동양미래대학교

데이터분석입문

Lecture 11. NumPy 라이브러리 시작하기

동양미래대학교
인공지능소프트웨어학과
강 환수

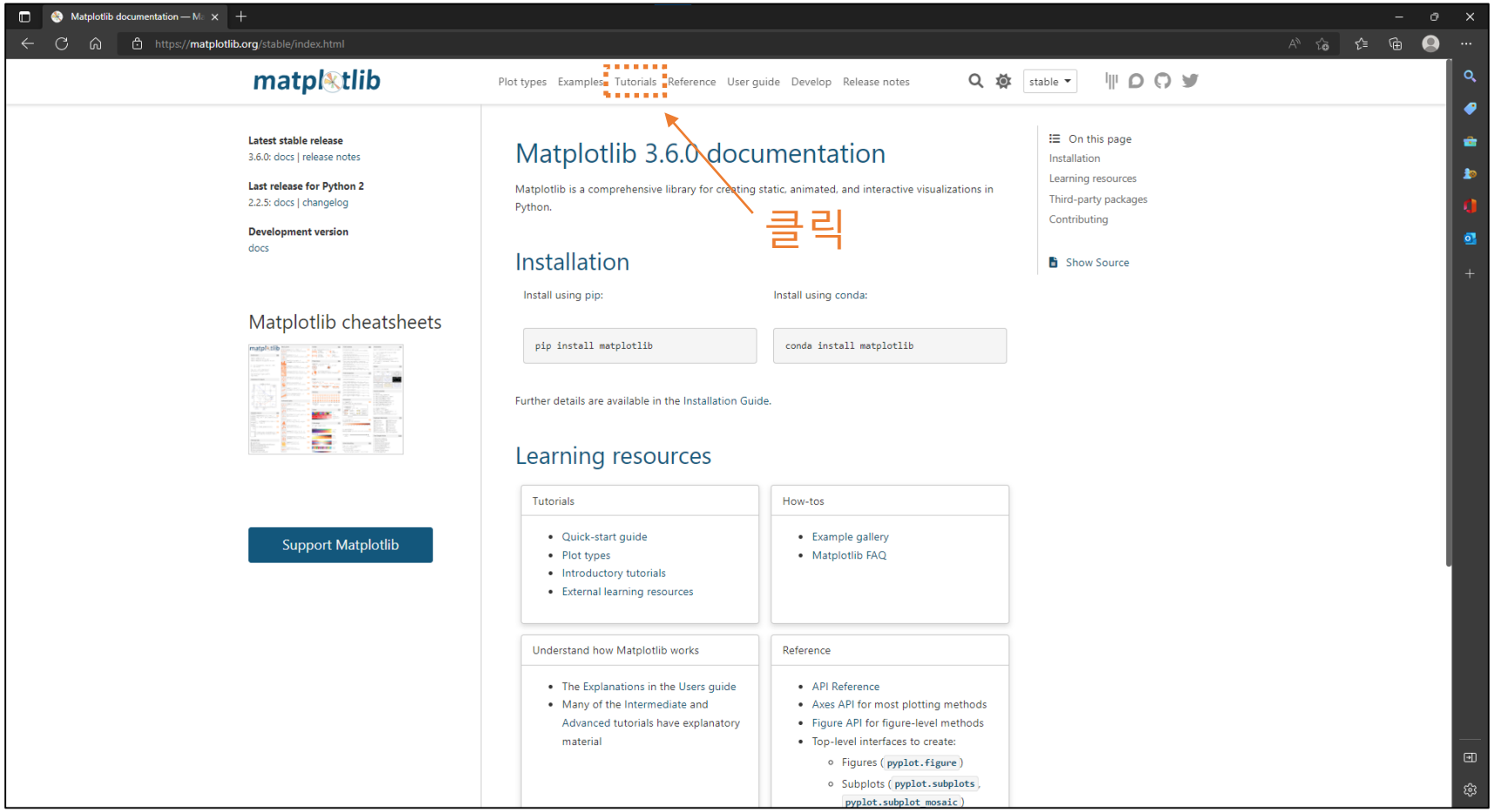
목차

01. 숫자 데이터를 쉽게 다루도록 돕는 numpy 라이브러리

02. numpy array 생성 및 활용 방법

