
Introduction To Data Analysis
데이터분석입문
Lecture 24. Numpy 배열 연산과 난수 생성
인공지능소프트웨어학과 강환수 교수

학습개요

- 배열과 스칼라 연산
- 배열 구조가 동일한 배열 연산
- 배열 구조가 비슷한 브로드캐스팅 배열 연산
- 난수 발생 함수

학습목표

- 배열의 사칙 연산을 수행할 수 있다.
- 브로드캐스팅을 이해하고 구조가 비슷한 두 배열의 연산을 수행할 수 있다.
- 다양한 난수 함수를 사용해 다차원 배열을 생성할 수 있다.

LESSON 01

Numpy 배열 연산

배열 연산

배열에서 산술 연산자는 요소별로 적용

- [x] 새로운 배열이 생성되고 결과 반환

```
a = np.array([20, 30, 40, 50])
a
```

✓ 0.0s

```
array([20, 30, 40, 50])
```

```
b = np.arange(4)
b
```

✓ 0.0s

```
array([0, 1, 2, 3])
```

```
c = a - b
c
```

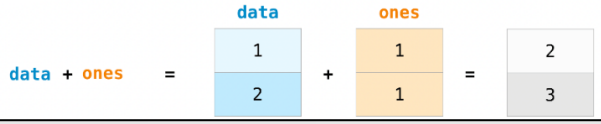
✓ 0.0s

```
array([20, 29, 38, 47])
```

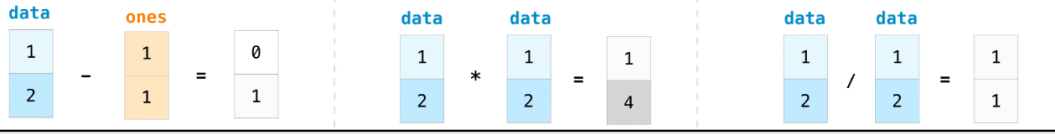
배열 기본 연산

+, −, *, /

```
>>> data = np.array([1, 2])
>>> ones = np.ones(2, dtype=int)
>>> data + ones
array([2, 3])
```




```
>>> data - ones
array([0, 1])
>>> data * data
array([1, 4])
>>> data / data
array([1., 1.])
```



배열 합 arr.sum()

arr.sum()

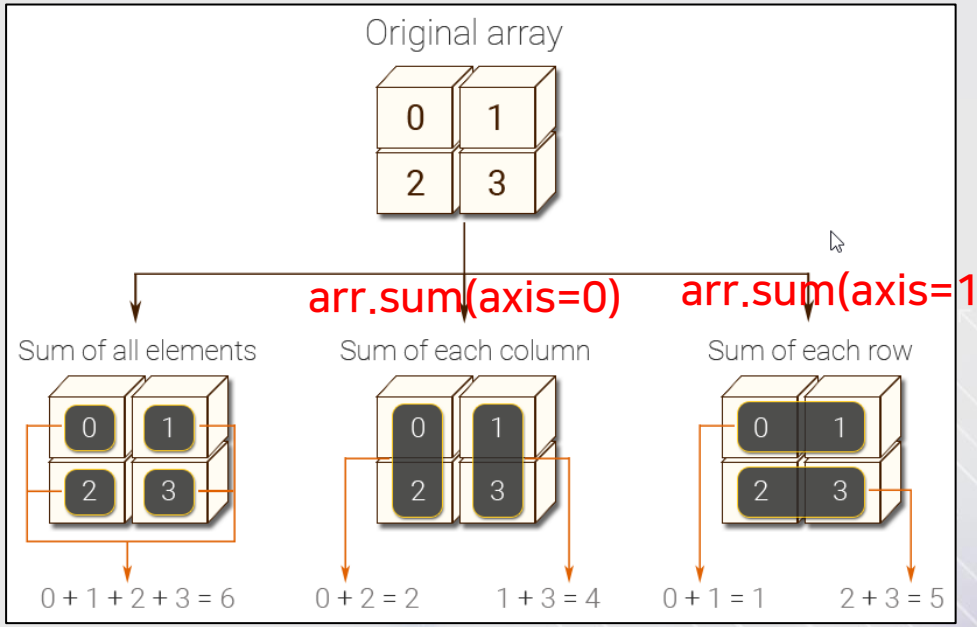
- 전체 원소 합

arr.sum(axis=0)

