



데이터시각화이해와실습

Lecture 07. numpy 라이브러리를 활용한 프로젝트

ndarray에서 제공하는 메서드 + 브로드캐스팅

리스트(List).append --> matplotlib

동덕여자대학교
데이터사이언스 전공
권 범

목차

❖ 01. 숫자 데이터를 쉽게 다루도록 돕는 numpy 라이브러리

❖ 02. numpy를 활용한 나만의 프로젝트 만들기

~~list a = [1, 2, 3] +, -, *, / 3 (리터럴 상수 정의 X)~~

for j in range(3):

 a[j] = a[j] ** 3

ndarray b = [4, 5, 6] +, **, / 3



브로드캐스팅

수치의, 수의

Numerical Python --> 1. math 2. random

01. 숫자 데이터를 쉽게 다루도록 돕는 numpy 라이브러리

값이 한 개: 스칼라

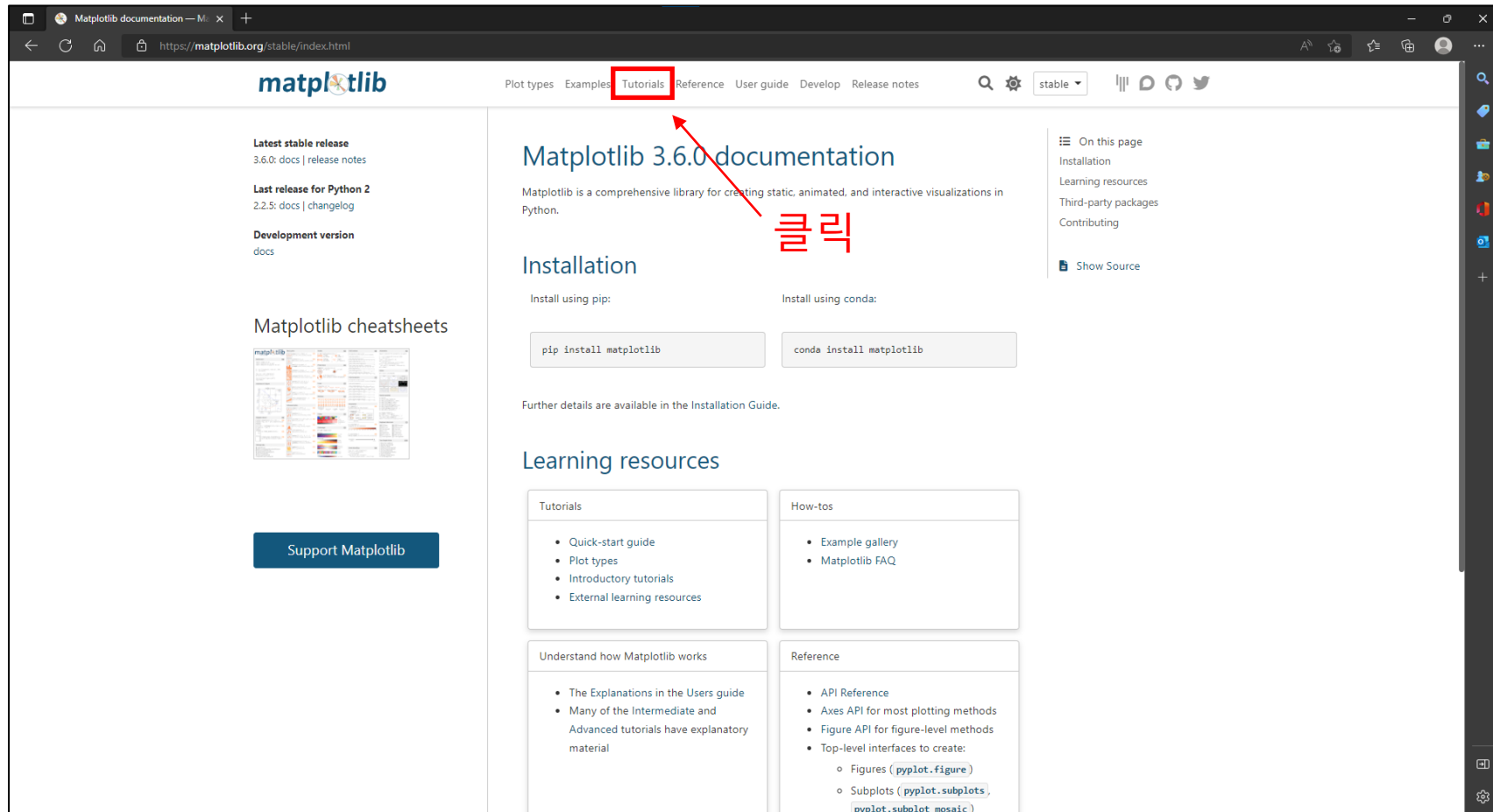
값이 여러 개: 배열(벡터, 행렬) --> 프로그래밍 언어에서의 Array

02. numpy를 활용한 나만의 프로젝트 만들기

01. 숫자 데이터를 쉽게 다루도록 돕는 numpy 라이브러리

❖ ① matplotlib 홈페이지 (1/6)

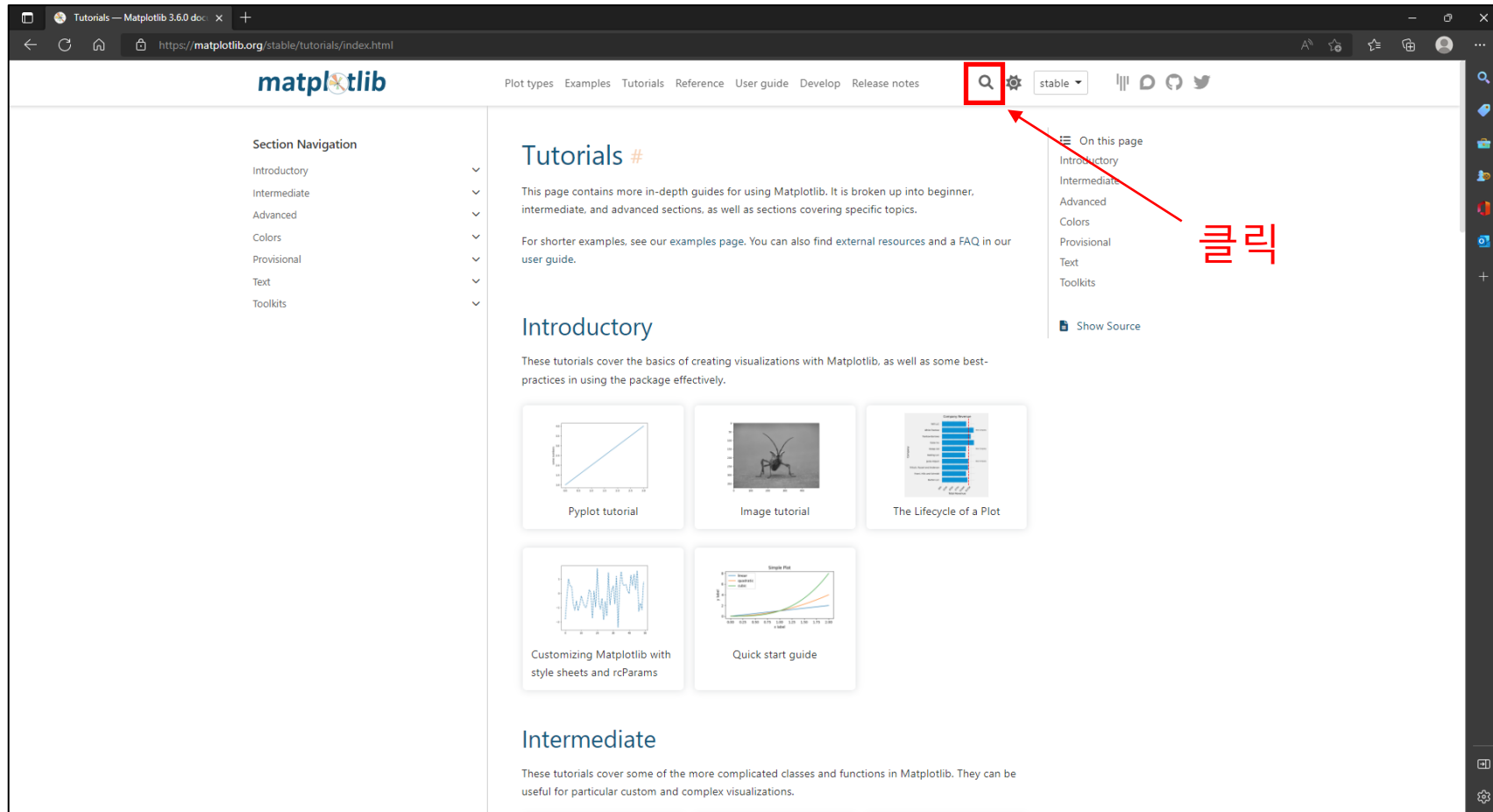
- <https://matplotlib.org>에서 데이터 시각화와 관련된 다양한 예시를 확인 할 수 있음
- [Tutorials] 버튼을 클릭



01. 숫자 데이터를 쉽게 다루도록 돕는 numpy 라이브러리

❖ ① matplotlib 홈페이지 (2/6)

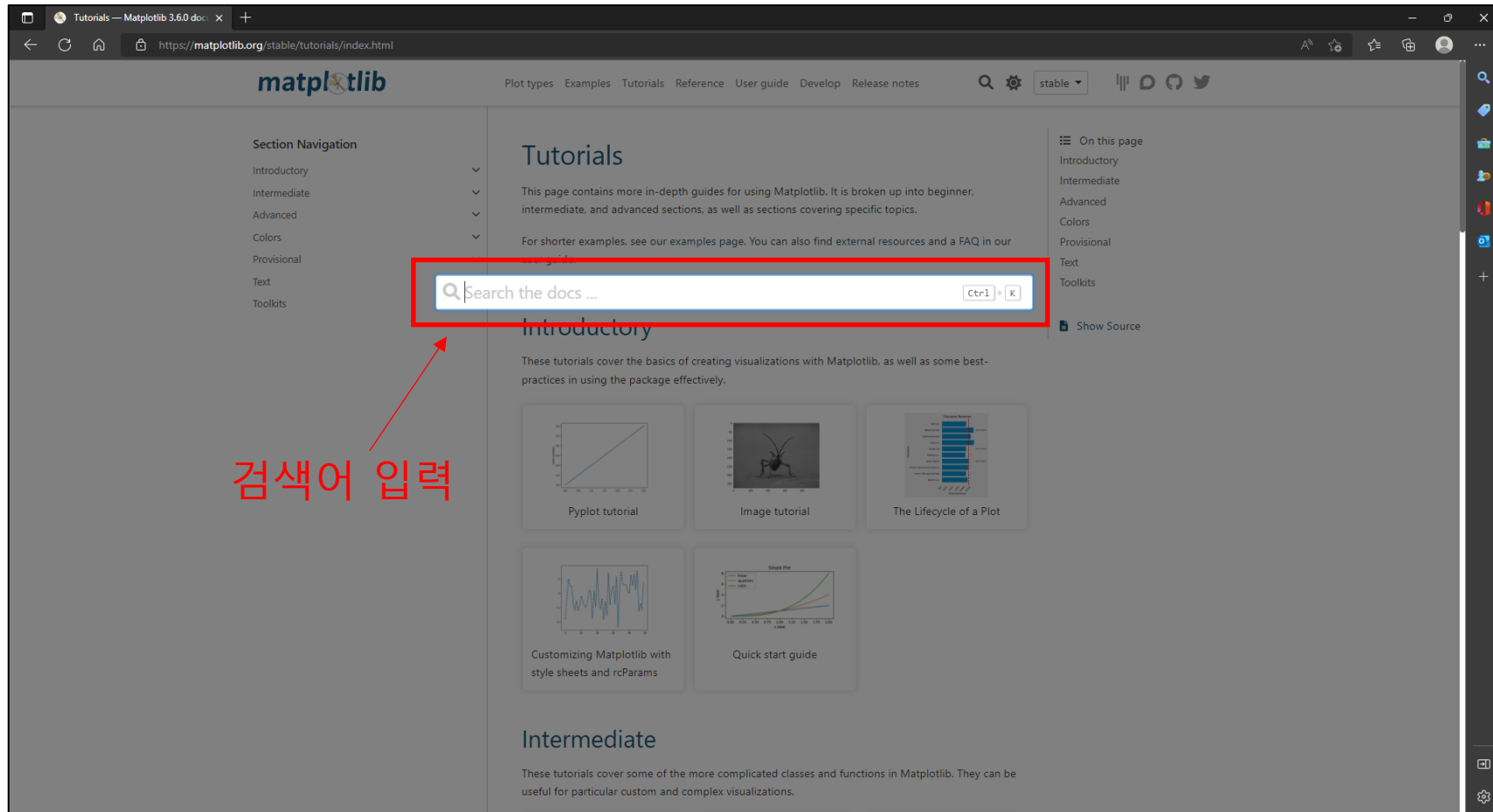
- 우측 상단에 돋보기 버튼을 클릭



01. 숫자 데이터를 쉽게 다루도록 돕는 numpy 라이브러리

❖ ① matplotlib 홈페이지 (3/6)

