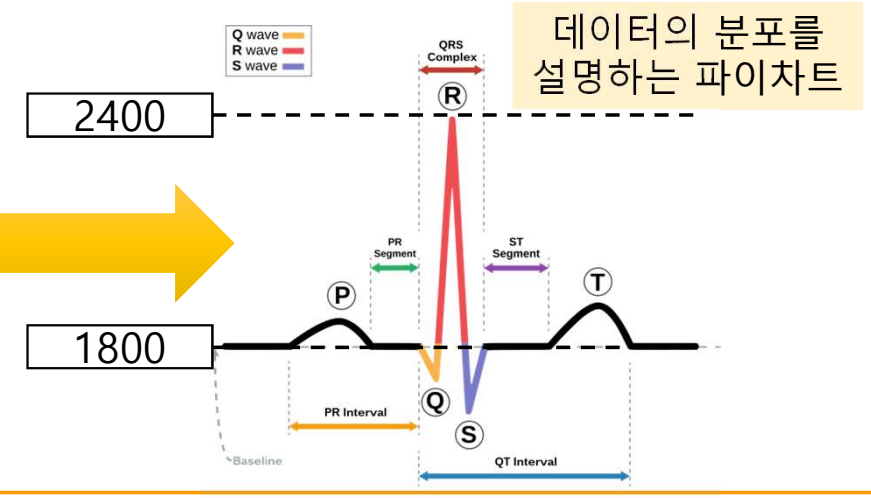
11장 차트를 멋지게 그려보자

11.1 데이터 시각화

- 데이터 시각화data visualization는 점이나 선, 막대 그래프등의 시각적 이미지를 사용하여 데이터를 화면에 표시하는 기술
- 효과적인 시각화는 사용자가 데이터를 분석하고 추론하는 데 도움이 된다. **사람들은 시각적으로 보이는 데이터를 직관적으로 이해**할 수 있기 때문이다.

1829.0	1829.0	1875.0	1875.0	1875.0	1850.0	1850.0	1843.0
1850.0	1851.0	1851.0	1864.0	1864.0	1864.0	1877.0	1877.0
1877.0	1877.0	1863.0	1863.0	1863.0	1887.0	1887.0	1884.0
1855.0	1879.0	1879.0	1879.0	1902.0	1902.0	1884.0	1884.0
1895.0	1895.0	1895.0	1913.0	1913.0	1929.0	1929.0	1929.0
1931.0	1931.0	1935.0	1935.0	1964.0	1964.0	1964.0	1986.0
1987.0	2003.0	2003.0	2027.0	2027.0	2027.0	2032.0	2032.0
2078.0	2078.0	2102.0	2102.0	2102.0	2078.0	2078.0	2092.0
2146.0	2135.0	2135.0	2135.0	2152.0	2152.0	2187.0	2187.0
2196.0	2196.0	2196.0	2195.0	2195.0	2226.0	2226.0	2226.
2237.0	2237.0	2227.0	2227.0	2274.0	2274.0	2274.0	2256.
2252.0	2275.0	2275.0	2297.0	2297.0	2297.0	2286.0	2286.
	2297.0	2307.0	2307.0	2307.0	2323.0	2323.0	2299.0
	2290.0	2290.0	2290.0	2321.0	2321.0	2303.0	2303.0
	2328.0	2328.0	2312.0	2312.0	2303.0	2303.0	2303.0
	2285.0	2307.0	2307.0	2336.0	2336.0	2336.0	2324.0
	2292.0	2292.0	2303.0	2303.0	2303.0	2279.0	2279.0
2299.0	2299.0	2310.0	2310.0	2310.0	2301.0	2301.0	2313.0
2310.0	2279.0	2279.0	2279.0	2315.0	2315.0	2302.0	2302.0
2290.0	2290.0	2231.0	2231.0	2231.0	2254.0	2254.0	2223.0
2243.0	2234.0	2234.0	2234.0	2225.0	2225.0	2234.0	2234.0
2224.0	2224.0	2224.0	2205.0	2205.0	2187.0	2187.0	2187.0

가공되지 않은 데이터

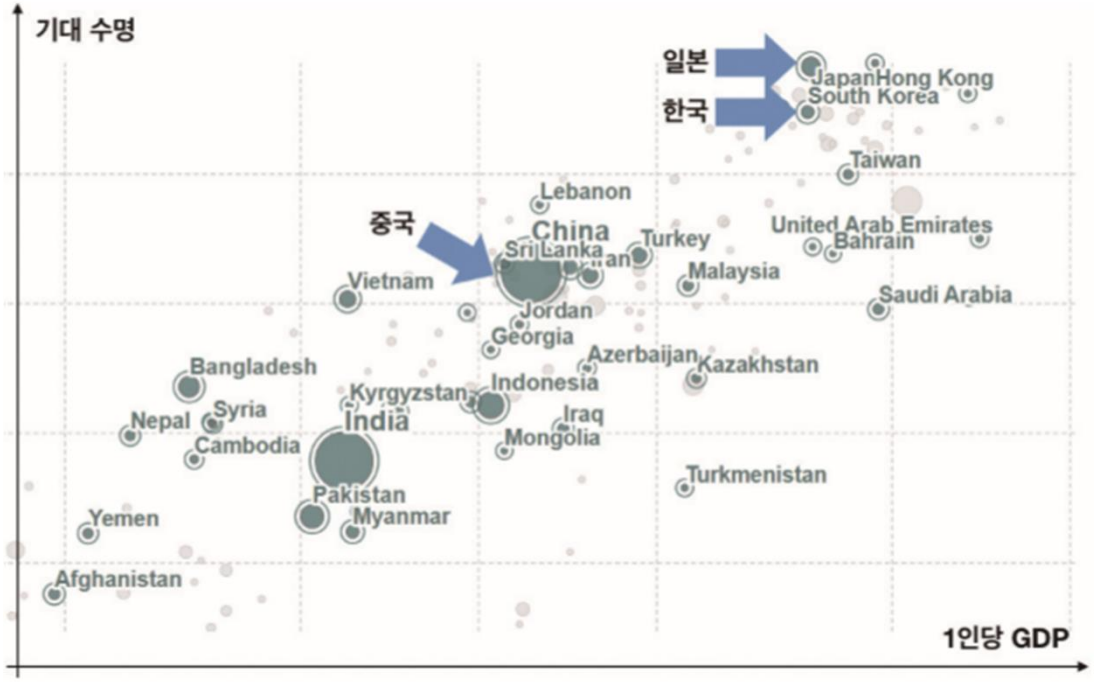


데이터의 분포를 설명하는 파이차트

1.사람은 시각화된 정보를 통해 데이터의 비교 분석이나 인과 관계 이해를 더 쉽게 할 수 있다. 데이터 시각화는 인간이 통찰을 통해 **데이터에 내재하는 패턴**을 알아내는 데도 유용하다.
2.데이터 시각화는 교육이나 홍보 등의 분야에서도 중요한 의사 소통의 도구로 간주된다.

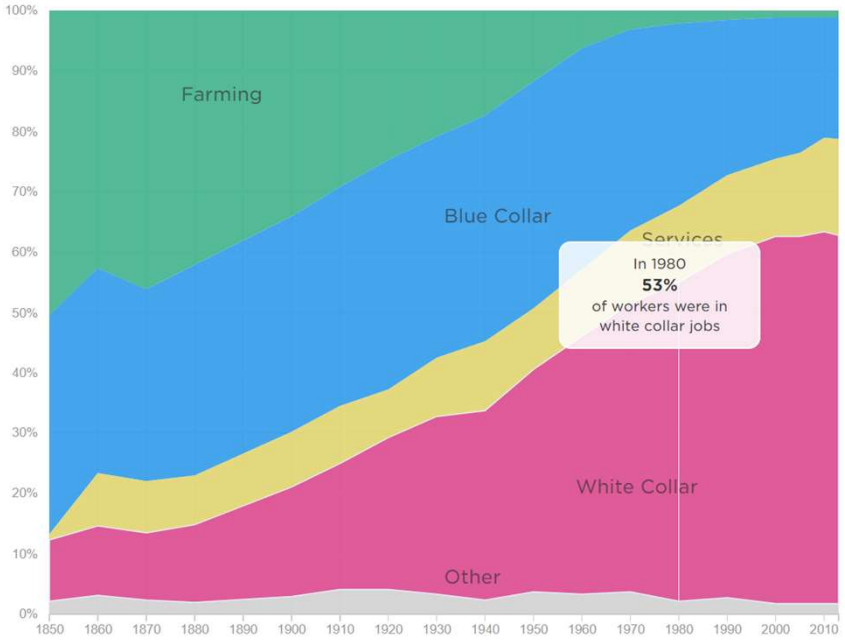
11.2 데이터 시각화를 통해 가치있는 정보를 만드는 사례

- 아래 그림은 스웨덴의 **한스 로슬링**Hans Rosling 교수가 만든 차트
- 이 차트는 GDP(국내총생산)와 기대 수명 사이의 상관 관계를 잘 보여주고 있다. 그림의 수평 축은 달러로 표시한 1인당 GDP이며 수직 축은 기대 수명



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

- 미국의 비영리 미디어 기관인 국립 공영 라디오(NPR)의 웹사이트는 다음과 같은 대화식 다이어그램을 통해서 1850년에서 부터 지금까지의 노동인구 구성의 변화를 표시하고 있다.
- 사용자는 마우스를 임의의 지점에 올릴 수 있으며 이 경우 풍선 도움말이 나타나서 사용자가 원하는 정보를 효과적으로 전달해 주고 있다.