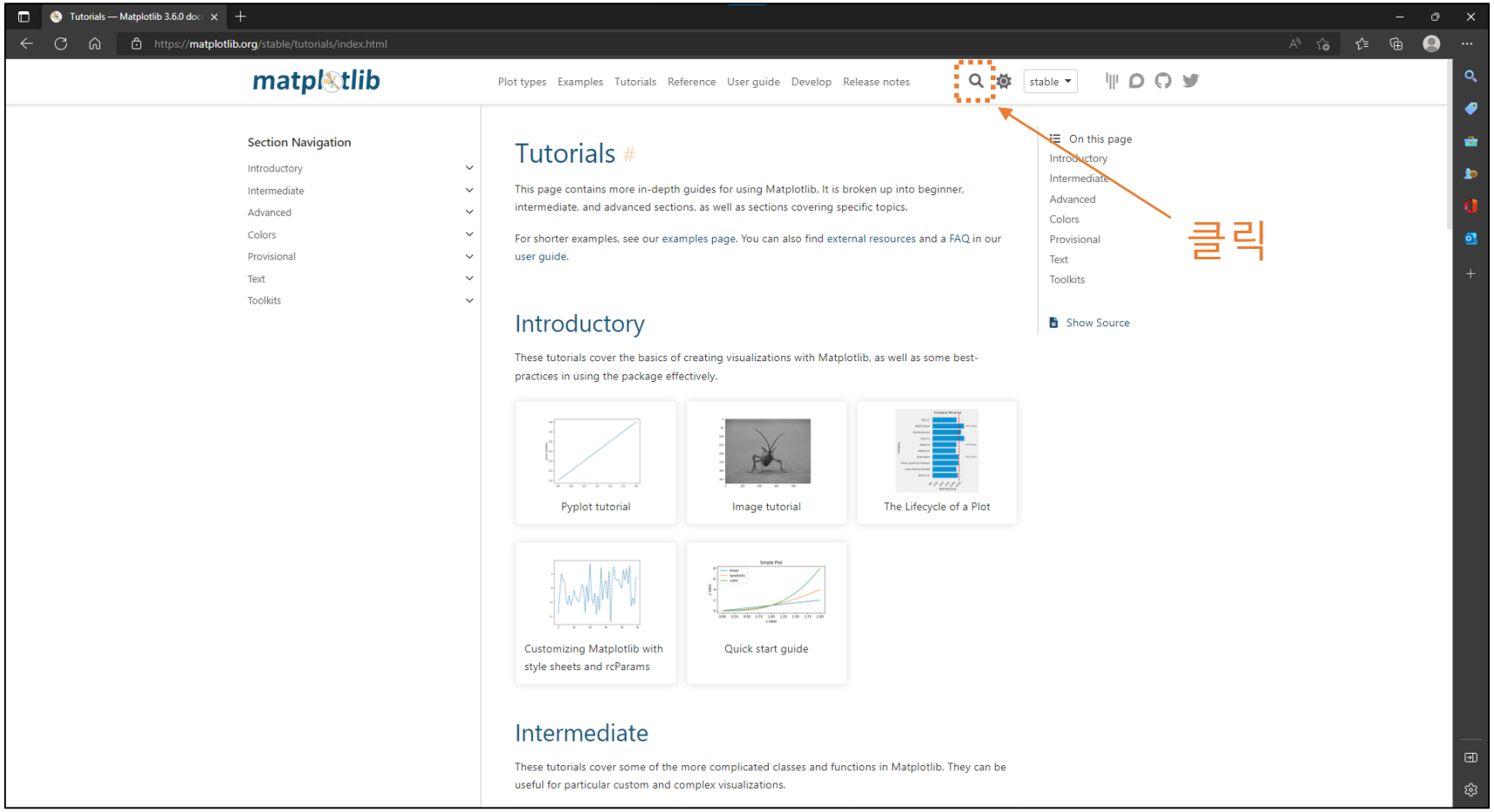
❖ ① matplotlib 홈페이지 (1/5)

- https://matplotlib.org에서 데이터 시각화와 관련된 다양한 예시를 확인 할 수 있습니다.
- [Tutorials] 버튼을 클릭하세요.



