데이터분석 프로그래밍
데이터 시각화

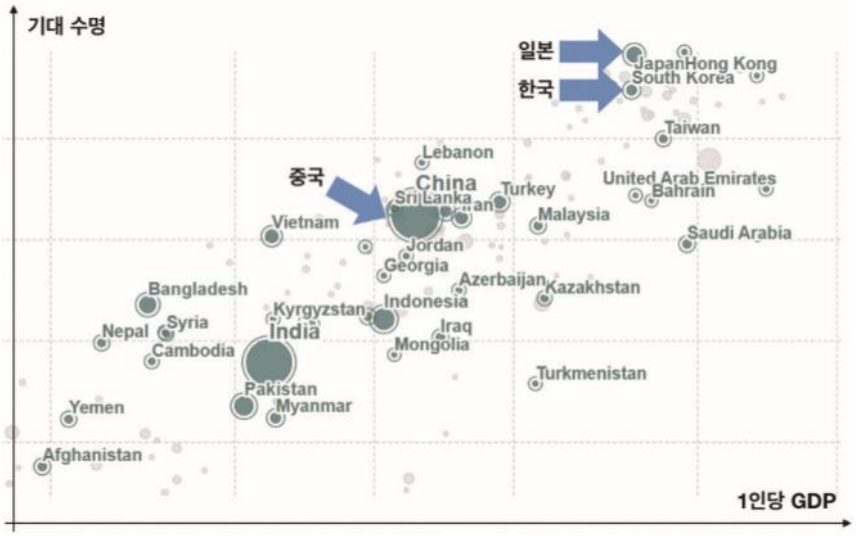
임현기

데이터 시각화

- 데이터 시각화는 점, 선, 막대 그래프 등의 시각적 이미지를 사용하여 데이터를 화면에 표현하는 기술
  - 데이터를 직관적으로 이해할 수 있게 함