출처 : 미 국립 공영 라디오,
https://www.npr.org/
직접 접속해 보세요

11.3 matplotlib 무작정 사용해 보기

- matplotlib은 가장 널리 사용되는 시각화 도구이다.
- 간단한 막대 그래프, 선 그래프, 산포도를 그리는 용도로는 matplotlib가 제격이다.

```
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()
```

이 코드의 실행 결과는?

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

11.4 맷플롯립 코드 살펴 보기

- 첫 번째 단계는 pyplot 모듈을 불러와서 plt라는 별칭으로 지정하는 것

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

- 연도별 GDP의 변화
  - 연도는 years 리스트에 담고, x축 데이터로 사용
  - GDP 값은 gdp라는 이름의 리스트를 만들어 데이터를 담게 하고, 이 값이 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]
```

- 선형 차트를 만들려면 plt의 plot() 함수를 호출. plot()은 x축 데이터와 y축 데이터를 인수로 받음
- 선의 색, 마크의 표시방법, 선의 두께 등을 키워드 인자로 줄 수 있다.

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

11.4 맷플롯립 코드 살펴 보기

- 차트의 제목 레이블을 설정한다. 차트의 최상단에 표시된다

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

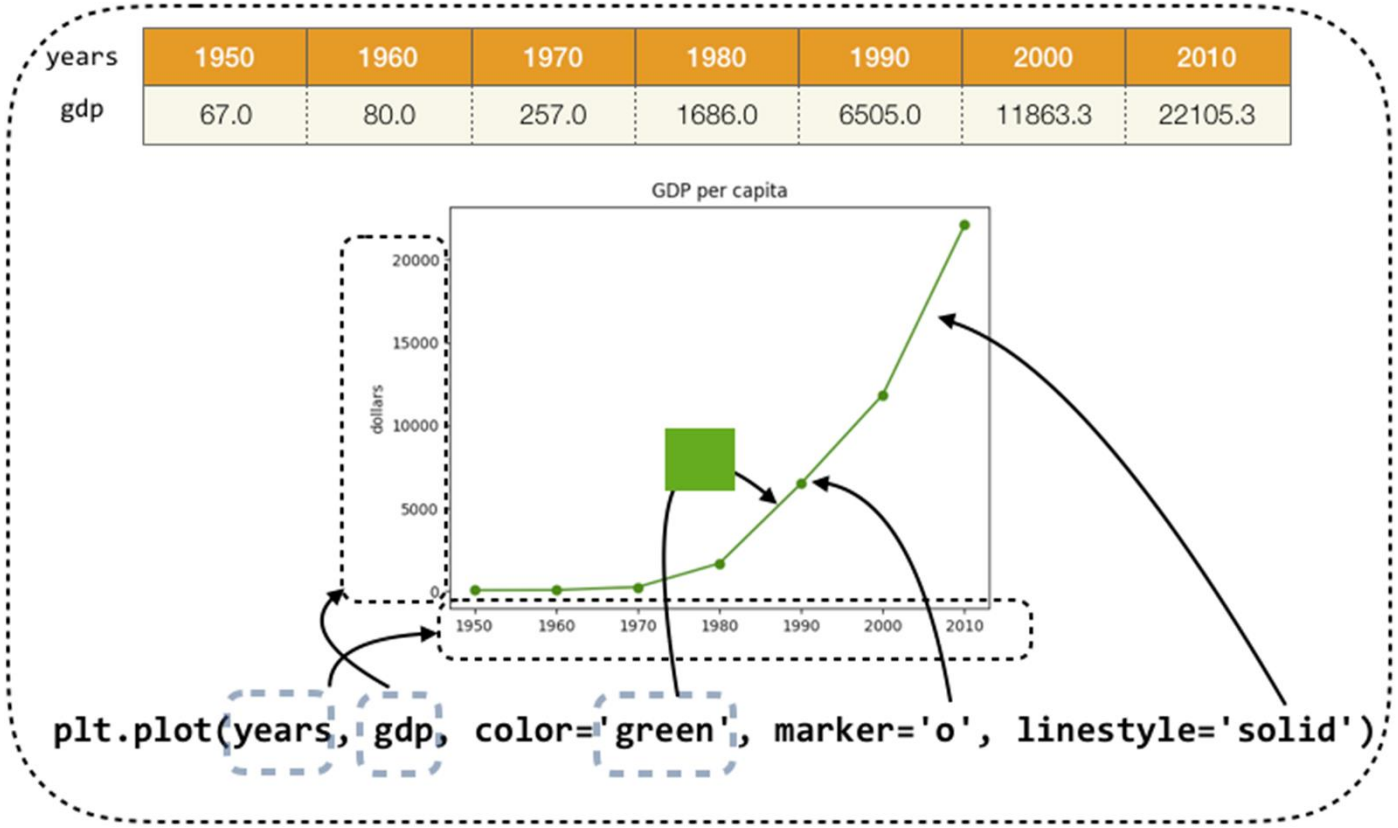
- y축의 레이블을 지정, savefig() 함수를 통해서 파일을 저장함. 해상도는 dpi를 통해 지정
  - dpi는 dot per inch로 1인치(약 2.54cm)당 점의 수를 의미함
- 반드시 plt.show()함수를 사용해야 차트가 화면에 출력된다.

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

```python
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)
plt.show()
```

GDP

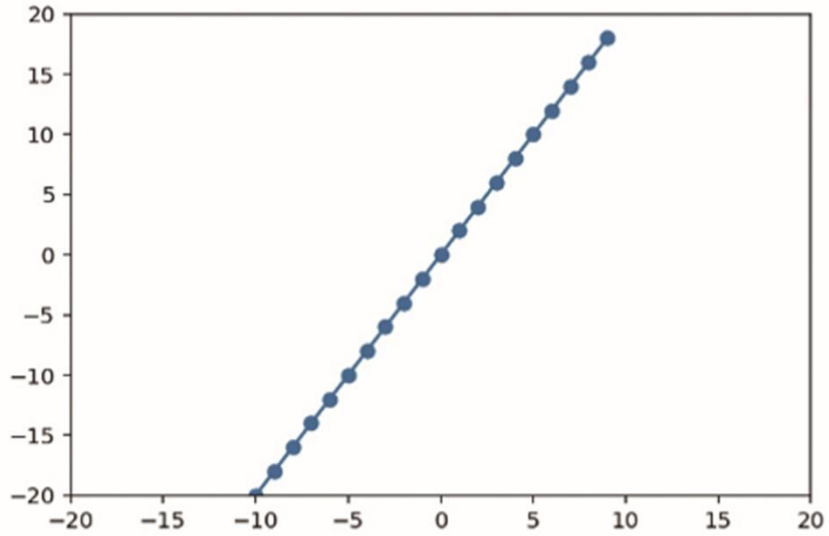
years	1950	1960	1970	1980	1990	2000	2010
gdp	67.0	80.0	257.0	1686.0	6505.0	11863.3	22105.3

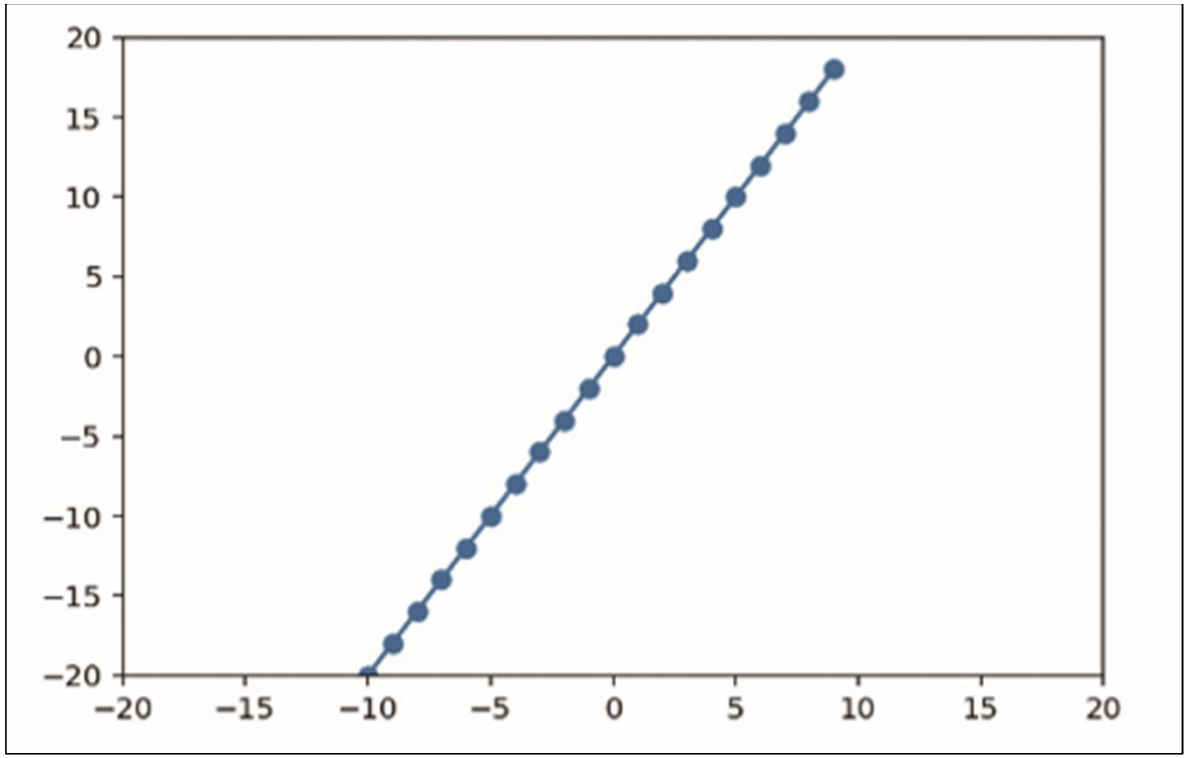


LAB 11-1 수학 함수도 쉽게 그려보자

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





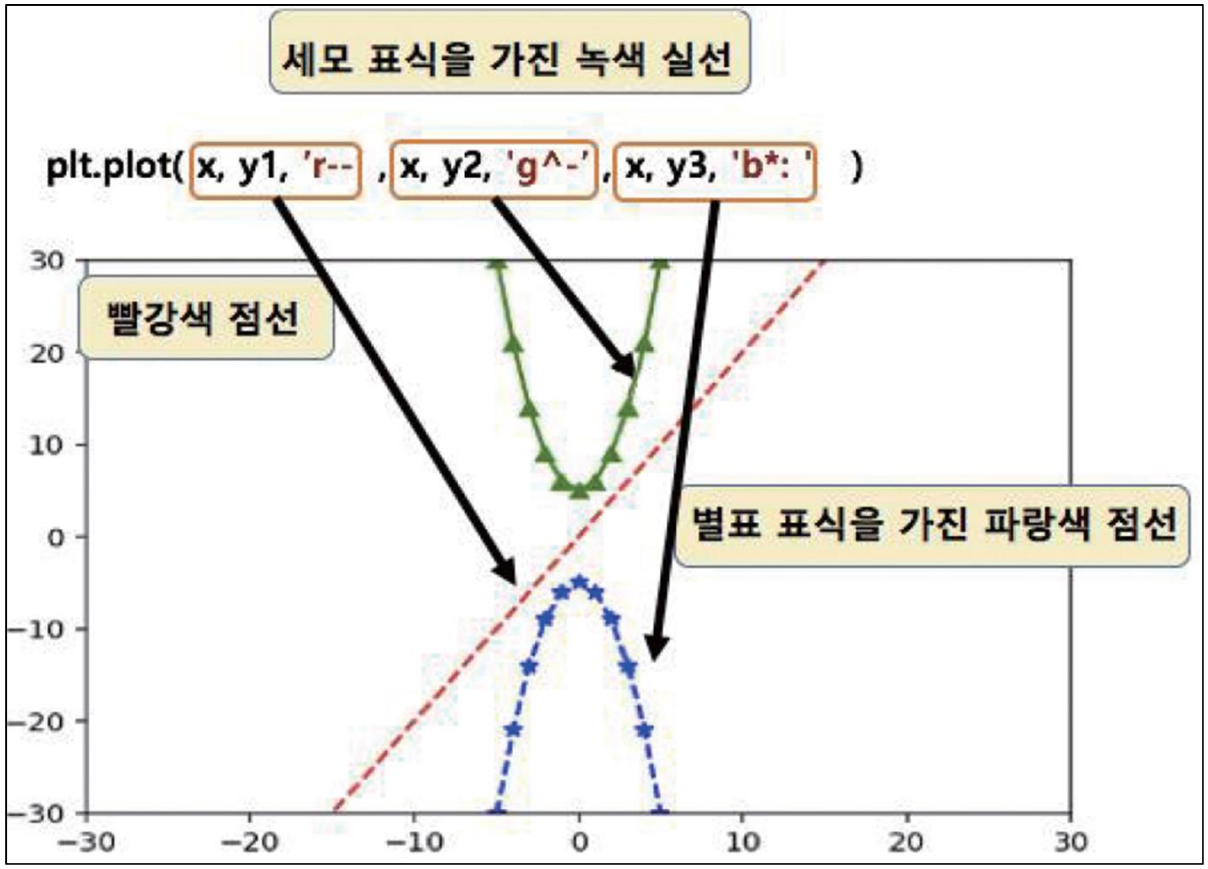