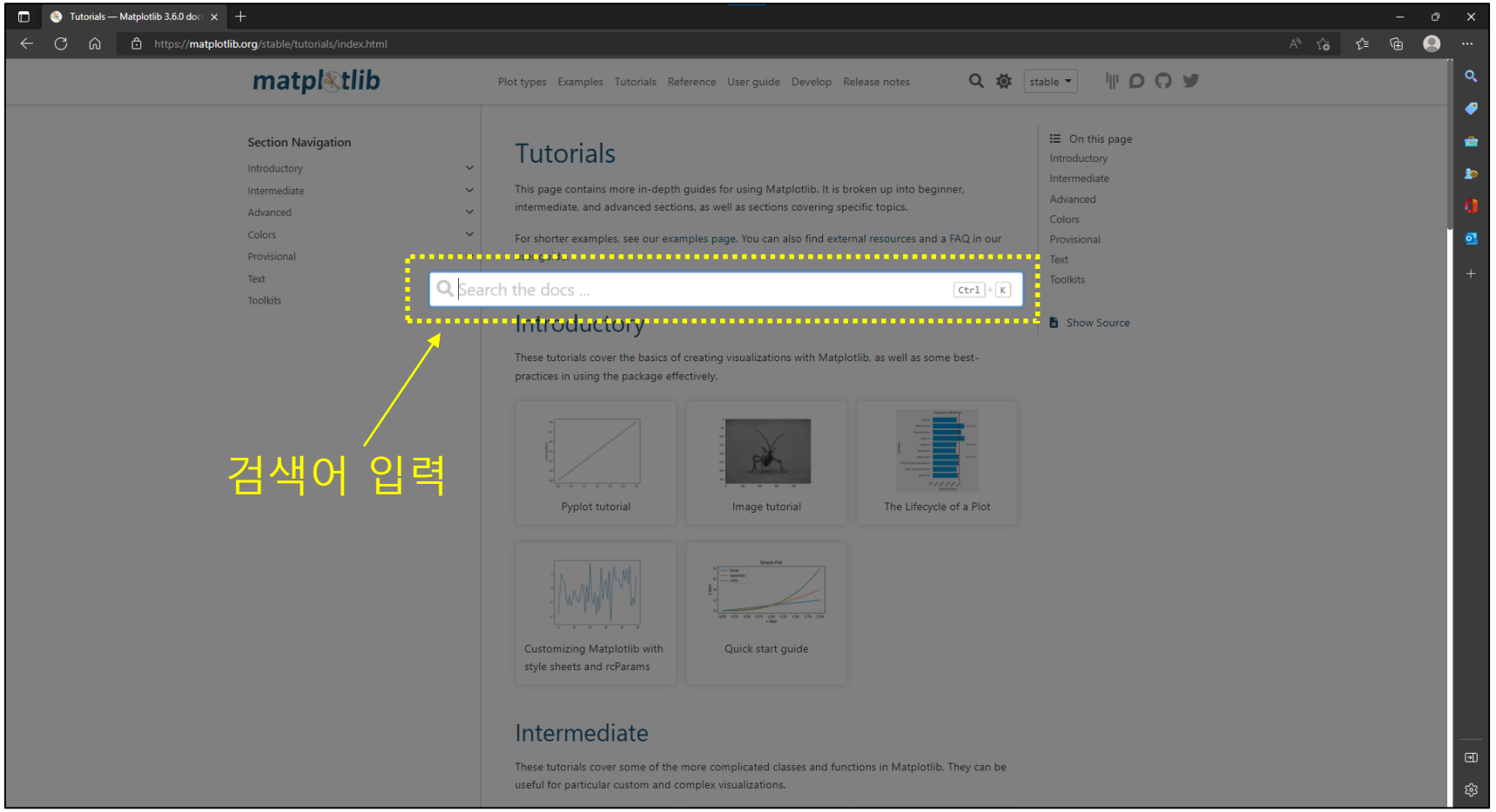
❖ ① matplotlib 홈페이지 (2/5)

- 우측 상단에 돋보기 버튼을 클릭하세요.



