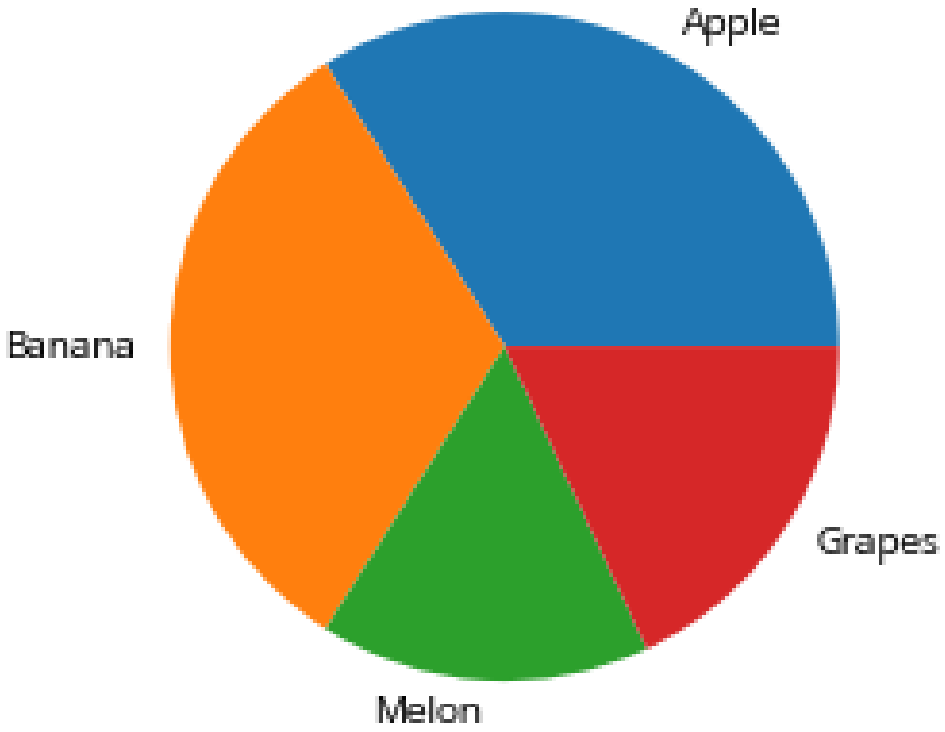
```
sales = [25, 35, 15]
menu = ['아메리카노', '카페라테', '마키아또']
plt.title("Barh Chart")
plt.xlabel("판매수")
plt.ylabel("메뉴")
# 수평 막대 그래프 : 인자로 넘겨주는 순서 주의
plt.barh( menu, sales )
plt.show()
```



파이 차트 (Pie chart, 원 그래프)

- 범주별 구성 비율을 원형으로 표현한 그래프

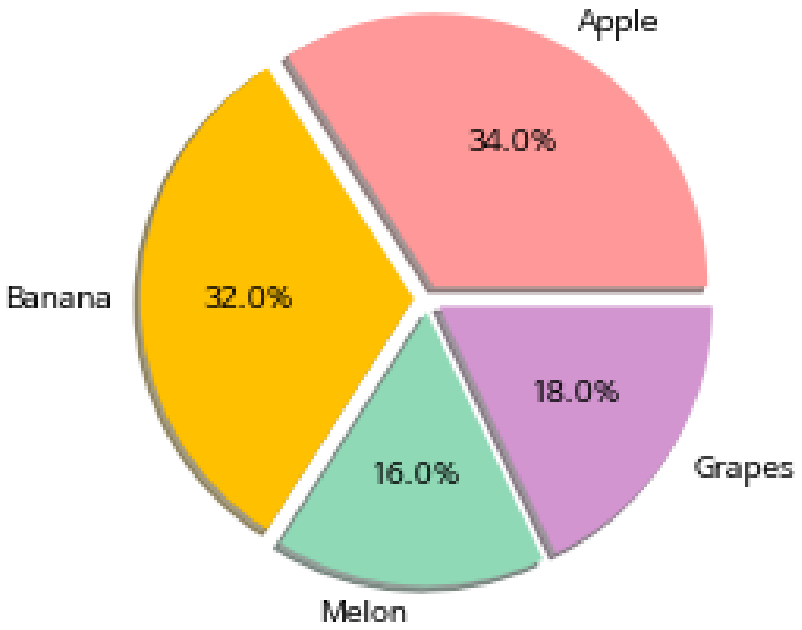
```
ratio = [34, 32, 16, 18]
labels = ['Apple', 'Banana', 'Melon', 'Grapes']

plt.pie(ratio, labels=labels)
plt.show()
```



