

시각화 도구 - Matplotlib

CONTENTS

- 1 데이터 시각화
 - 2 matplotlib 무작정 사용해 보기
 - 3 차트 장식을 도와주는 다양한 기법들
 - LAB 1 수학 함수도 쉽게 그려보자
 - 4 하나의 차트에 여러 개의 데이터를 그려보자
 - LAB 2 삼각함수의 기본인 사인 그래프 그리기
 - 5 막대형 차트도 손쉽게 그려보자
 - 6 데이터를 점으로 표현하는 산포도 그래프 그리기
 - 7 히스토그램으로 자료의 분포를 한눈에 살펴보자
 - LAB 3 정규분포로 생성된 난수를 눈으로 확인하기
 - LAB 4 차종별 판매량을 파이 차트로 표현하자
 - 8 데이터를 효율적으로 표현하는 상자 차트를 알아보자
 - 9 한 화면에 여러 그래프 그리기 : subplots()
 - LAB 5 서브플롯 이용해 보기
-

선형 그래프

막대형 그래프

산포도

히스토 그램

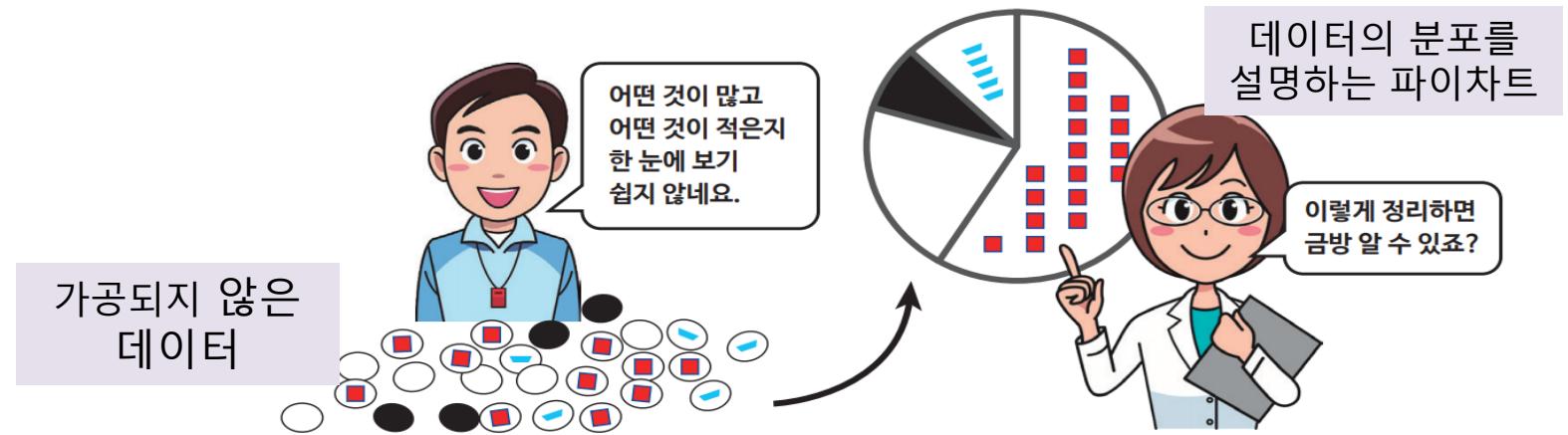
상자 차트

이 장에서 배울 것들

- 수치정보를 시각적인 정보로 표현해 봅시다.
- 맷플롯립 패키지의 다양한 기능을 알아보도록 하자.
- 데이터를 시각화하는 다양한 방법을 살펴봅시다.
- 하나의 차트에 여러 시각화 방법을 함께 사용할 수 있도록 연습해 봅시다.
- 데이터 범위와 중앙값을 효율적으로 가시화하고 이상치도 파악해 봅시다.
- 가시화를 통해 데이터들을 상호 비교하는 방법을 알아봅시다.
데이터 분석의 결과를 효율적으로 전달하는 연습을 해 봅시다.

1 데이터 시각화

- 데이터 시각화 **data visualization**는 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 이해할 수 있기 때문이다.**



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

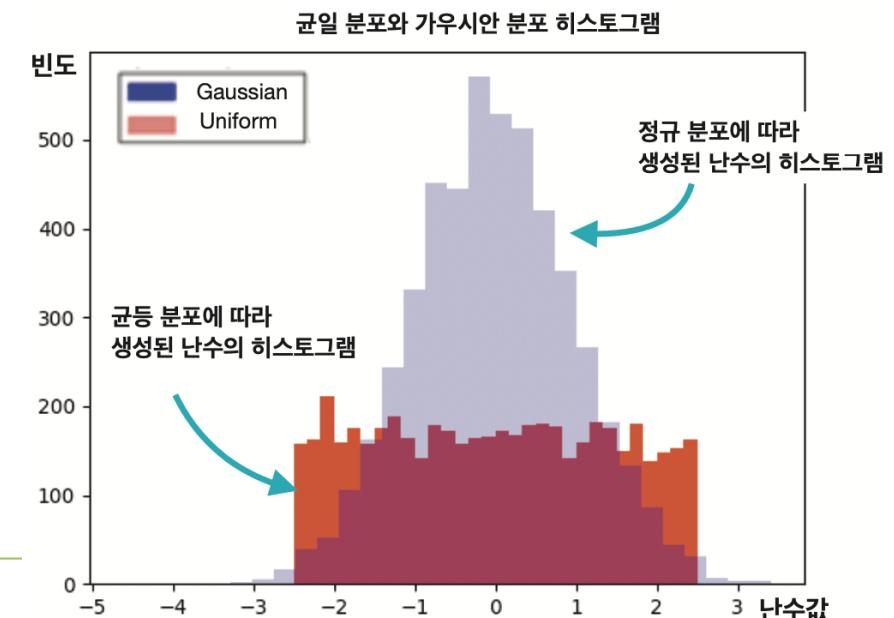
1 데이터 시각화



Uniform

Gaussian

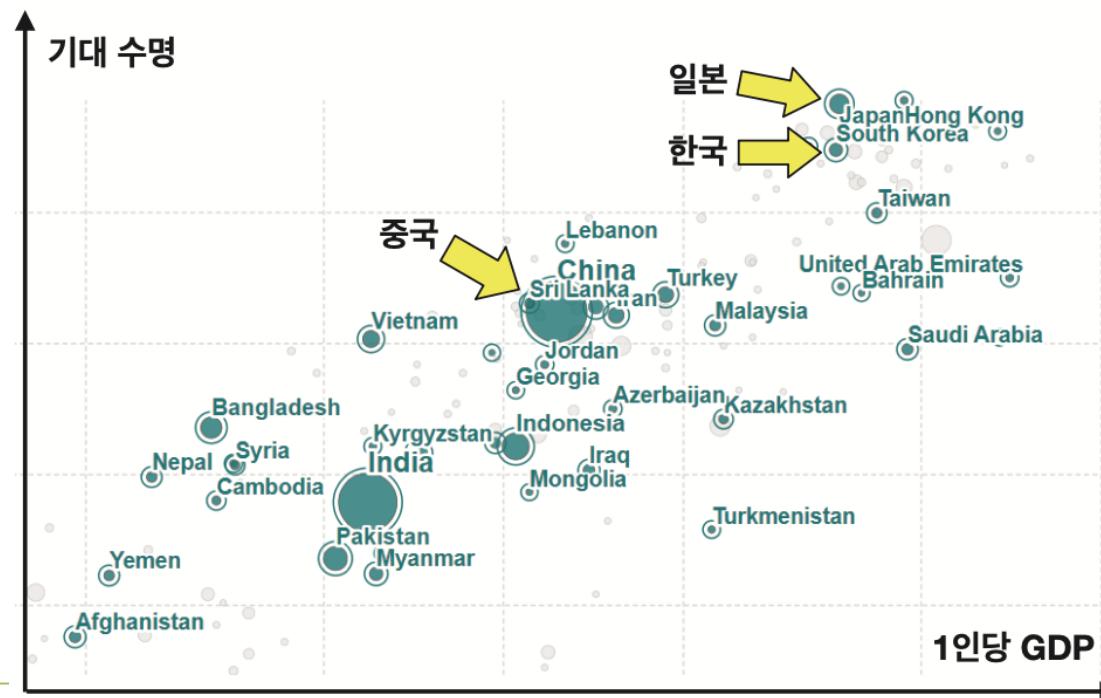
- 확률 분포의 하나인 균일 분포와 가우시안 분포를 그래프로 이해해보자.
- 아래 그림을 이것을 말이나 표로 설명하려면 얼마나 복잡할까를 생각해보자.
- 그림의 x 축은 데이터 값이며 y축은 데이터의 출현 확률/빈도이다.
- 이 그림을 통해 균등 분포가 데이터가 값에 상관없이 균등한 확률로 출현하는 성질을 가진다는 것을 쉽게 이해할 수 있을 것이다.
- 반면 가우시안 분포는 각 데이터 값의 출현 확률이 서로 다르며, 평균값 주위의 데이터가 많이 나타나는 분포라는 것을 쉽게 알 수 있다.