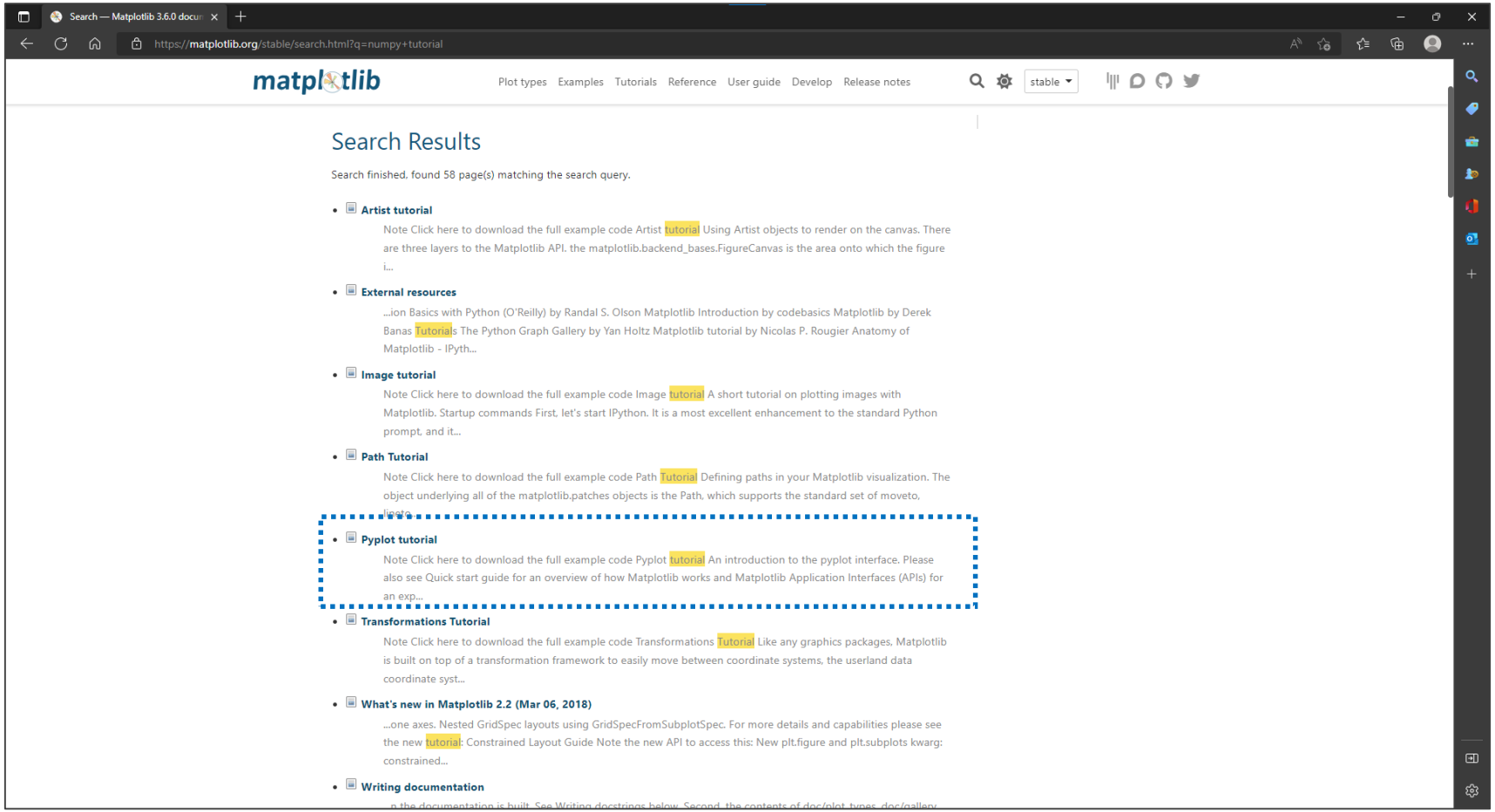
❖ ① matplotlib 홈페이지 (3/5)

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



❖ ① matplotlib 홈페이지 (4/5)

- "Pyplot tutorial"에 지금까지 배운 내용들이 잘 정리되어 있습니다.
- 내용을 같이 살펴보겠습니다.



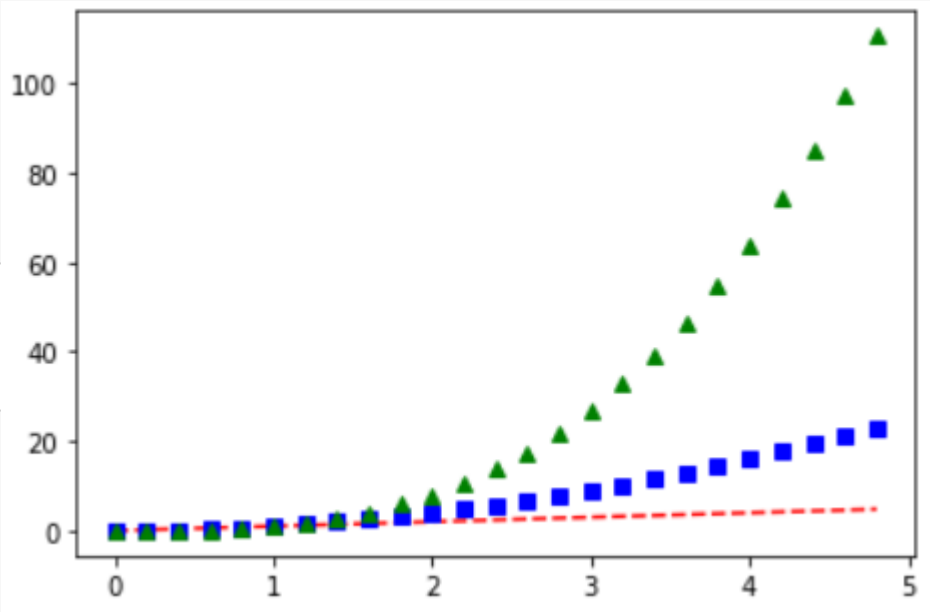
❖ ① matplotlib 홈페이지 (5/5)