```
import matplotlib.pyplot as plt

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()
```

11.5 차트 장식을 도와주는 다양한 기법들

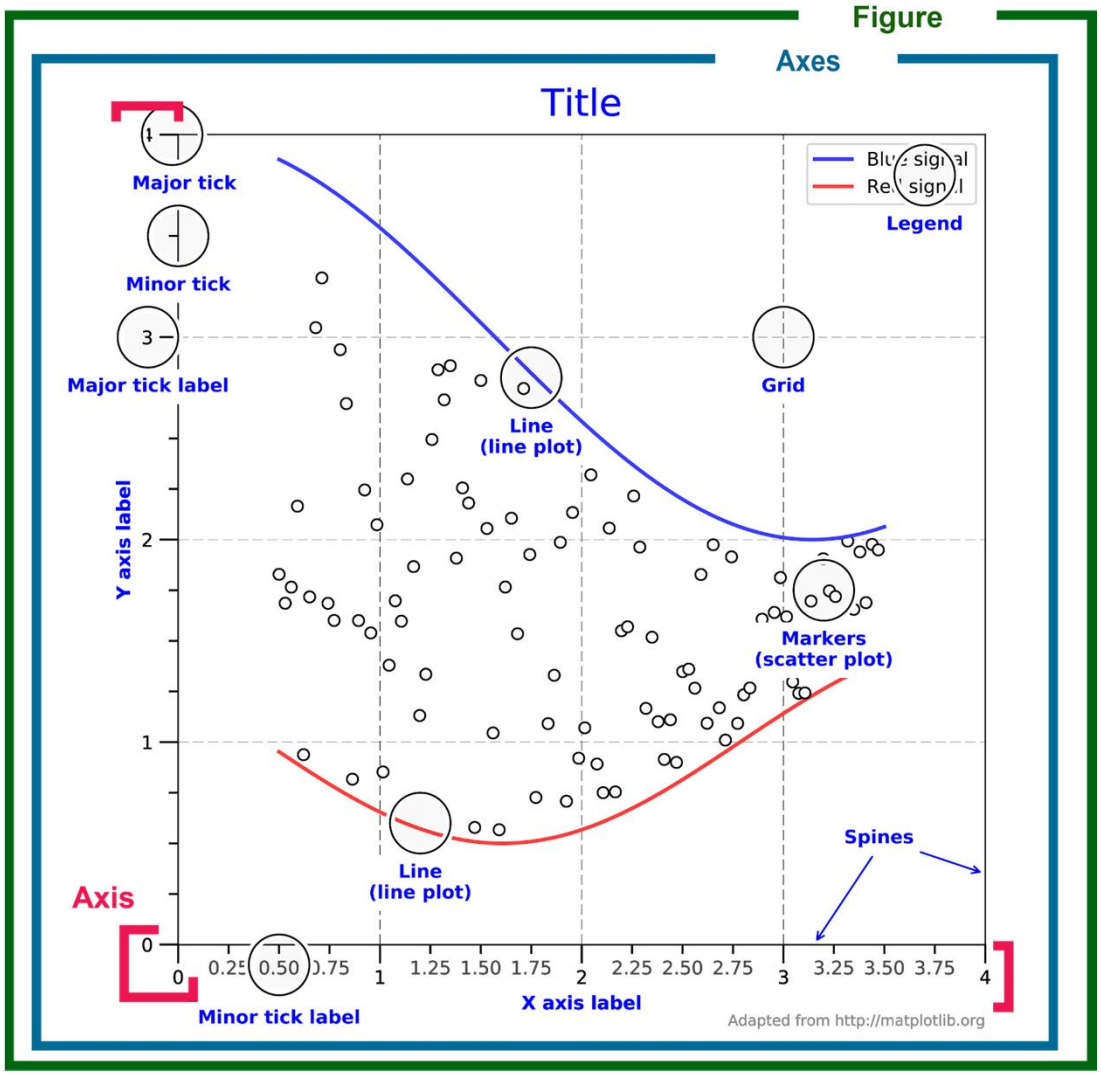
- plot() 함수를 수정하여 한 번의 호출만으로 $2x$, x^2+5, $-x^2-5$ 의 세 함수를 그려보도록 하자.