데이터 시각화

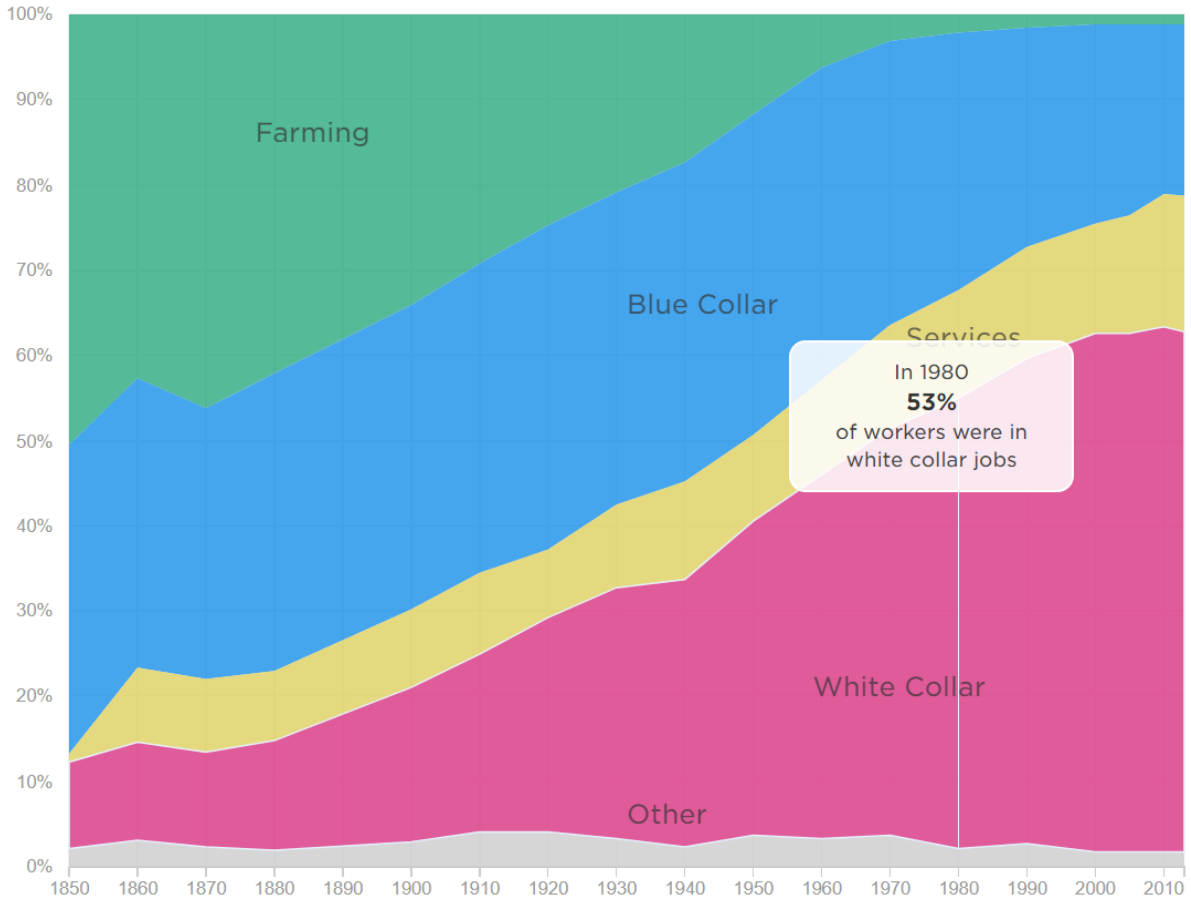
- 국내총생산(GDP)와 기대 수명 사이의 상관 관계



출처 : Our World in Data, https://ourworldindata.org/

데이터 시각화

- 1850년~지금까지의 노동 인구 구성 변화



matplotlib

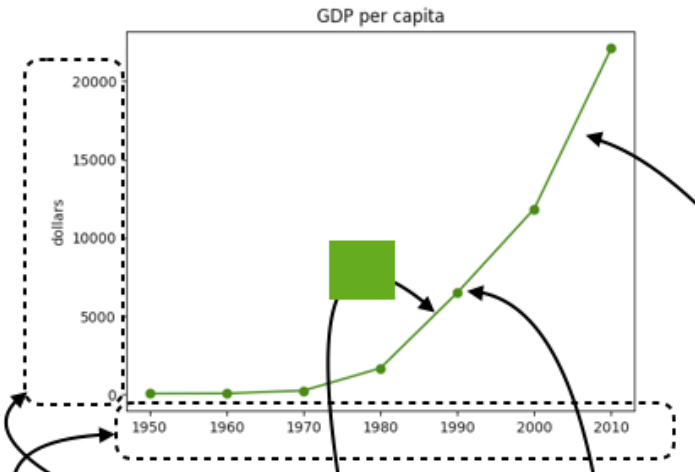
- 간단한 막대 그래프, 선 그래프, 산포도를 그림

```
import matplotlib.pyplot as plt

# 우리나라의 연간 1인당 국민소득을 각각 years, gdp에 저장
years = [1950, 1960, 1970, 1980, 1990, 2000, 2010]
gdp = [67.0, 80.0, 257.0, 1686.0, 6505, 11865.3, 22105.3]

# 선 그래프를 그린다. x축에는 years값, y축에는 gdp 값이 표시된다.
plt.plot(years, gdp, color='green', marker='o', linestyle='solid')

# 제목을 설정한다.
plt.title("GDP per capita") # 1인당 국민소득

# y축에 레이블을 붙인다.
plt.ylabel("dollars")
plt.savefig("gdp_per_capita.png", dpi=600)  # png 이미지로 저장 가능
plt.show()
```

GDP 데이터 출처: 한국은행 경제통계시스템, https://ecos.bok.or.kr/

matplotlib

years	1950	1960	1970	1980	1990	2000	2010
gdp	67.0	80.0	257.0	1686.0	6505.0	11863.3	22105.3




```
plt.plot(years, gdp, color='green', marker='o', linestyle='solid')
```

matplotlib

- pyplot 모듈을 불러와 plt로 별칭

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

- 연도별 GDP 변화
  - 연도는 years 리스트에 담고, x축 데이터로 사용
  - GDP는 gdp 리스트에 담고, y축 데이터로 사용

```
years = [1950, 1960, 1970, 1980, 1990, 2000, 2010]
gdp = [67.0, 80.0, 257.0, 1686.0, 6505, 11865.3, 22105.3]
```

- 선의 색, 마크의 표시방법, 선의 형태 설정

```
plt.plot(years, gdp, color='green', marker='o', linestyle='solid')
```

matplotlib

- 차트의 제목 레이블을 설정

```
plt.title("GDP per capita")
```

- y축의 레이블을 지정
- savefig() 함수를 통해서 파일을 저장

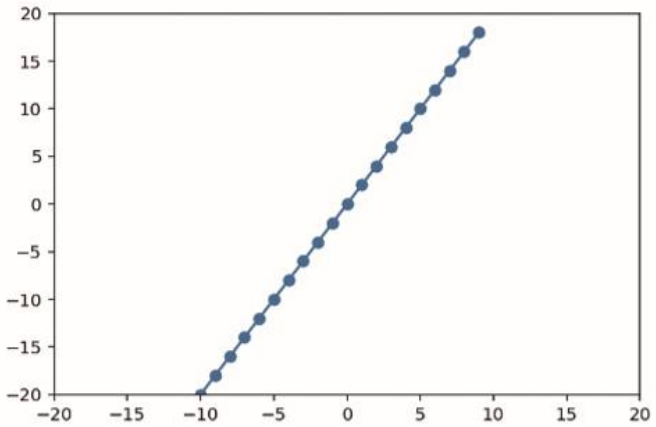
```
plt.ylabel("dollars")
plt.savefig("gdp_per_capita.png", dpi=600)
plt.show()
```

- 반드시 plt.show() 함수를 사용하여야 화면에 차트가 나타남

```
plt.plot(gdp, years, color='red',
marker='o', linestyle='solid')
```

이 실습에서는 $y = 2x$ 그래프를 그려보자. 많은 방법이 있지만, 여기서는 첫 번째 리스트 x에 -10에서 10사이의 수 담고, 두 번째 리스트는 이 첫 번째 리스트의 원소를 이용하여 y 값을 만들어 보자.

원하는 결과



```
import matplotlib.pyplot as plt

x = [x for x in range(-10, 10)]
y = [2*t for t in x]            # 2*x를 원소로 가지는 y 함수
plt.plot(x, y, marker='o')      # 선 그래프에 동그라미 표식을 출력

plt.axis([-20, 20, -20, 20])    # 그림을 그릴 영역을 지정함
plt.show()
```

도전문제 11.1

넘파이의 난수 생성을 이용하여 1,000개의 난수를 생성하고 생성된 순서대로 화면에 다음과 같이 그리는 일을 해 보라. 가로축은 생성된 순서, 세로축은 생성된 값이 될 것이다.