1 데이터 시각화

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

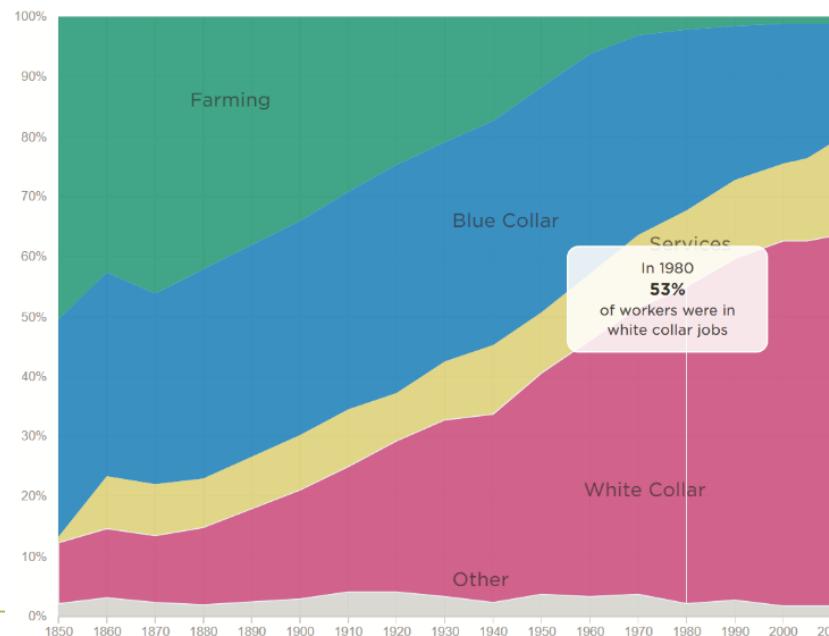


출처: Our World in Data, <https://ourworldindata.org/>

1 데이터 시각화

데이터 시각화를 통해 가치있는 정보를 만드는 사례 (계속)

- 미국의 비영리 미디어 기관인 국립 공영 라디오(NPR)의 웹사이트는 다음과 같은 대화식 다이어그램을 통해서 1850년에서부터 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/>

2 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
```

이 코드의 실행 결과는?

```
# 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()
```

2 matplotlib 무작정 사용해 보기

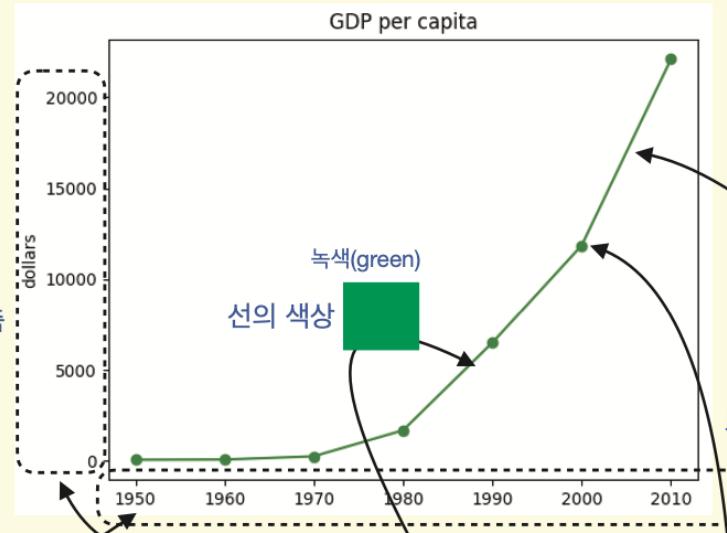
years

1950	1960	1970	1980	1990	2000	2010
67.0	80.0	257.0	1686.0	6505.0	11865.3	22105.3

gdp

제공되는 데이터

gdp
값
: 세로축



선의 색상
녹색(green)

선의 스타일 : 실선(solid)

표식의 스타일 : o
years : 가로축

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

제공되는 데이터와 matplotlib.pyplot 모듈을 이용한 시각화 방법

각각의 인자들의 의미는 정보를 시각화하는 방법

2 matplotlib 무작정 사용해 보기

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]
```

2 matplotlib 무작정 사용해 보기

맷플롯립 코드 살펴 보기 (계속)

- 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')
```

2 matplotlib 무작정 사용해 보기

맷플롯립 코드 살펴 보기 (계속)

- 차트의 제목 레이블을 설정한다. 차트의 최상단에 표시된다.

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

- y축의 레이블을 지정, savefig() 함수를 통해서 파일을 저장함. 해상도는 dpi dot per inch를 통해 지정
 - dpi는 1인치(약 2.54cm)당 점의 수를 의미함

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

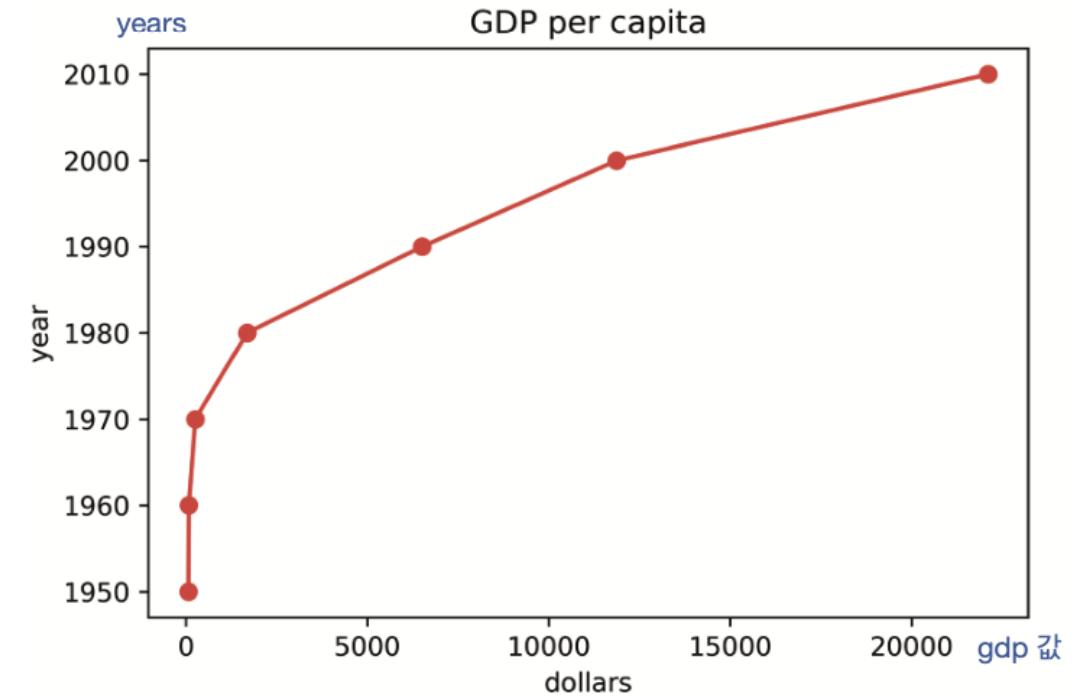
2 matplotlib 무작정 사용해 보기

matplotlib 코드 살펴 보기 (계속)

- 반드시 plt.plot() 함수를 사용하여야 화면에 차트가 나타난다

```
plt.xlabel("dollars")
plt.ylabel("gdp")