파이 차트 스타일 지정

- 스타일 지정으로 시각화 효과를 높일 수 있음

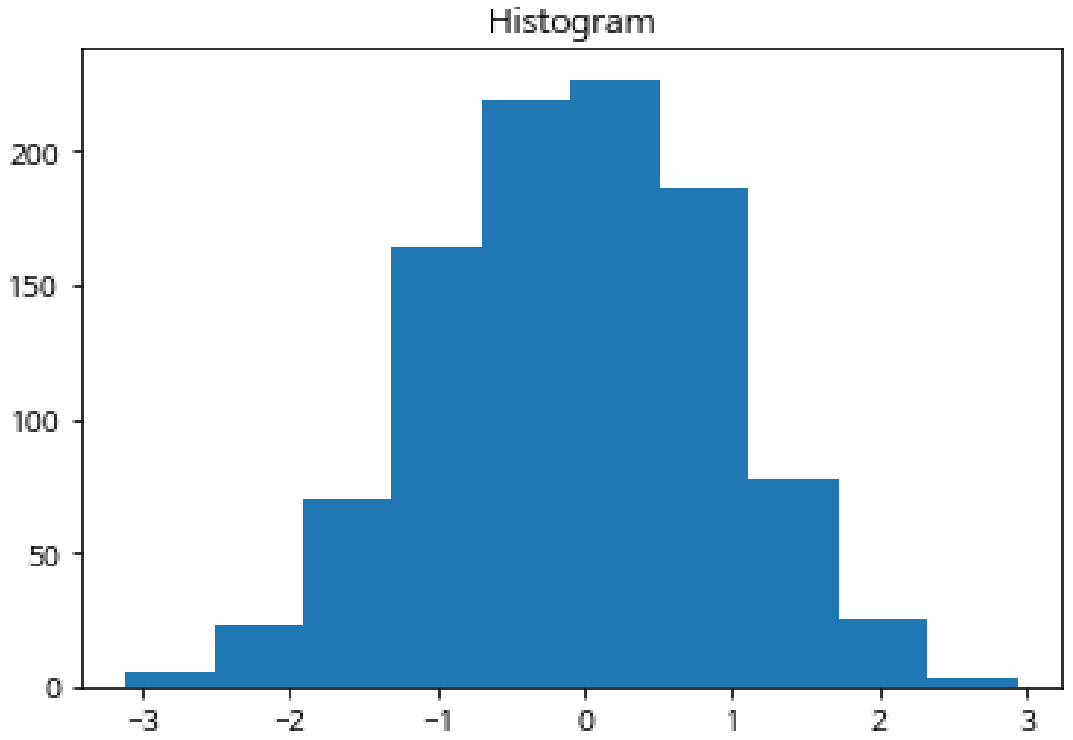
```
ratio = [34, 32, 16, 18]
labels = ['Apple', 'Banana', 'Melon', 'Grapes']
explode = [0.05, 0.05, 0.05, 0.05]
colors = ['#ff9999', '#ffc000',
          '#8fd9b6', '#d395d0']
plt.pie(ratio, labels=labels, autopct='%.1f%%',
 explode=explode, shadow=True, colors=colors)
plt.show()
```



히스토그램 (Histogram)