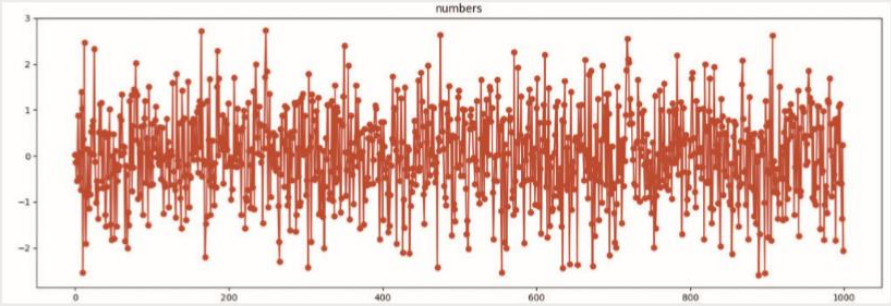


차트 장식

- 색깔, 마커, 선모양을 한번에 설정

  예) r-- g^- b*:

```
import matplotlib.pyplot as plt          # 지면 효율을 위해서 앞으로 이 줄은 생략함

x  = [x for x in range(-20, 20)]         # -20에서 20사이의 수를 1의 간격으로 생성
y1 = [2*t for t in x]                    # 2*x를 원소로 가지는 y1 함수
y2 = [t**2 + 5 for t in x]               # x**2 + 5를 원소로 가지는 y2 함수
y3 = [-t**2 - 5 for t in x]              # -x**2 - 5를 원소로 가지는 y3 함수
# 빨간색 점선, 녹색 실선과 세모기호, 파란색 별표와 점선으로 각각의 함수를 표현
plt.plot(x, y1, 'r--', x, y2, 'g^-', x, y3, 'b*:')
plt.axis([-30, 30, -30, 30])             # 그림을 그릴 영역을 지정함
plt.show()
```

  - rgb는 빨강, 녹색, 파랑
  - -- 점선, - 실선, : 짧은 점선
  - ^ 세모, * 별표

차트 장식

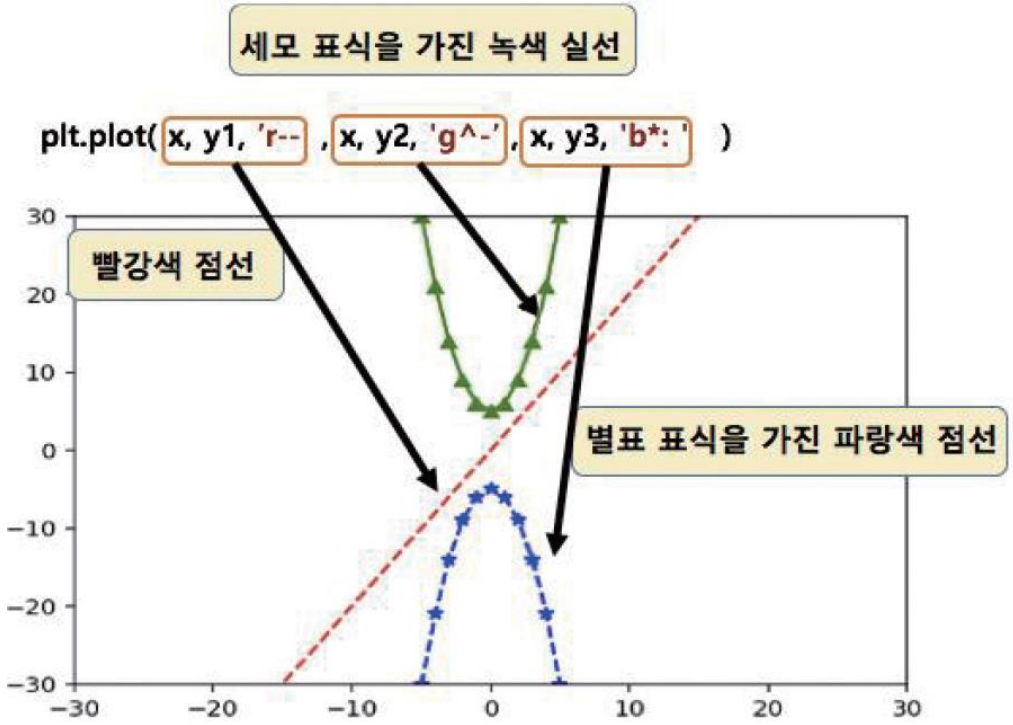

세모 표식을 가진 녹색 실선
plt.plot(x, y1, 'r--' , x, y2, 'g^-', x, y3, 'b*: ')
빨강색 점선
별표 표식을 가진 파랑색 점선
30
20
10
0
−10
−20
−30
−30
−20
−10
0
10
20
30