```
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()
```

- ‘r--’나 ‘g^-’, ‘b*:’와 같은 포맷 지정명령을 통해 선의 색깔, 표식의 종류, 선의 형태를 지정할 수도 있다.
- r은 빨강, g는 녹색, b는 파랑을 의미한다.
- --는 점선, -는 실선, :는 짧은 점선을 의미한다.
- 표식 기호 ^는 세모, *는 별표 표식을 각각 의미



matplotlib에서 차트를 구성하는 여러 가지 용어들



- **Figure**: 그림이 그려질 영역을 의미
- **Axes**: 그림 안에 작은 그림을 의미 (여러 개 가능)
- **Title:** 차트의 제목을 의미
- **Legend:** 차트의 정보를 담은 범례를 의미
- **Grid:** 차트의 격자무늬를 의미
- **Major tick:** 차트의 큰 눈금을 의미
- **Major tick label:** 차트 큰 눈금의 값을 의미
- **Minor tick:** 차트의 작은 눈금을 의미
- **Minor tick label:** 차트 작은 눈금 값을 의미
- **Axis:** 차트의 축을 의미(x, y 등)
- **X axis label**: x 축의 이름을 의미
- **Y axis label**: y 축의 이름을 의미
- **Spines:** 차트의 축의 모양을 의미 (테두리)
- **Line:** 선 그래프의 선을 의미
- **Markers:** scatter 그래프의 마커를 의미

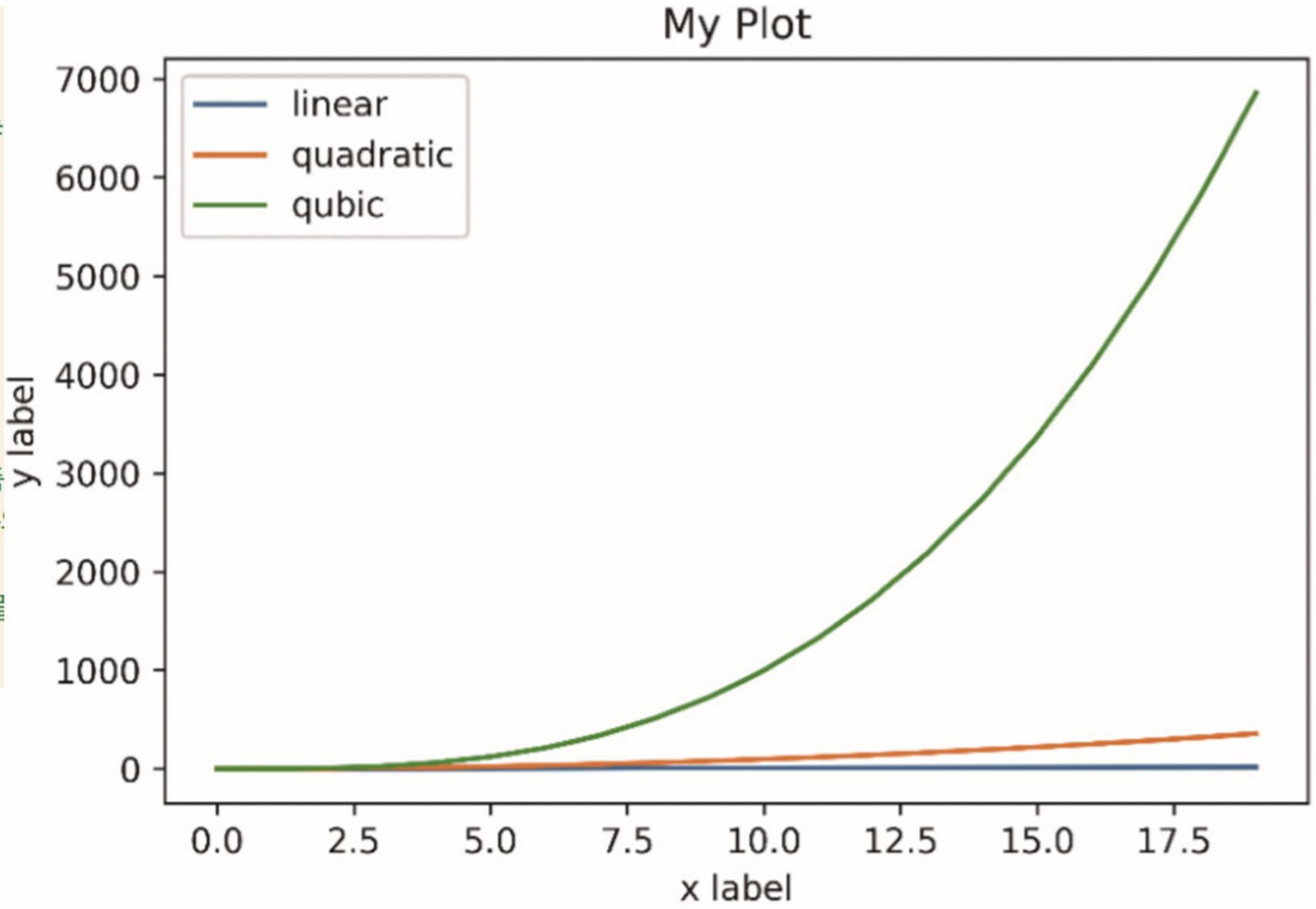
11.6 하나의 차트에 여러 개의 데이터를 그려보자

- 리스트 x를 다음과 같이 선언하고 이를 리스트 x, y, z와 함께 그려보자.
- 여기에 각 차트를 설명하는 범례(legend)를 추가하면 다음과 같다.

```
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, y, label='cubic')

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()
```

코드의 실행 결과는?

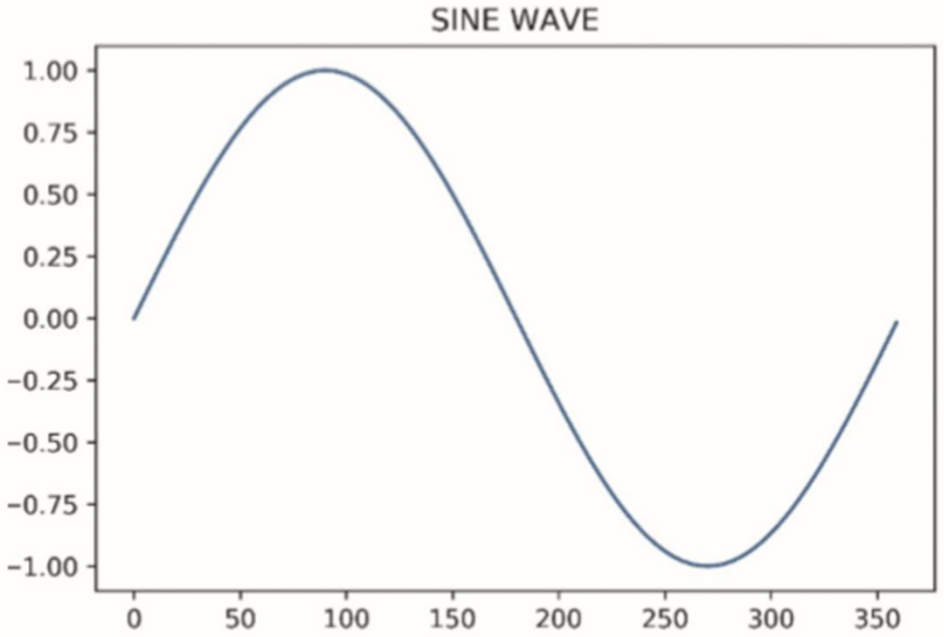
```
import matplotlib.pyplot as plt

x = [x for x in range(20)]      #
y = [x**2 for x in range(20)]   #
z = [x**3 for x in range(20)]   #

plt.plot(x, x, label='linear')
plt.plot(x, y, label='quadratic')
plt.plot(x, y, label='cubic')

plt.xlabel('x label')       # x 축
plt.ylabel('y label')       # y 축
plt.title("My Plot")
plt.legend()                 # 디폴
plt.show()
```



LAB 11-2 삼각함수의 기본인 사인 그래프 그리기

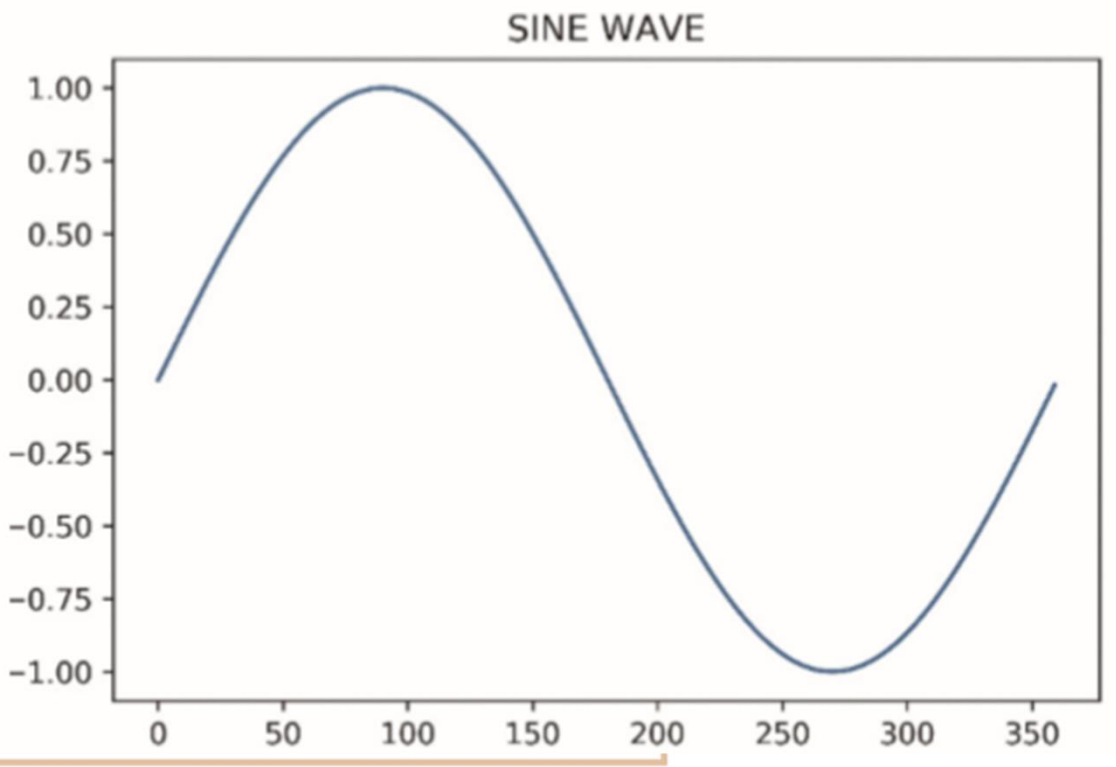
다음과 같은 사인 그래프를 그려보자. 사인값은 math 모듈의 sin() 함수로 계산할 수 있다.

원하는 결과



LAB 11-2 삼각함수의 기본인 사인 그래프 그리기