- 도수분포표 그래프. 가로축은 집계 구간, 세로축은 도수 (횟수나 개수 등)를 나타냄

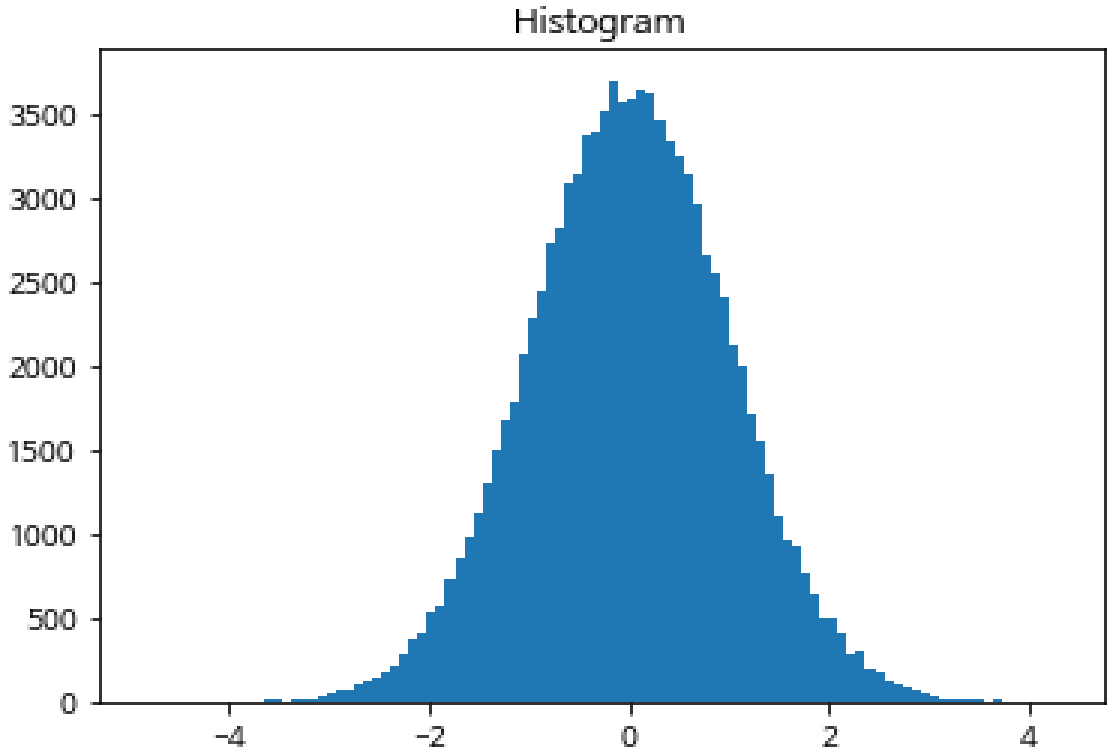
```
# 가우시안 표준 정규 분포 난수 생성
x = np.random.randn(1000)
plt.title("Histogram")
plt.hist(x)
plt.show()
```



히스토그램 (Histogram)

- 구간 개수에 따라 히스토그램 분포의 형태가 달라질 수 있음

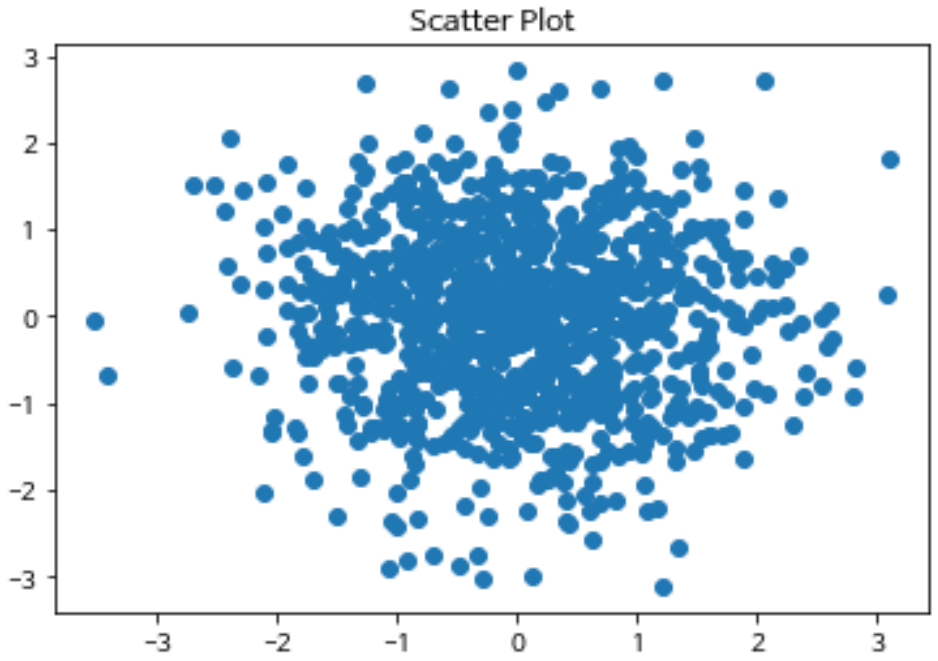
```
# 가우시안 표준 정규 분포 난수 생성
x = np.random.randn(100000)
plt.title("Histogram")

# 구간 개수 지정
plt.hist(x, bins=100)
plt.show()
```