차트 장식

- 주요 키워드

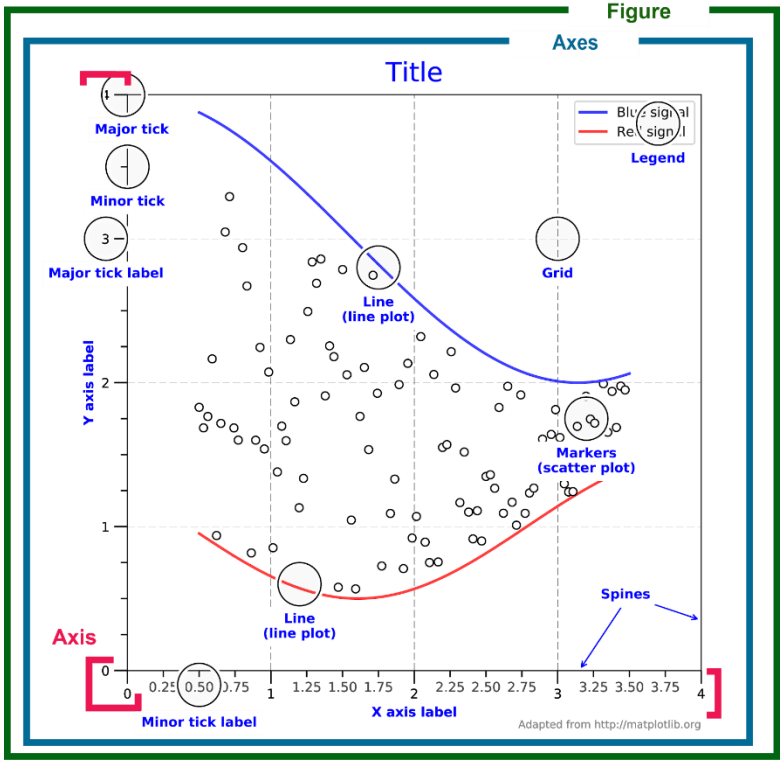
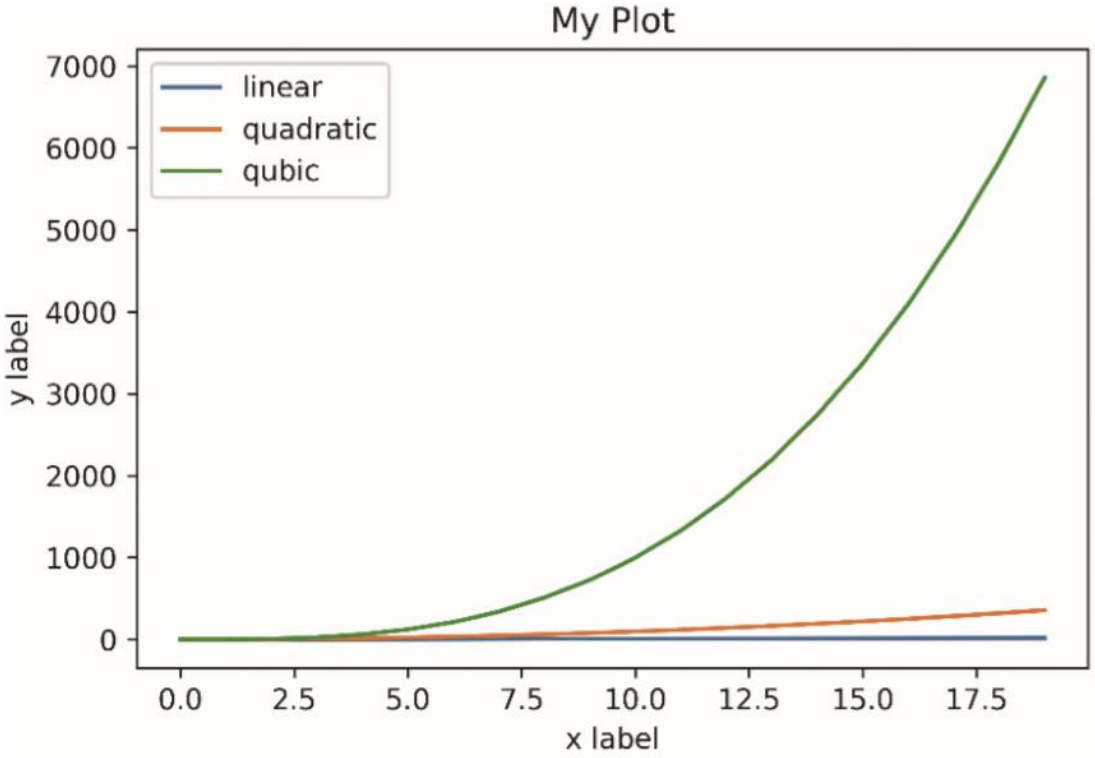


Figure
Axes
Title
Major tick
Minor tick
Major tick label
Legend
Grid
Line
(line plot)
Markers
(scatter plot)
Y axis label
Line
(line plot)
Spines
Axis
Minor tick label
X axis label
Adapted from http://matplotlib.org

하나의 차트에 그리기

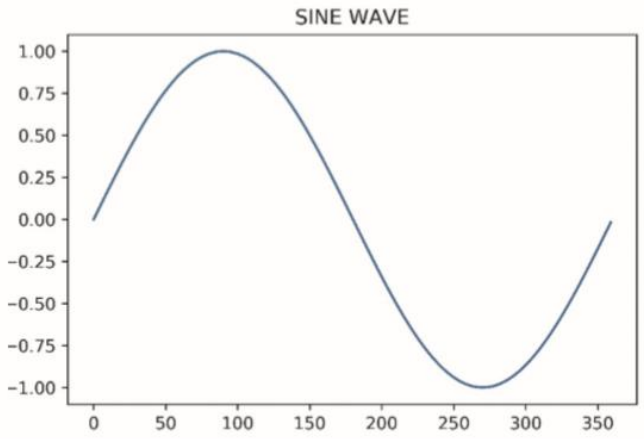
```
import matplotlib.pyplot as plt

x = [x for x in range(20)]      # 0에서 20까지의 정수를 생성
y = [x**2 for x in range(20)]   # 0에서 20까지의 정수 x에 대해 x 제곱값을 생성
z = [x**3 for x in range(20)]   # 0에서 20까지의 정수 x에 대해 x 세제곱값을 생성

plt.plot(x, x, label='linear')    # 각 선에 대한 레이블
plt.plot(x, y, label='quadratic')
plt.plot(x, z, label='qubic')

plt.xlabel('x label')       # x 축의 레이블
plt.ylabel('y label')       # y 축의 레이블
plt.title("My Plot")
plt.legend()                # 디폴트 위치에 범례를 표시한다
plt.show()
```

하나의 차트에 그리기


My Plot
linear
quadratic
qubic
7000
6000
5000
4000
3000
2000
1000
0
y label
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
x label

다음과 같은 사인 그래프를 그려보자. 사인값은 math 모듈의 sin() 함수로 계산할 수 있다.

