03_Numpy 배열 연산

Numpy 배열 연산

- 산술연산
  - 배열은 기본적으로 요소의 연산이 가능

> 리스트의 덧셈

```
list1 = [1,2,3]
list2 = [4,5,6]
print(list1 + list2)
```

Out : [1, 2, 3, 4, 5, 6]

> 배열의 덧셈

```
arr1 = np.array([1,2,3])
arr2 = np.array([4,5,6])
print(arr1 + arr2)
```

Out : [5 7 9]

Numpy 배열 연산

- 산술연산
  - shape이 같은 배열은 연산 가능

> **모양이 같은 배열 연산**

```
arr1 = np.arange(1, 7, dtype=np.float32).reshape(2, 3)
arr2 = np.arange(11,17, dtype=np.int32).reshape(2, 3)
print(arr1 + arr2)  # 더하기
```

Out : [[12. 14. 16.]
[18. 20. 22.]]

```
print(arr2 - arr1)  # 빼기
```

Out : [[10. 10. 10.]
[10. 10. 10.]]

Numpy 배열 연산

- 산술연산
  - shape이 같은 배열은 연산 가능

> **모양이 같은 배열 연산**

```
print(arr1 * arr2)  # 곱하기
```

Out : [[11. 24. 39.]
[56. 75. 96.]]

```
print(arr2 / arr1)  # 나누기
```

Out : [[11. 6. 4.33333333]
[3.5 3. 2.66666667]]

```
print(arr1 ** arr2)  # 제곱
```

Out : [[1.00000000e+00 4.09600000e+03
1.59432300e+06]
[2.68435456e+08 3.05175781e+10
2.82110991e+12]]

Numpy 배열 연산

- 산술연산
  - shape이 같은 배열은 연산 가능

> 모양이 같은 배열 연산

```
print(arr2 // arr1)  # 몫
```

Out : [[11. 6. 4.]
 [3. 3. 2.]]

```
print(arr2 % arr1)  # 나머지
```

Out : [[0. 0. 1.]
 [2. 0. 4.]]

Numpy 배열 연산

- 산술연산
  - Numpy의 연산 함수

> **np.add(arr1, arr2)**

```
print(arr1 + arr2)
```

Out : [[12. 14. 16.]
[18. 20. 22.]]

```
print(np.add(arr1, arr2))
```

Out : [[12. 14. 16.]
[18. 20. 22.]]

Numpy 배열 연산

- 산술연산
  - Numpy의 연산 함수

> **np.subtract(arr1, arr2)**

```
print(arr1 - arr2)
```

Out : [[-10. -10. -10.]
[-10. -10. -10.]]

```
print(np.subtract(arr1, arr2))
```

Out : [[-10. -10. -10.]
[-10. -10. -10.]]

Numpy 배열 연산

- 산술연산
  - Numpy의 연산 함수

> **np.multiply(arr1, arr2)**

```
print(arr1 * arr2)
```

Out : [[11. 24. 39.]
[56. 75. 96.]]

```
print(np.multiply(arr1, arr2))
```

Out : [[11. 24. 39.]
[56. 75. 96.]]

Numpy 배열 연산

- 산술연산
  - Numpy의 연산 함수

> **np.divide(arr1, arr2)**

```
print(arr1 / arr2)
```

Out : [[0.09090909 0.16666667 0.23076923]
[0.28571429 0.33333333 0.375]]

```
print(np.divide(arr1, arr2))
```

Out : [[0.09090909 0.16666667 0.23076923]
[0.28571429 0.33333333 0.375]]

Numpy 배열 연산

- 산술연산
  - Broadcasting
  - 연산하고자 하는 배열의 모양이 다른 경우의 연산

> 일반적인 상황

```
arr3 = np.arange(10).reshape(5, 2)
arr4 = np.arange(10).reshape(2, 5)
print(arr3 + arr4)
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-35-1a8162fbbf08> in <module>
      1 arr3 = np.arange(10).reshape(5, 2)
      2 arr4 = np.arange(10).reshape(2, 5)
----> 3 print(arr3 + arr4)

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

Numpy 배열 연산

- 산술연산
  - Broadcasting : shape이 다른 배열 간 연산 지원

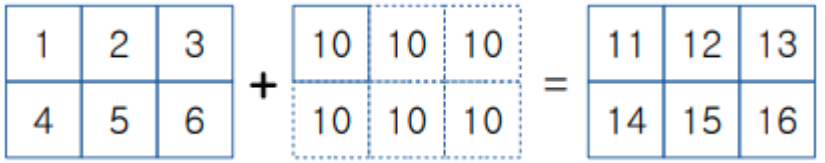
> **배열과 스칼라의 연산**