산점도 (scatter) 그래프

- 두 변수의 상관 관계를 직교 좌표계의 평면에 점으로 표현하는 그래프

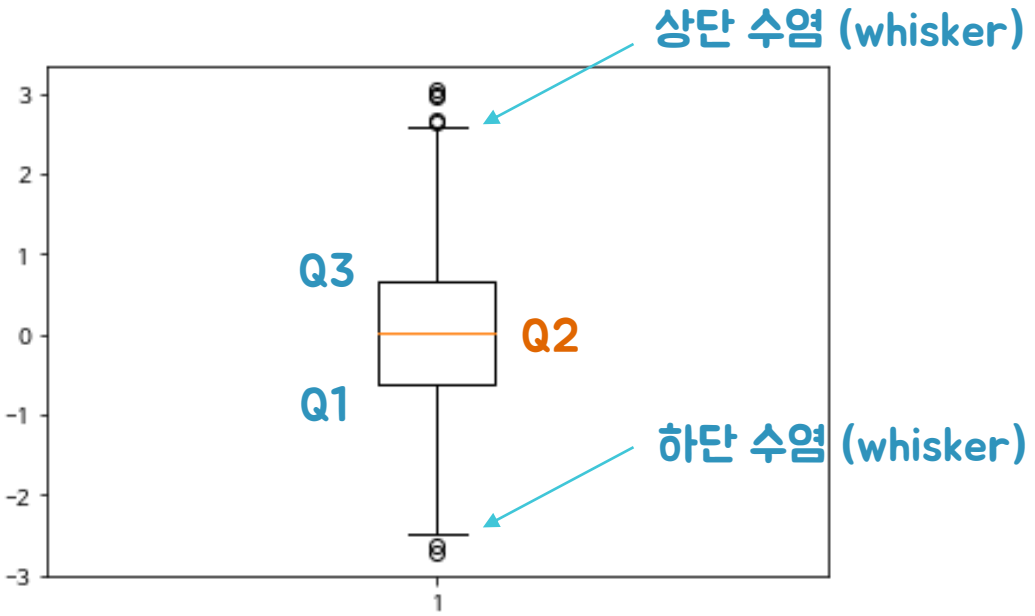
```
X = np.random.randn(1000)
Y = np.random.randn(1000)
plt.title("Scatter Plot")
plt.scatter(X, Y)
plt.show()
```



박스 (Box) 그래프

- 최소값, 제 1사분위 수 (Q1), 제 2사분위 수 (중앙값), 제 3사분위 수 (Q3), 최대값

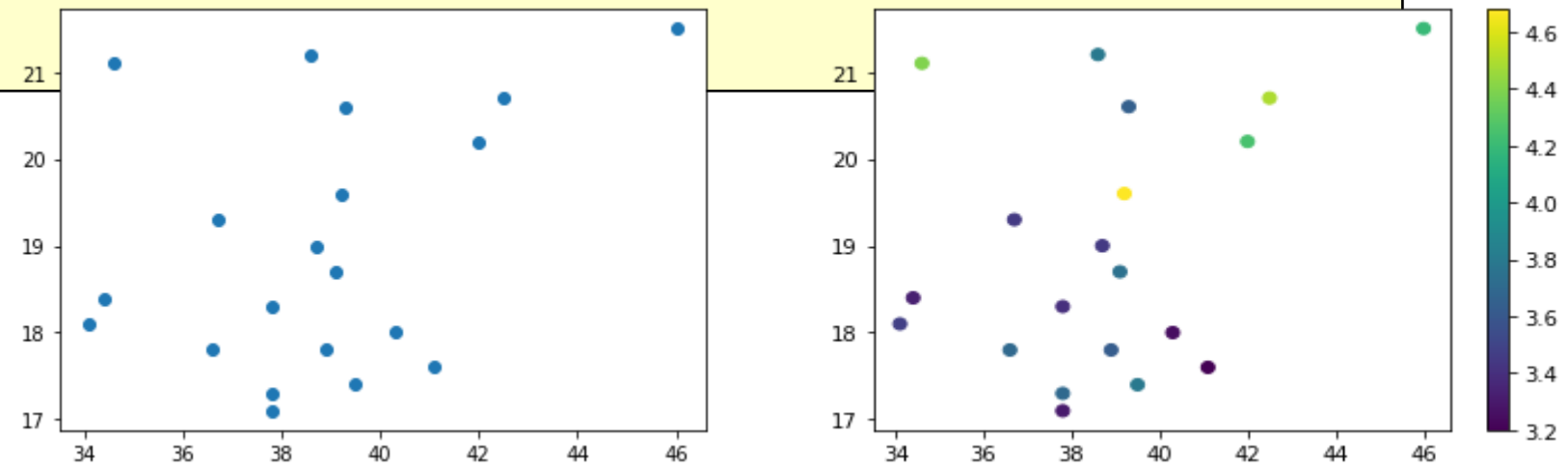
```
data = np.random.randn(1000)
plt.boxplot( data )
plt.show()
```