원하는 결과



```python
import math
import matplotlib.pyplot as plt

x = []
y = []

for angle in range(360):
    x.append(angle)
    y.append(math.sin(math.radians(angle)))

plt.plot(x, y)
plt.title("SINE WAVE")
plt.show()
```

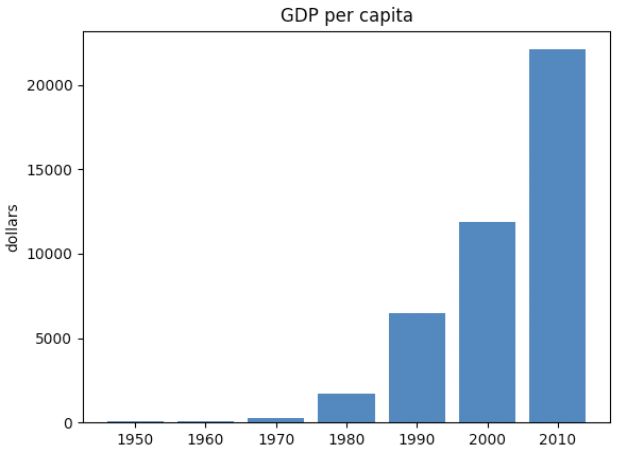
도전문제 11.3

사인 그래프 코드를 수정하여 동일한 그림위에 코사인 그래프도 그려보라. 코사인 그래프는 붉은색 실선으로 표기하여라.

막대형 차트 그리기

- plt.bar() 함수 호출

```
from matplotlib import pyplot as plt

# 1인당 국민소득
years = [1950, 1960, 1970, 1980, 1990, 2000, 2010]
gdp = [67.0, 80.0, 257.0, 1686.0, 6505, 11865.3, 22105.3]

plt.bar(range(len(years)), gdp)

plt.title("GDP per capita")    # 제목을 설정한다.
plt.ylabel("dollars")          # y축에 레이블를 붙인다.

# y축에 틱을 붙인다.
plt.xticks(range(len(years)), years)
plt.show()
```



막대형 차트 그리기

- bar() 함수의 가로축 범위 지정

```
plt.bar(range(len(years)), gdp)
```

- xtick() 함수를 이용하여 가로축 눈금 지정

```
plt.xticks(range(len(years)), years)
```

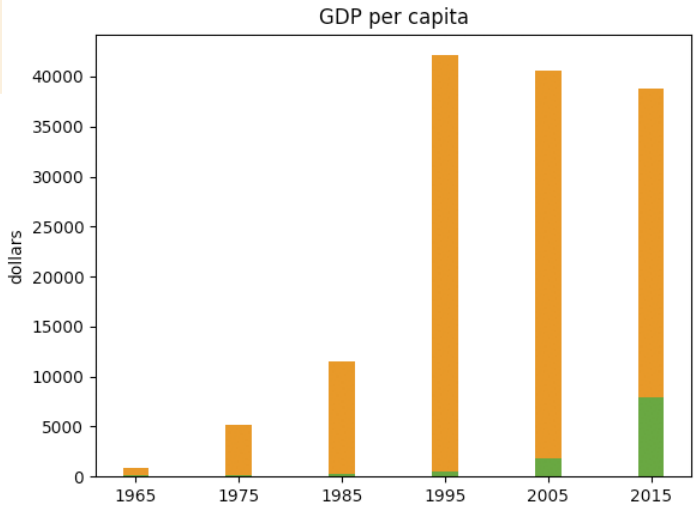
다중 막대형 차트 그리기

```
# 1인당 국민소득
years = [1965, 1975, 1985, 1995, 2005, 2015]
ko = [130, 650, 2450, 11600, 17790, 27250]
jp = [890, 5120, 11500, 42130, 40560, 38780]
ch = [100, 200, 290, 540, 1760, 7940]
```

- 두께를 0.25로 줄이고, 3개의 차트를 공유

```
x_range = range(len(years))
plt.bar(x_range, ko, width = 0.25)
plt.bar(x_range, jp, width = 0.25)
plt.bar(x_range, ch, width = 0.25)
plt.show()
```