```
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()
```



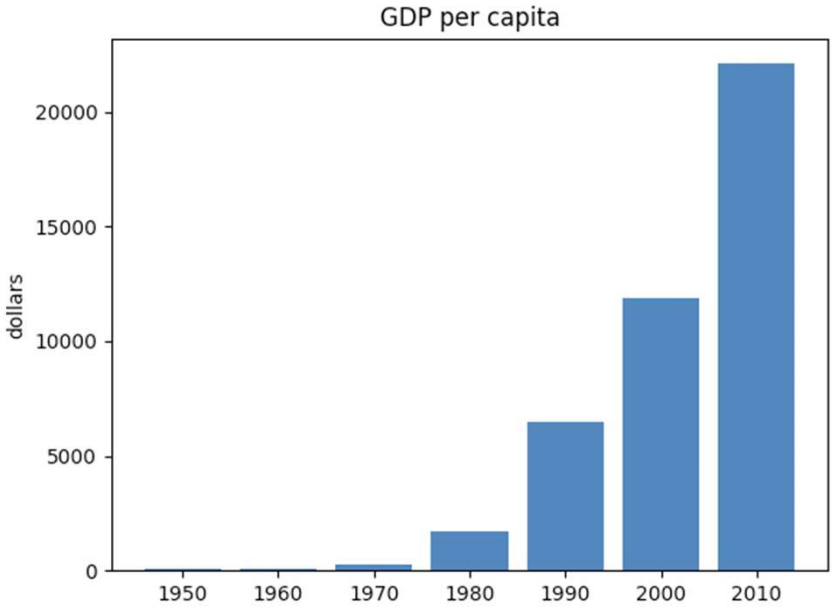
도전문제 11.3

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

11.7 막대형 차트도 손쉽게 그려보자

- plt.bar() 함수를 호출하여 화면에 막대형 차트를 그릴 수 있다.
- 앞의 1인당 국민소득을 막대 그래프로 그려보자.

```
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.

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

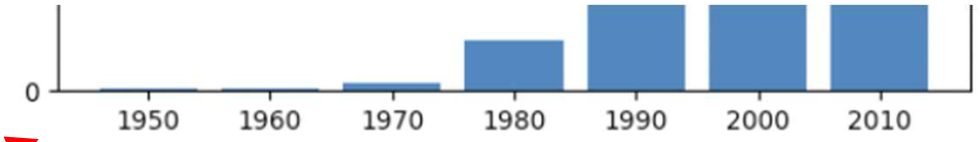


11.7 막대형 차트도 손쉽게 그려보자

- 아래의 range() 함수는 정수의 범위를 생성하는 함수이다.
- 이 경우 years 리스트가 7 개의 항목을 가지고 있으므로 0에서 6까지의 정수 범위를 만든다.
- 이 범위가 막대형 차트의 가로축 범위가 되는 것이다.

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

- xtick() 함수를 사용하여 가로축 범위의 눈금마다 부여할 눈금값을 지정
- 아래와 같이 years 리스트의 항목들을 사용




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

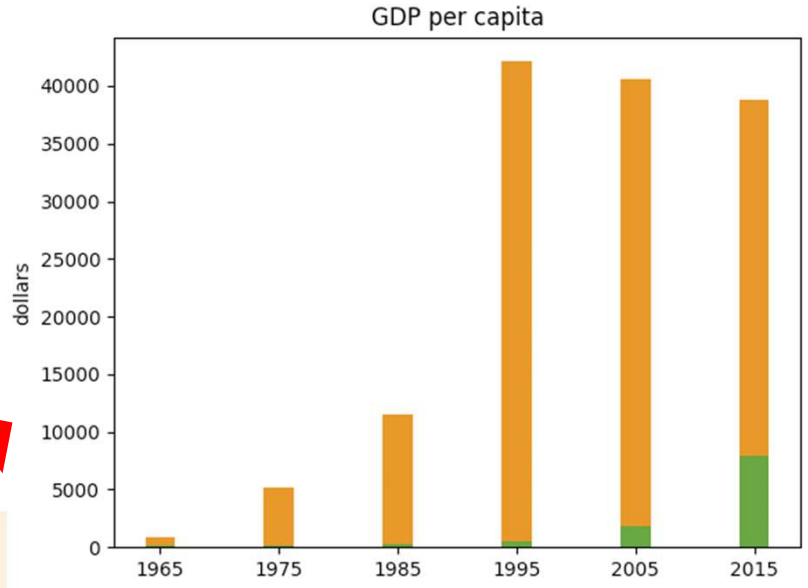
11.8 여러나라의 국민소득 추이를 다중 막대형 차트로 그리자

- 한국$_{kr}$뿐만 아니라 일본$_{jp}$, 중국$_{ch}$의 국민소득 추이를 다중 막대형 차트로 그려보자
- 어떤 방법을 사용할까?
- 아래와 같이 데이터 리스트를 먼저 만들자

```
# 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 줄여보자. 그리고 가로축의 범위는 세 개의 막대형 차트가 공유해야 하는 것이므로 아래와 같이 변수에 담아 공유해서 사용하자

```
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()
```

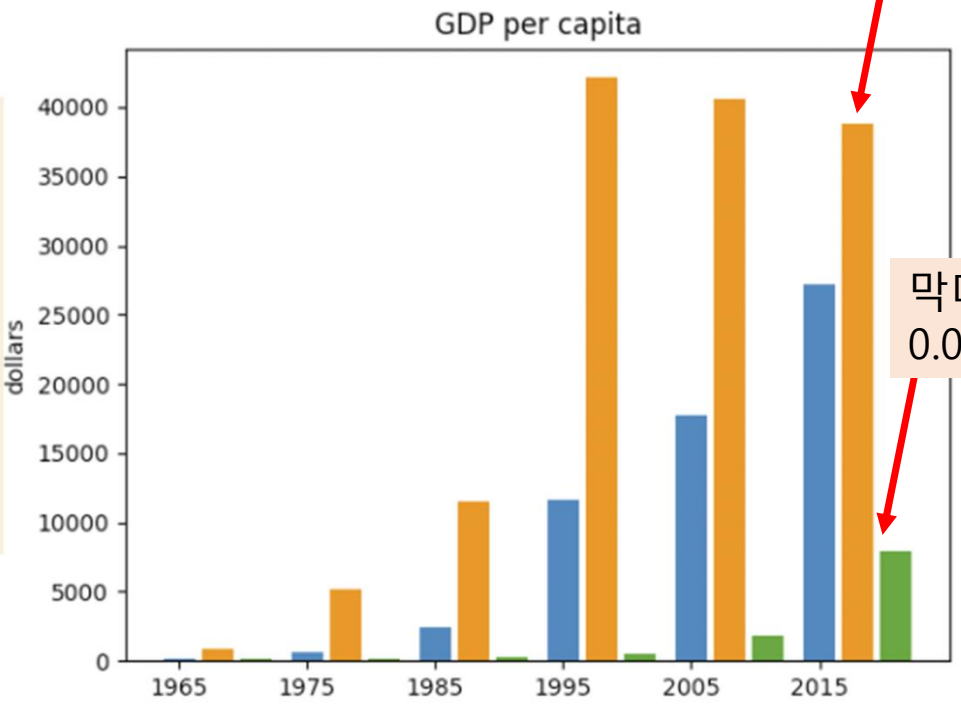


11.8 여러나라의 국민소득 추이를 다중 막대형 차트로 그리자

- 이전 페이지의 표는 알아보기가 힘든 단점이 있다.
- 각 막대의 가로축 범위를 조금씩 차이를 두는 방법을 사용
- 그런데 정수 범위를 생성하는 range() 함수의 결과에 실수값을 더하는 것은 불가능
- 실수 범위를 생성하는 넘파이의 범위로 변경하여 다음과 같이 완성