- numpy 라이브러리를 활용하여 작성한 코드

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

t = np.arange(0., 5., 0.2)

plt.plot(t, t, 'r--', t, t**2, 'bs', t, t**3, 'g^')
plt.show()
```



- 파이썬 리스트를 활용하여 작성한 코드

```
import matplotlib.pyplot as plt

a = []
b = []
c = []

for i in range(0, 50, 2):
    a.append(i / 10)
    b.append((i / 10) ** 2)
    c.append((i / 10) ** 3)

plt.plot(a, a, 'r--', a, b, 'bs', a, c, 'g^')
plt.show()
```

numpy 라이브러리를 활용할 경우
보다 적은 수의 코드로 그리고 간결하게
원하는 결과를 얻을 수 있습니다.

❖ ② numpy 라이브러리 시작하기 (1/4)

- 제곱근(Square Root) 출력하기

```
import numpy

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

```
1.4142135623730951
```

- alias를 활용하여 제곱근 출력하기

```
import numpy as np

print(np.sqrt(2))
```

```
1.4142135623730951
```

❖ ② numpy 라이브러리 시작하기 (2/4)

- 파이와 삼각함수 활용하기

```
import numpy as np

print(np.pi)
print(np.sin(0))
print(np.cos(np.pi))
```

```
3.141592653589793
0.0
-1.0
```

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

```
import numpy as np

a = np.random.rand(5)
print(a)
print(type(a))
```

```
[0.66718898 0.66502759 0.2474137  0.91645236 0.24887168]
<class 'numpy.ndarray'>
```

random 라이브러리의
randint( ) 함수의 실수 버전이라고
생각하시면 됩니다.

❖ ② numpy 라이브러리 시작하기 (3/4)

- random 서브 라이브러리의 choice( ) 함수

```
import numpy as np

print(np.random.choice(6, 10))
```

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

0이상 6미만인 정수 중 10개를 뽑는다(중복 허용)

- random 서브 라이브러리의 choice( ) 함수(중복 금지)

```
import numpy as np

print(np.random.choice(10, 6, replace=False))
```

```
[5 8 6 7 2 9]
```

만약 한 번 뽑은 숫자를
다시 뽑지 못하게 하고 싶다면
replace 속성을 False로 설정하면 됩니다.

0이상 10미만인 정수 중 6개를 뽑는다(중복 허용 X)

❖ ② numpy 라이브러리 시작하기 (4/4)

- random 서브 라이브러리의 choice( ) 함수(확률 설정)

```
import numpy as np

print(np.random.choice(6, 10, p=[0.1, 0.2, 0.3, 0.2, 0.1, 0.1]))
```

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

p 속성은 각 경우의 수가 발생할 확률을 정할 수 있습니다.
주의할 점은 확률의 합은 반드시 1이어야 합니다.

02. numpy array 생성 및 활용 방법

01. 숫자 데이터를 쉽게 다루도록 돕는 numpy 라이브러리

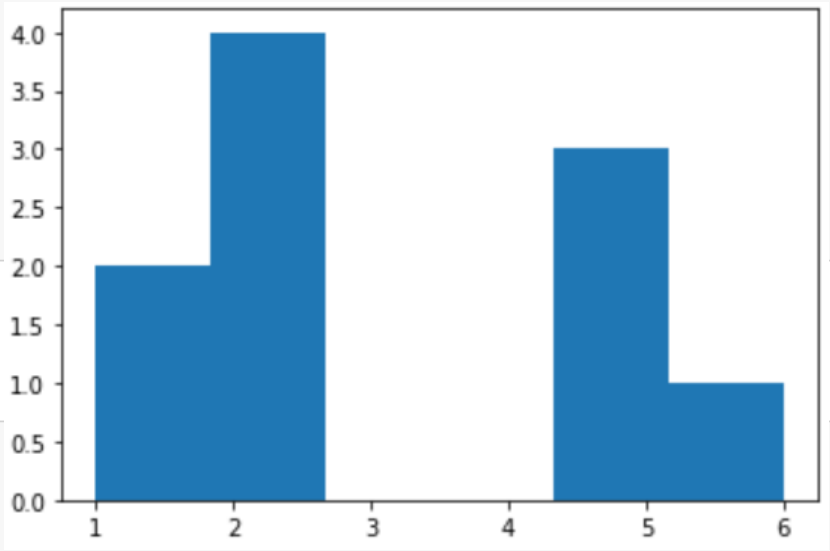
❖ ① numpy 라이브러리를 활용해 그래프 그리기 (1/3)