```
scalar = 10
vector = np.array([1, 2, 3])
print(scalar + vector)
```

Out : [11 12 13]


```
scalar = 10
matrix = np.array([[1, 2, 3], [4, 5, 6]])
print(scalar + matrix)
```

Out : [[11 12 13]
[14 15 16]]



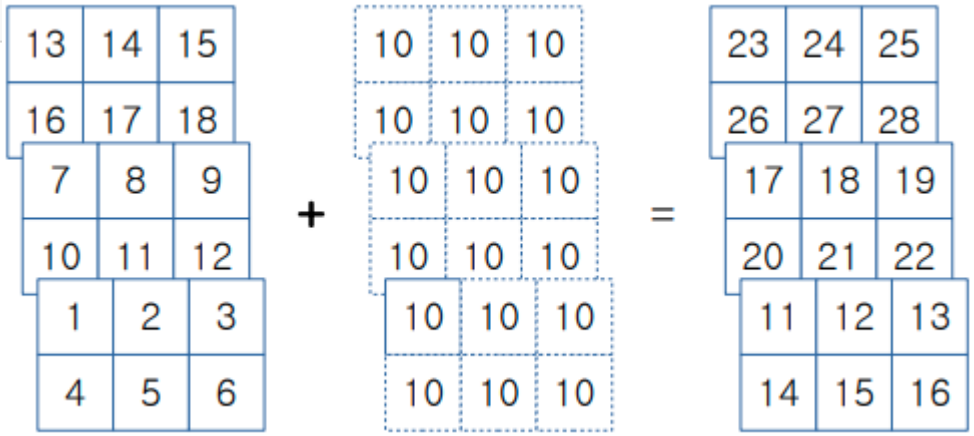
Numpy 배열 연산

- 산술연산
  - Broadcasting : shape이 다른 배열 간 연산 지원

> **배열과 스칼라의 연산**

```
scalar = 10
tensor = np.array([[[1, 2, 3],[4, 5, 6]],
                   [[7, 8, 9], [10, 11, 12]],
                   [[13, 14, 15], [16, 17, 18]]])
print(scalar + tensor)
```

Out : [[[11 12 13]
  [14 15 16]]
 [[17 18 19]
  [20 21 22]]
 [[23 24 25]
  [26 27 28]]]



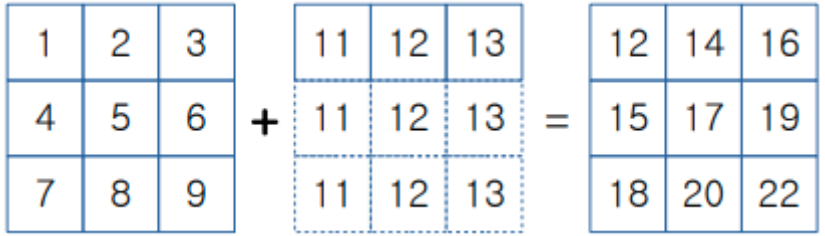
Numpy 배열 연산

- 산술연산

> 백터와 매트릭스의 연산

```
matrix = np.arange(1,10).reshape(3, 3)
vector = np.arange(11,14).reshape(1, 3)
print(matrix + vector)
```

Out : [[12 14 16]
[15 17 19]
[18 20 22]]



Numpy 배열 연산

- 산술연산

  > 백터와 매트릭스의 연산

```
matrix = np.arange(1,10).reshape(3, 3)
vector = np.arange(11,14).reshape(3, 1)
print(matrix + vector)
```

Out : [[12 13 14]
[16 17 18]
[20 21 22]]

Numpy 배열 연산

- 산술연산

  > 백터와 백터의 브로드캐스팅 연산

```
arr1 = np.array([1,2,3])
arr2 = np.array([1,2,3]).reshape(3, 1)
print(arr1)
print(arr2)
```

Out : [1 2 3]

[[1]

[2]

[3]]