- IQR (Inter Quartile Range) = Q3 - Q1
- 상단 수염 (whisker) : Q3 + 1.5 x IQR 보다 작은 데이터 중 가장 큰 값
- 하단 수염 (whisker) : Q1 - 1.5 x IQR 보다 큰 데이터 중 가장 작은 값
- 수염 표시를 경계로 이상치 (outlier) 데이터를 구분

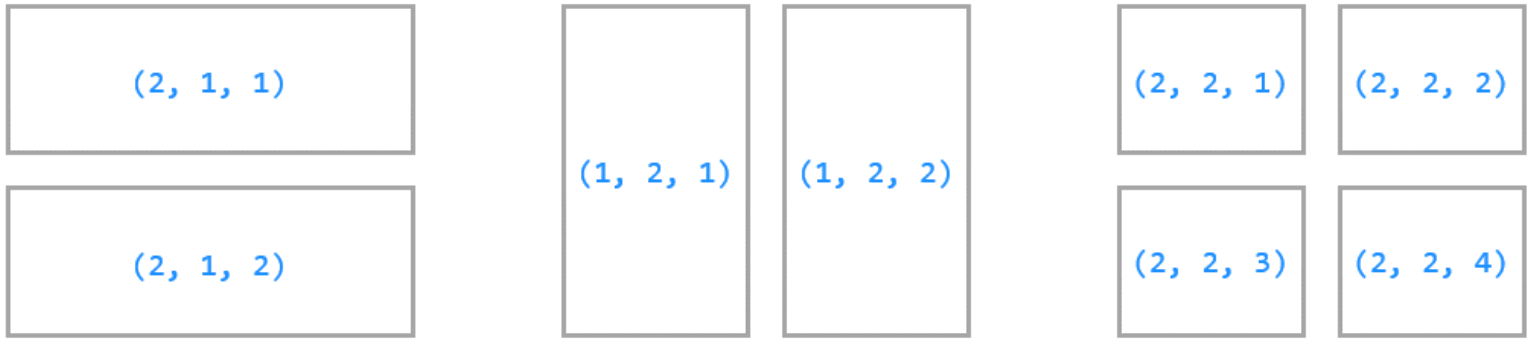
산점도 그래프에서 색상 활용

```
bill_lengh = [39.1, 39.5, 40.3, 36.7, 39.3, 38.9, 39.2, 34.1, 42.0, 37.8,
              37.8, 41.1, 38.6, 34.6, 36.6, 38.7, 42.5, 34.4, 46.0, 37.8],
bill_depth = [18.7, 17.4, 18.0, 19.3, 20.6, 17.8, 19.6, 18.1, 20.2,  17.1,
              17.3, 17.6, 21.2, 21.1, 17.8, 19.0, 20.7, 18.4, 21.5,  18.3],
body_mass = [3.75, 3.80, 3.25, 3.45, 3.65, 3.63, 4.68, 3.48, 4.25, 3.30,
             3.70, 3.20, 3.80, 4.40, 3.70, 3.45, 4.50, 3.33, 4.20, 3.40]

plt.scatter(bill_lengh, bill_depth)  # 펭귄 부리 길이와 두께

plt.show()

plt.scatter(bill_lengh, bill_depth, c=body_mass)  # 컬러로 펭귄 몸무게를 함께 표현

plt.colorbar()

plt.show()
```