- 행(axis=0)으로 따라가면서 각 열의 합

arr.sum(axis=1)

- 열(axis=1)로 따라가면서 각 행의 합



배열 합 arr.sum()

```
b = np.array([[1, 1], [2, 2]])
b
```

✓ 0.0s

```
array([[1, 1],
       [2, 2]])
```

```
print(b.sum())
print(b.sum(axis=0))
```

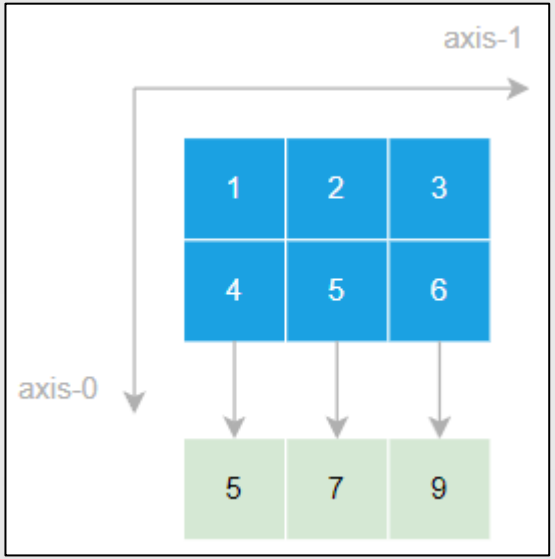
✓ 0.0s

```
6
[3 3]
```

```
print(b.sum(axis=1))
```

✓ 0.0s

```
[2 4]
```



브로드캐스팅(broadcasting)

배열과 단일 숫자 사이의 연산(벡터와 스칼라 사이의 연산 이라고도 함)

- 또는, 서로 다른 구조의 두 배열 사이에서 연산을 수행

NumPy가 다양한 모양의 배열에 대해 작업을 수행할 수 있도록 하는 메커니즘

- 연산이 각 셀에서 발생
- 두 배열의 차원이 동일하거나 둘 중 하나가 1인 경우 배열의 차원은 호환
  - 차원이 호환되지 않으면 ValueError 발생

```
>>> data = np.array([1.0, 2.0])
>>> data * 1.6
array([1.6, 3.2])
```

배열 열 수가 동일

```
data = np.array([[1, 2], [3, 4], [5, 6]])
data
```

✓ 0.0s

```
array([[1, 2],
       [3, 4],
       [5, 6]])
```

```
ones_row = np.array([[1, 1]])
ones_row
```

✓ 0.0s

```
array([[1, 1]])
```

```
data + ones_row
```

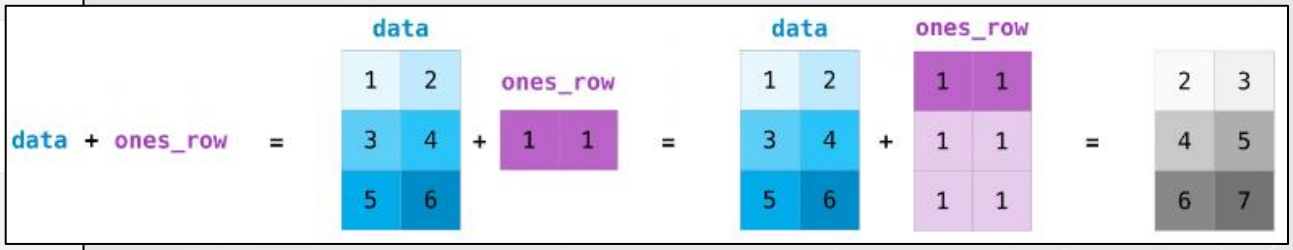
✓ 0.0s

```
array([[2, 3],
       [4, 5],
       [6, 7]])
```

```
data * ones_row
```

✓ 0.0s

```
array([[1, 2],
       [3, 4],
       [5, 6]])
```



배열 확장

두 배열을 모두 확장하여 초기 배열 중 하나보다 큰 출력 배열을 형성

```
a = np.array([0.0, 10.0, 20.0, 30.0])
a = a.reshape(4, 1)
a
```

✓ 0.0s

```
array([[ 0.],
       [10.],
       [20.],
       [30.]])
```

```
b = np.array([1.0, 2.0, 3.0])
b
```