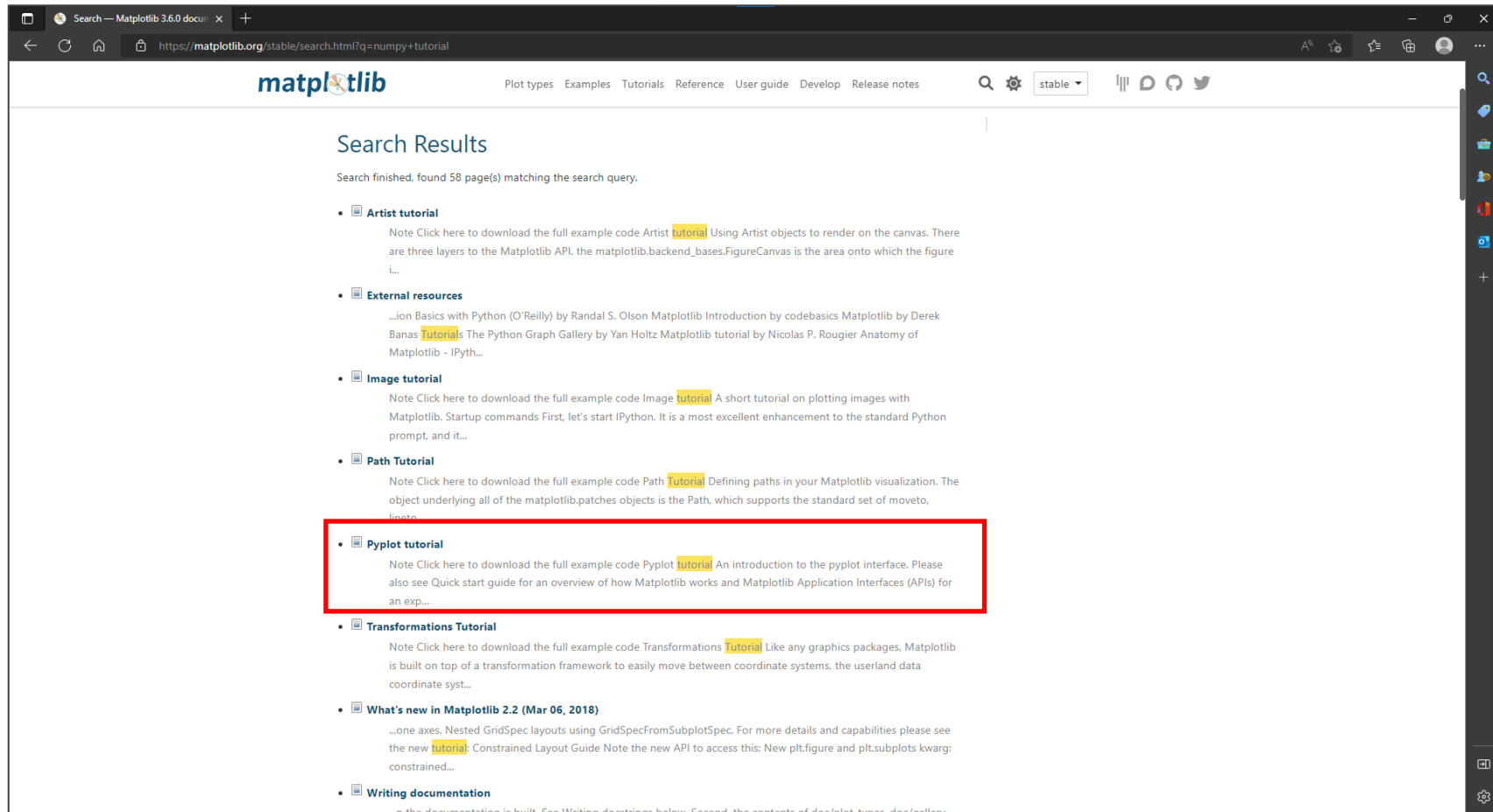
- [Search the docs ...]에 "(찾고자 하는)키워드"로 검색하면 관련 내용을 확인할 수 있음
- "numpy tutorial"이라고 검색어를 입력하고 검색해 보자



01. 숫자 데이터를 쉽게 다루도록 돕는 numpy 라이브러리

❖ ① matplotlib 홈페이지 (4/6)

- “Pyplot tutorial”에 지금까지 배운 내용들이 잘 정리되어 있음
- 내용을 같이 살펴 보자



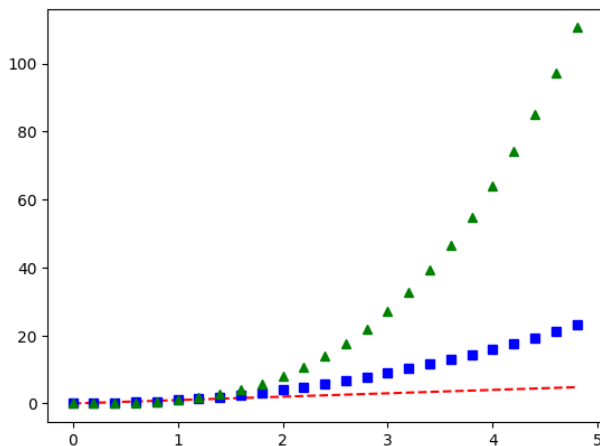
01. 숫자 데이터를 쉽게 다루도록 돕는 numpy 라이브러리

❖ ① matplotlib 홈페이지 (5/6)

- numpy 라이브러리를 활용하여 작성한 코드

```
1 import matplotlib.pyplot as plt
2 import numpy as np
3
4 t = np.arange(0., 5., 0.2)
5
6 plt.figure()
7 plt.plot(t, t, "r--", t, t**2, "bs", t, t**3, "g^")
8 plt.show()
```

실행결과



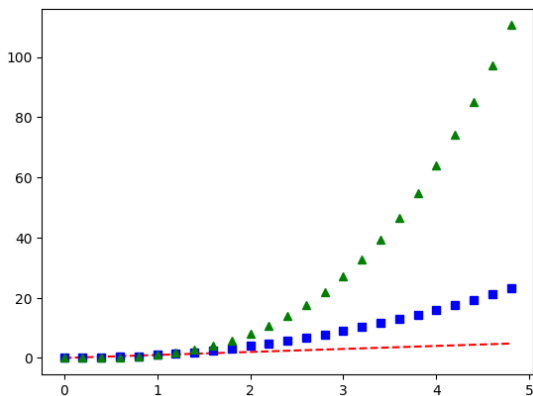
01. 숫자 데이터를 쉽게 다루도록 돕는 numpy 라이브러리

❖ ① matplotlib 홈페이지 (6/6)

- 파이썬 리스트를 활용하여 작성한 코드

```
1 import matplotlib.pyplot as plt
2 a, b, c = [], [], []
3 for j in range(0, 50, 2):
4     a.append(j / 10)
5     b.append((j / 10) ** 2)
6     c.append((j / 10) ** 3)
7 plt.figure()
8 plt.plot(a, a, "r--", a, b, "bs", a, c, "g^")
9 plt.show()
```

실행결과



numpy 라이브러리를 활용할 경우,
보다 적은 수의 코드로 그리고 간결하게
원하는 결과를 얻을 수 있음

01. 숫자 데이터를 쉽게 다루도록 돕는 numpy 라이브러리

❖ ② numpy 라이브러리 시작하기 (1/6)

- 제곱근(Square Root) 출력하기

```
1 import numpy
2 print(numpy.sqrt(2))
```

실행결과

```
1.4142135623730951
```

- alias를 활용하여 제곱근 출력하기

```
1 import numpy as np
2 print(np.sqrt(2))
```

실행결과

```
1.4142135623730951
```

01. 숫자 데이터를 쉽게 다루도록 돕는 numpy 라이브러리

❖ ② numpy 라이브러리 시작하기 (2/6)

- 파이와 삼각함수 활용하기

```
1 import numpy as np
2 print(np.pi)
3 print(np.sin(0))
4 print(np.cos(np.pi))
```

실행결과

```
3.141592653589793
0.0
-1.0
```

01. 숫자 데이터를 쉽게 다루도록 돕는 numpy 라이브러리

❖ ② numpy 라이브러리 시작하기 (3/6)

- random 서브 라이브러리의 rand() 함수 / numpy의 ndarray 타입

```
1 import numpy as np
2 a = np.random.rand(5)
3 print(a)
4 print(type(a))
```

실행결과

```
[0.92527466 0.42716173 0.40530289 0.43800242 0.28079498]
<class 'numpy.ndarray'>
```

random 라이브러리의 randint() 함수의
실수 버전이라고 생각하면 됨

01. 숫자 데이터를 쉽게 다루도록 돕는 numpy 라이브러리

❖ ② numpy 라이브러리 시작하기 (4/6)

- random 서브 라이브러리의 choice() 함수

```
1 import numpy as np
2 print(np.random.choice(6, 10))
```

실행결과

```
[4 4 5 3 5 2 4 1 0 0]
```

0이상 6미만인 정수 중 10개를 뽑음(중복 허용)

01. 숫자 데이터를 쉽게 다루도록 돕는 numpy 라이브러리

❖ ② numpy 라이브러리 시작하기 (5/6)

- random 서브 라이브러리의 choice() 함수(중복 금지)

```
1 import numpy as np
2 print(np.random.choice(10, 6, replace=False))
```

실행결과

```
[7 3 8 9 1 6]
```

- ✓ 만약 한 번 뽑은 숫자를 다시 뽑지 못하게 하고 싶다면, replace 속성을 False로 설정하면 됨
- ✓ 0이상 10미만인 정수 중 6개를 뽑음(중복 허용)

01. 숫자 데이터를 쉽게 다루도록 돕는 numpy 라이브러리

❖ ② numpy 라이브러리 시작하기 (6/6)

- random 서브 라이브러리의 choice() 함수(확률 설정)

```
1 import numpy as np
2 print(np.random.choice(6, 10, p=[0.1, 0.2, 0.3, 0.2, 0.1, 0.1]))
```

