

---

# Lecture 2

NumPy 배열 브로드캐스팅

---

# 브로드캐스팅(Broadcasting)

- 브로드캐스트(Broadcast)는 일반적으로 아는 라디오나 텔레비전 방송을 얘기하기도 하지만 ‘**흩뿌리다, 널리 퍼뜨리다**’라는 뜻도 가지고 있다
- 선형대수에서 행렬의 연산을 할 때 기본적으로 행렬의 크기(shape)가 동일해야 한다
- 하지만 넘파이에서는 차원이나 크기가 다르더라도 일정한 조건을 충족하면 **자동으로 확장하여 연산**하게 되며 이를 **브로드캐스팅**이라고 한다 아래는 **스칼라 브로드캐스팅** 예제이다 (스칼라와 배열과의 연산은 브로드캐스팅을 통해 **element-wise** 연산을 수행한다)

```
In [1]: 1 import numpy as np
        2 x = np.array([[1, 2],[3,4]]) # (2,2) 배열
        3 print(x+5) # addition
        4 print(x-3.) # subtraction
        5 print(x*2) # product
        6 print(x/2) # division
        7 print(x//2) # floor division
        8 print(x%2) # modulus
        9 print(x**2) # power
       10 print(np.sqrt(x)) # square root
```



```
[[6 7]
 [8 9]]
[[-2. -1.]
 [ 0.  1.]]
[[2 4]
 [6 8]]
[[0.5 1.]
 [1.5 2.]]
```

```
[[0 1]
 [1 2]]
[[1 0]
 [1 0]]
[[ 1  4]
 [ 9 16]]
[[1.      1.41421356]
 [1.73205081 2.      ]]
```

# 브로드캐스팅(Broadcasting)

- 브로드캐스팅을 수행함에 있어 준수해야 하는 특정한 규칙은 다음과 같다:

**연산하는 배열들의 뒤에서부터 대응하는 축의 길이가 동일 혹은 1이어야만 한다**

스칼라 브로드캐스팅은 스칼라 자체의 축의 길이가 1이어서 신경을 쓰지 않아도 되지만 그 외는 아니다  
위 조건이 충족되지 않으면 배열의 모양이 호환되지 않기에 ValueError 가 발생한다  
연산 결과로 반환되는 배열은 입력 배열들의 차원 중에 **가장 큰 크기**로 반환된다

```
In [2]: 1 arr1 = np.random.randn(30).reshape(2,3,5)
        2 arr2 = np.random.randn(10).reshape(2,1,5)
        3 print(arr1+arr2) # (2,3,5) 배열
```

비동일, but 1

동일

```
[[[-2.28725091  3.56465639  0.27991111 -0.14250442 -1.64735105]
  [-2.10667604  1.53749222 -1.39135849  1.31438412 -2.8883705 ]
  [-2.17472212  1.02945185  1.5797158  0.38095287 -2.18460983]]

 [[ 0.75263738  1.13693959 -0.6152639 -1.21473688  3.07764824]
  [-0.36495636  0.69729974 -0.9228876 -2.02962868 -0.65365283]
  [-0.26415015 -0.34657002  1.12693211 -1.25688473  1.95278461]]]
```

# 브로드캐스팅(Broadcasting)

- 연산하는 배열들의 뒤에서부터 대응하는 축의 길이가 동일 혹은 1이어야만 한다  
연산 결과로 반환되는 배열은 입력 배열들의 차원 중에 가장 큰 크기로 반환된다

```
In [3]: 1 arr1 = np.random.randn(30).reshape(2,3,5)
        2 arr2 = np.random.randn(10).reshape(2,1,5)
        3 arr3 = np.random.randn(15).reshape(1,3,5)
        4 print(arr1+arr2+arr3)
        5 print((arr1+arr2+arr3).shape)
```

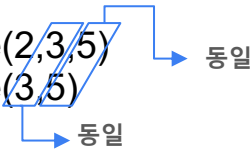
```
[[[-3.19150668 -0.35951265  1.74444888  0.73718809 -0.80957821]
 [ 0.5099595  0.5030344 -0.34121533  1.70730631 -0.76728266]
 [-1.73470171  0.3602853  2.10827702 -1.46173185  0.67195242]]

 [[-1.01013676 -0.30672908  1.04568408  2.81337037 -0.1697908 ]
 [ 1.86528779  0.53533074 -0.44193719  2.77112381 -0.38174986]
 [-1.72666955  1.29752381  2.17987642  0.10876395  0.91903391]]]
(2, 3, 5)
```

# 브로드캐스팅(Broadcasting)

- 연산하는 배열들의 뒤에서부터 대응하는 축의 길이가 동일 혹은 1이어야만 한다  
연산 결과로 반환되는 배열은 입력 배열들의 차원 중에 가장 큰 크기로 반환된다

```
In [4]: 1 arr1 = np.arange(30).reshape(2,3,5)
        2 arr2 = np.arange(15).reshape(3,5)
        3 print(arr1*arr2)
        4 print((arr1*arr2).shape)
```



```
[[[ 0  1  4  9 16]
   [ 25 36 49 64 81]
   [100 121 144 169 196]]

 [[ 0 16 34 54 76]
   [100 126 154 184 216]
   [250 286 324 364 406]]]
(2, 3, 5)
```

# 브로드캐스팅(Broadcasting)

- 연산하는 배열들의 뒤에서부터 대응하는 축의 길이가 동일 혹은 1이어야만 한다  
연산 결과로 반환되는 배열은 입력 배열들의 차원 중에 **가장 큰 크기**로 반환된다  
위 조건이 충족되지 않으면 배열의 모양이 호환되지 않기에 `ValueError` 가 발생한다

```
In [5]: 1 arr1 = np.random.randn(30).reshape(2,3,5)
        2 arr2 = np.random.randn(10).reshape(2,1,5)
        3 arr3 = np.random.randn(12).reshape(2,3,2)
        4 print(arr1+arr2+arr3)
```

`ValueError: operands could not be broadcast together with shapes (2,3,5) (2,3,2)`