- numpy 라이브러리를 활용하여 작성한 코드

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

dice = np.random.choice(6, 10)

plt.hist(dice, bins=6)
plt.show()
```



- 파이썬 리스트를 활용하여 작성한 코드

```
import matplotlib.pyplot as plt
import random

dice = []
for i in range(10):
    dice.append(random.randint(1, 6))

plt.hist(dice, bins=6)
plt.show()
```

numpy 라이브러리를 활용할 경우 장점!
① 코드가 간결해짐
② 코드 실행 속도가 빠름

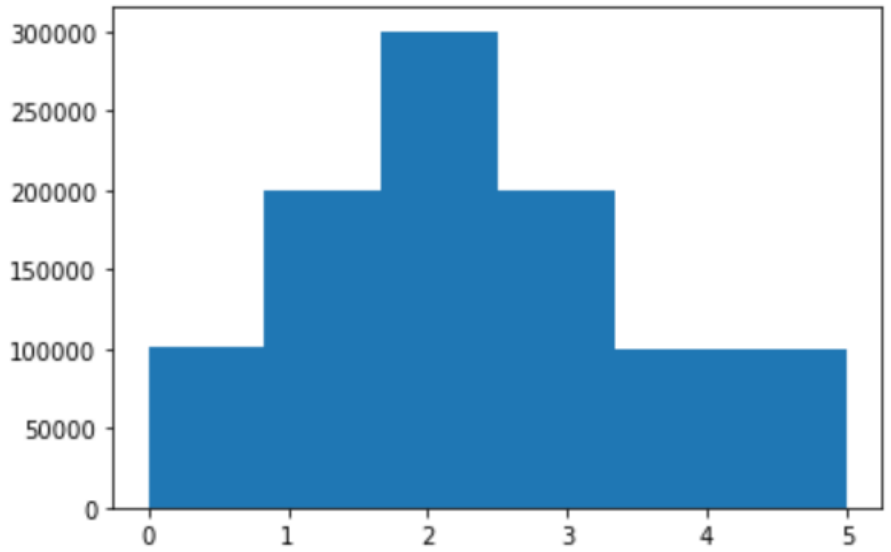
❖ ① numpy 라이브러리를 활용해 그래프 그리기 (2/3)

- 1부터 6까지 숫자를 랜덤으로 추출한 결과를 히스토그램으로 시각화하기

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

dice = np.random.choice(6, 1000000, p=[0.1, 0.2, 0.3, 0.2, 0.1, 0.1])

plt.hist(dice, bins=6)
plt.show()
```



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

❖ ① numpy 라이브러리를 활용해 그래프 그리기 (3/3)

- numpy 라이브러리를 활용하여 작성한 코드

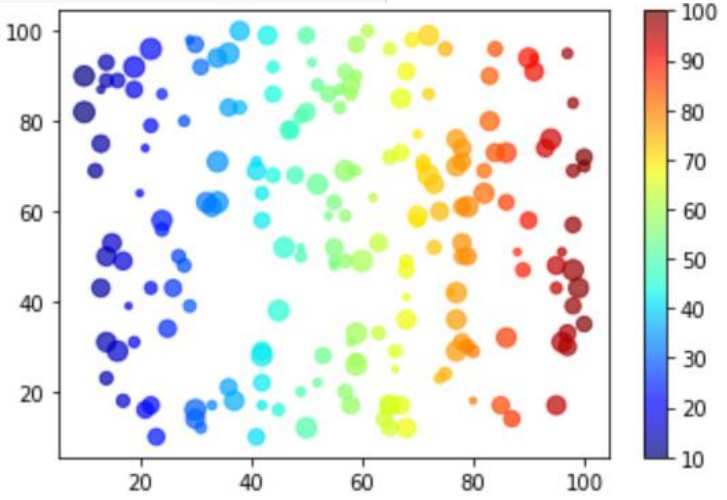
```
import matplotlib.pyplot as plt
import numpy as np

x = np.random.randint(10, 100, 200)
y = np.random.randint(10, 100, 200)
size = np.random.rand(200) * 100

plt.scatter(x, y, s = size, c = x, cmap = 'jet', alpha = 0.7)
plt.colorbar()
plt.show()
```

10이상 100미만의 정수 중 200개를 뽑는다.

0과 1사이의 실수 200개를 뽑는다.



- 파이썬 리스트를 활용하여 작성한 코드

```
import matplotlib.pyplot as plt
import random

x = []
y = []
size = []

for i in range(200):
    x.append(random.randint(10, 100))
    y.append(random.randint(10, 100))
    size.append(random.randint(10, 100))

plt.scatter(x, y, s = size, c = x, cmap = 'jet', alpha = 0.7)
plt.colorbar()
plt.show()
```

10이상 100이하의 정수를 임의로 뽑는다.