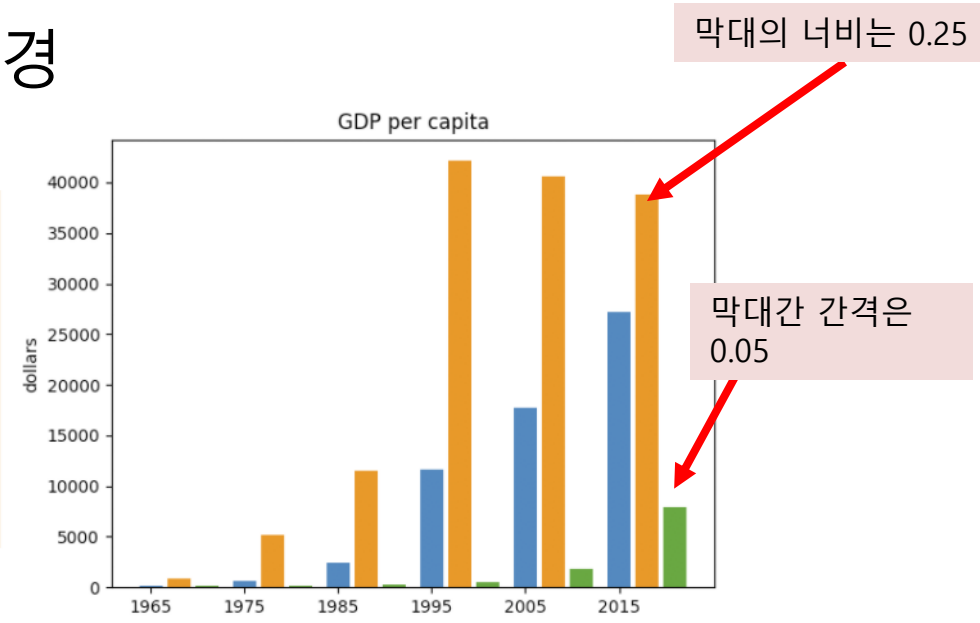
  - 알아보기 힘든 케이스



다중 막대형 차트 그리기

- 가로축 범위에 차이를 두기
  - range() 함수는 정수 범위만 가능
  - 넘파이를 이용하여 변경

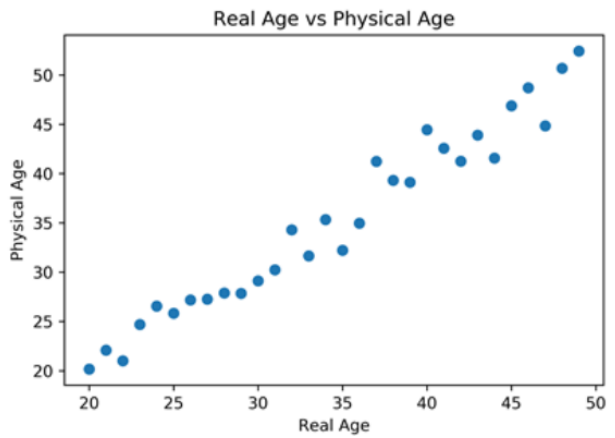
```
import numpy as np

x_range = np.arange(len(years))
plt.bar(x_range + 0.0, ko, width = 0.25)
plt.bar(x_range + 0.3, jp, width = 0.25)
plt.bar(x_range + 0.6, ch, width = 0.25)
plt.show()
```



산포도 그래프 그리기

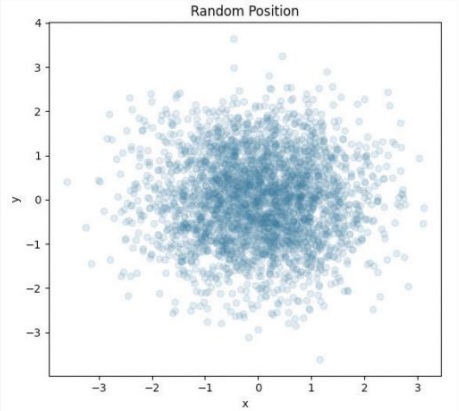
- scatter() 함수 사용

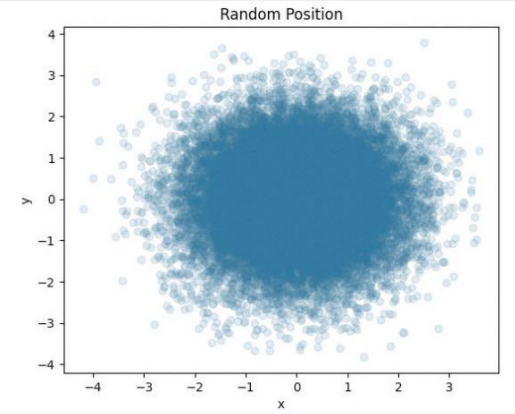
```
import matplotlib.pyplot as plt
import numpy as np

xData = np.arange(20, 50)
yData = xData + 2*np.random.randn(30)     # xData에 randn() 함수로 잡음을 섞는다.
                                          # 잡음은 정규분포로 만들어 질 것이다.

plt.scatter(xData, yData)
plt.title('Real Age vs Physical Age')
plt.xlabel('Real Age')
plt.ylabel('Physical Age')

plt.savefig("age.png", dpi=600)
plt.show()
```



도전문제 11.4

난수를 발생시켜 임의의 2차원 좌표를 생성해 그려보자. 이때 좌표의 x와 y 값은 표준정규분포를 따르도록 할 것이다. 그러면 생성된 좌표는 원점 (0,0)에 밀집한 모양을 가질 것이다. scatter() 함수 내에 불투명도를 의미하는 alpha 키워드 매개변수에 1보다 작은 값을 주어 점이 많이 겹칠 때 더 진하게 보이게 만들 수 있다.



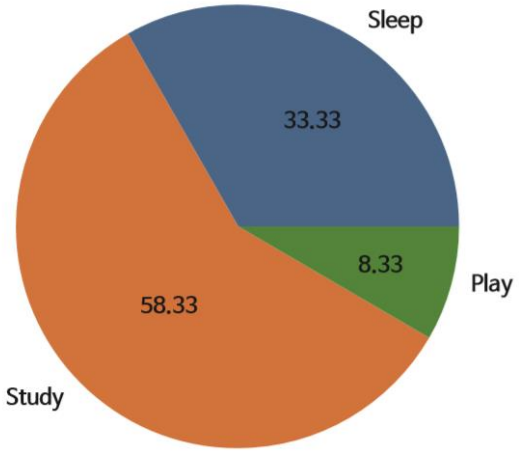


파이 차트 그리기

```
import matplotlib.pyplot as plt
times = [8, 14, 2]
```

```
timelabels = ["Sleep", "Study", "Play"]
```

```
# autopct로 백분율을 표시할 때 소수점 2번째 자리까지 표시하게 한다.
# labels 매개 변수에 timelabels 리스트를 전달한다.
plt.pie(times, labels = timelabels, autopct = "%.2f")
plt.show()
```



히스토그램

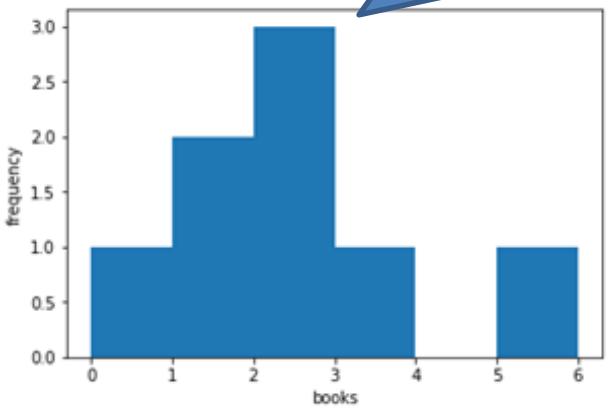
- 히스토그램
  - 주어진 자료를 몇 개의 구간으로 나누고, 각 구간의 도수를 나타낸 막대 그래프

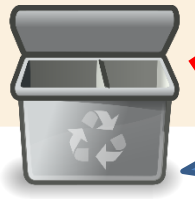
```
books = [ 1, 6, 2, 3, 1, 2, 0, 2 ]
```