실행결과

```
[5 3 3 2 2 0 2 4 2 2]
```

- ✓ p 속성은 각 경우의 수가 발생할 확률을 정할 수 있음
- ✓ 주의할 점은 확률의 합은 반드시 1이어야 함

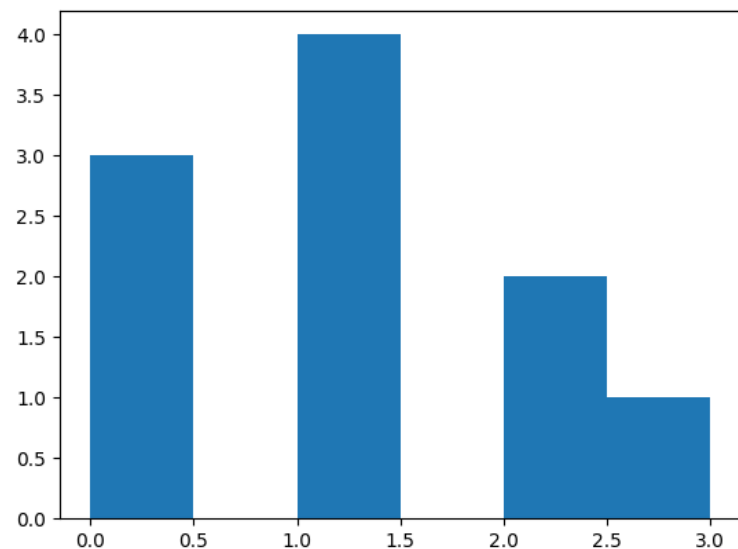
01. 숫자 데이터를 쉽게 다루도록 돕는 numpy 라이브러리

❖ ③ numpy 라이브러리를 활용해 그래프 그리기 (1/8)

- numpy 라이브러리를 활용하여 작성한 코드

```
1 import matplotlib.pyplot as plt
2 import numpy as np
3 dice = np.random.choice(6, 10)
4 plt.figure()
5 plt.hist(dice, bins=6)
6 plt.show()
```

실행결과



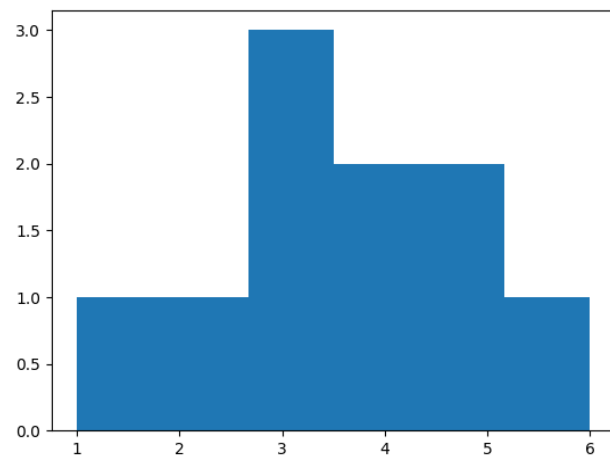
01. 숫자 데이터를 쉽게 다루도록 돕는 numpy 라이브러리

❖ ③ numpy 라이브러리를 활용해 그래프 그리기 (2/8)

- 파이썬 리스트를 활용하여 작성한 코드

```
1 import matplotlib.pyplot as plt
2 import random
3 dice = []
4 for _ in range(10):
5     dice.append(random.randint(1, 6))
6 plt.figure()
7 plt.hist(dice, bins=6)
8 plt.show()
```

실행결과



✓ numpy 라이브러리를 활용할 경우 장점:

- ① 코드가 간결해짐
- ② 코드 실행 속도가 빠름

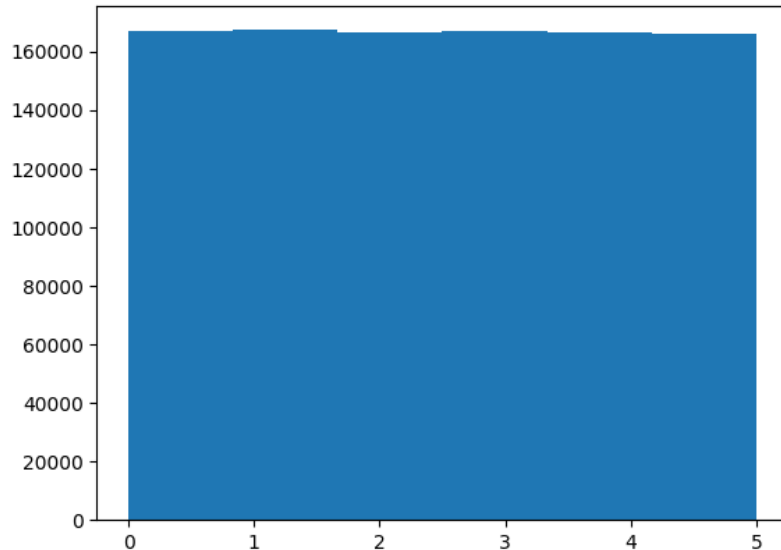
01. 숫자 데이터를 쉽게 다루도록 돕는 numpy 라이브러리

❖ ③ numpy 라이브러리를 활용해 그래프 그리기 (3/8)

- 1부터 6까지 숫자를 동일한 확률로 랜덤으로 추출한 결과를, 히스토그램으로 시각화하기

```
1 import matplotlib.pyplot as plt
2 import numpy as np
3 dice = np.random.choice(6, 1000000)
4 plt.figure()
5 plt.hist(dice, bins=6)
6 plt.show()
```

실행결과



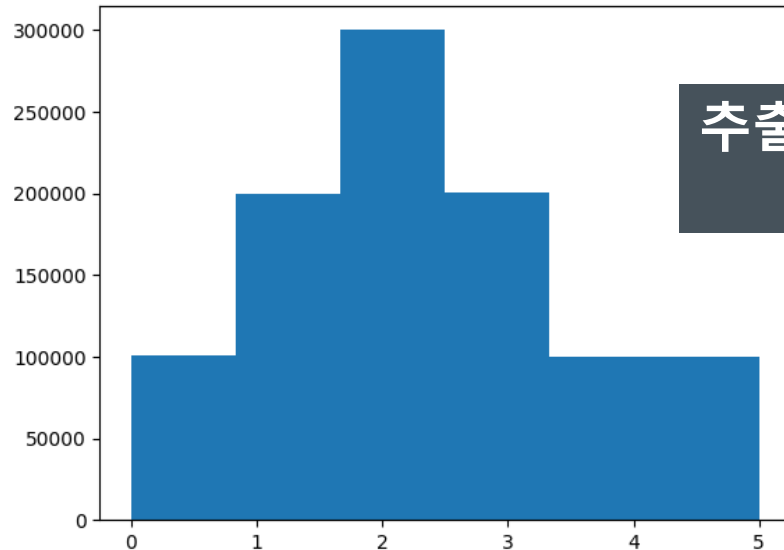
01. 숫자 데이터를 쉽게 다루도록 돕는 numpy 라이브러리

❖ ③ numpy 라이브러리를 활용해 그래프 그리기 (4/8)

- 1부터 6까지 숫자를 확률 값에 따라 랜덤으로 추출한 결과를, 히스토그램으로 시각화하기

```
1 import matplotlib.pyplot as plt
2 import numpy as np
3 dice = np.random.choice(6, 1000000, p=[0.1, 0.2, 0.3, 0.2, 0.1, 0.1])
4 plt.figure()
5 plt.hist(dice, bins=6)
6 plt.show()
```

실행결과



추출 시행 횟수를 늘리니, p 속성에 설정했던 확률 값에 따라 각 숫자가 추출됨을 확인 할 수 있음(큰 수의 법칙)

01. 숫자 데이터를 쉽게 다루도록 돕는 numpy 라이브러리

❖ ③ numpy 라이브러리를 활용해 그래프 그리기 (5/8)

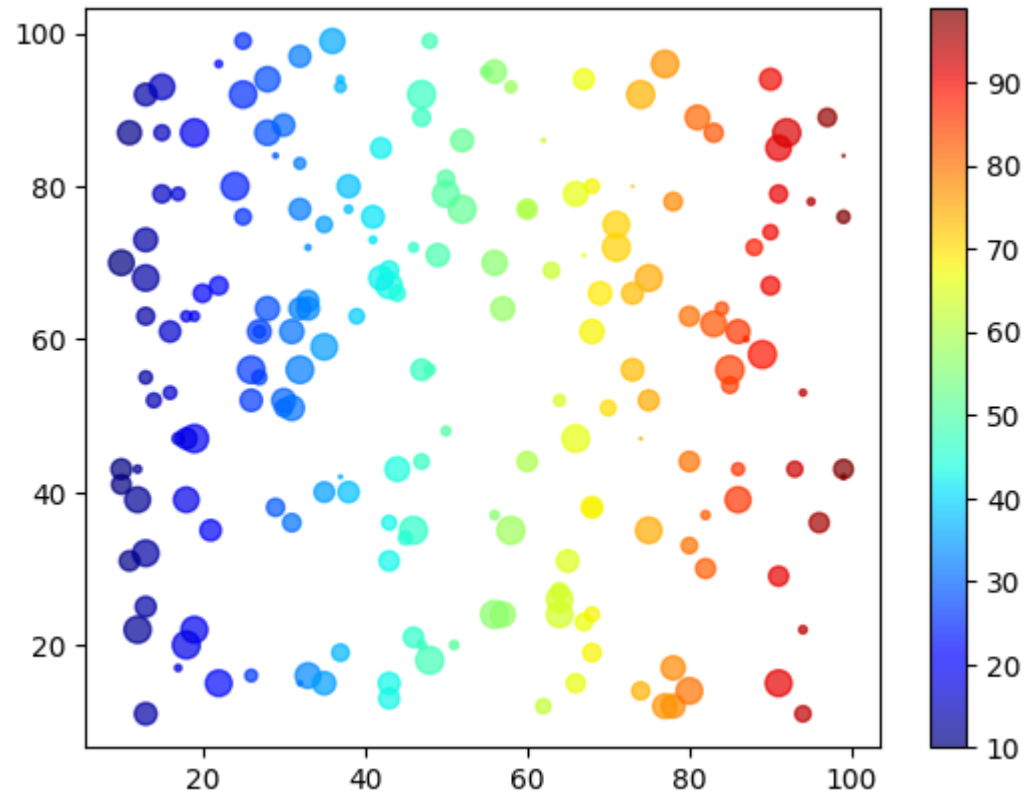
- numpy 라이브러리를 활용하여 작성한 코드

```
1 import matplotlib.pyplot as plt
2 import numpy as np
3
4 x = np.random.randint(10, 100, 200)      # 10이상 100미만의 정수 중 200개를 뽑음
5 y = np.random.randint(10, 100, 200)
6 size = np.random.rand(200) * 100        # 0과 1사이의 실수 200개를 뽑음
7
8 plt.figure()
9 plt.scatter(x, y, s=size, c=x, cmap="jet", alpha=0.7)
10 plt.colorbar()
11 plt.show()
```

01. 숫자 데이터를 쉽게 다루도록 돕는 numpy 라이브러리

❖ ③ numpy 라이브러리를 활용해 그래프 그리기 (6/8)

실행결과



버블 차트

01. 숫자 데이터를 쉽게 다루도록 돕는 numpy 라이브러리

❖ ③ numpy 라이브러리를 활용해 그래프 그리기 (7/8)