여러 개의 그래프 그리기

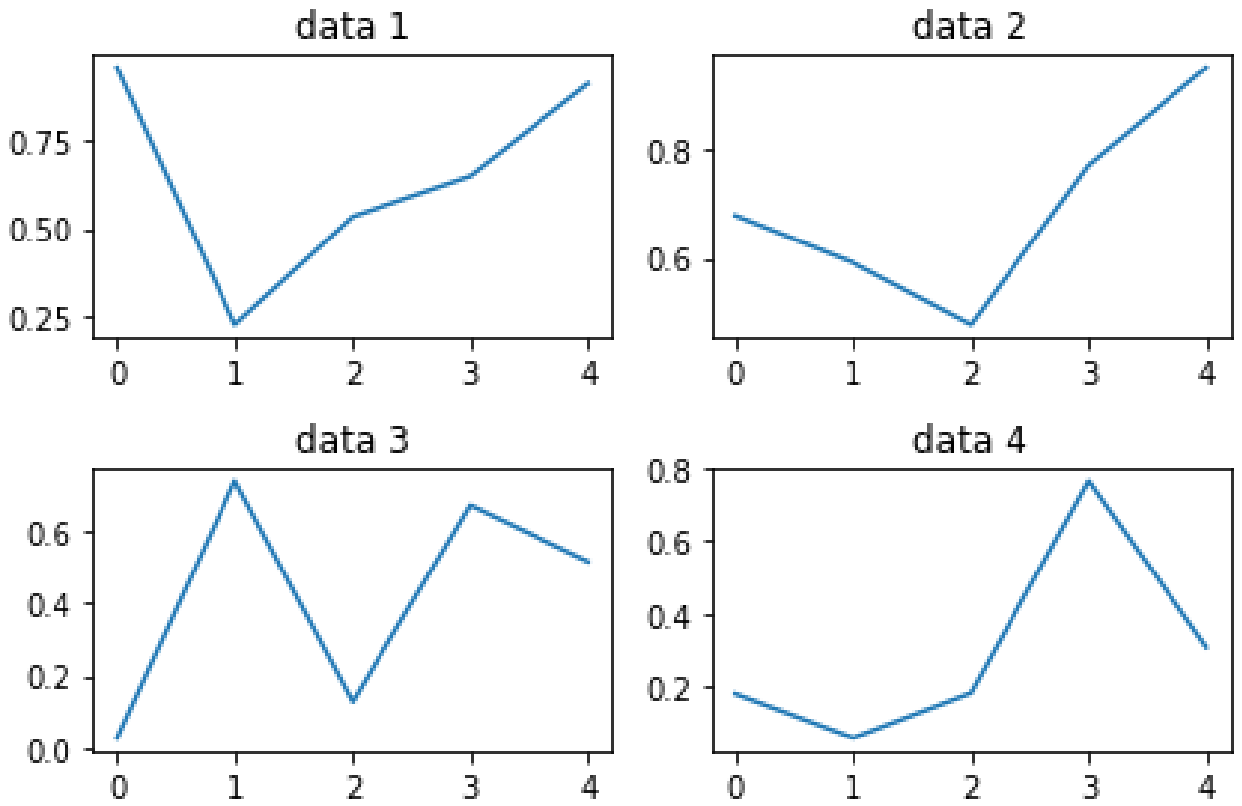
- subplot() 함수는 여러 개의 그래프를 하나의 그림으로 생성




```
plt.subplot(row, column, index)
```

subplot () 사용 예시

```
plt.subplot(2,2,1)
plt.title("data 1")
plt.plot(np.random.rand(5))

plt.subplot(2,2,2)
plt.title("data  2")
plt.plot(np.random.rand(5))

plt.subplot(2,2,3)
plt.title("data  3")
plt.plot(np.random.rand(5))

plt.subplot(2,2,4)
plt.title("data  4")
plt.plot(np.random.rand(5))

plt.tight_layout()
plt.show()
```



subplot () 사용 예시

```
plt.subplot(2,2,1)
ratio = [34, 32, 16, 18]
labels = ['Apple', 'Banana', 'Melon', 'Grapes']
explode = [0.05, 0.05, 0.05, 0.05]
colors = ['#ff9999', '#ffc000', '#8fd9b6', '#d395d0']
plt.pie(ratio, labels=labels, colors=colors)
plt.title("data 1")

plt.subplot(2,2,2)
sales = [25, 35, 15]
menu = ['아메리카노', '카페라테', '마키아또']
plt.barh(menu, sales)
plt.title("data 2")

plt.subplot(2,2,3)
x = np.random.randn(1000)
plt.hist(x, bins=100)
plt.title("data 3")

plt.subplot(2,2,4)
data1 = np.random.normal(0, 2.0, 1000)
data2 = np.random.normal(-3.0, 1.5, 1000)
data3 = np.random.normal(1.2, 1.5, 1000)
plt.boxplot( [data1, data2, data3] )
plt.title("data 4")
plt.tight_layout()
plt.show()
```