학생 8명이 1년동안 읽은 책의 수

```
import matplotlib.pyplot as plt

books = [ 1, 6, 2, 3, 1, 2, 0, 2 ]

# 6개의 빈을 이용하여 books 안에 저장된 자료의 히스토그램 그리기
plt.hist(books, bins = 6)

plt.xlabel("books")
plt.ylabel("frequency")
plt.show()
```

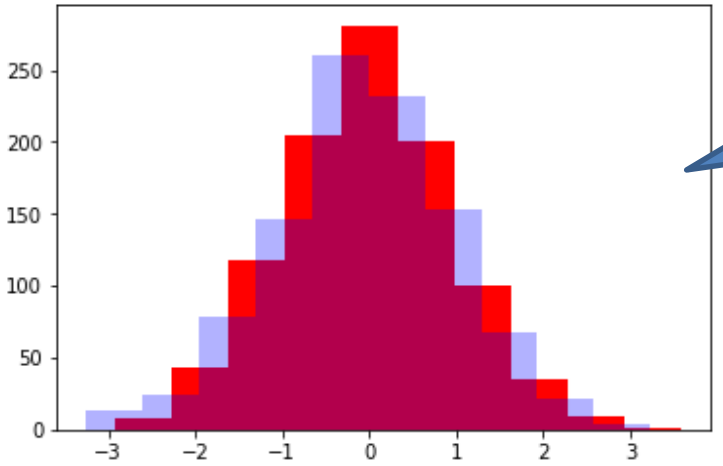
데이터의 분포를 설명하는 차트.





데이터가 들어가는 통을 bin이라 하자.

히스토그램



정규분포 함수가 만든
임의의 값의 분포

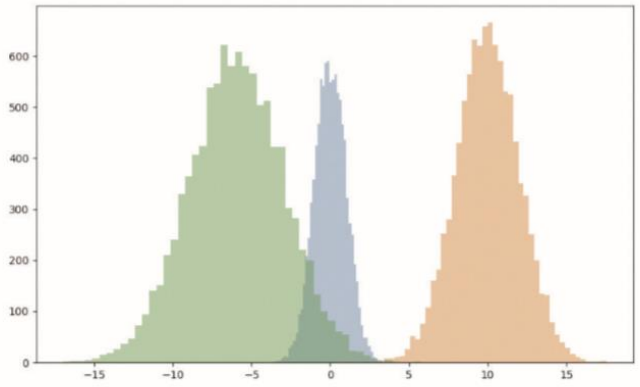
```
import numpy as np
import matplotlib.pyplot as plt

n_bins = 10
x = np.random.randn(1000)
y = np.random.randn(1000)

plt.hist(x, n_bins, histtype='bar', color='red')
plt.hist(y, n_bins, histtype='bar', color='blue', alpha=0.3)
plt.show()
```

10개의 구간으로
나누어서 분포를 그림

파란색 히스토그램의
투명도는 0.3
alpha 1 : 불투명
alpha 0 : 완전 투명

정규 분포로 난수를 생성하고 이 수들이 어떠한 방식으로 흩어져 있는지를 확인해 보고자 한다. 표준 정규 분포로 난수를 하나 생성해 보고, 다음으로는 평균 10, 표준편차 2인 정규분포로 난수를 생성해 보라. 그리고 마지막으로 평균 -6, 표준편차 3인 정규분포로 난수를 또 생성한다. 그리고 이렇게 생성된 난수들이 어떤 분포를 보이는지 히스토그램을 통해 확인해 보자.

원하는 결과



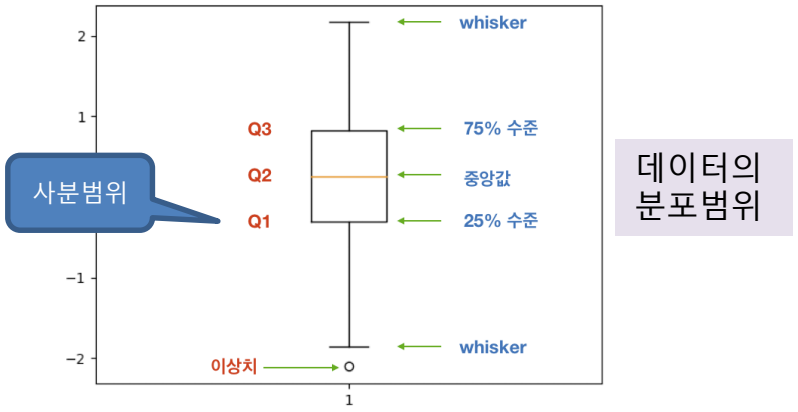
```python
import numpy as np
import matplotlib.pyplot as plt

mu1, sigma1 = 10, 2
mu2, sigma2 = -6, 3

standard_Gauss = np.random.randn(10000)
Gauss1 = mu1 + sigma1 * np.random.randn(10000)
Gauss2 = mu2 + sigma2 * np.random.randn(10000)

plt.figure(figsize=(10,6))
plt.hist(standard_Gauss, bins=50, alpha=0.4)
plt.hist(Gauss1, bins=50, alpha=0.4)
plt.hist(Gauss2, bins=50, alpha=0.4)

plt.show()
```