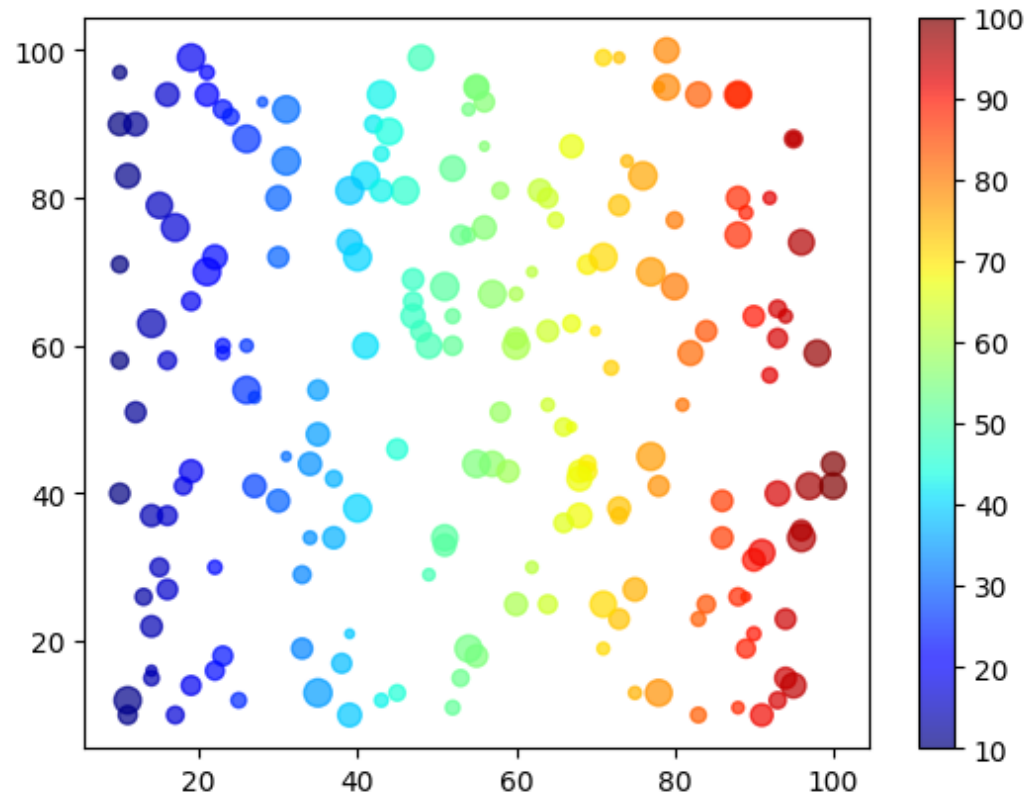
- 파이썬 리스트를 활용하여 작성한 코드

```
1 import matplotlib.pyplot as plt
2 import random
3
4 x = []
5 y = []
6 size = []
7
8 for _ in range(200):
9     x.append(random.randint(10, 100))      # 10이상 100이하의 정수를 임의로 뽑음
10    y.append(random.randint(10, 100))
11    size.append(random.randint(10, 100))
12
13 plt.figure()
14 plt.scatter(x, y, s=size, c=x, cmap="jet", alpha=0.7)
15 plt.colorbar()
16 plt.show()
```

01. 숫자 데이터를 쉽게 다루도록 돕는 numpy 라이브러리

❖ ③ numpy 라이브러리를 활용해 그래프 그리기 (8/8)

실행결과



01. 숫자 데이터를 쉽게 다루도록 돕는 numpy 라이브러리

❖ ④ numpy array 생성하기 (1/6)

- 리스트로 ndarray(N-Dimensional Array) 만들기

```
1 import numpy as np
2 a = np.array([1, 2, 3, 4])
3 print(a)
```

실행결과

```
[1 2 3 4]
```


01. 숫자 데이터를 쉽게 다루도록 돕는 numpy 라이브러리

❖ ④ numpy array 생성하기 (2/6)

- numpy array의 인덱싱(Indexing), 슬라이싱(Slicing)

```
1 import numpy as np
2 a = np.array([1, 2, 3, 4])
3 print(a[1], a[-1])
4 print(a[1:])
```

실행결과

```
2 4
[2 3 4]
```

01. 숫자 데이터를 쉽게 다루도록 돕는 numpy 라이브러리

❖ ④ numpy array 생성하기 (3/6)

- zeros(), ones(), eye() 함수로 numpy array 만들기

```
1 import numpy as np
2
3 a = np.zeros(10)
4 print(a)
5
6 b = np.ones(10)
7 print(b)
8
9 c = np.eye(3)    # 3행 x 3열의 단위 행렬 생
10 print(c)
```

실행결과

```
[0.  0.  0.  0.  0.  0.  0.  0.  0.  0.]
[1.  1.  1.  1.  1.  1.  1.  1.  1.  1.]
[[1.  0.  0.]
 [0.  1.  0.]
 [0.  0.  1.]]
```

01. 숫자 데이터를 쉽게 다루도록 돕는 numpy 라이브러리

❖ ④ numpy array 생성하기 (4/6)

- 연속된 숫자의 numpy array 만들기

```
1 import numpy as np
2 print(np.arange(3))
3 print(np.arange(3, 7))
4 print(np.arange(3, 7, 2))
```

실행결과

```
[0 1 2]
[3 4 5 6]
[3 5]
```

01. 숫자 데이터를 쉽게 다루도록 돕는 numpy 라이브러리

❖ ④ numpy array 생성하기 (5/6)

- 연속된 실수의 numpy array 만들기

```
1 import numpy as np
2 a = np.arange(1, 2, 0.1)      # 0이상 2미만 구간에서 0.1 간격으로 실수 생성
3 b = np.linspace(1, 2, 11)    # 1부터 2까지를 11개의 구간으로 나눈 실수 생성
4 print(a)
5 print(b)
```

실행결과

```
[1.  1.1  1.2  1.3  1.4  1.5  1.6  1.7  1.8  1.9]
[1.  1.1  1.2  1.3  1.4  1.5  1.6  1.7  1.8  1.9  2. ]
```

01. 숫자 데이터를 쉽게 다루도록 돕는 numpy 라이브러리

❖ ④ numpy array 생성하기 (6/6)

- 연속된 실수의 numpy array 만들기

```
1 import numpy as np
2 a = np.arange(-np.pi, np.pi, np.pi/10)
3 b = np.linspace(-np.pi, np.pi, 20)
4 print(a)
5 print(b)
```

실행결과

```
[-3.14159265 -2.82743339 -2.51327412 -2.19911486 -1.88495559 -1.57079633 -
 1.25663706 -0.9424778 -0.62831853 -0.31415927 0. 0.31415927 0.62831853
 0.9424778 1.25663706 1.57079633 1.88495559 2.19911486 2.51327412 2.82743339]
[-3.14159265 -2.81089869 -2.48020473 -2.14951076 -1.8188168 -1.48812284 -
 1.15742887 -0.82673491 -0.49604095 -0.16534698 0.16534698 0.49604095 0.82673491
 1.15742887 1.48812284 1.8188168 2.14951076 2.48020473 2.81089869 3.14159265]
```

01. 숫자 데이터를 쉽게 다루도록 돕는 numpy 라이브러리

❖ ⑤ numpy array의 다양한 활용 (1/9)

- numpy array에 값을 한꺼번에 더하기

```
1 import numpy as np
2 a = np.zeros(10) + 5
3 print(a)
```

실행결과

```
[5. 5. 5. 5. 5. 5. 5. 5. 5. 5.]
```

01. 숫자 데이터를 쉽게 다루도록 돕는 numpy 라이브러리

❖ ⑤ numpy array의 다양한 활용 (2/9)

- numpy array에 함수 적용하기

```
1 import numpy as np
2 a = np.linspace(1, 2, 11)
3 print(np.sqrt(a))
```

실행결과

```
[1.  1.04880885 1.09544512 1.14017543 1.18321596 1.22474487 1.26491106
 1.30384048 1.34164079 1.37840488 1.41421356]
```

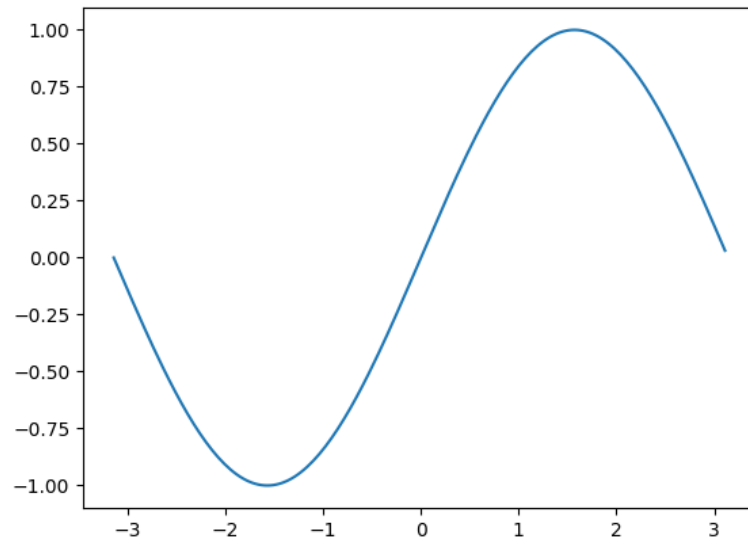
01. 숫자 데이터를 쉽게 다루도록 돕는 numpy 라이브러리

❖ ⑤ numpy array의 다양한 활용 (3/9)

- numpy를 활용한 그래프 그리기

```
1 import matplotlib.pyplot as plt
2 import numpy as np
3 a = np.arange(-np.pi, np.pi, np.pi/100)
4 plt.figure()
5 plt.plot(a, np.sin(a))
6 plt.show()
```

실행결과



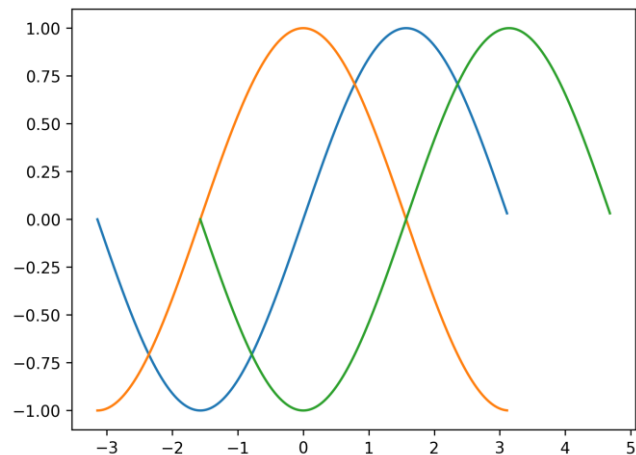
01. 숫자 데이터를 쉽게 다루도록 돕는 numpy 라이브러리

❖ ⑤ numpy array의 다양한 활용 (4/9)

- numpy를 활용한 다양한 그래프 그리기

```
1 import matplotlib.pyplot as plt
2 import numpy as np
3 a = np.arange(-np.pi, np.pi, np.pi/100)
4 plt.figure()
5 plt.plot(a, np.sin(a))
6 plt.plot(a, np.cos(a))
7 plt.plot(a + np.pi/2, np.sin(a))
8 plt.show()
```

실행결과



01. 숫자 데이터를 쉽게 다루도록 돕는 numpy 라이브러리

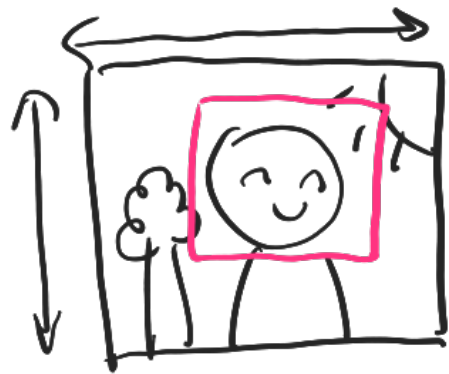
❖ ⑤ numpy array의 다양한 활용 (5/9)