```
print(arr1 + arr2)
```

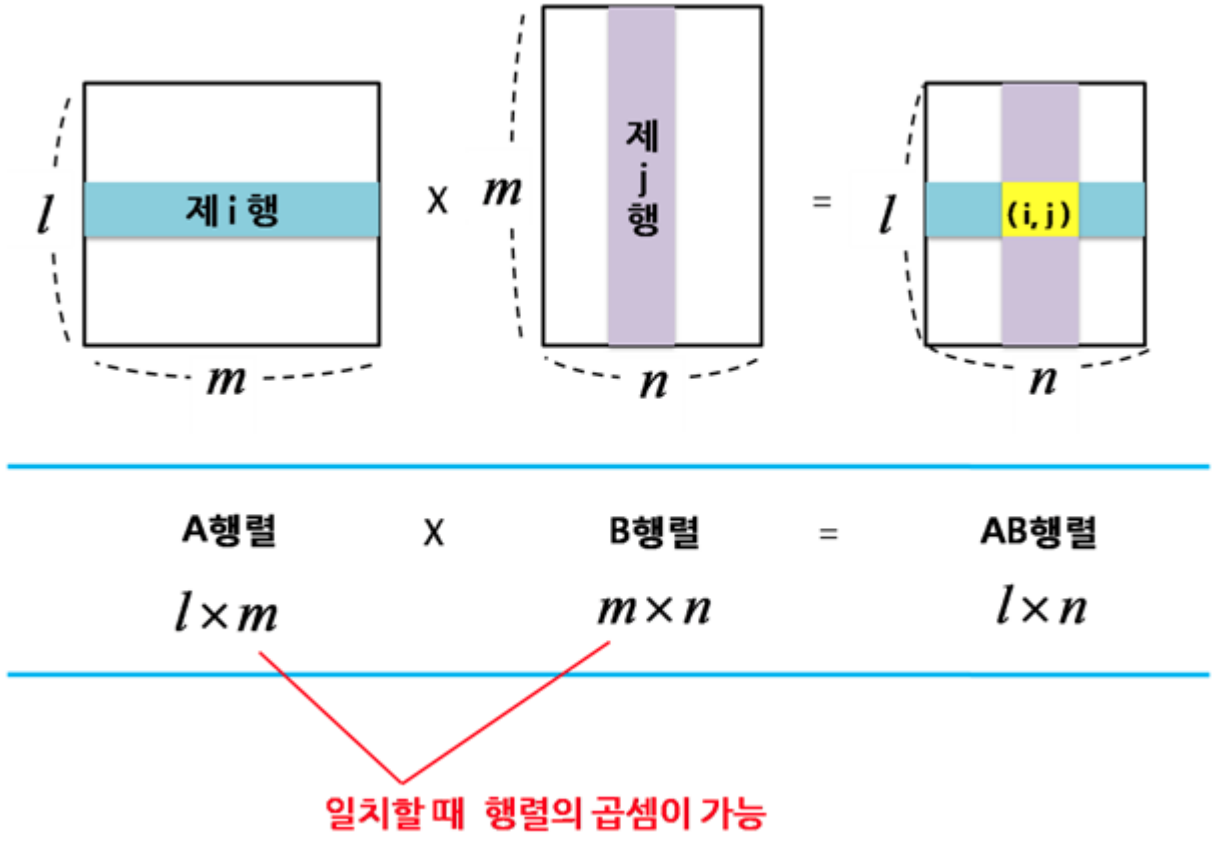
Out : [[2 3 4]

[3 4 5]

[4 5 6]]

Numpy 배열 연산

- 행렬 연산
  - 수학에서의 행렬의 곱셈



Numpy 배열 연산

- 행렬 연산
  - np.dot(arr1, arr2) 함수 사용

> 일반적인 행렬곱

```
arr1 = np.array([1,2,3])
arr2 = np.array([1,2,3])
print(arr1 * arr2)
```

Out : [1 4 9]

> np.dot 사용 행렬곱

```
arr1 = np.array([1,2,3])
arr2 = np.array([1,2,3]).T
print(np.dot(arr1, arr2))
```

Out : 14

Numpy 배열 연산

- 행렬 연산
  - np.dot(arr1, arr2) 함수

> **np.dot(arr1, arr2)**

```
print(np.dot(arr1, arr2))
```

Out : 14

> **arr1.dot(arr2)**

```
print(arr1.dot(arr2))
```

Out : 14

> **arr1 @ arr2**

```
print(arr1 @ arr2)
```

Out : 14

Numpy 배열 연산

- 행렬 연산

> 2차원 배열 행렬곱

```
arr3 = np.arange(1, 7).reshape(2, 3)
arr4 = np.arange(11, 17).reshape(3, 2)
print(np.dot(arr3, arr4))
```

Out : [[82 88]
 [199 214]]

1	2	3
4	5	6

×

11	12
13	14
15	16

=

82	88
199	214