상자 차트

- 상자 차트
  - 데이터의 최대, 최소, 중간값, 사분위 수 등을 가시화

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

random_data = np.random.randn(100)
```

```
plt.boxplot(random_data)
plt.show()
```

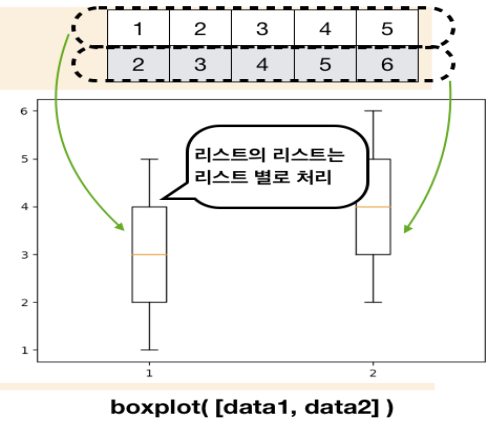


상자 차트

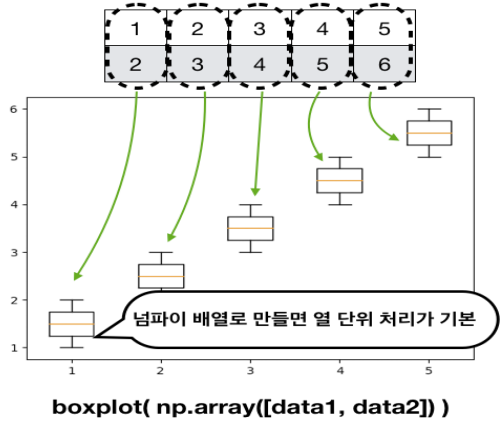
```
import numpy as np
import matplotlib.pyplot as plt
data1 = [1, 2, 3, 4, 5]
data2 = [2, 3, 4, 5, 6]
```

```
plt.boxplot([ data1, data2 ] )
```

```
plt.boxplot(np.array([ data1, data2 ]) )
```



boxplot([data1, data2])

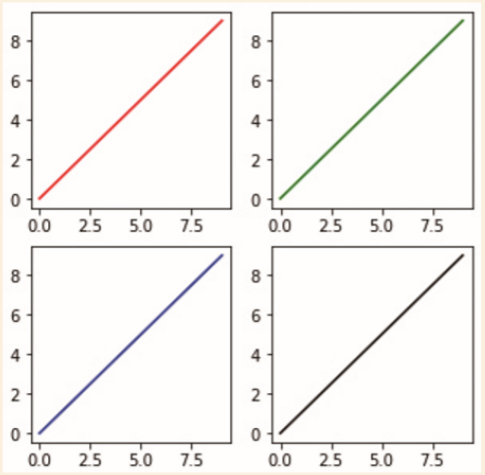


boxplot(np.array([data1, data2]))

여러 그래프 그리기

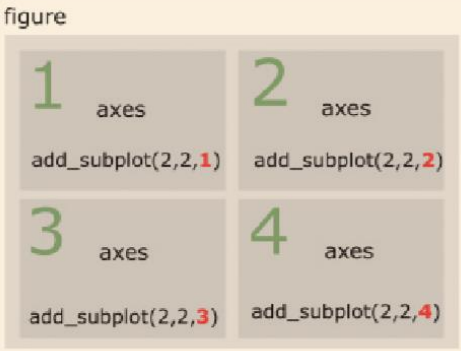
- subplots()

```
import matplotlib.pyplot as plt

# 개수가 2 x 2개, 크기가 (5, 5) 인치
fig, ax = plt.subplots(2, 2, figsize=(5, 5))

ax[0, 0].plot(range(10), 'r') # row=0, col=0
ax[1, 0].plot(range(10), 'b') # row=1, col=0
ax[0, 1].plot(range(10), 'g') # row=0, col=1
ax[1, 1].plot(range(10), 'k') # row=1, col=1
plt.show()
```



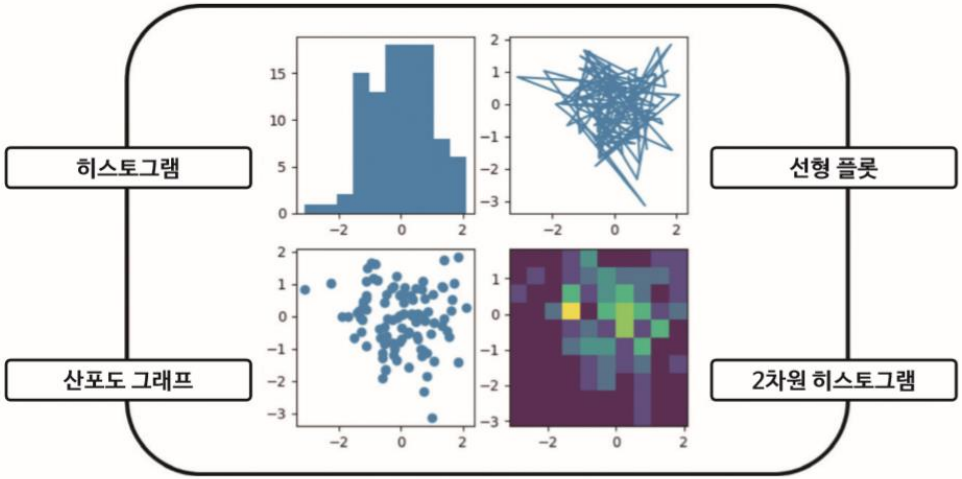
여러 그래프 그리기

```
fig = plt.figure()
ax1 = fig.add_subplot(2, 2, 1)
ax2 = fig.add_subplot(2, 2, 2)
ax3 = fig.add_subplot(2, 2, 3)
ax4 = fig.add_subplot(2, 2, 4)
```



다양한 차트를 하나의 그림에 나타내어서 강력하고 유연한 데이터 표현을 얻을 수 있다. 다음과 같이 4개의 차트를 함께 그려보자. 데이터는 data[0]과 data[1]에 각각 100개가 있으며 이들은 정규 분포를 따라 생성된 난수이다.

원하는 결과



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

np.random.seed(19680801)
data = np.random.randn(2, 100)

fig, axs = plt.subplots(2, 2, figsize=(5, 5))

axs[0, 0].hist(data[0])
axs[1, 0].scatter(data[0], data[1])
axs[0, 1].plot(data[0], data[1])
axs[1, 1].hist2d(data[0], data[1])

plt.show()
```