❖ ② numpy array 생성하기 (1/4)

- 리스트로 ndarray(N–Dimensional Array) 만들기

```
import numpy as np

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

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

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

```
import numpy as np

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

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

❖ ② numpy array 생성하기 (2/4)

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

```
import numpy as np

a = np.zeros(10)
print(a)

b = np.ones(10)
print(b)

c = np.eye(3)
print(c)
```

3행 × 3열의 단위 행렬 생성

```
[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.]]
```

❖ ② numpy array 생성하기 (3/4)

- 연속된 숫자의 numpy array 만들기

```
import numpy as np

print(np.arange(3))
print(np.arange(3, 7))
print(np.arange(3, 7, 2))
```

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

- 연속된 실수의 numpy array 만들기

```
import numpy as np

a = np.arange(1, 2, 0.1)    # 1이상 2미만 구간에서 0.1 간격으로 실수 생성
b = np.linspace(1, 2, 11)   # 1부터 2까지 11개 구간으로 나눈 실수 생성
print(a)
print(b)
```

```
[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. ]
```

❖ ② numpy array 생성하기 (4/4)

- 연속된 실수의 numpy array 만들기

```
import numpy as np

a = np.arange(-np.pi, np.pi, np.pi/10)
b = np.linspace(-np.pi, np.pi, 20)
print(a)
print(b)
```

```
[-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]
```

❖ ③ numpy array의 다양한 활용 (1/6)

- numpy array에 값을 한꺼번에 더하기

```
import numpy as np

a = np.zeros(10) + 5
print(a)
```

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

- numpy array에 함수 적용하기

```
import numpy as np

a = np.linspace(1, 2, 11)
print(np.sqrt(a))
```

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

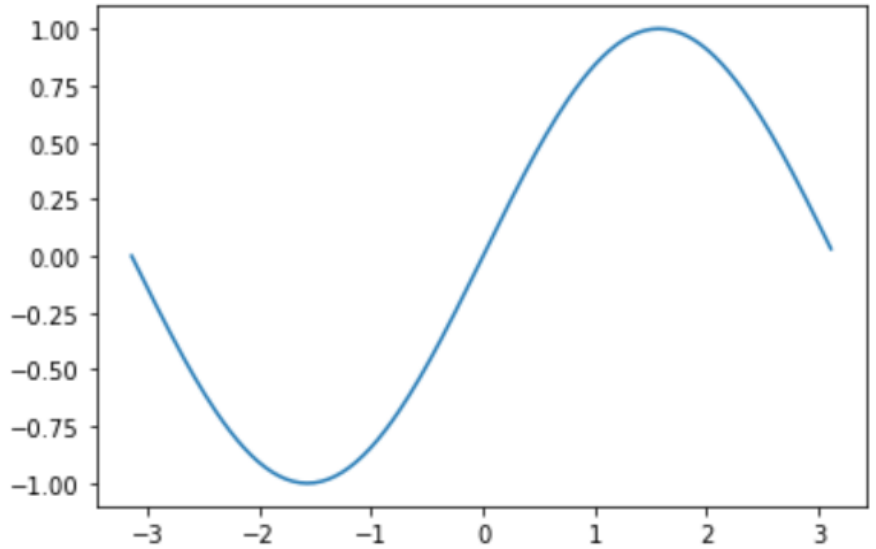
❖ ③ numpy array의 다양한 활용 (2/6)

- numpy를 활용한 그래프 그리기

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

a = np.arange(-np.pi, np.pi, np.pi/100)

plt.plot(a, np.sin(a))
plt.show()
```



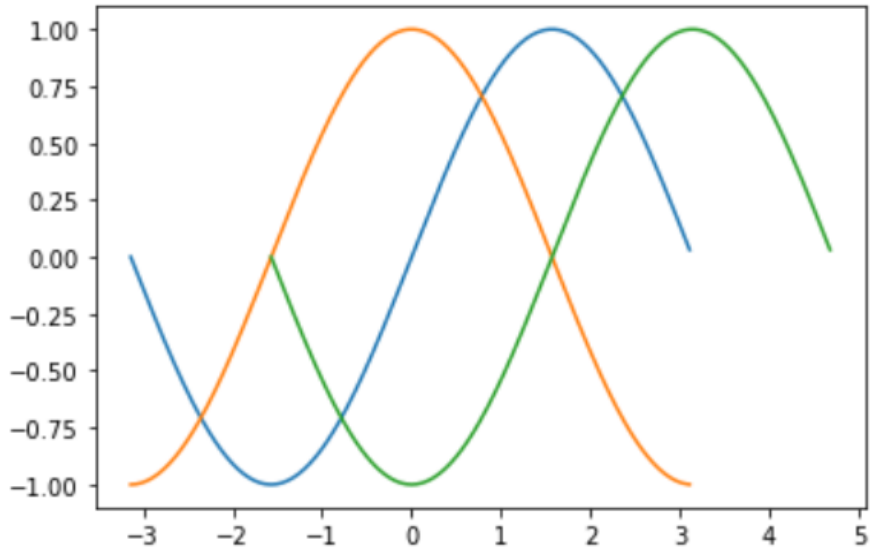
❖ ③ numpy array의 다양한 활용 (3/6)

- numpy를 활용한 다양한 그래프 그리기

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

a = np.arange(-np.pi, np.pi, np.pi/100)

plt.plot(a, np.sin(a))
plt.plot(a, np.cos(a))
plt.plot(a + np.pi/2, np.sin(a))
plt.show()
```