- 마스크(Mask) 만들고 적용하기

```
1 import numpy as np
2 a = np.arange(-5, 5)      # 데이터 만들기
3 print(a)
4 print(a < 0)              # 마스크(Mask) 만들기
5 print(a[a < 0])           # 마스크 적용하기
```

실행결과

```
[-5 -4 -3 -2 -1  0  1  2  3  4]
[ True True True True True False False False False False]
[-5 -4 -3 -2 -1]
```



2D Array = $\begin{bmatrix} 0 & 100 & \dots \\ 80 & & \end{bmatrix}$

B

mask = $\begin{bmatrix} 000 & 111 & 000 \\ 000 & 111 & 000 \\ 000 & 111 & 000 \\ 000 & 000 & 000 \end{bmatrix}$

B[mask] = Face

01. 숫자 데이터를 쉽게 다루도록 돕는 numpy 라이브러리

❖ ⑤ numpy array의 다양한 활용 (6/9)

- 마스크(Mask) 연결해서 사용하기

```
1 import numpy as np
2 a = np.arange(-5, 5)      # 데이터 만들기
3 print(a)
4
5 mask1 = abs(a) > 3
6 print(mask1)
7 print(a[mask1])
8
9 mask2 = abs(a) % 2 == 0
10 print(mask2)
11 print(a[mask2])
12
13 print(a[mask1 + mask2])  # OR 연산 (둘 중 하나의 조건이라도 참이면 참)
14 print(a[mask1 * mask2]) # AND 연산 (두 가지 조건이 모두 참이면 참)
```

01. 숫자 데이터를 쉽게 다루도록 돕는 numpy 라이브러리

❖ ⑤ numpy array의 다양한 활용 (7/9)

실행결과

```
[-5 -4 -3 -2 -1 0 1 2 3 4]
[ True True False False False False False False False True]
[-5 -4 4]
[False True False True False True False True False True]
[-4 -2 0 2 4]
[-5 -4 -2 0 2 4]
[-4 4]
```

01. 숫자 데이터를 쉽게 다루도록 돕는 numpy 라이브러리

❖ ⑤ numpy array의 다양한 활용 (8/9)

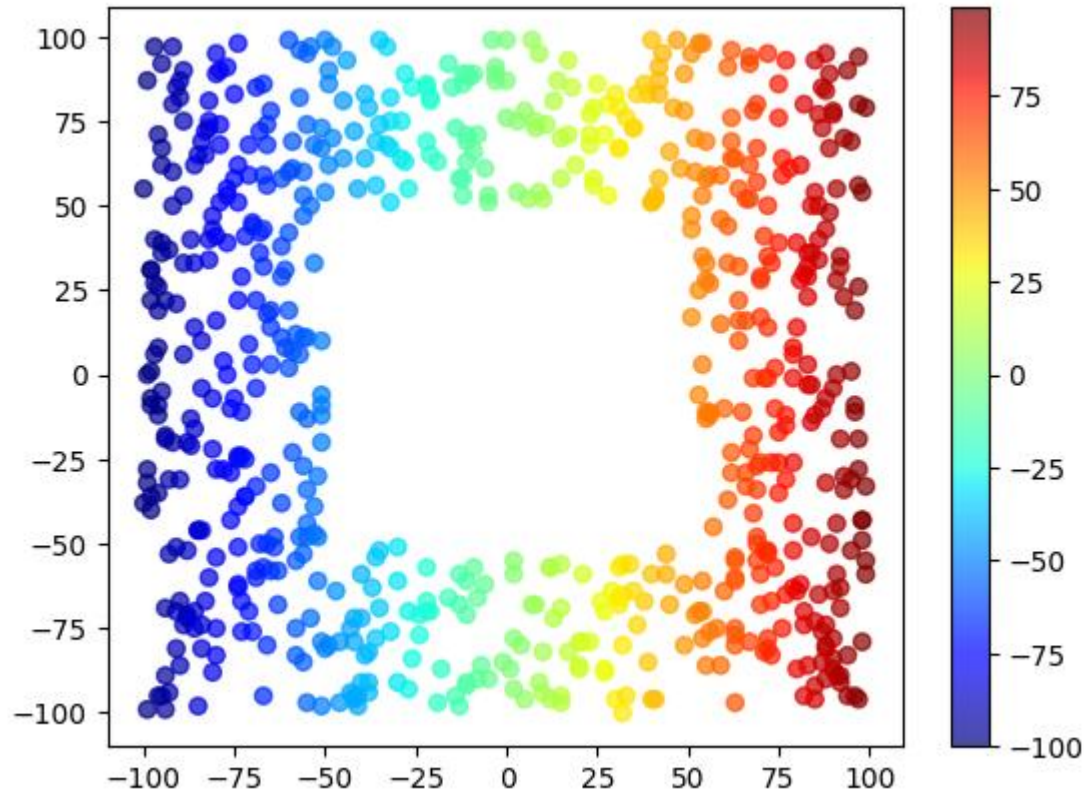
- numpy 라이브러리를 사용하여 버블 차트를 그려 보자

```
1 import matplotlib.pyplot as plt
2 import numpy as np
3 x = np.random.randint(-100, 100, 1000)    # 1000개의 랜덤 값 추출
4 y = np.random.randint(-100, 100, 1000)    # 1000개의 랜덤 값 추출
5
6 mask1 = abs(x) > 50                        # x에 저장된 값 중 절댓값이 50보다 큰 값 걸러 냄
7 mask2 = abs(y) > 50                        # y에 저장된 값 중 절댓값이 50보다 큰 값 걸러 냄
8 x = x[mask1 + mask2]                      # 둘 중 하나라도 참이면 포함
9 y = y[mask1 + mask2]
10
11 plt.figure()
12 plt.scatter(x, y, c=x, cmap="jet", alpha = 0.7)
13 plt.colorbar()
14 plt.show()
```

01. 숫자 데이터를 쉽게 다루도록 돕는 numpy 라이브러리

❖ ⑤ numpy array의 다양한 활용 (9/9)

실행결과



02. numpy를 활용한 나만의 프로젝트 만들기

01. 숫자 데이터를 쉽게 다루도록 돕는 numpy 라이브러리

02. numpy를 활용한 나만의 프로젝트 만들기

❖ ① 관심 있는 데이터 찾기

- 공공데이터포털(<https://www.data.go.kr/>) 등에서 관심 있는 데이터 찾기



02. numpy를 활용한 나만의 프로젝트 만들기

❖ ② 데이터 살펴보고 질문하기 (1/2)

- 데이터를 살펴보는 방법
 - ◆ 엑셀 같은 스프레드시트 프로그램 등을 활용해 데이터 자세히 살펴보기
 - ◆ 데이터가 담고 있는 내용(또는 담고 있지 않은 내용)
 - ◆ 데이터가 기록된 기간은 언제부터 언제까지인지
 - ◆ 어떤 형태로 시각화해보면 어떤 정보들을 알 수 있을지 생각해보기
 - ◆ 데이터를 보며 궁금한 내용들 자유롭게 질문하기

02. numpy를 활용한 나만의 프로젝트 만들기

❖ ② 데이터 살펴보며 질문하기 (2/2)

- age.csv의 인구 데이터를 보며 떠오른 질문들 예시
 - ◆ 전국에서 영유아들이 가장 많이 사는 지역은 어디일까?
 - ◆ 보통 학군이 좋다고 알려진 지역에는 청소년들이 많이 살까?
 - ◆ 광역시 데이터를 10년 단위로 살펴보면 청년 비율이 줄고 있다는 사실을 알 수 있을까?
 - ◆ 서울에서 지난 5년간 인구가 가장 많이 증가한 구는 어디일까?
 - ◆ 우리 동네의 인구 구조와 가장 비슷한 동네는 어디일까?

영유아들?

청소년들?

청년비율?


인구 구조와 가장 비슷한 동네?

문제를 명확히 정의할 필요가 있음

02. numpy를 활용한 나만의 프로젝트 만들기

❖ ③ 질문을 명확한 문제로 정의하기

- 예를 들어, 아래와 같이 문제를 좀 더 명확하게 정의할 수 있음
 - ◆ 전국에서 영유아들이 가장 많이 사는 지역은 어디일까?
 - 전국에 있는 읍면동 중 만 0세 이상 6세 이하의 인구 비율이 높은 상위 10곳은?
 - ◆ 우리 동네의 인구 구조와 가장 비슷한 동네는 어디일까?
 - 전국에서 우리 동네의 연령별 인구 구조와 가장 형태가 비슷한 지역은 어디일까?



위 문제를 해결하기 위해서,
알고리즘을 어떻게 설계해야 할까?

02. numpy를 활용한 나만의 프로젝트 만들기

❖ ④ 알고리즘 설계하기

- [문제] 전국에서 우리 동네의 연령별 인구 구조와 가장 형태가 비슷한 지역은 어디일까?
 - ◆ Step 1) 데이터를 읽음
 - ◆ Step 2) 궁금한 지역의 이름을 입력 받음
 - ◆ Step 3) 궁금한 지역의 인구 구조를 저장함
 - ◆ Step 4) 궁금한 지역의 인구 구조와 가장 비슷한 인구 구조를 가진 지역을 찾음
 - ◆ Step 5) 가장 비슷한 곳의 인구 구조와 궁금한 지역의 인구 구조를 시각화함

numpy를 활용하여 궁금한 지역의 인구 데이터를 출력하는
알고리즘을 코드로 작성해 보자

02. numpy를 활용한 나만의 프로젝트 만들기

❖ Step 1) 데이터 읽어 오기 (1/2)

```
1 import csv
2
3 f = open("age.csv", encoding="cp949")
4 data = csv.reader(f)
5 header = next(data)
6 print(header)
7
8 for row in data:
9     print(row)
10    break
11
12 f.close()
```

02. numpy를 활용한 나만의 프로젝트 만들기

❖ Step 1) 데이터 읽어 오기 (2/2)

실행결과

```
[ '행정구역', '2022년08월_계_총인구수', '2022년08월_계_연령구간인구수', '2022년08월_계_0세', '2022년08월_계_1세', '2022년08월_계_2세', '2022년08월_계_3세', '2022년08월_계_4세', '2022년08월_계_5세', '2022년08월_계_6세', '2022년08월_계_7세', '2022년08월_계_8세', '2022년08월_계_9세', '2022년08월_계_10세', '2022년08월_계_11세', ... (중략) ... '2022년08월_계_94세', '2022년08월_계_95세', '2022년08월_계_96세', '2022년08월_계_97세', '2022년08월_계_98세', '2022년08월_계_99세', '2022년08월_계_100세 이상' ] [ '서울특별시 (1100000000)', '9,488,454', '9,488,454', '40,931', '44,731', '45,511', '49,245', '52,278', '57,054', '64,993', '67,389', '66,458', '69,728', ... (중략) ... '6,636', '5,631', '4,113', '2,990', '1,896', '1,333', '1,013', '950', '1,706' ]
```

문자열 자료형이고 숫자 사이에 쉼표(,)가 있음

02. numpy를 활용한 나만의 프로젝트 만들기

❖ Step 2) 궁금한 지역의 이름 입력 받기

```
1 import csv
2
3 f = open("age.csv", encoding="cp949")
4 data = csv.reader(f)
5 header = next(data)
6
7 name = input("인구 구조가 알고 싶은 지역의 이름(읍면동 단위)를 입력하세요: ")
8 for row in data:
9     print(row)
10    break
11
12 f.close()
```