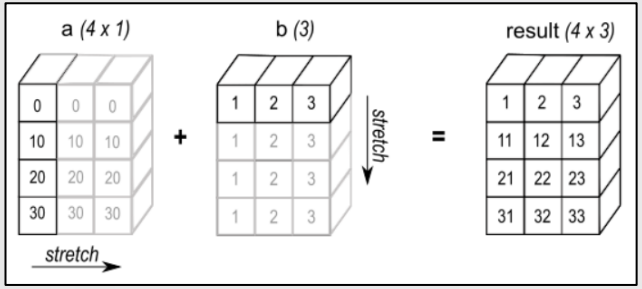
✓ 0.0s

```
array([1., 2., 3.])
```

```
a + b
```

✓ 0.0s

```
array([[ 1.,  2.,  3.],
       [11., 12., 13.],
       [21., 22., 23.],
       [31., 32., 33.]])
```



행 확장

```
a = np.array([[ 0.0,  0.0,  0.0],
              [10.0, 10.0, 10.0],
              [20.0, 20.0, 20.0],
              [30.0, 30.0, 30.0]])
a
```

✓ 0.0s

```
array([[ 0.,  0.,  0.],
       [10., 10., 10.],
       [20., 20., 20.],
       [30., 30., 30.]])
```

```
b = np.array([1.0, 2.0, 3.0])
b
```

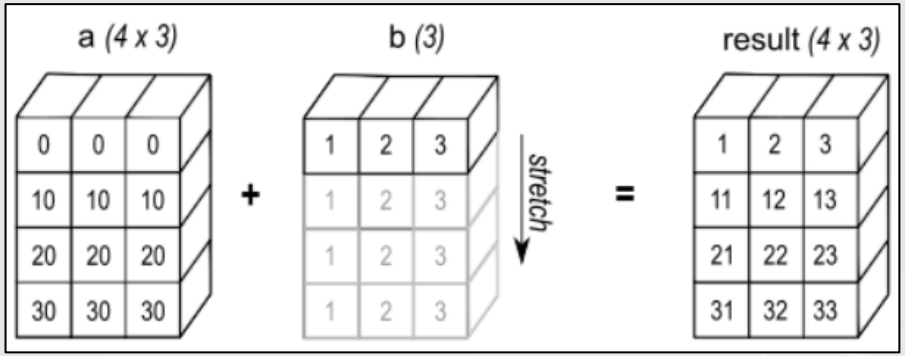
✓ 0.0s

```
array([1., 2., 3.])
```

```
a + b
```

✓ 0.0s

```
array([[ 1.,  2.,  3.],
       [11., 12., 13.],
       [21., 22., 23.],
       [31., 32., 33.]])
```



브로드캐스팅 오류

ValueError

```
a
```

✓ 0.0s Python

```
array([[ 0.,  0.,  0.],
       [10., 10., 10.],
       [20., 20., 20.],
       [30., 30., 30.]])
```

```
b = np.array([1.0, 2.0, 3.0, 4.0])
b
```

✓ 0.0s Python

```
array([1., 2., 3., 4.])
```

```
a + b
```

⊗ 0.0s Python

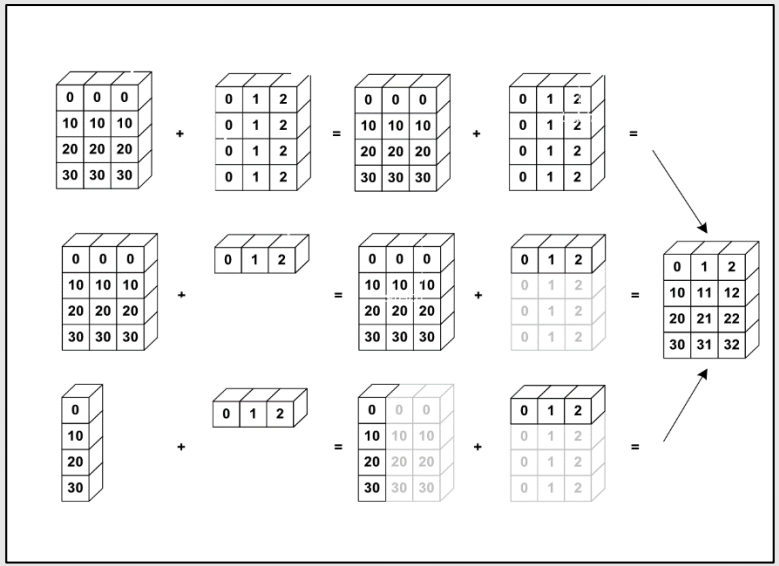
```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
d:\(1 Drive)\m365\OneDrive - 동양미래대학\55. 콘텐츠제작\2023 데이터분석입문\fina
----> 1 a + b

ValueError: operands could not be broadcast together with shapes (4,3) (4,)
```