❖ ③ numpy array의 다양한 활용 (4/6)

- 마스크(Mask) 만들고 적용하기

```
import numpy as np

a = np.arange(-5, 5)     # 데이터 만들기
print(a)

print(a<0)               # 마스크 (Mask) 만들기
print(a[a<0])            # 마스크 적용하기
```

```
[-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]
```

❖ ③ numpy array의 다양한 활용 (5/6)

- 마스크(Mask) 연결해서 사용하기

```
import numpy as np

a = np.arange(-5, 5)    # 데이터 만들기
print(a)

mask1 = (abs(a) > 3)
print(mask1)
print(a[mask1])

mask2 = ((abs(a) % 2) == 0)
print(mask2)
print(a[mask2])

print(a[mask1 + mask2])  # OR 연산 (둘 중 하나의 조건이라도 참이면 참)
print(a[mask1 * mask2])  # AND 연산 (두 가지 조건이 모두 참이면 참)
```

```
[-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]
```

❖ ③ numpy array의 다양한 활용 (6/6)

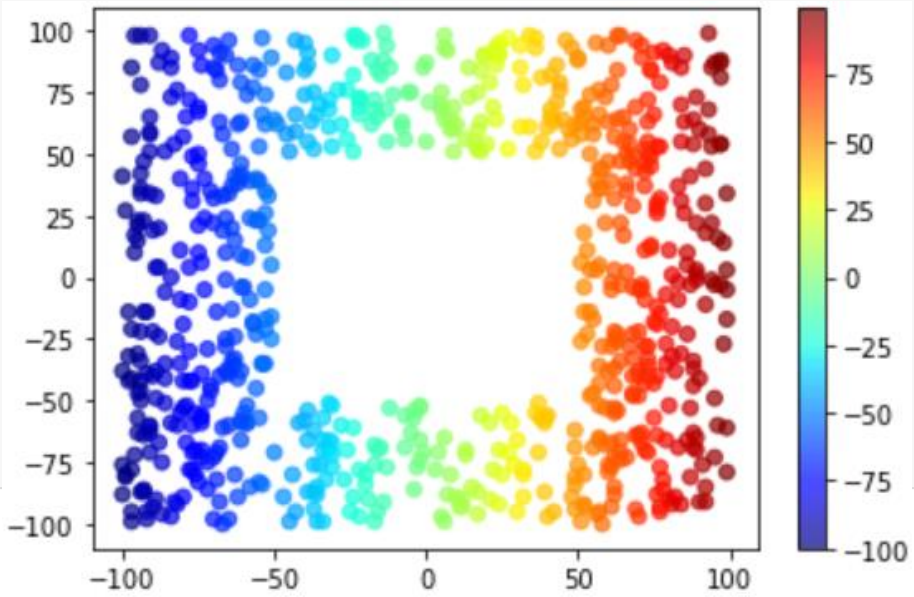
- numpy 라이브러리를 사용하여 재미있는 버블 차트 그리기

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

x = np.random.randint(-100, 100, 1000)
y = np.random.randint(-100, 100, 1000)

mask1 = (abs(x) > 50)
mask2 = (abs(y) > 50)
x = x[mask1 + mask2]    # 둘 중 하나라도 참이면 포함
y = y[mask1 + mask2]

plt.scatter(x, y, c=x, cmap='jet', alpha=0.7)
plt.colorbar()
plt.show()
```



끝맺음

- 01. 숫자 데이터를 쉽게 다루도록 돕는 numpy 라이브러리
- 02. numpy array 생성 및 활용 방법

THANK YOU!
Q & A

- Name: 강환수
- Office: 동양미래대학교 2호관 706호 (02-2610-1941)
- E-mail: hsknag@dongyang.ac.kr
- Homepage: https://github.com/ai7dnn/2023-DA