02. numpy를 활용한 나만의 프로젝트 만들기

❖ Step 3) 궁금한 지역의 인구 구조를 리스트에 저장하기

```
1 import csv
2
3 f = open("age.csv", encoding="cp949")
4 data = csv.reader(f)
5 header = next(data)
6
7 home = []          # 입력받은 지역의 데이터를 저장할 리스트 생성
8 name = input("인구 구조가 알고 싶은 지역의 이름(읍면동 단위)를 입력하세요: ")
9 for row in data:
10     if name in row[0]:
11         for j in row[3:]:
12             home.append(int(j))
13
14 f.close()
15 print(home)
```

리스트가 아니라
numpy를 활용해 보자

02. numpy를 활용한 나만의 프로젝트 만들기

❖ Step 3) 궁금한 지역의 인구 구조를 numpy array에 저장하기

```
1 import csv
2 import numpy as np
3
4 f = open("age.csv", encoding="cp949")
5 data = csv.reader(f)
6 header = next(data)
7
8 name = input("인구 구조가 알고 싶은 지역의 이름(읍면동 단위)를 입력하세요: ")
9 for row in data:
10     if name in row[0]:
11         home = np.array(row[3:], dtype="int")
12
13 f.close()
14 print(home)
```

문자열 자료형을 정수형 자료형으로
변환하기 위해서 dtype 속성을 int로 설정함

실행결과

인구 구조가 알고 싶은 지역의 이름(읍면동 단위)를 입력하세요: [신도림](#)
[253 270 269 312 291 313 390 412 386 369 396 404 380 373 387 324 324 319 322
... (중략) ...
112 93 92 83 72 75 46 43 31 19 22 6 15 13 4 8 8 2 3 2]

02. numpy를 활용한 나만의 프로젝트 만들기

❖ Step 3) 궁금한 지역의 인구 구조 시각화하기 (1/2)

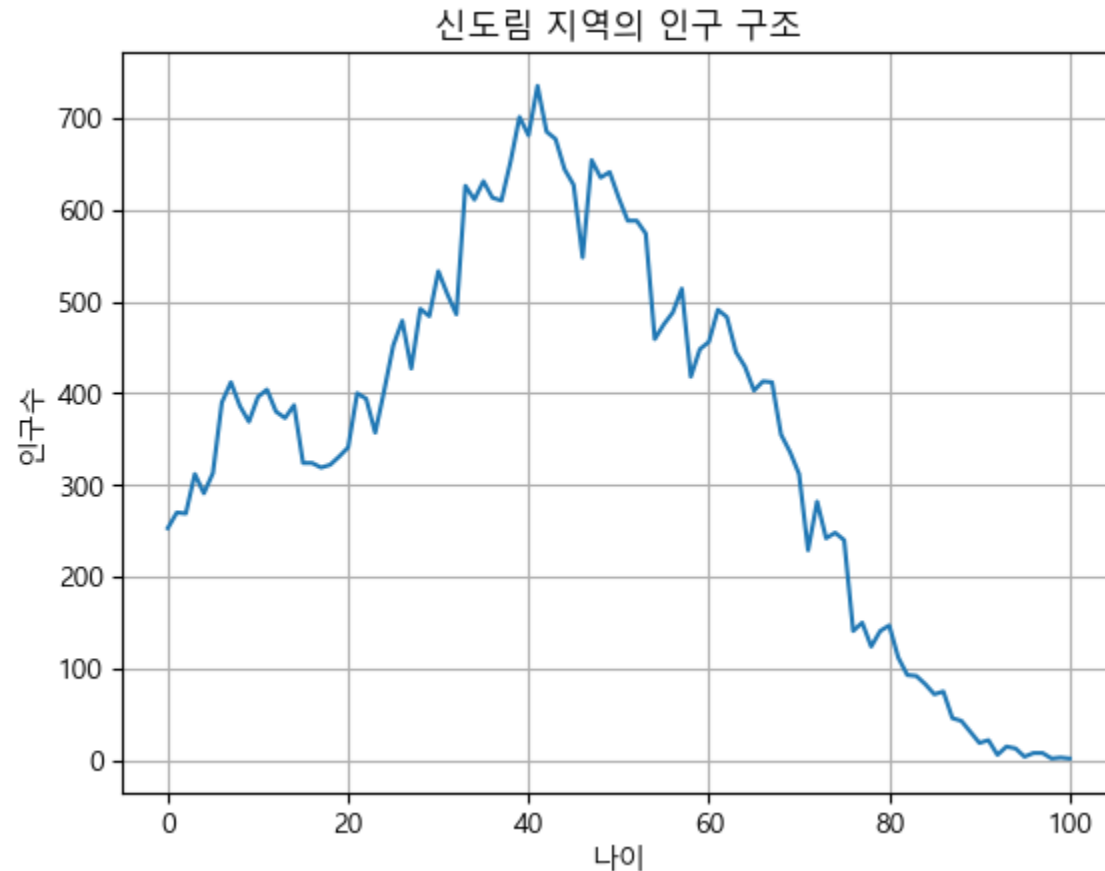
```
1 import csv
2 import numpy as np
3 import matplotlib.pyplot as plt
4 f = open("age.csv", encoding="cp949")
5 data = csv.reader(f)
6 header = next(data)
7 name = input("인구 구조가 알고 싶은 지역의 이름(읍면동 단위)를 입력하세요: ")
8 for row in data:
9     if name in row[0]:
10         home = np.array(row[3:], dtype="int")
11
12 f.close()
13 plt.rc("font", family="Malgun Gothic")
14 plt.figure()
15 plt.title(name + " 지역의 인구 구조")
16 plt.plot(home)
17 plt.xlabel("나이")
18 plt.ylabel("인구수")
19 plt.grid()
20 plt.show()
```

02. numpy를 활용한 나만의 프로젝트 만들기

❖ Step 3) 궁금한 지역의 인구 구조 시각화하기 (2/2)

실행결과

인구 구조가 알고 싶은 지역의 이름(읍면동 단위)를 입력하세요: [신도림](#)



02. numpy를 활용한 나만의 프로젝트 만들기

❖ Step 4) 궁금한 지역의 인구 구조와 가장 비슷한 인구 구조를 가진 지역 찾기 (1/17)

- 내가 알고자 하는 궁금한 지역: A
- 비교할 지역(= A지역을 제외한 나머지 지역): B

A와 B의 인구 구조가 비슷하다는 것을
어떻게 알 수 있을까?

02. numpy를 활용한 나만의 프로젝트 만들기

❖ Step 4) 궁금한 지역의 인구 구조와 가장 비슷한 인구 구조를 가진 지역 찾기 (2/17)

알고리즘 버전 1

- ① 전국의 모든 지역 중 한 곳(B)을 선택함
- ② 궁금한 지역 A의 0세 인구수에서 B의 0세 인구수를 뺌
- ③ ②를 "100세 이상 인구수"에 해당하는 값까지 반복한 후 각각의 차이를 모두 더함
- ④ 전국의 모든 지역에 대해 반복하며, 그 차이가 가장 작은 지역을 찾음

비슷한 인구 구조를 찾기 위한 간단한 방법으로,
우선 두 지역의 연령별 인구수 차이를 구해 모두 더해보면 어떨까?

02. numpy를 활용한 나만의 프로젝트 만들기

❖ Step 4) 궁금한 지역의 인구 구조와 가장 비슷한 인구 구조를 가진 지역 찾기 (3/17)

```
1 import csv
2 import numpy as np
3 import matplotlib.pyplot as plt
4
5 f = open("age.csv", encoding="cp949")
6 data = csv.reader(f)
7 header = next(data)
8
9 name = input("인구 구조가 알고 싶은 지역의 이름(읍면동 단위)를 입력하세요: ")
10 for row in data:
11     if name in row[0]:
12         home = np.array(row[3:], dtype="int")
13         if "화춘면" in row[0]:
14             homeB = np.array(row[3:], dtype="int")
15
16 f.close()
17
18
19
20
```

02. numpy를 활용한 나만의 프로젝트 만들기

❖ Step 4) 궁금한 지역의 인구 구조와 가장 비슷한 인구 구조를 가진 지역 찾기 (4/17)

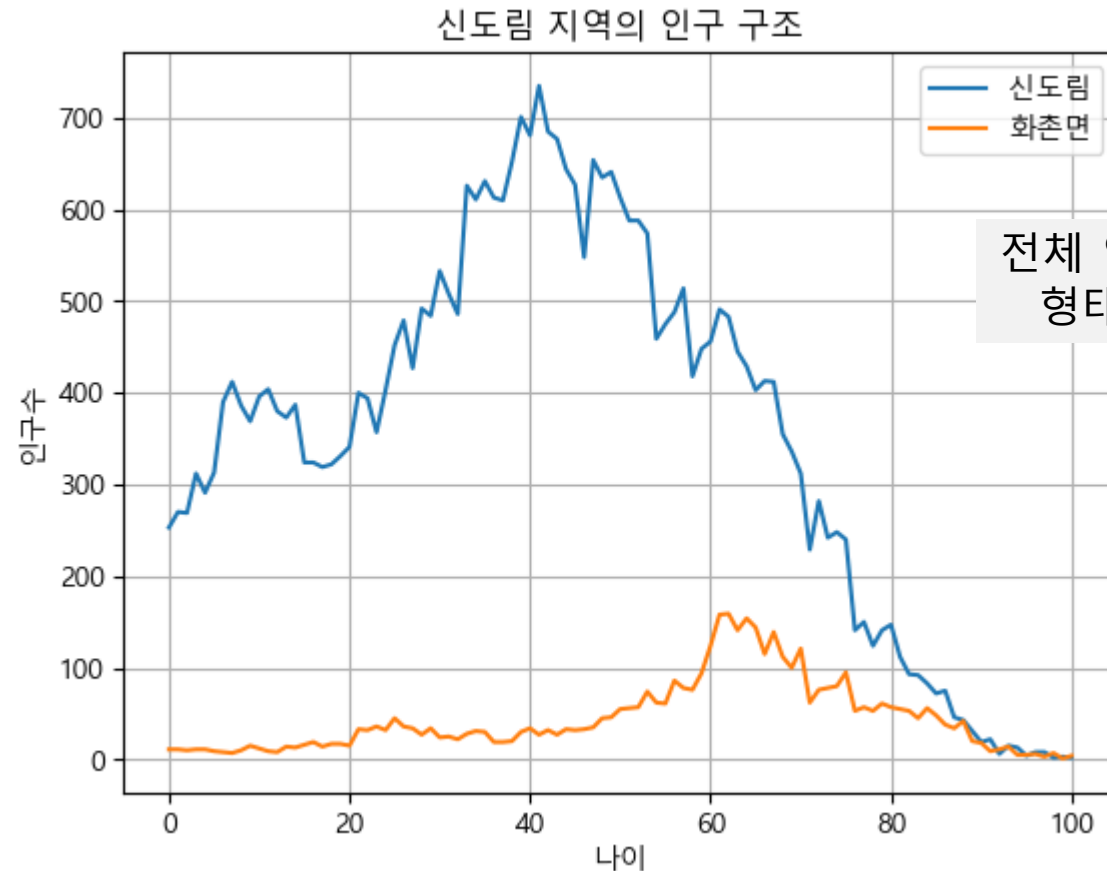
```
21 plt.rc("font", family="Malgun Gothic")
22 plt.figure()
23 plt.title(name + " 지역의 인구 구조")
24 plt.plot(home, label=name)
25 plt.plot(homeB, label="화춘면")
26 plt.xlabel("나이")
27 plt.ylabel("인구수")
28 plt.legend()
29 plt.grid()
30 plt.show()
```

02. numpy를 활용한 나만의 프로젝트 만들기

❖ Step 4) 궁금한 지역의 인구 구조와 가장 비슷한 인구 구조를 가진 지역 찾기 (5/17)