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



3 차트 장식을 도와주는 다양한 기법들

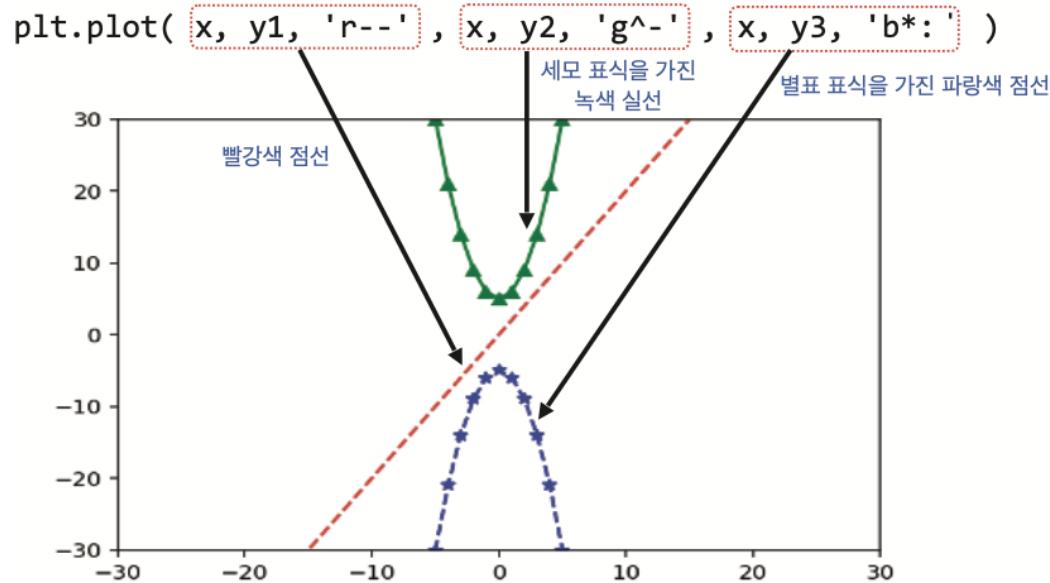
- `plot()` 함수를 수정하여 한 번의 호출만으로 $2x, x^2 + 5, -x^2 - 5$ 의 세 함수를 그려보도록 하자.

```
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
# 빨강색 점선, 녹색 실선과 세모기호, 파랑색 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()
```

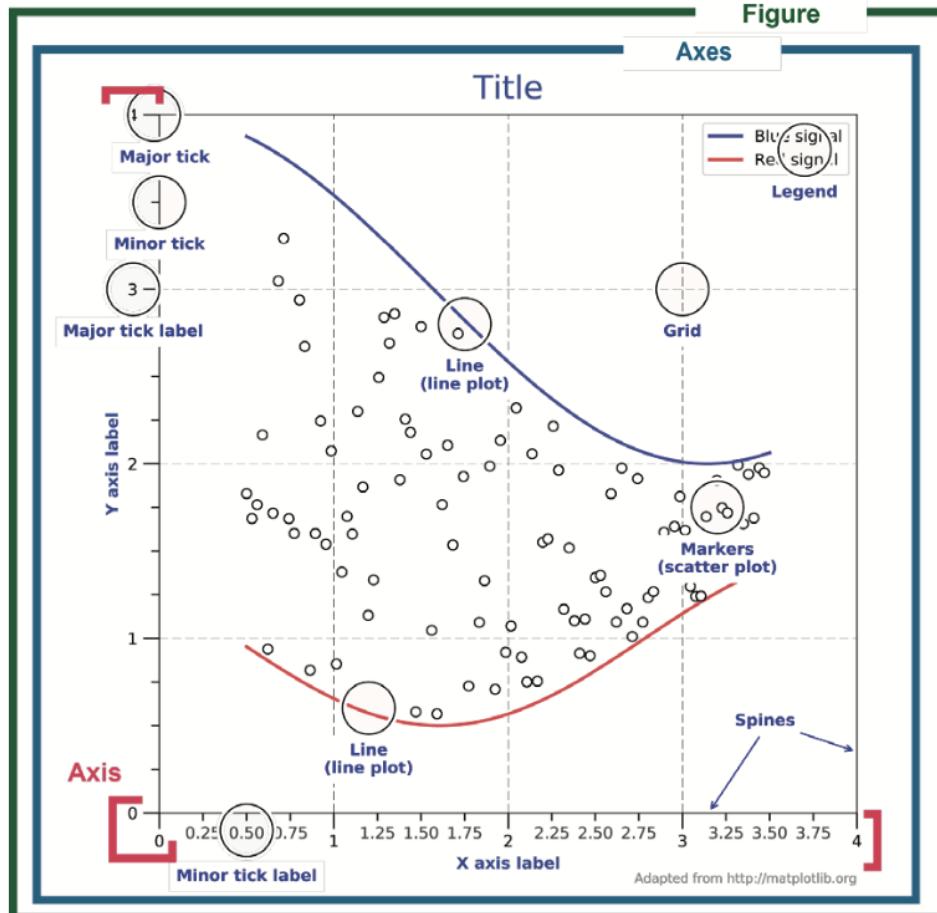
3 차트 장식을 도와주는 다양한 기법들

- '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

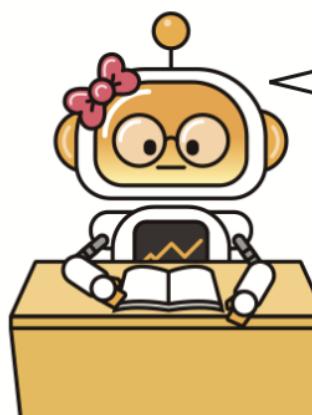


3 차트 장식을 도와주는 다양한 기법들

matplotlib에서 차트를 구성하는 여러 가지 용어들



- Figure : (여기에서) 그림을 포함하는 가장 바깥쪽 차트
- Axes : 데이터가 그려지는 공간 (x, y, z axes) \rightarrow axes
x axis
y axis
z axis
- Axis \leftarrow Tick \leftarrow Major
Tick labels
Minor
- Title : 그림의 제목
- Legend : 범례
- Grid : 그림 위에 선
- Line : 선
- Marker : 원형의 표시기
- Spline : 그림 경계



matplotlib 라이브러리에는 그림과 같이 다양한 기능이 제공됩니다. 예를 들어 오른쪽 상단의 Legend는 범례 표시에 해당하는 속성 이지요.

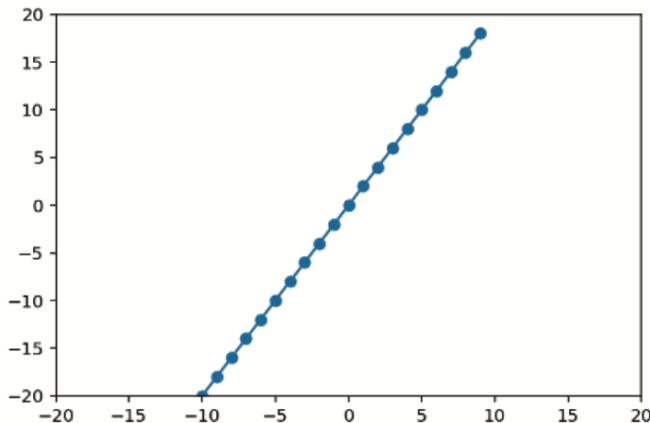
LAB¹ 수학 함수도 쉽게 그려보자

실습 시간



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

원하는 결과



힌트



이를 위하여 `range(-10, 10)` 함수를 사용할 것이며, 이 값을 이용하여 $y = 2x$ 를 얻기 위하여 다음과 같은 방법을 사용할 수 있다.

```
x = [x for x in range(-10, 10)] # -10에서 10사이의 수를 1의 간격으로 생성  
y = [2*t for t in x]           # 2*x를 원소로 갖는 y 함수
```