브로드캐스팅 정리

Shape이 같은 두 배열에 대한 이항 연산은 배열의 요소별로 수행

두 배열 간의 Shape이 다를 경우

- 크기가 다른 배열 간의 연산
- 두 배열 간의 형상을 맞추는 Broadcasting 과정을 거쳐 계산



LESSON 02

Numpy 난수 생성

np.random.rand(d0, d1, ..., dn)

주어진 차원의 0에서 1 미만 사이 [0.0, 1.0)의 실수인 난수 또는 배열을 생성

```
from numpy import random

print(random.rand()) # [0, 1) 난수 하나
print(random.rand(3))
print(random.rand(2, 3))
```

✓ 0.0s

```
0.6114724886049473
[0.99022192 0.60048418 0.98224925]
[[0.03080039 0.86239107 0.34515384]
 [0.67110614 0.25327017 0.00345886]]
```

np.random.random(size=None)

0에서 1 미만 사이 [0.0, 1.0)의 실수인 난수 또는 배열을 생성

```
from numpy import random

print(random.random()) # [0, 1) 난수 하나
print(random.random(size=(2,)))
print(random.random(size=2))
print(random.random((2, 3)))
```

✓ 0.0s

```
0.7573745558538069
[0.98777801 0.13770937]
[0.82099894 0.5895019 ]
[[0.86575809 0.49705386 0.91797815]
 [0.54963282 0.81145833 0.60026703]]
```

비교 rand()와 random()

다차원 배열의 지정 방법이 다름

```
from numpy import random

print(random.rand(2, 3))
print(random.random((2, 3)))
```

✓ 0.0s

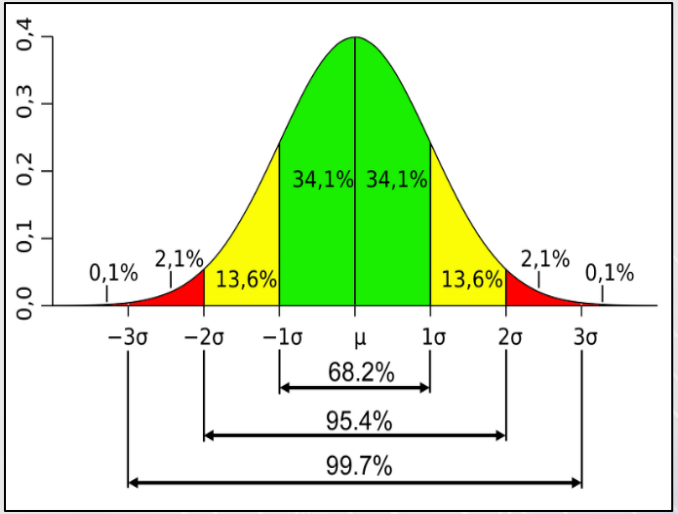
```
[[0.73816928 0.99684857 0.60786187]
 [0.14365811 0.66123217 0.09070631]]
[[0.11688829 0.70637377 0.4439654 ]
 [0.29883582 0.69668008 0.80457289]]
```

np.random.randn(d0, d1, ..., dn)

주어진 차원의 표준 정규분포에서 난수 또는 배열을 생성

```
from numpy import random

print(random.randn()) # 표준 정규분포 N(0, 1)의 난수 발생
print(random.randn(2)) # 표준 정규분포의 난수 2개 생성
print(random.randn(2, 3)) # 표준 정규분포의 난수 6개 생성
```

```
0.2908671341667035
[2.19241702 0.31201965]
[[ 0.42097069 -0.59104511 -2.37324937]
 [-0.92058848  1.10121038  0.80513957]]
```



np.random.randint(low=0, high=None, size=None)