실행결과

인구 구조가 알고 싶은 지역의 이름(읍면동 단위)를 입력하세요: **신도림**



02. numpy를 활용한 나만의 프로젝트 만들기

❖ Step 4) 궁금한 지역의 인구 구조와 가장 비슷한 인구 구조를 가진 지역 찾기 (6/17)

알고리즘 버전 2

- ① 전국의 모든 지역 중 한 곳(B)을 선택함
- ② 궁금한 지역 A의 0세 **인구 비율**에서 B의 0세 **인구 비율**을 뺌
- ③ ②를 "100세 이상 인구수"에 해당하는 값까지 반복한 후 각각의 차이를 모두 더함
- ④ 전국의 모든 지역에 대해 반복하며, 그 차이가 가장 작은 지역을 찾음

"인구수" → "인구 비율"로 변경

02. numpy를 활용한 나만의 프로젝트 만들기

❖ Step 4) 궁금한 지역의 인구 구조와 가장 비슷한 인구 구조를 가진 지역 찾기 (7/17)

```
1 import csv
2 import numpy as np
3 import matplotlib.pyplot as plt
4
5 f = open("age.csv", encoding="cp949")
6 data = csv.reader(f)
7 header = next(data)
8
9 name = input("인구 구조가 알고 싶은 지역의 이름(읍면동 단위)를 입력하세요: ")
10 for row in data:
11     for j in range(0, 102, 1):
12         row[j + 2] = int(row[j + 2].replace(', ', ''))
13     if name in row[0]:
14         home = np.array(row[3:], dtype="int") / row[2]
15     if "화춘면" in row[0]:
16         homeB = np.array(row[3:], dtype="int") / row[2]
17
18 f.close()
19
20
```

02. numpy를 활용한 나만의 프로젝트 만들기

❖ Step 4) 궁금한 지역의 인구 구조와 가장 비슷한 인구 구조를 가진 지역 찾기 (8/17)

```
21 plt.rc("font", family="Malgun Gothic")
22 plt.figure()
23 plt.title(name + " 지역의 인구 구조")
24 plt.plot(home, label=name)
25 plt.plot(homeB, label="화춘면")
26 plt.xlabel("나이")
27 plt.ylabel("인구 비율")
28 plt.legend()
29 plt.grid()
30 plt.show()
```

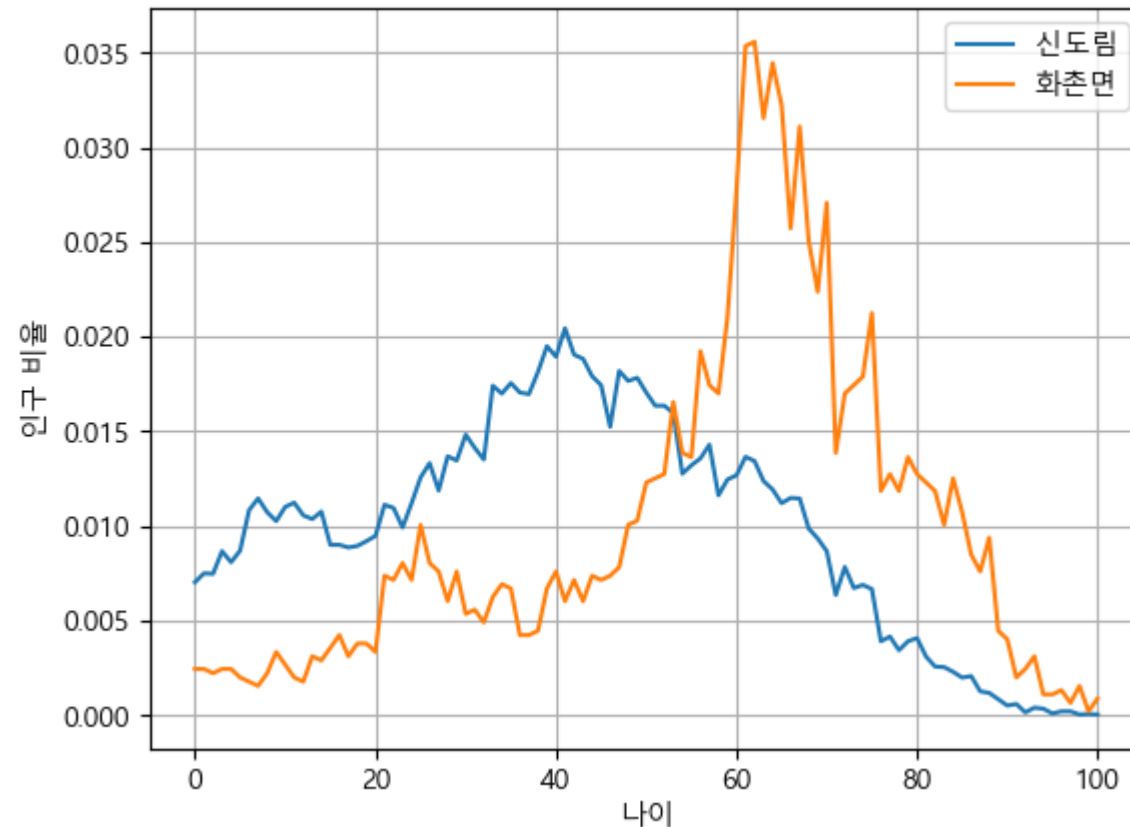
02. numpy를 활용한 나만의 프로젝트 만들기

❖ Step 4) 궁금한 지역의 인구 구조와 가장 비슷한 인구 구조를 가진 지역 찾기 (9/17)

실행결과

인구 구조가 알고 싶은 지역의 이름(읍면동 단위)를 입력하세요: **신도림**

신도림 지역의 인구 구조



전체 인구수가 다르더라도
인구 구조를 비교할 수 있을 것 같음

02. numpy를 활용한 나만의 프로젝트 만들기

❖ Step 4) 궁금한 지역의 인구 구조와 가장 비슷한 인구 구조를 가진 지역 찾기 (10/17)

```
1 import csv
2 import numpy as np
3 import matplotlib.pyplot as plt
4
5 f = open("age.csv", encoding="cp949")
6 data = csv.reader(f)
7 header = next(data)
8
9 name = input("인구 구조가 알고 싶은 지역의 이름(읍면동 단위)를 입력하세요: ")
10 for row in data:
11     for j in range(0, 102, 1):
12         row[j + 2] = int(row[j + 2].replace(', ', ''))
13     if name in row[0]:
14         home = np.array(row[3:], dtype="int") / row[2]
15
16 for row in data:
17     print(row)
```

실행해 보면, 아무것도 출력되지 않음!

02. numpy를 활용한 나만의 프로젝트 만들기

❖ Step 4) 궁금한 지역의 인구 구조와 가장 비슷한 인구 구조를 가진 지역 찾기 (11/17)

```
1 import csv
2 import numpy as np
3 import matplotlib.pyplot as plt
4
5 f = open("age.csv", encoding="cp949")
6 data = csv.reader(f)
7 header = next(data)
8
9 print(type(data))
10 data = list(data)
11 print(type(data))
12
13 name = input("인구 구조가 알고 싶은 지역의 이름(읍면동 단위)를 입력하세요: ")
14 for row in data:
15     for j in range(0, 102, 1):
16         row[j + 2] = int(row[j + 2].replace(',',''))
17     if name in row[0]:
18         home = np.array(row[3:], dtype="int") / row[2]
```

`_csv.reader: 일회용 <--> list: 다회용`

`data`의 자료형을 확인하고,
리스트로 저장하자

02. numpy를 활용한 나만의 프로젝트 만들기

❖ Step 4) 궁금한 지역의 인구 구조와 가장 비슷한 인구 구조를 가진 지역 찾기 (12/17)

```
21 for row in data:
22     print(row)
23
24 f.close()
```

실행결과

```
<class '_csv.reader'>
```

```
<class 'list'>
```

인구 구조가 알고 싶은 지역의 이름(읍면동 단위)를 입력하세요: [신도림](#)

```
['서울특별시 (1100000000)', '9,488,454', 9488454, 40931, 44731, 45511, 49245,
52278, 57054, 64993, 67389, 66458, 69728, 73178, 74211, 70537, 70854, 77007,
76198, 71798, 72776, 81292, 84479, 92605, 109564, 122792, 128581, 140859,
151748, 160960, 163689, 168698, 171640, 169823, 155282, 148124, 142339, 134772,
132666, 132963, 131530, 136684, 145139, 153385, 154158, 153713, 142498, 133753,
136239, 134513, 143303, 155439, 158351, 161030, 169789, 159870, 162879, 151879,
141272, 138092, 141719, 132797, 146195, 136622, 157609, 153686, 140019, 132035,
130199, 120257, 125862, 102156, 95014, 96639, 72168, 81170, 79765, 78781, 77931,
55964, 58999, 55081, 57654, 58697, 46452, 40065, 37030, 31920, 27601, 23645,
20122, 15883, 12847, 10513, 7820, 6636, 5631, 4113, 2990, 1896, 1333, 1013, 950,
1706] ... (중략) ...
```

02. numpy를 활용한 나만의 프로젝트 만들기

❖ Step 4) 궁금한 지역의 인구 구조와 가장 비슷한 인구 구조를 가진 지역 찾기 (13/17)

```
1 import csv
2 import numpy as np
3 import matplotlib.pyplot as plt
4 f = open("age.csv", encoding="cp949")
5 data = csv.reader(f)
6 header = next(data)
7 data = list(data)
8 name = input("인구 구조가 알고 싶은 지역의 이름(읍면동 단위)를 입력하세요: ")
9 for row in data:
10     for j in range(0, 102, 1):
11         row[j + 2] = int(row[j + 2].replace(',', ''))
12     if name in row[0]:
13         home = np.array(row[3:], dtype="int") / row[2]
14
15 for row in data:
16     if name not in row[0]:
17         away = np.array(row[3:], dtype="int") / row[2]
18         print(home - away)
19
20 f.close()
```


02. numpy를 활용한 나만의 프로젝트 만들기

❖ Step 4) 궁금한 지역의 인구 구조와 가장 비슷한 인구 구조를 가진 지역 찾기 (14/17)

실행결과

인구 구조가 알고 싶은 지역의 이름(읍면동 단위)를 입력하세요: [신도림](#)

```
[ 2.72476195e-03  2.79721999e-03  2.68719454e-03  3.48993570e-03
 2.58605815e-03  2.69475574e-03  4.00021626e-03  4.35974502e-03
 3.73453696e-03  2.91696281e-03  3.30451059e-03  3.41820368e-03
 3.13772438e-03  2.90957339e-03  2.65058498e-03  9.83168793e-04
 1.44689028e-03  1.20471624e-03  3.90665219e-04  3.05165843e-04
-2.73040621e-04 -4.18974202e-04 -1.98001130e-03 -3.61947163e-03
-3.63373244e-03 -3.41814396e-03 -3.63786054e-03 -5.37212786e-03
-4.09171436e-03 -4.62433765e-03 -3.06964799e-03 -2.23266089e-03
-2.09031653e-03  2.41421118e-03  2.79440253e-03  3.57276212e-03
 3.04069586e-03  3.10826067e-03  3.73352599e-03  4.20563676e-03
 2.78017492e-03  4.20100266e-03  2.85688771e-03  3.81628830e-03
 3.81986548e-03  3.08491807e-03  1.06902116e-03  3.09158188e-03
... (중략) ...
```