LAB¹수학 함수도 쉽게 그려보자

해답 코드



```
import matplotlib.pyplot as plt

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

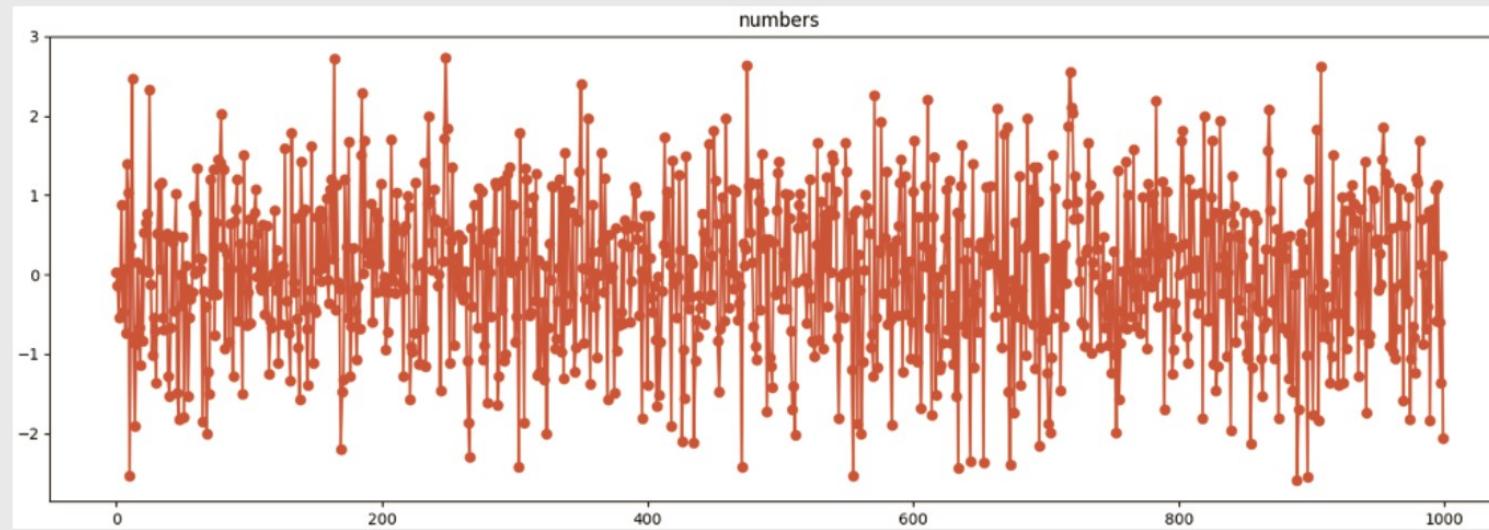
plt.axis([-20, 20, -20, 20])               # 그림을 그릴 영역을 지정함
plt.show()
```

LAB¹수학 함수도 쉽게 그려보자



도전문제 11.1

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



4 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.

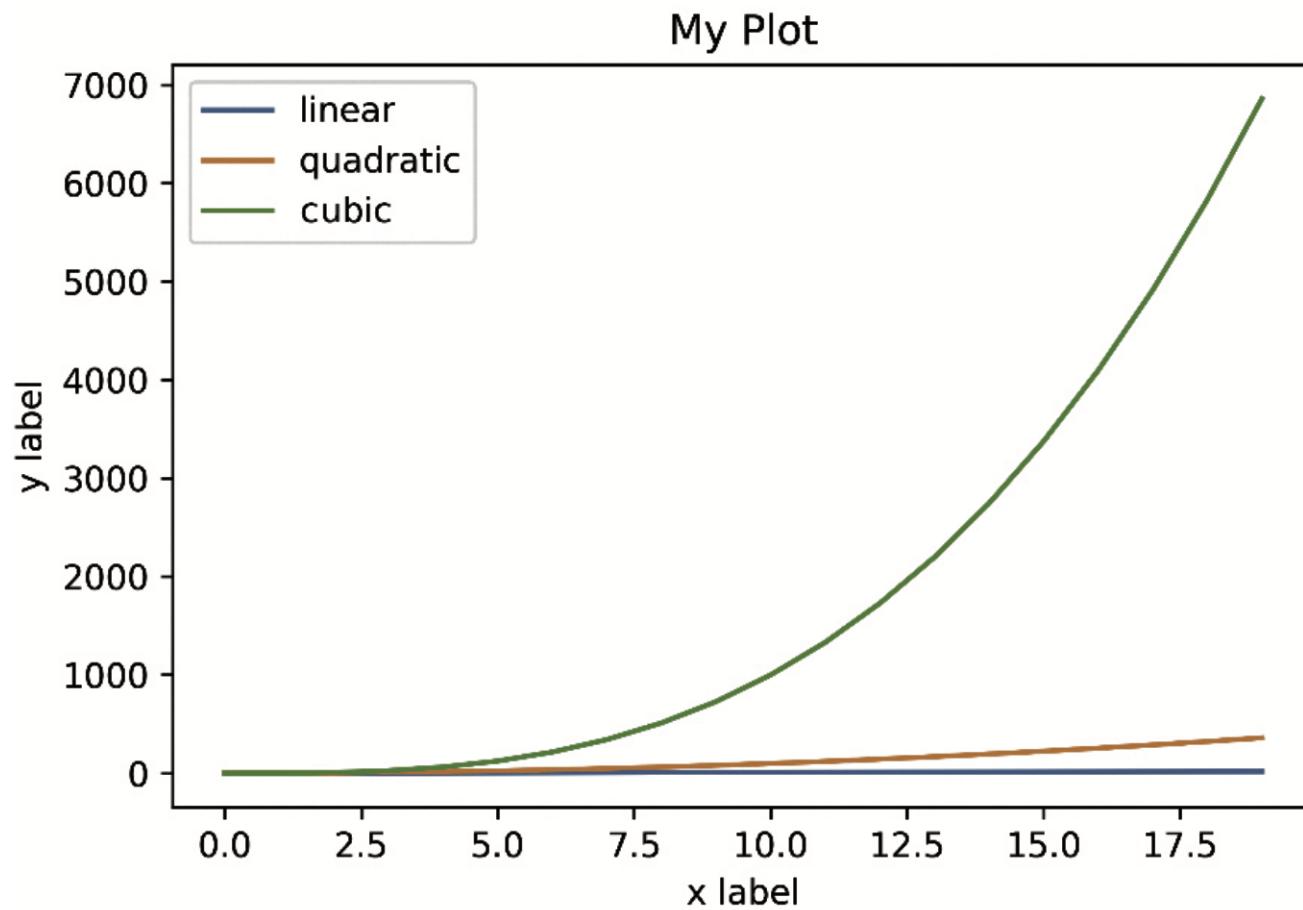
```
import matplotlib.pyplot as plt
x = [x for x in range(20)] # 0에서 19까지의 정수를 생성
y = [x**2 for x in range(20)] # 0에서 20까지의 정수 x에 대해 x 제곱값을 생성
z = [x**3 for x in range(20)] # 0에서 20까지의 정수 x에 대해 x 세제곱값을 생성

plt.plot(x, x, label='linear') # 각 선에 대한 레이블
plt.plot(x, y, label='quadratic') # y=x^2
plt.plot(x, z, label='cubic') # y=x^3

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

4 하나의 차트에 여러 개의 데이터를 그려보자

코드의 실행 결과는?

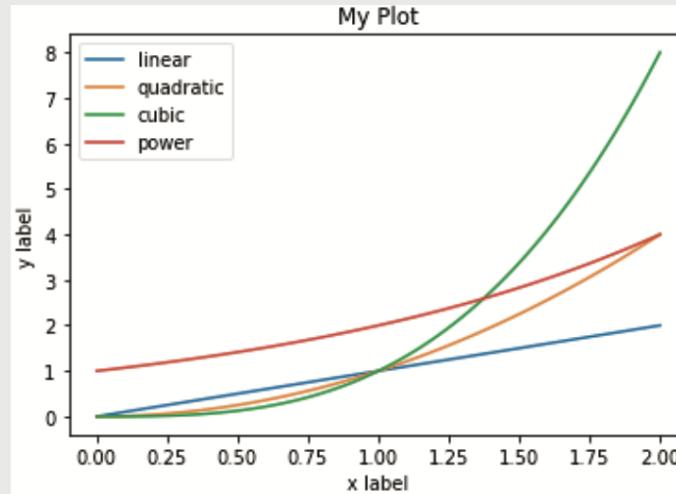


4 하나의 차트에 여러 개의 데이터를 그려보자



도전문제 11.2

위의 차트에 다음과 같이 2^x 함수도 추가하여 표시해 보자. 이 때 범례에 나타낼 문자열은 power라고 설정하자.



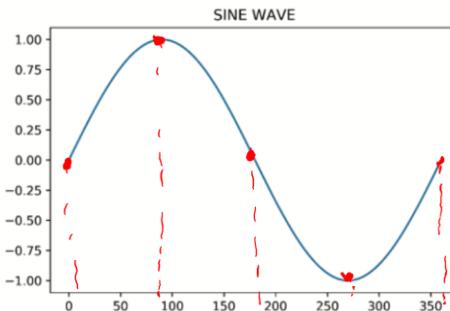
LAB² 삼각함수의 기본인 사인 그래프 그리기

실습 시간



다음과 같은 사인 그래프를 그려보자. 0에서 360까지의 값을 넘파이 `arange()` 함수를 이용하여 생성하도록 하자. 이 값을 사인값으로 고치면 될 것이다. 이를 위하여 numpy 모듈의 `sin()` 함수를 사용하면 된다.

원하는 결과



힌트



넘파이를 이용해서 0에서 360까지의 값을 만들자. 이를 위하여 `arange()` 함수를 사용하자. 그림과 같은 각을 도(degree)로 표현하면 0도에서 360도까지 범위이다. `math` 모듈이나 `numpy` 모듈의 `sin()` 함수와 `cos()` 함수는 라디안(radian)을 입력으로 받는다. 0에서 360도로 표현된 각 x 을 라디안으로 바꾸는 방법은 넘파이의 `radians()` 함수를 사용한다. 참고할 코드는 다음과 같다.