```
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()
```

막대의 너비는 0.25

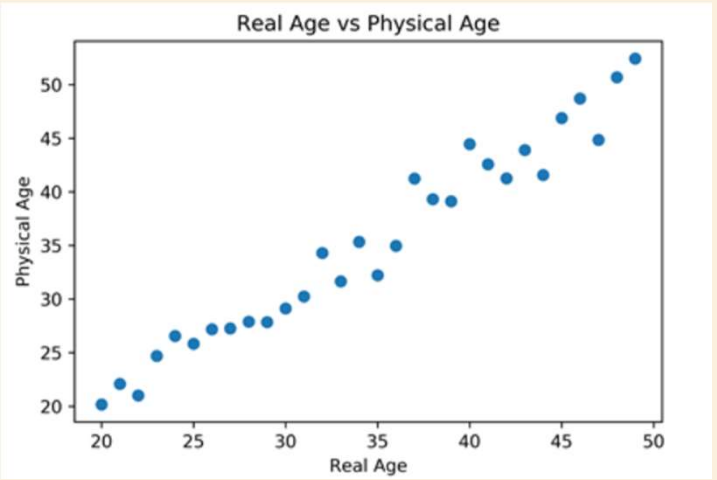


막대간 간격은 0.05

11.9 데이터를 점으로 표현하는 산포도 그래프 그리기

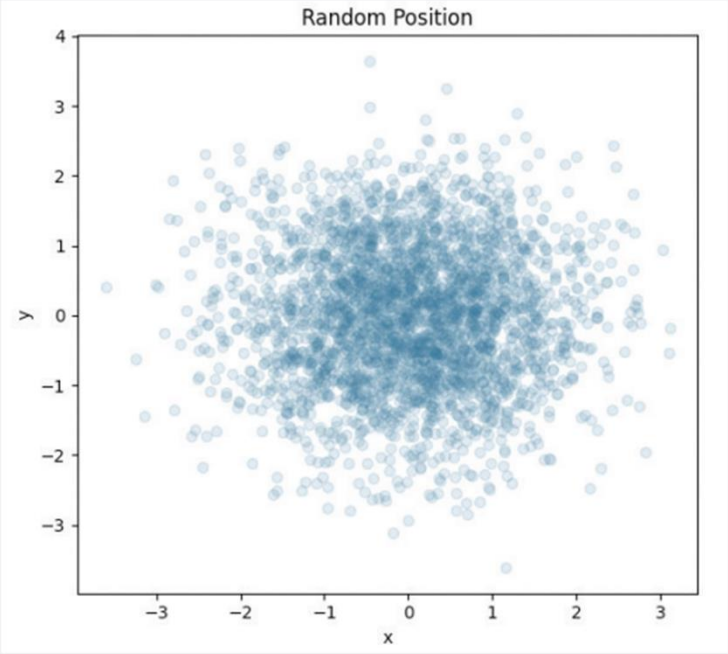
- **산포도 플롯**scatter plot은 개별 데이터 포인트를 그리는 차트. 산포도를 그릴 때는 각 데이터 포인트가 연결되지 않는다. 산포도 그래프를 그릴 때는 scatter() 함수를 사용함

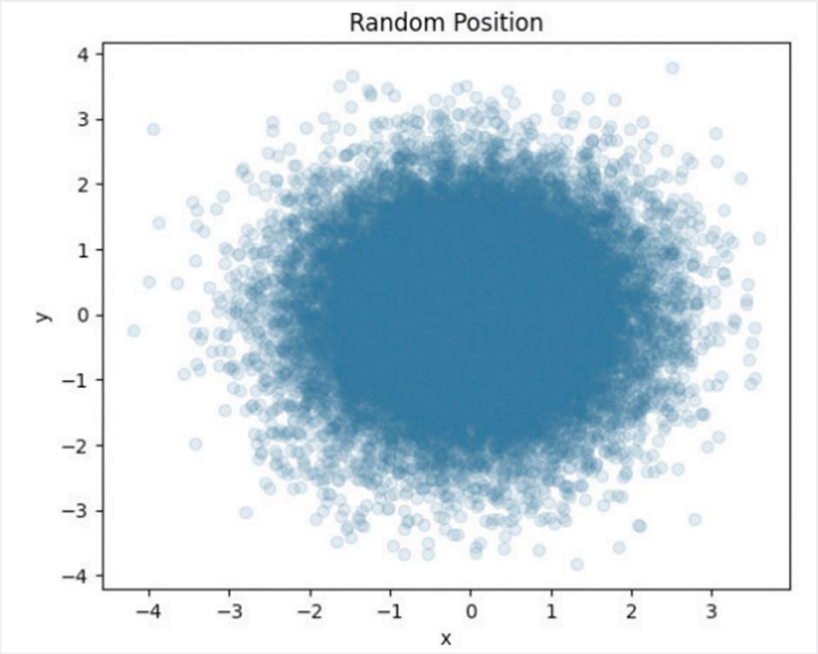
```
import matplotlib.pyplot as plt
import numpy as np

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.





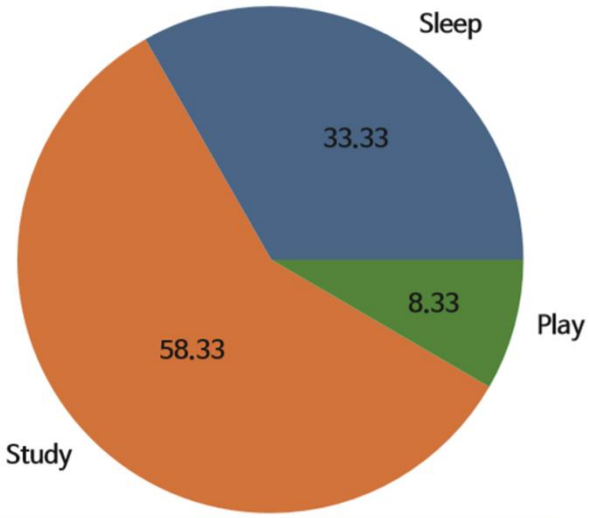
11.10 맛있는 피자가 생각나는 파이 차트

- **파이 차트**pie chart는 데이터의 값에 따라서 원형 비율로 나누어져 있는 차트
- 파이 차트는 비즈니스 세계에서 널리 사용(예들들어, 상품의 시장 점유율을 표시)

```
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()
```



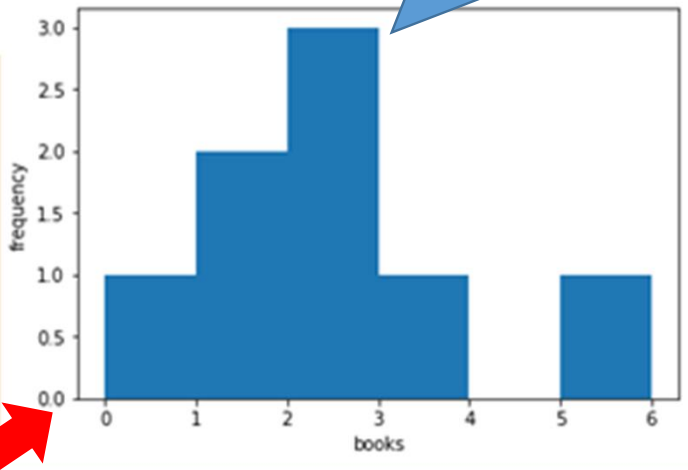
11.11 히스토그램으로 자료의 분포를 한눈에 보아요

- 히스토그램은 주어진 자료를 몇 개의 구간으로 나누고 각 구간의 도수frequency를 조사하여 나타낸 막대 그래프이다

학생 8명이 1년동안 읽은 책의 수

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

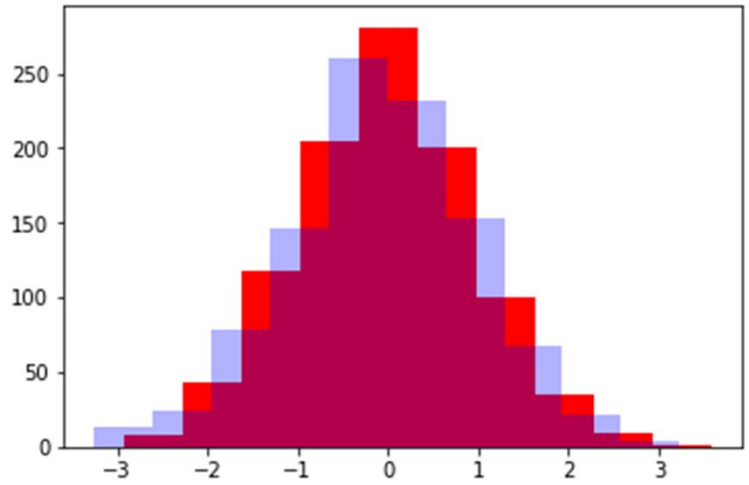
데이터의 분포를 설명하는 차트.




```
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()
```

데이터가 들어가는 통을 bin이라 하자.