3.05066470e-03은 $3.05066470 \times 10^{-3}$ 이라는 의미로
0.00305066470을 의미함

02. numpy를 활용한 나만의 프로젝트 만들기

❖ Step 4) 궁금한 지역의 인구 구조와 가장 비슷한 인구 구조를 가진 지역 찾기 (15/17)

```
1 import csv
2 import numpy as np
3 import matplotlib.pyplot as plt
4
5 f = open("age.csv", encoding="cp949")
6 data = csv.reader(f)
7 header = next(data)
8 data = list(data)
9
10 min_val = 1          # 최솟값을 저장할 변수
11 result_name = ''     # 최솟값을 갖는 지역의 이름을 저장할 변수
12 result = 0           # 최솟값을 갖는 지역의 연령대별 인구 비율을 저장할 배열
13
14 name = input("인구 구조가 알고 싶은 지역의 이름(읍면동 단위)를 입력하세요: ")
15 for row in data:
16     for j in range(0, 102, 1):
17         row[j + 2] = int(row[j + 2].replace(',', ''))
18     if name in row[0]:
19         home = np.array(row[3:], dtype="int") / row[2]
20
```

02. numpy를 활용한 나만의 프로젝트 만들기

❖ Step 4) 궁금한 지역의 인구 구조와 가장 비슷한 인구 구조를 가진 지역 찾기 (16/17)

```
21 for row in data:
22     if name not in row[0] and row[2] != 0: ← 총 인구수가 0명인 행정구역도 존재함
23         away = np.array(row[3:], dtype="int") / row[2]
24         s = np.sum(home - away)
25         if s < min_val:
26             min_val = s
27             result_name = row[0]
28             result = away
29
30 f.close()
31 plt.rc("font", family="Malgun Gothic")
32 plt.figure()
33 plt.title(name + " 지역과 가장 비슷한 인구 구조를 가진 지역: " + result_name)
34 plt.plot(home, label=name)
35 plt.plot(result, label=result_name)
36 plt.xlabel("나이")
37 plt.ylabel("인구 비율")
38 plt.legend()
39 plt.grid()
40 plt.show()
```

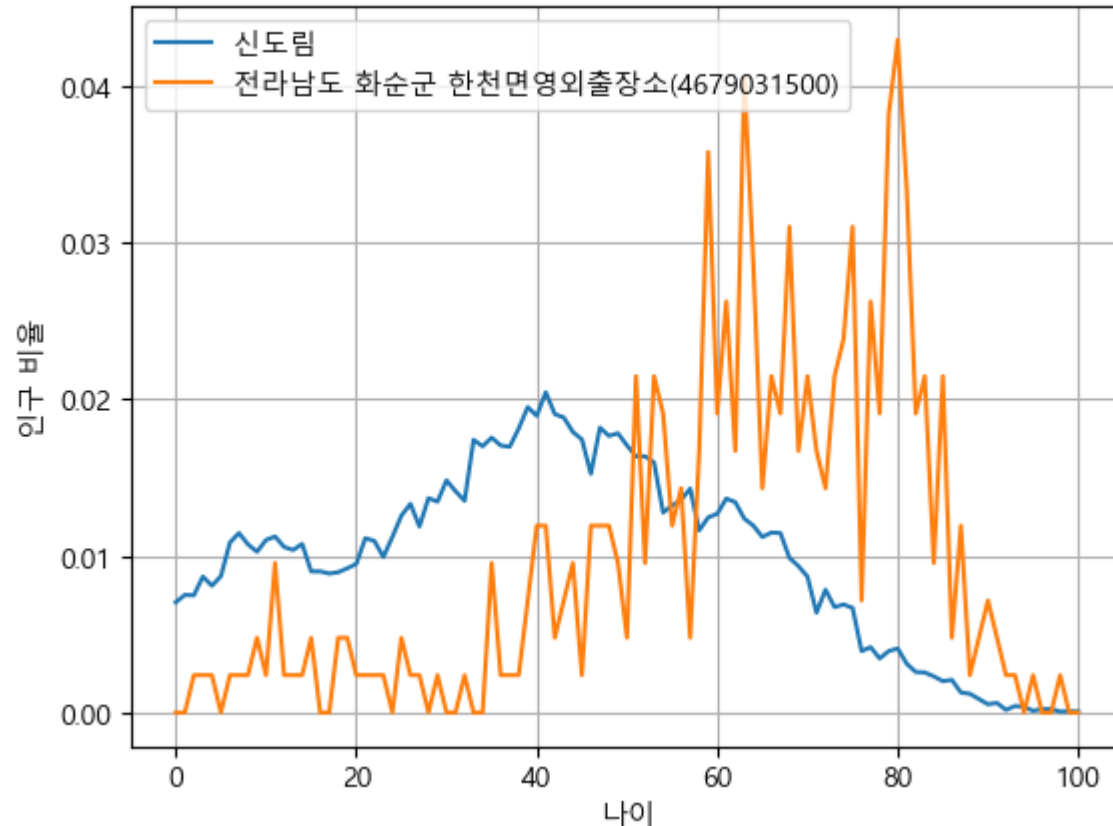
02. numpy를 활용한 나만의 프로젝트 만들기

❖ Step 4) 궁금한 지역의 인구 구조와 가장 비슷한 인구 구조를 가진 지역 찾기 (17/17)

실행결과

인구 구조가 알고 싶은 지역의 이름(읍면동 단위)를 입력하세요: **신도림**

신도림 지역과 가장 비슷한 인구 구조를 가진 지역: 전라남도 화순군 한천면영외출장소(4679031500)



인구 구조가 비슷해 보이지 않는데,
알고리즘에 문제가 있었던 걸까?

02. numpy를 활용한 나만의 프로젝트 만들기

❖ Step 5) 가장 비슷한 곳의 인구 구조와 궁금한 지역의 인구 구조 시각화하기 (1/4)

알고리즘 버전 2

- ① 전국의 모든 지역 중 한 곳(B)을 선택함
- ② 궁금한 지역 A의 0세 **인구 비율**에서 B의 0세 **인구 비율**을 뺌
- ③ ②를 "100세 이상 인구수"에 해당하는 값까지 반복한 후 각각의 차이를 모두 더함
- ④ 전국의 모든 지역에 대해 반복하며, 그 차이가 가장 작은 지역을 찾음

문제 원인을 발견했음!

예를 들어

- ✓ ②에서 0세에 대해 계산한 결과가 음수이고, 1세에 대해 계산한 결과가 양수였다고 가정해 보자
- ✓ 그러면 ③에서 차이를 더하는 과정에서 음수와 양수가 상쇄되는 현상이 발생함

02. numpy를 활용한 나만의 프로젝트 만들기

❖ Step 5) 가장 비슷한 곳의 인구 구조와 궁금한 지역의 인구 구조 시각화하기 (2/4)

알고리즘 버전 3

- ① 전국의 모든 지역 중 한 곳 (B)을 선택함
- ② 궁금한 지역 A의 0세 인구 비율에서 B의 0세 인구 비율을 뺌
- ③ ②를 "100세 이상 인구수"에 해당하는 값까지 반복한 후 각각의 **차이의 제곱**의 합을 구함
- ④ 전국의 모든 지역에 대해 반복하며, 그 차이가 가장 작은 지역을 찾음

“차이의 합” → “차이의 제곱의 합”으로 변경

02. numpy를 활용한 나만의 프로젝트 만들기

❖ Step 5) 가장 비슷한 곳의 인구 구조와 궁금한 지역의 인구 구조 시각화하기 (3/4)

```
21 for row in data:
22     if name not in row[0] and row[2] != 0:
23         away = np.array(row[3:], dtype="int") / row[2]
24         s = np.sum((home - away)**2)
25         if s < min_val:
26             min_val = s
27             result_name = row[0]
28             result = away
29
30 f.close()
31 plt.rc("font", family="Malgun Gothic")
32 plt.figure()
33 plt.title(name + " 지역과 가장 비슷한 인구 구조를 가진 지역: " + result_name)
34 plt.plot(home, label=name)
35 plt.plot(result, label=result_name)
36 plt.xlabel("나이")
37 plt.ylabel("인구 비율")
38 plt.legend()
39 plt.grid()
40 plt.show()
```

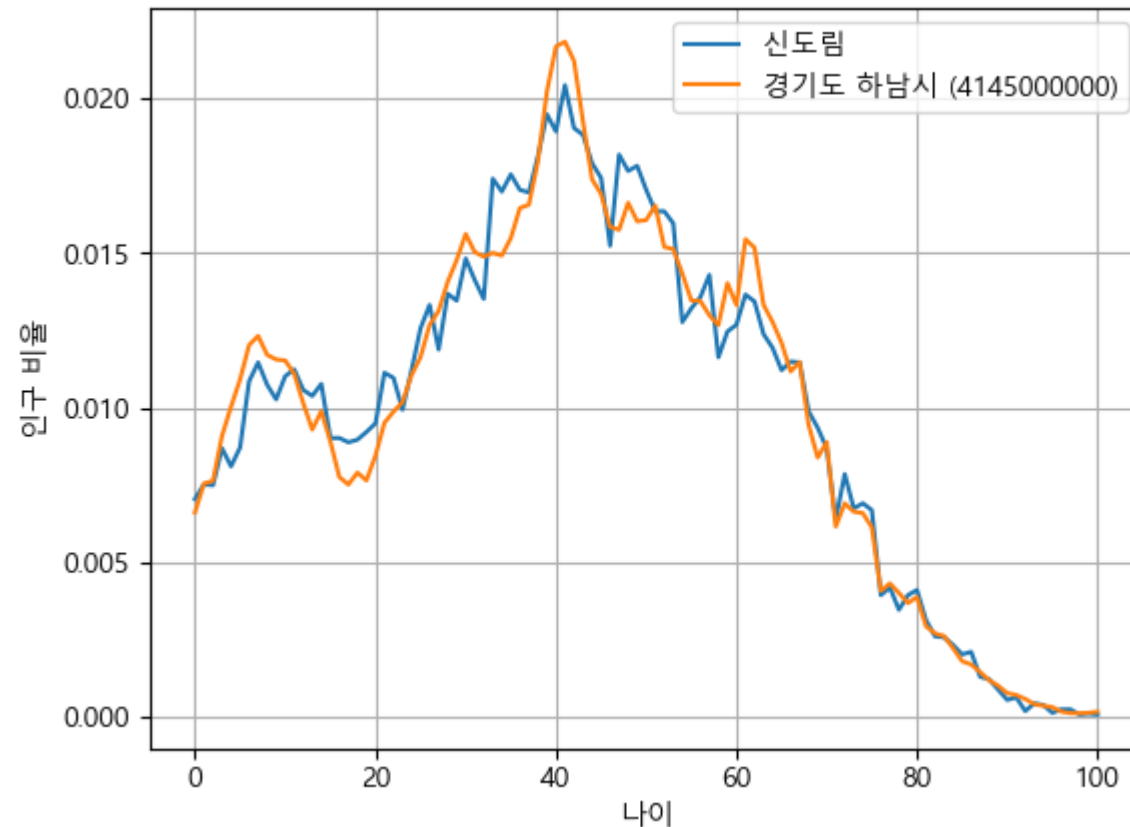
02. numpy를 활용한 나만의 프로젝트 만들기

❖ Step 5) 가장 비슷한 곳의 인구 구조와 궁금한 지역의 인구 구조 시각화하기 (4/4)

실행결과

인구 구조가 알고 싶은 지역의 이름(읍면동 단위)를 입력하세요: **신도림**

신도림 지역과 가장 비슷한 인구 구조를 가진 지역: 경기도 하남시 (4145000000)



인구 구조가 비슷해 보임!

- ❖ 01. 숫자 데이터를 쉽게 다루도록 돕는 numpy 라이브러리
- ❖ 02. numpy를 활용한 나만의 프로젝트 만들기

THANK YOU!

Q & A

- Name: 권범
- Office: 동덕여자대학교 인문관 B821호
- Phone: 02-940-4752
- E-mail: bkwon@dongduk.ac.kr