```
import numpy as np  
  
x = np.arange(0, 360)  
y = np.sin(np.radians(x))
```

다음으로 이 x 와 y 를 `plt.plot()` 함수로 그리기만 하면 될 것이다.

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

$$\text{radian} = \text{degree} \times \frac{\pi}{180}$$

$$0, 180, 360 \Rightarrow 0$$

$$90 \rightarrow 1$$

$$270 \rightarrow -1$$

LAB² 삼각함수의 기본인 사인 그래프 그리기

해답 코드



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

x = np.arange(0, 360)
y = np.sin(np.radians(x))

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

LAB² 삼각함수의 기본인 사인 그래프 그리기



도전문제 11.3

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

5 막대형 차트도 손쉽게 그려보자

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

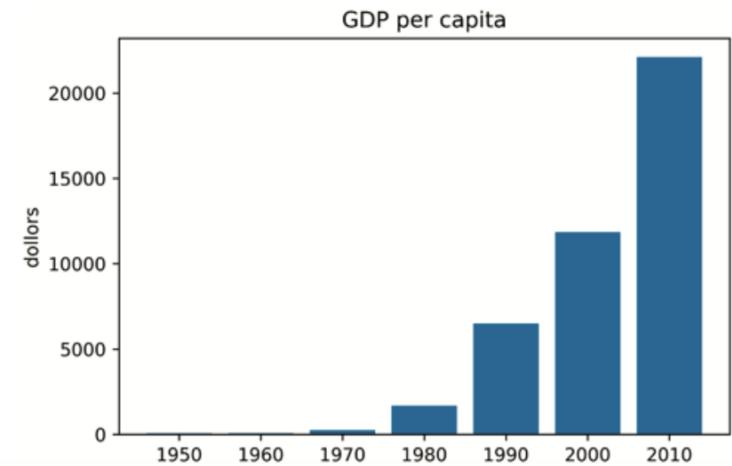
```
from matplotlib import pyplot as plt

# 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 레이블을 붙인다.

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



5 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)
```

5 막대형 차트도 손쉽게 그려보자

여러 나라의 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.

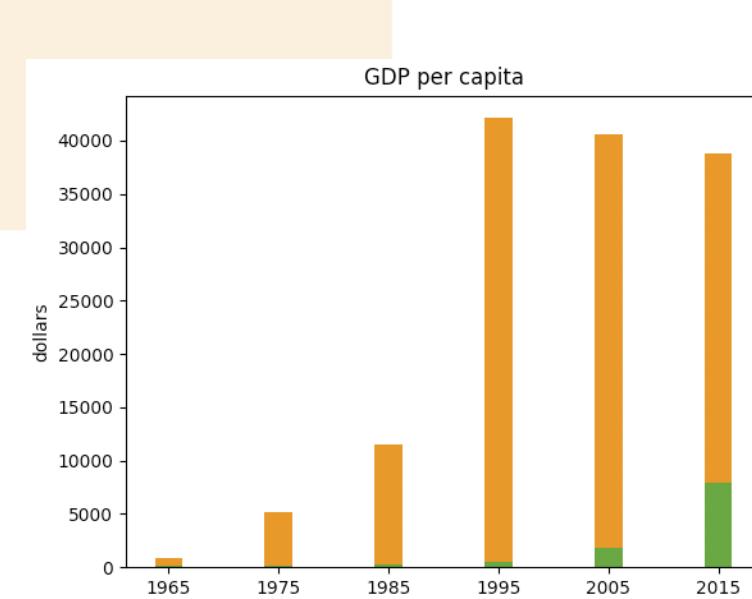
```
# 1인당 국민소득  
years = [1965, 1975, 1985, 1995, 2005, 2015]  
ko = [130, 650, 2450, 11600, 17790, 27250]  
jp = [890, 5120, 11500, 42130, 40560, 38780]  
ch = [100, 200, 290, 540, 1760, 7940]
```

5 막대형 차트도 손쉽게 그려보자

여러 나라의 국민소득 추이를 다중 막대형 차트로 그리자 (계속)

- 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()
```



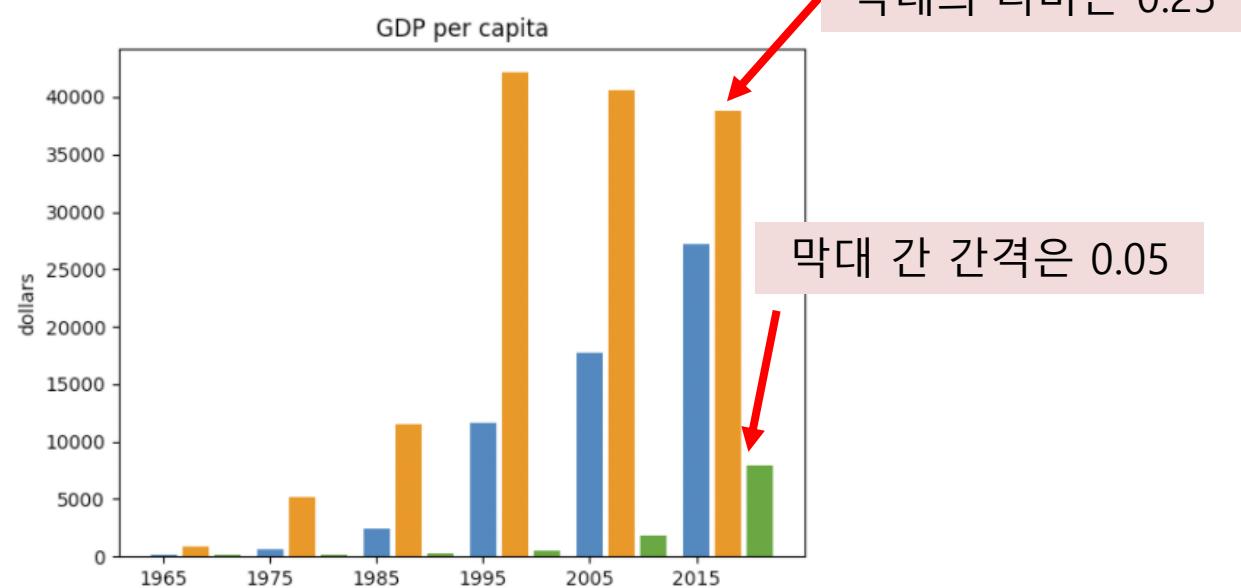
5 막대형 차트도 손쉽게 그려보자

여러 나라의 국민소득 추이를 다중 막대형 차트로 그리자 (계속)