일정 범위 low <= x < high 내의 정수를 생성

- 정수 인자가 하나면 [0, 인자) 사이의 정수 난수

```
from numpy import random

print(random.randint(2)) # [0, 2) 정수 난수
print(random.randint(1, 7)) # [1, 7) 정수 난수
print(random.randint(1, 7, 3)) # [1, 7) 정수 난수 3개
print(random.randint(1, 7, size=(2, 3)))
```

✓ 0.0s Python

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

np.random.choice(a, size=None, replace=True, p=None)

주어진 1차원 배열 a에서 요소 size 형태로 선택

- 옵션 replace: 중복 허용 여부

```python
from numpy import random

print(sorted(random.choice(range(1, 46), 6)))
print(sorted(random.choice(range(1, 46), 6, replace=False)))
```

✓ 0.0s Python

```
[14, 18, 28, 33, 35, 38]
[10, 13, 14, 23, 36, 44]
```

```python
from numpy import random

print(random.choice(5, 3))
print(random.choice(5, 3, replace=False, p=[0.1, 0, 0.3, 0.6, 0]))
```

✓ 0.0s Python

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

옵션 replace=True

모집단의 수가 작으면 오류

```
from numpy import random

print(random.choice(5, (3, 2)))
print(random.choice(range(1, 46), size=(5, 6), replace=False))
```

✓ 0.0s Python

```
[[3 2]
 [0 1]
 [1 0]]
[[28 42 35 25 16 30]
 [ 9  6 23 21 41  5]
 [ 3 31 33 15 12 44]
 [19 38 10  4 11 20]
 [34 39 13  7 40 27]]
```

```
random.choice(range(1, 30), size=(5, 6), replace=False) # 오류
```

⊗ 0.0s Python

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
d:\(1 Drive)\m365\OneDrive - 동양미래대학\55. 콘텐츠제작\2023 데이터분석입문\fina
----> 1 random.choice(range(1, 30), size=(5, 6), replace=False) # 오류

File mtrand.pyx:984, in numpy.random.mtrand.RandomState.choice()

ValueError: Cannot take a larger sample than population when 'replace=False'
```

Numpy 난수 생성 함수

이름	설명	예제 코드
rand(d0, d1, ..., dn)	주어진 차원의 0에서 1 사이 [0.0, 1.0)의 실수인 난수 배열을 생성	np.random.rand(2, 3)
random(size=None)	0에서 1 사이 [0.0, 1.0)의 실수인 난수를 생성	np.random.random((3, 3))
randn(d0, d1, ..., dn)	주어진 차원의 표준 정규 분포에서 난수 배열을 생성	np.random.randn(4, 5)
randint(low=0, high=None, size=None)	일정 범위 low <= x < high 내의 정수를 생성, 정수 인자가 하나면 [0, 인자] 사이의 정수 난수	np.random.randint(1, 100, 10)
uniform(low=0, high=1, size=None)	주어진 범위 내 [low, high)의 실수인 난수를 생성 half-open interval: [low, high)	np.random.uniform(2, 5, (2, 2))
shuffle(x)	배열의 요소를 무작위로 섞기	mylist = [1, 2, 3, 4, 5] np.random.shuffle(mylist)
choice(a, size=None, replace=True, p=None)	주어진 배열에서 무작위로 요소를 선택	np.random.choice([1, 2, 3, 4, 5], 3)

SUMMARY

학습정리

구조가 동일한 배열 연산

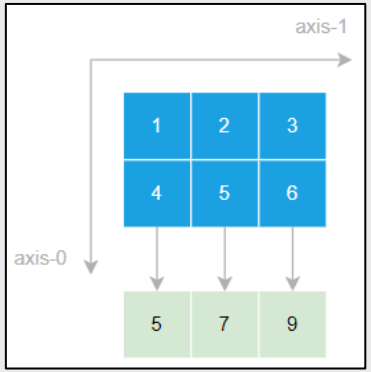
- 각 항목 연산

배열 합

- arr.sum()
- arr.sum(axis=0)
- arr.sum(axis=1)

구조가 다른 배열 연산

- 배열을 확장해 연산
- ValueError



난수 생성 함수

- rand(d0, d1, ..., dn)
- random(size=None)
- randn(d0, d1, ..., dn)
- randint(low=0, high=None, size=None)
- choice(a, size=None, replace=True, p=None)