11.12 겹쳐진 히스토그램도 그리자 : 다중 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()
```

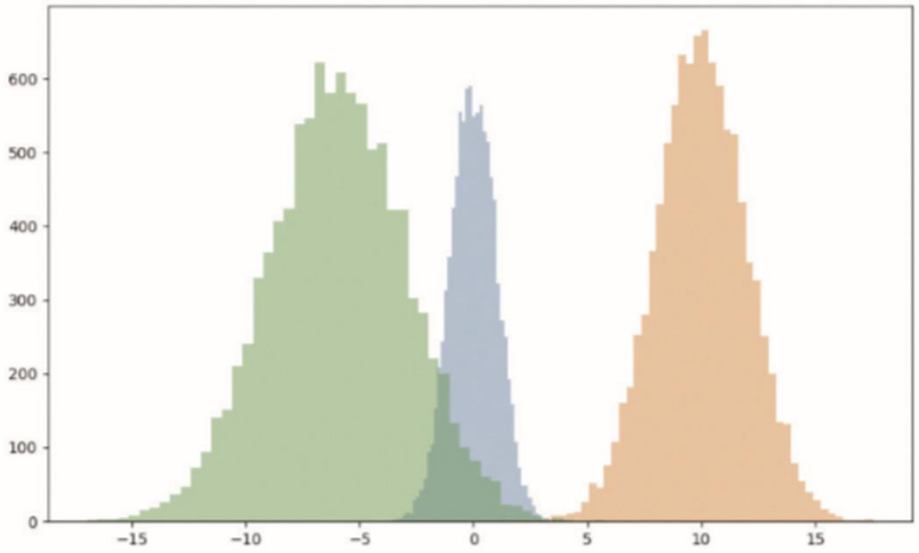
10개의 구간으로 나누어서 분포를 그림

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

LAB11-3 정규분포로 생성된 난수를 눈으로 확인하기

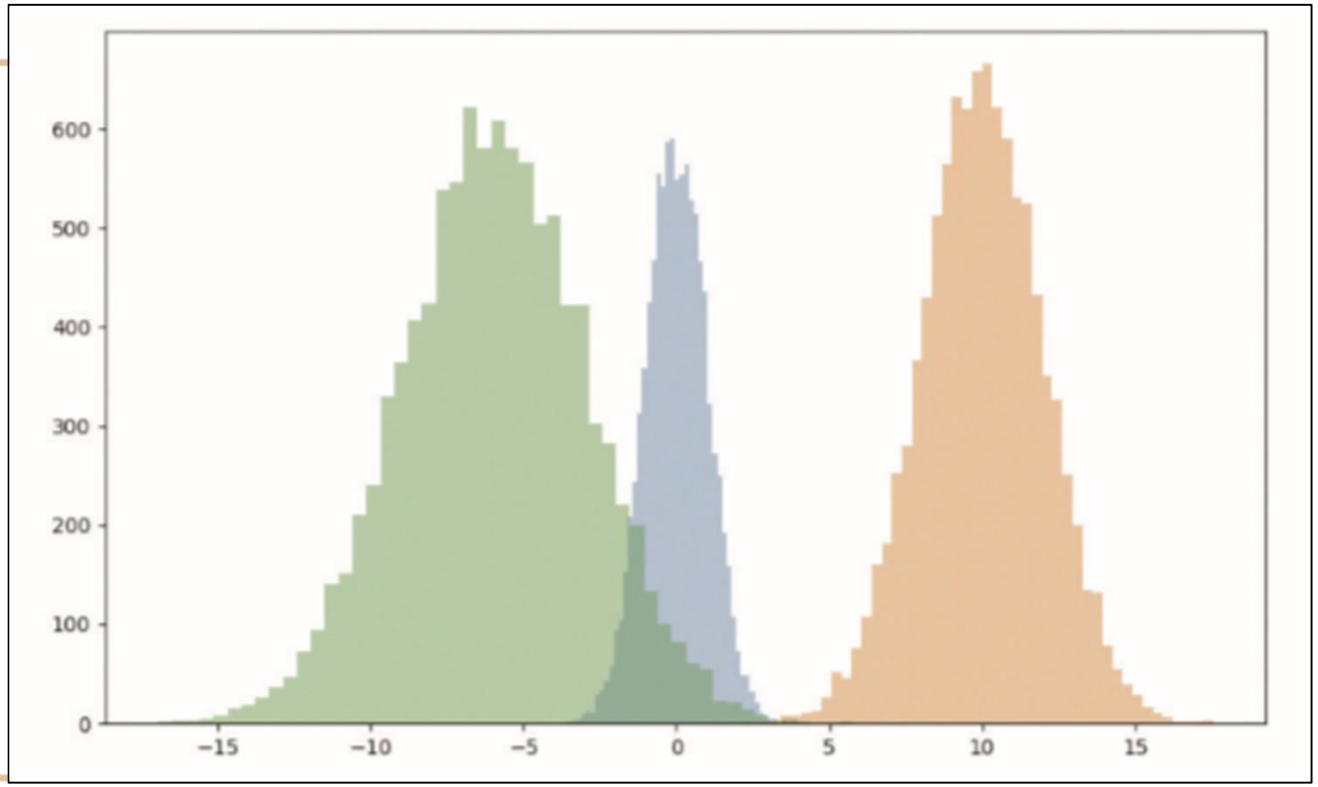
정규 분포로 난수를 생성하고 이 수들이 어떠한 방식으로 흩어져 있는지를 확인해 보고자 한다. 표준 정규 분포로 난수를 하나 생성해 보고, 다음으로는 평균 10, 표준편차 2인 정규분포로 난수를 생성해 보라. 그리고 마지막으로 평균 -6, 표준편차 3인 정규분포로 난수를 또 생성한다. 그리고 이렇게 생성된 난수들이 어떤 분포를 보이는지 히스토그램을 통해 확인해 보자.

원하는 결과



LAB11-3정규분포로 생성된 난수를 눈으로 확인하기

```
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()
```

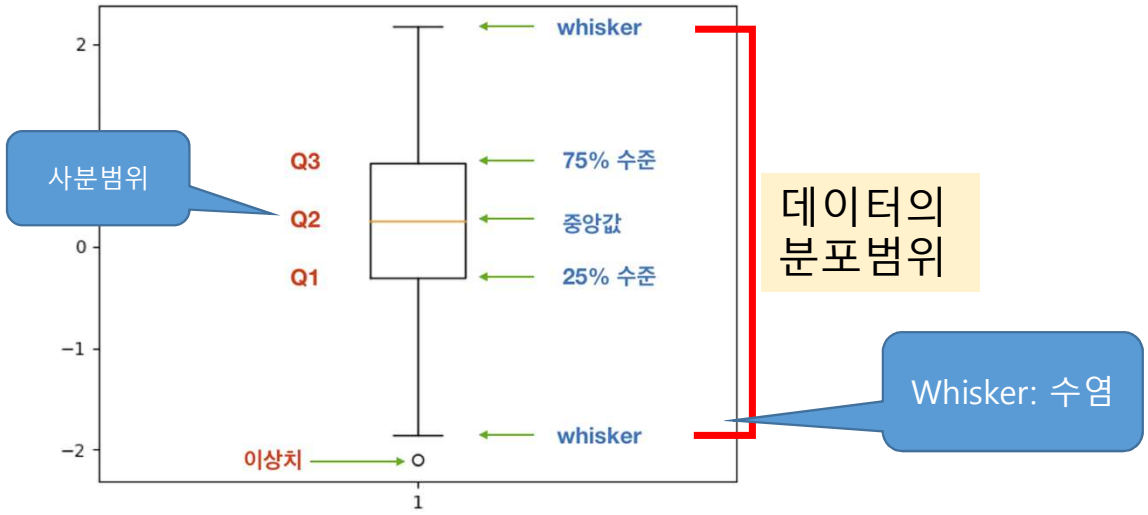


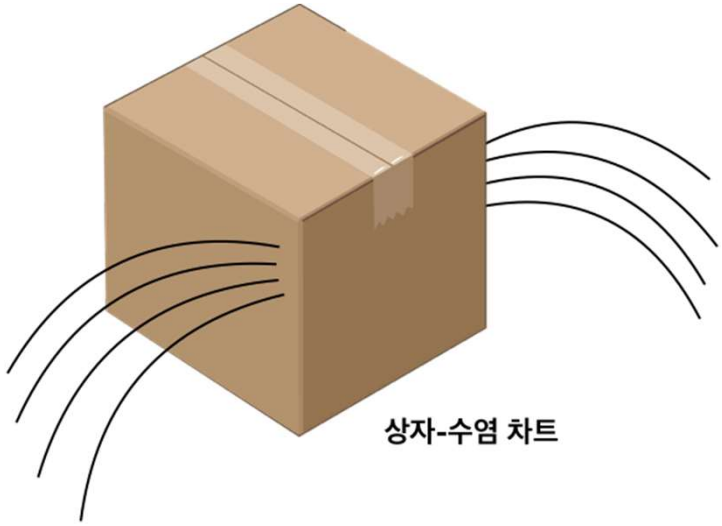
11.13 데이터를 효율적으로 표현하는 상자 차트를 알아보자

- 상자 차트box plot은 데이터의 최대, 최소, 중간값과 사분위 수 등을 효율적으로 가시화 할 수 있는 방법이다.

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

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

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



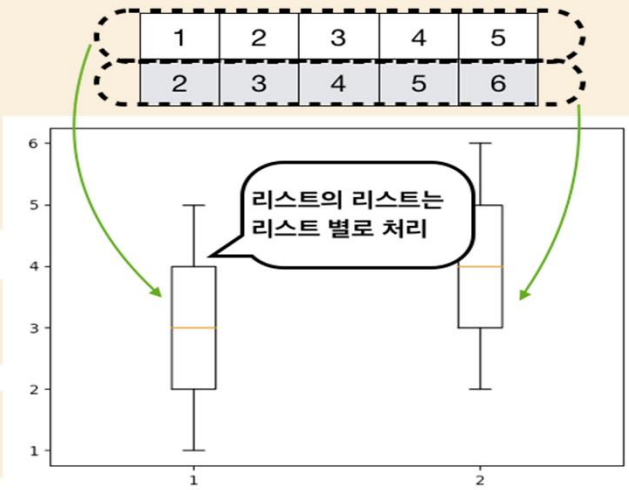


11.14 여러 개의 상자 차트 그리기

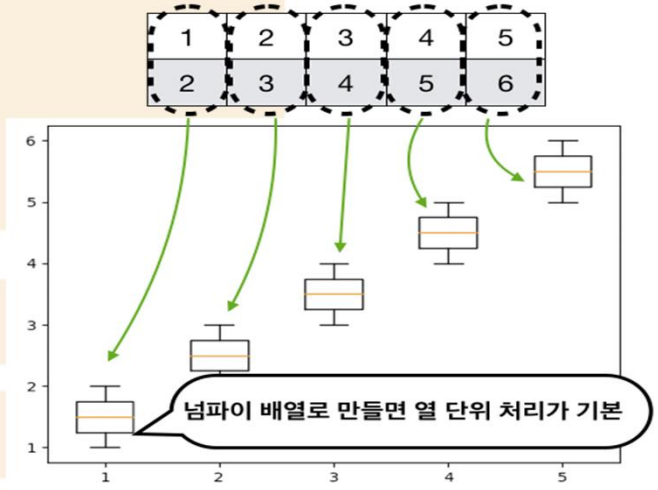
```
import numpy as np
import matplotlib.pyplot as plt
data1 = [1, 2, 3, 4, 5]
data2 = [2, 3, 4, 5, 6]
```

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

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

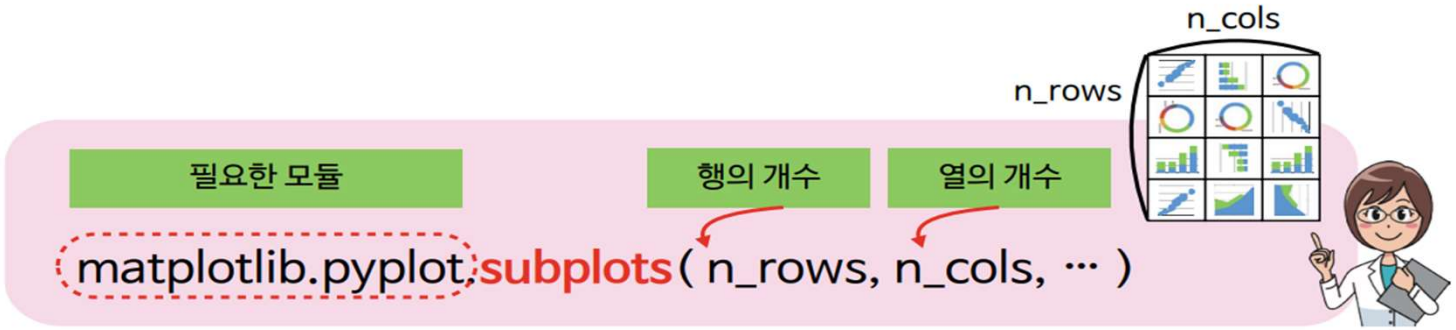


boxplot([data1, data2])

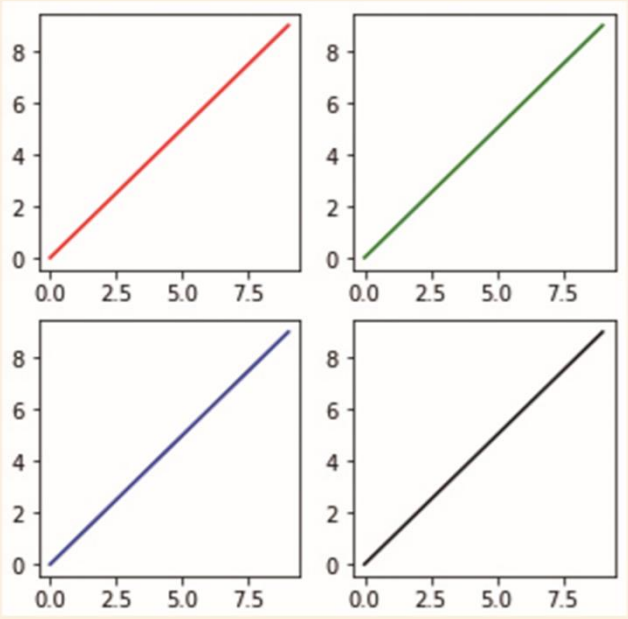


boxplot(np.array([data1, data2]))

11.15 한 화면에 여러 그래프 그리기 : subplots()



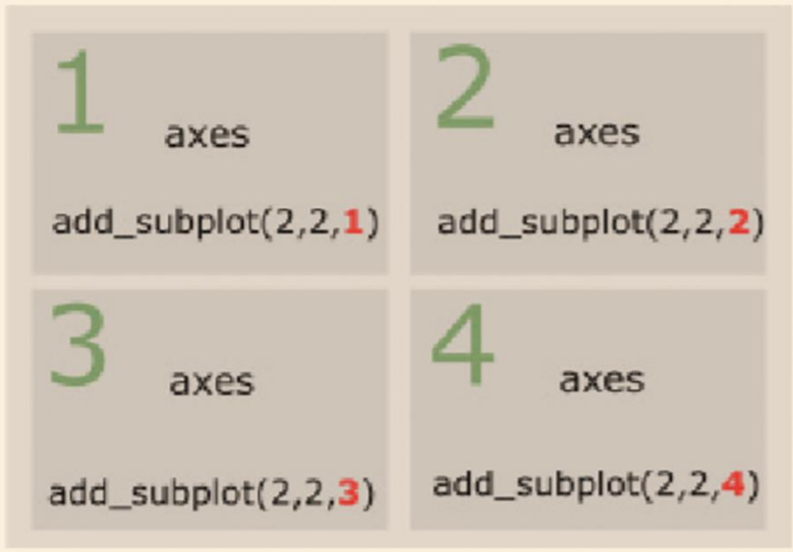