- 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

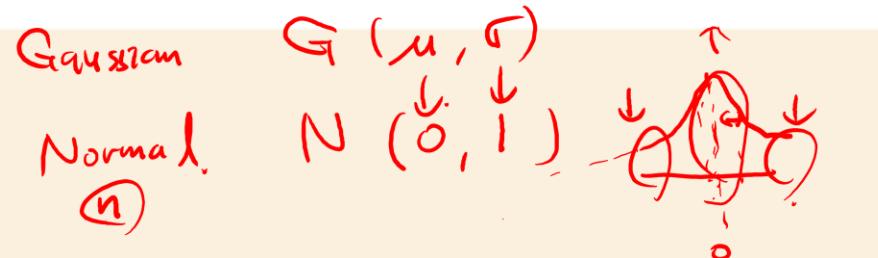
x_range = np.arange(len(years))
plt.bar(x_range + 0.0, ko, width = 0.25)
plt.bar(x_range + 0.3, jp, width = 0.25)
plt.bar(x_range + 0.6, ch, width = 0.25)
plt.show()
```



6 데이터를 점으로 표현하는 산포도 그래프 그리기

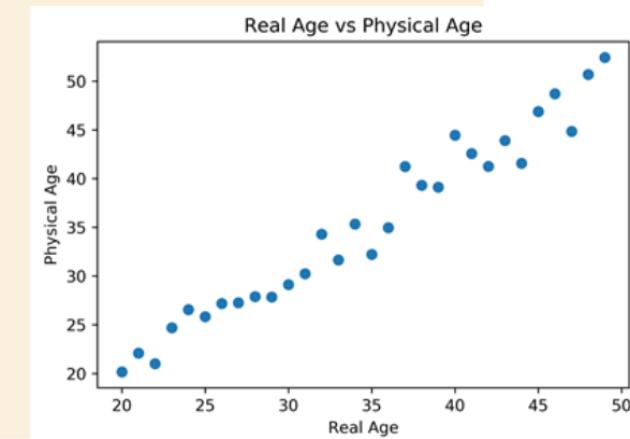
- 산포도 플롯 scatter plot은 개별 데이터 포인트를 그리는 차트. 산포도를 그릴 때는 각 데이터 포인트가 연결되지 않는다. 산포도 그래프를 그릴 때는 scatter() 함수를 사용함

```
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()
```



xData에 randn() 함수로 잡음을 섞는다.
잡음은 정규분포로 만들어 질 것이다.

$\sim (-0.5) \sim 0.5$

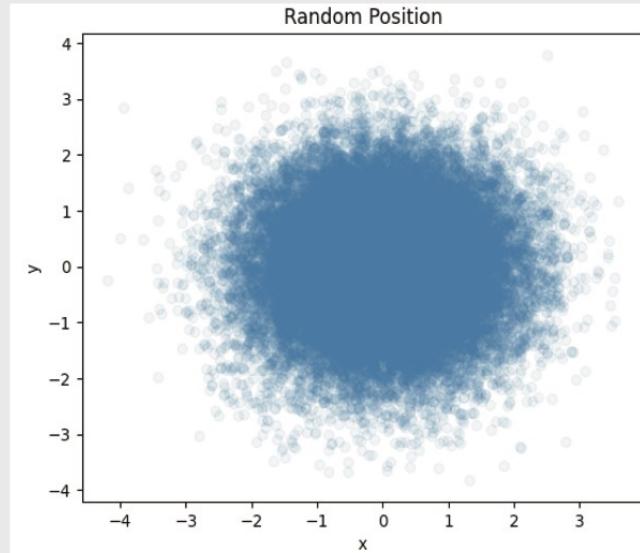
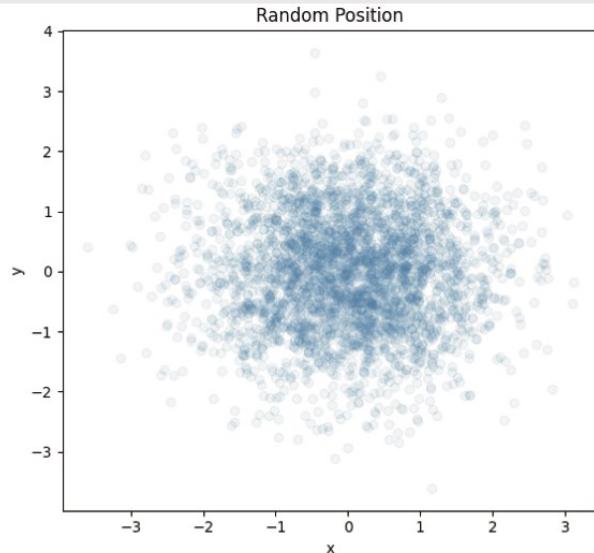


6 데이터를 점으로 표현하는 산포도 그래프 그리기



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 (이 값은 0에서 1 사이 값을 가지며, 0이면 투명, 1이면 불투명이다).



6 데이터를 점으로 표현하는 산포도 그래프 그리기

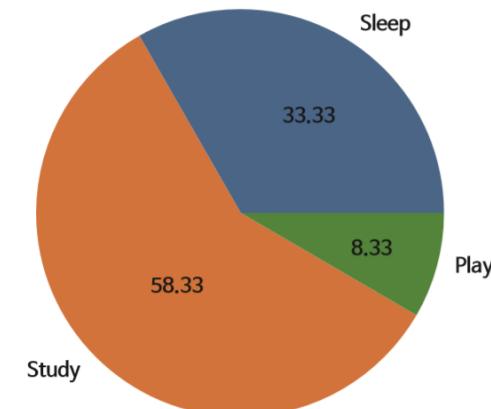
원형 비율을 표시하는 파이 차트 만들기

- 파이 차트 pie chart는 데이터의 값에 따라서 원형 비율로 나누어져 있는 차트
- 파이 차트는 비즈니스 세계에서 널리 사용(예들 들어, 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()
```



7 히스토그램으로 자료의 분포를 한눈에 살펴보자

- **히스토그램histogram**은 주어진 자료를 몇 개의 구간으로 나누고 각 구간의 **도수frequency**를 조사하여 나타낸 막대 그래프이다

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

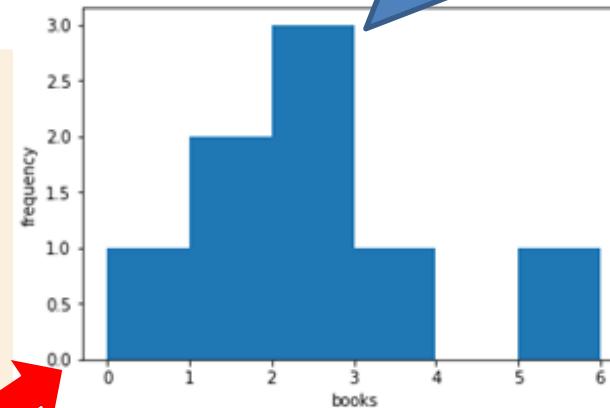
학생 8명이 1년 동안
읽은 책의 수

```
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()
```



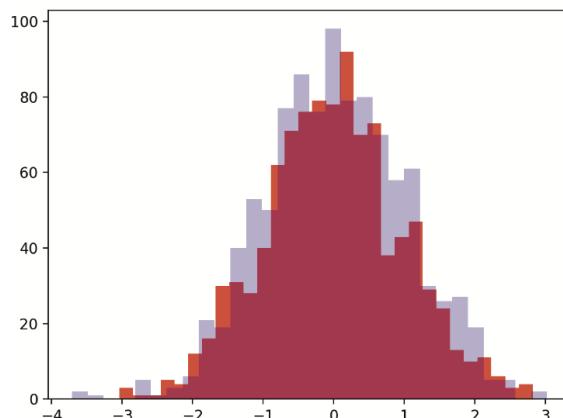
데이터의 분포를
설명하는 차트.



데이터가 들어가는
통을 bin이라 하자.

7 히스토그램으로 자료의 분포를 한눈에 살펴보자

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

7 히스토그램으로 자료의 분포를 한눈에 살펴보자

겹쳐진 히스토그램도 그리자 : 다중 히스토그램 (계속)



도전문제 11.5

다음 코드에서 빈의 개수를 5로 설정해 히스토그램을 그려보라.

```
import matplotlib.pyplot as plt

x = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
plt.hist(x)
plt.show()
```

LAB³ 정규분포로 생성된 난수를 눈으로 확인하기

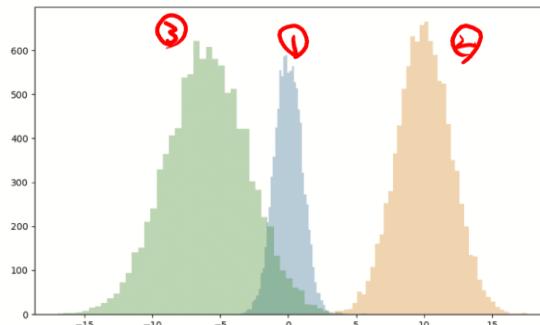
실습 시간



정규분포로 난수를 생성하고 이 수들이 어떠한 방식으로 흩어져 있는지를 확인해 보고자 한다. 표준 정규분포로 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.

①

원하는 결과



힌트



충분한 규모의 난수를 발생시키기 위해 10,000개의 난수를 각각의 분포로 생성해 보자. 표준 정규분포를 생성하는 것은 간단히 다음과 같이 생성할 수 있다.

`standard_Gauss = np.random.randn(10000)`

그리고 평균값 μ 의 값을 담은 변수 `mu`와 표준편차 σ 를 저장한 변수 `sigma`를 이용하여 해당 평균값과 표준편차를 가진 정규분포를 따르는 값들을 생성하는 코드는 다음과 같다.

`Gauss = mu + sigma * np.random.randn(10000)`

이렇게 생성된 난수들을 맷플롯립의 히스토그램 그리기로 가시화하면 된다. 위 결과는 `bins`을 50개로 설정한 결과이다.

LAB³ 정규분포로 생성된 난수를 눈으로 확인하기

해답 코드



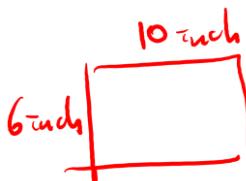
```
import numpy as np
import matplotlib.pyplot as plt

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()
```



LAB⁴ 차종별 판매량을 파이 차트로 표현하자

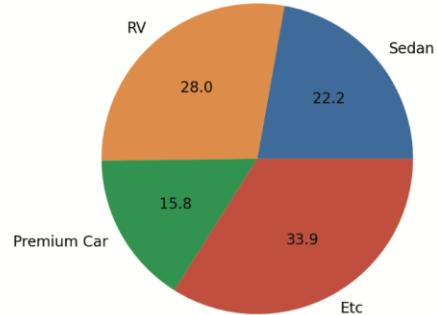
실습 시간



국내 최대 자동차 회사인 H사는 2022년 4월달에 차종별로 다음과 같은 판매량을 달성하였다. 이 자료를 바탕으로 아래 그림과 같은 파이 차트를 그려보도록 하자(판매량의 단위는 천대이다).

차종	세단	RV	프리미엄 차	기타
판매량	15.7	19.8	11.2	24

원하는 결과



힌트



표를 바탕으로 자동차의 판매량을 다음과 같은 리스트의 원소로 넣을 수 있다.

```
volume = [15.7, 19.8, 11.2, 24.0]
```

다음으로 레이블을 리스트의 원소로 넣도록 하자.

```
name_labels = ['Sedan', 'RV', 'Premium Car', 'Etc']
```

이렇게 생성된 판매량 자료와 레이블을 plt.pie() 함수의 인자로 넣도록 한다.

LAB⁴ 차종별 판매량을 파이 차트로 표현하자

해답 코드



```
import matplotlib.pyplot as plt

volume = [15.7, 19.8, 11.2, 24.0]
name_labels = ['Sedan', 'RV', 'Premium Car', 'Etc']

plt.pie(volume, labels = name_labels, autopct = "%.1f")
plt.show()
```

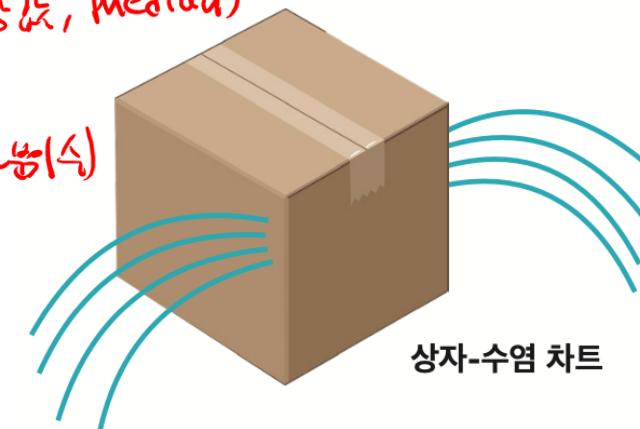
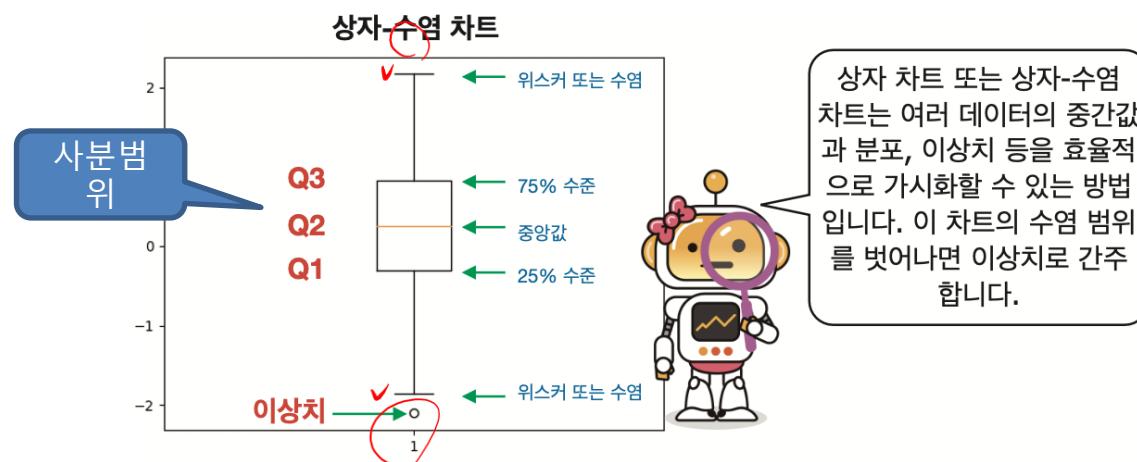
8 데이터를 효율적으로 표현하는 상자 차트를 알아보자

- **상자 차트** box plot 는 데이터의 최대, 최소, 중간값과 사분위 수 등을 효율적으로 가시화 할 수 있는 방법이다. 데이터의 범위를 표시하는 선을 수염이라고 불러 **상자-수염 whisker** 차트라고도 한다.

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

random_data = np.random.randn(100)

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



Inter Quartile Range (IQR)

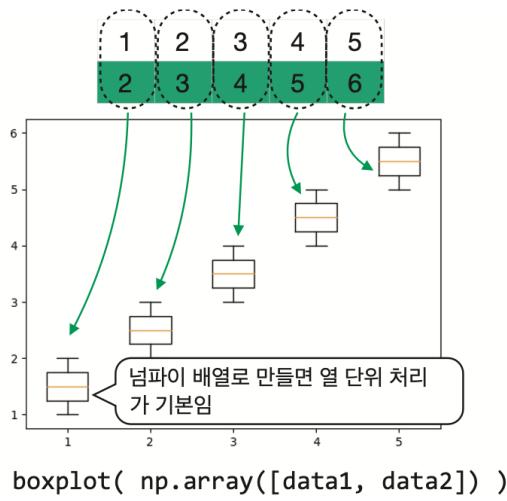
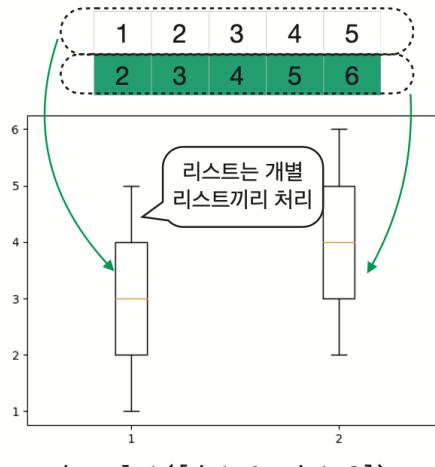
8 데이터를 효율적으로 표현하는 상자 차트를 알아보자