```
import matplotlib.pyplot as plt

# 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()
```



11.15 한 화면에 여러 그래프 그리기 : subplots()

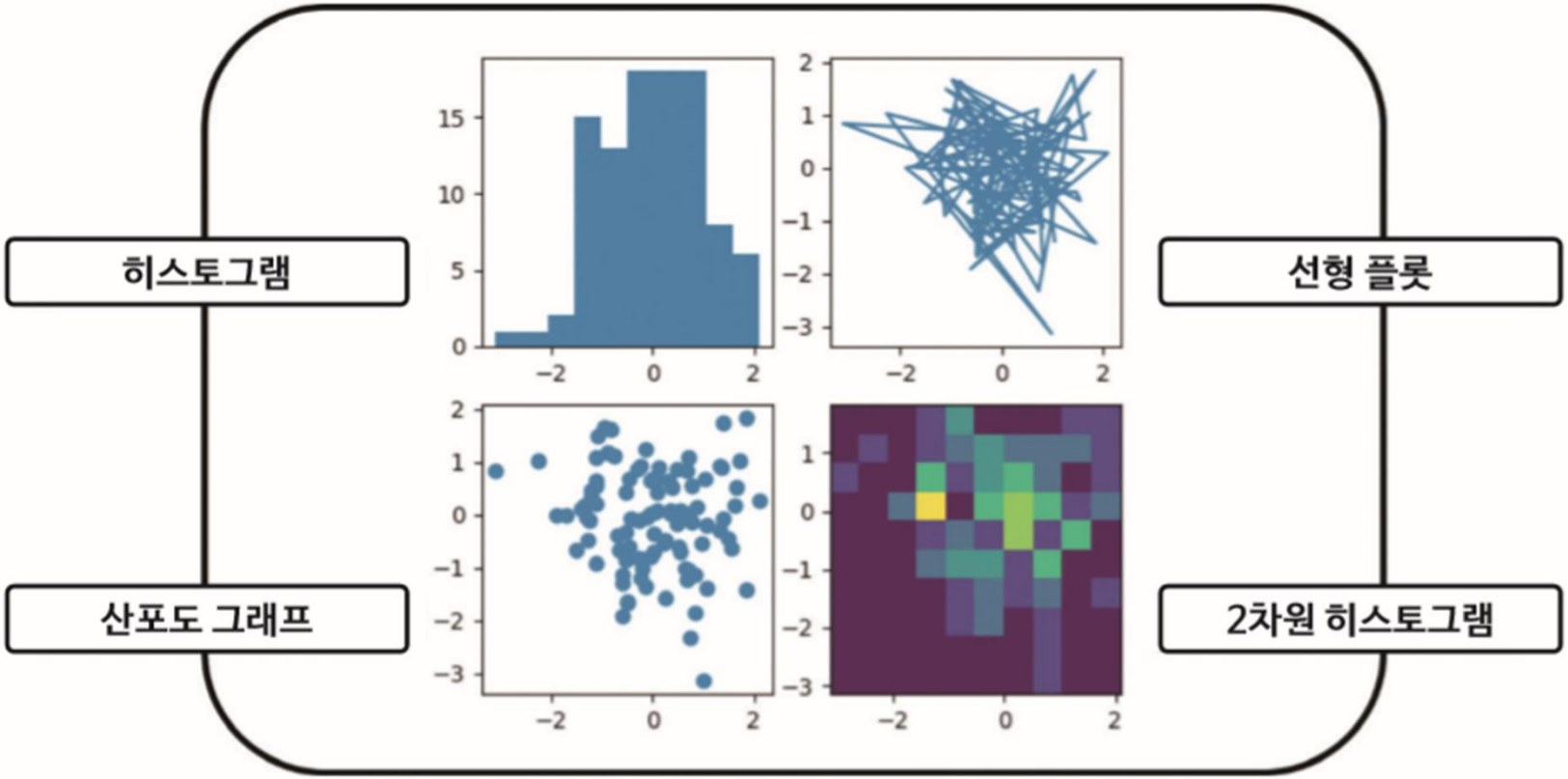
```
fig = plt.figure()
ax1 = fig.add_subplot(2, 2, 1)
ax2 = fig.add_subplot(2, 2, 2)
ax3 = fig.add_subplot(2, 2, 3)
ax4 = fig.add_subplot(2, 2, 4)
```

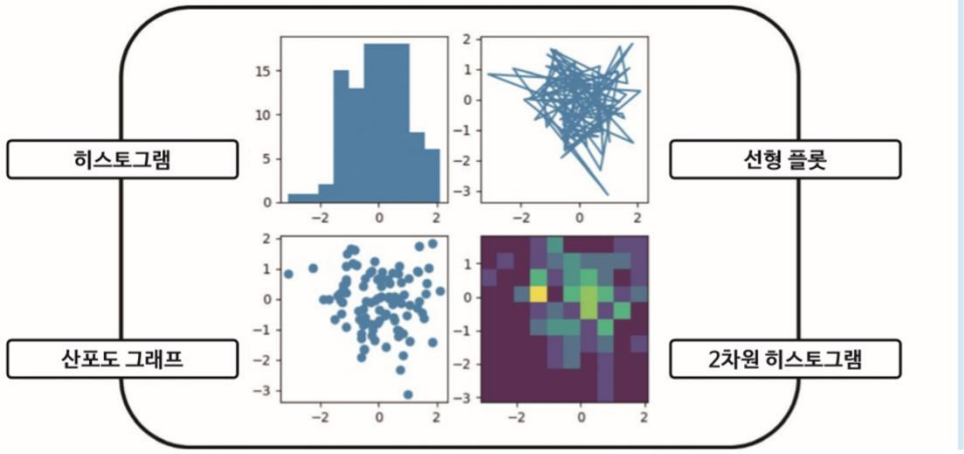
다른 방법으로 부분 그래프를 그려볼 수 있어요



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()
```