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 ]))
```

List-based
Box Plot



Numpy-based
Box Plot

8 데이터를 효율적으로 표현하는 상자 차트를 알아보자

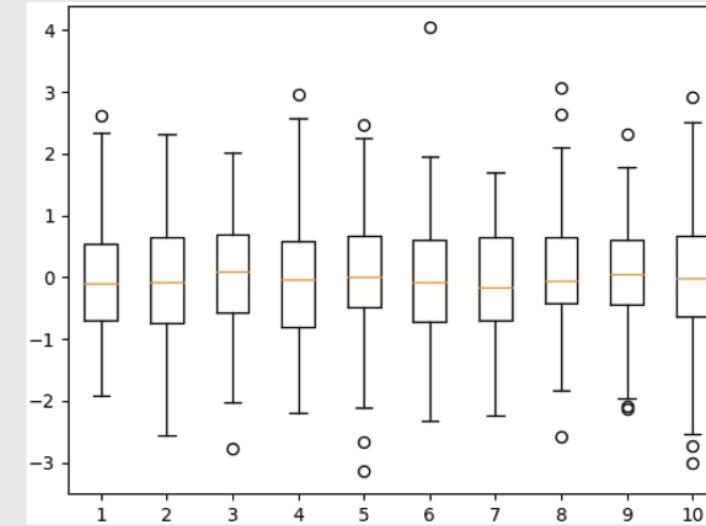
여러 개의 상자 차트 그리기 (계속)



도전문제 11.6

오른쪽 그림과 같이 100개의 난수를 가진 데이터 10종을 만들어 각각의 데이터 범위를 가시화하는 코드를 작성해 보라. 난수를 생성하는 일은 넘파이의 `random` 모듈이 가지고 있는 `randn(d0, d1)`을 사용하면 될 것이다.

이때 생성되는 데이터는 넘파이 배열이므로 차원을 어떻게 결정해야 하는지 유의해서 숫자 d_0 와 d_1 을 결정해야 할 것이다. 결과는 생성된 난수에 따라 오른쪽 그림과 조금 다를 것이다.



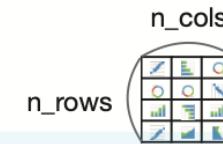
9 한 화면에 여러 그래프 그리기 : subplots()

필요한 모듈

`matplotlib.pyplot.subplots(n_rows, n_cols, ...)`

행의 수

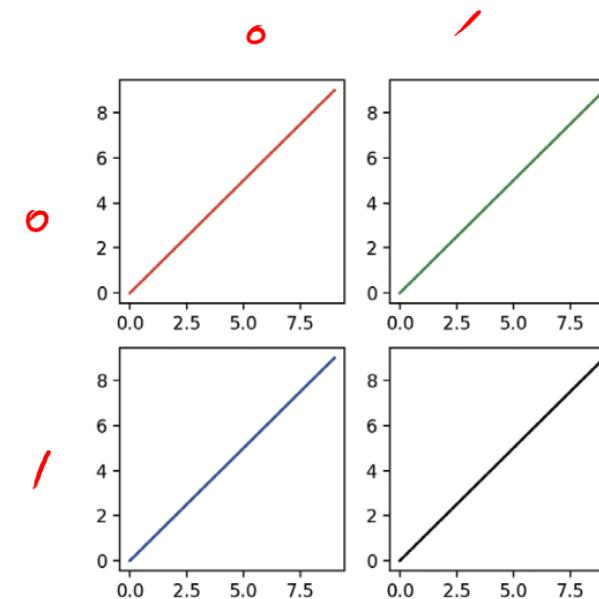
열의 수



```
import matplotlib.pyplot as plt

# 갯수가 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()
```

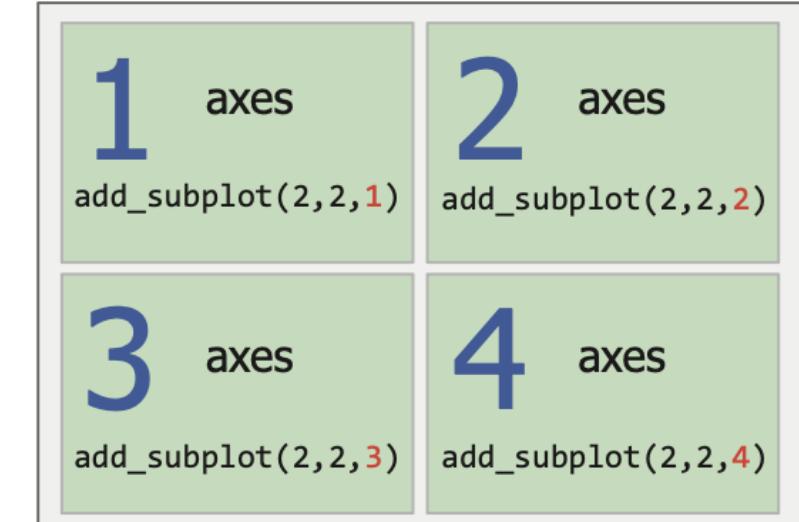


9 한 화면에 여러 그래프 그리기 : subplots()

```
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

figure



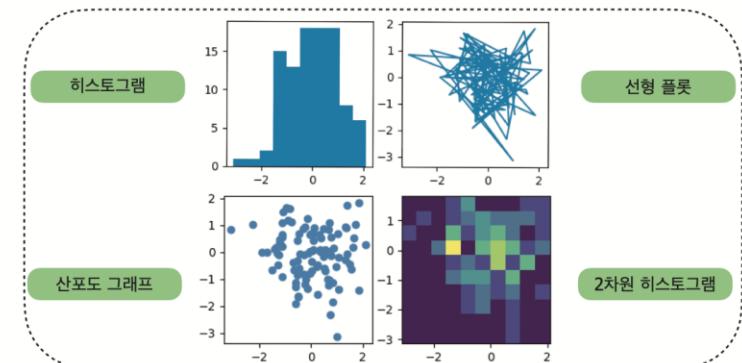
LAB⁵ 서브플롯 이용해 보기

실습 시간



다양한 차트를 하나의 그림에 나타내어 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



힌트



`data[0]`과 `data[1]`에 각각 100개의 난수를 정규 분포를 따라 생성하고 싶으면 행렬의 크기가 (2, 100)이 되도록 해야 한다. 정규 분포를 따라 난수를 발생시키는 함수는 `randn()`이므로 아래와 같이 생성할 수 있을 것이다.

```
data = np.random.randn(2, 100)
```

화면을 네 개로 쪼개기 위해 다음 코드를 사용할 수 있다.

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

생성된 네 개의 axis는 `axs`에 담겨 있으며 [row, col] 형태로 접근 가능하다.

LAB⁵ 서브플롯 이용해 보기

해답 코드



```
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()
```

- 데이터 시각화는 데이터가 가진 의미를 파악하는 데에 매우 중요한 역할을 한다.
- 좋은 시각화는 많은 양의 설명을 대체할 수 있는 효율적인 정보 전달 도구이다.
- 맷플롯립은 파이썬에서 가장 많이 사용되는 데이터 시각화 도구이다.
- 필요한 정보에 따라 다양한 시각화 기법을 골라서 사용할 수 있다.
- 하나의 그림을 쪼개어 여러 개의 차트를 표시할 수도 있다.

데이터를 이해하는 데에는 숫자도 좋은 정보가 되지만 아무래도 그림이 직관적이지요.
데이터 과학자들은 다양한 방식으로 데이터를 시각화합니다.



맷플롯립을 이용해서 다양한 그림을 그려 보았는데,
 이걸로 이제 어떤 그림이든 쉽게 가시화할 수 있는 것인가요?

다양한 방식으로 가시화할 수 있는 것은 맞지만 데이터를 잘 정리해야 하겠지요.
 그리고 여기서 배운 내용은 맷플롯립의 아주 간단한 시각화 예시에 불과하답니다.
 다양한 시각화 기술은 맷플롯립만을 다룬 다양한 자료를 참고하도록 하세요.





데이터를 가지고 다양한 시각 정보를 제공하는 것은 맷플롯립 뿐인가요?

그렇지 않아요 맷플롯립을 활용하여 더 다양한 시각화가 가능하도록 만든 모듈들도 있고요.

데이터를 다루는 데에 많이 사용되는 판다스 모듈은 맷플롯립과 결합된 시각화 방법을 제공합니다.

그리고 seaborn 모듈의 가시화 기능도 매우 뛰어나요. 뒤에서 조금씩 살펴보도록 해요.





Questions?