

11. 어레이 인덱싱 (2부)

주요 내용

- 어레이의 축
- 인덱싱과 슬라이싱
- 부울 인덱싱
- 팬시 인덱싱
- 어레이 쪼개기

11.3. 부울 인덱싱

1차원 부울 어레이 활용

In [1]:

```
import numpy as np

names = np.array(['Bob', 'Joe', 'Will', 'Bob', 'Will', 'Joe', 'Joe'])
```

Out[1]:

```
array(['Bob', 'Joe', 'Will', 'Bob', 'Will', 'Joe', 'Joe'], dtype='<U4')
```

In [2]:

```
np.random.seed(3)

data = np.random.randn(7, 4)
data
```

Out[2]:

```
array([[ 1.78862847,  0.43650985,  0.09649747, -1.8634927 ],
       [-0.2773882 , -0.35475898, -0.08274148, -0.62700068],
       [-0.04381817, -0.47721803, -1.31386475,  0.88462238],
       [ 0.88131804,  1.70957306,  0.05003364, -0.40467741],
       [-0.54535995, -1.54647732,  0.98236743, -1.10106763],
       [-1.18504653, -0.2056499 ,  1.48614836,  0.23671627],
       [-1.02378514, -0.7129932 ,  0.62524497, -0.16051336]])
```

가정: names와 data의 연관성

- 2차원 어레이 `data` 의 각 행이 `names` 어레이의 항목과 연관된다고 가정
- 예를 들어, '`'Bob'`' 은 0번 인덱스와 3번 인덱스의 행과 연관됨

data 어레이에서 'Bob' 과 연관된 행만 추출

In [3]:

```
name_Bob = (names == 'Bob')
name_Bob
```

Out[3]:

```
array([ True, False, False,  True, False, False, False])
```

In [4]:

```
data[name_Bob]
```

Out[4]:

```
array([[ 1.78862847,  0.43650985,  0.09649747, -1.8634927 ],
       [ 0.88131804,  1.70957306,  0.05003364, -0.40467741]])
```

부울 인덱싱과 일반 인덱싱/슬라이싱 혼합

- 행 기준: Bob이 포함된 행의 인덱스를 갖는 행
- 열 기준: 3번 열

In [5]:

```
data[name_Bob, 3]
```

Out[5]:

```
array([-1.8634927 , -0.40467741])
```

- 행 기준: Bob이 포함된 행의 인덱스를 갖는 행
- 열 기준: 2번 열 이후 전체

In [6]:

```
data[name_Bob, 2:]
```

Out[6]:

```
array([[ 0.09649747, -1.8634927 ],
       [ 0.05003364, -0.40467741]])
```

마스크 활용

마스크 mask: 논리 연산자(~, &, |)를 사용하는 부울 어레이 표현식

- 이름이 'Bob' 이 아닌 이름과 연관된 행만 가져오기

In [7]:

```
mask = names != 'Bob'  
data[mask]
```

Out[7]:

```
array([[-0.2773882 , -0.35475898, -0.08274148, -0.62700068],  
       [-0.04381817, -0.47721803, -1.31386475,  0.88462238],  
       [-0.54535995, -1.54647732,  0.98236743, -1.10106763],  
       [-1.18504653, -0.2056499 ,  1.48614836,  0.23671627],  
       [-1.02378514, -0.7129932 ,  0.62524497, -0.16051336]])
```

- 'Bob' 또는 'Will' 이 위치한 인덱스에 해당하는 행만 가져오기

In [8]:

```
mask = (names == 'Bob') | (names == 'Will')
data[mask]
```

Out[8]:

```
array([[ 1.78862847,  0.43650985,  0.09649747, -1.8634927 ],
       [-0.04381817, -0.47721803, -1.31386475,  0.88462238],
       [ 0.88131804,  1.70957306,  0.05003364, -0.40467741],
       [-0.54535995, -1.54647732,  0.98236743, -1.10106763]])
```

항목 업데이트

- 마스크를 이용하여 전체 행 또는 전체 열을 특정 값으로 변경 가능

In [9]:

```
mask = names != 'Joe'  
data[mask] = 7  
data
```

Out[9]:

```
array([[ 7.          ,  7.          ,  7.          ,  7.          ],  
       [-0.2773882 , -0.35475898, -0.08274148, -0.62700068],  
       [ 7.          ,  7.          ,  7.          ,  7.          ],  
       [ 7.          ,  7.          ,  7.          ,  7.          ],  
       [ 7.          ,  7.          ,  7.          ,  7.          ],  
       [-1.18504653, -0.2056499 ,  1.48614836,  0.23671627],  
       [-1.02378514, -0.7129932 ,  0.62524497, -0.16051336]])
```

다차원 마스크

- 음수 항목만 추출해서 1차원 어레이 생성하기

In [10]:

```
mask = data < 0  
mask
```

Out[10]:

```
array([[False, False, False, False],  
       [ True,  True,  True,  True],  
       [False, False, False, False],  
       [False, False, False, False],  
       [False, False, False, False],  
       [ True,  True, False, False],  
       [ True,  True, False,  True]])
```

In [11]:

```
data[mask]
```

Out[11]:

```
array([-0.2773882 , -0.35475898, -0.08274148, -0.62700068, -1.1850  
4653,  
      -0.2056499 , -1.02378514, -0.7129932 , -0.16051336])
```

- 모든 음수 항목을 0으로 변경하기

In [12]:

```
data[mask] = 0  
data
```

Out[12]:

```
array([[7.        , 7.        , 7.        , 7.        , 7.        ],  
       [0.        , 0.        , 0.        , 0.        , 0.        ],  
       [7.        , 7.        , 7.        , 7.        , 7.        ],  
       [7.        , 7.        , 7.        , 7.        , 7.        ],  
       [7.        , 7.        , 7.        , 7.        , 7.        ],  
       [0.        , 0.        , 1.48614836, 0.23671627],  
       [0.        , 0.        , 0.62524497, 0.        ]])
```

부울 인덱싱과 뷰

- 부울 인덱싱은 뷰를 이용하지 않음

In [13]:

```
data2 = data[names == 'Bob']  
data2
```

Out[13]:

```
array([[7., 7., 7., 7.],  
       [7., 7., 7., 7.]])
```

In [14]:

```
data2[0] = -1  
data2
```

Out[14]:

```
array([[-1., -1., -1., -1.],  
       [ 7.,  7.,  7.,  7.]])
```

In [15]:

data

Out[15]:

```
array([[7.        , 7.        , 7.        , 7.        , 7.        ],
       [0.        , 0.        , 0.        , 0.        , 0.        ],
       [7.        , 7.        , 7.        , 7.        , 7.        ],
       [7.        , 7.        , 7.        , 7.        , 7.        ],
       [7.        , 7.        , 7.        , 7.        , 7.        ],
       [0.        , 0.        , 1.48614836, 0.23671627],
       [0.        , 0.        , 0.62524497, 0.        ]])
```

11.4. 팬시 인덱싱: 인덱스 리스트 활용

0번 축 팬시 인덱싱

In [16]:

```
arr = np.arange(32).reshape((8, 4))
arr
```

Out[16]:

```
array([[ 0,  1,  2,  3],
       [ 4,  5,  6,  7],
       [ 8,  9, 10, 11],
       [12, 13, 14, 15],
       [16, 17, 18, 19],
       [20, 21, 22, 23],
       [24, 25, 26, 27],
       [28, 29, 30, 31]])
```

In [17]:

```
arr[[4, 3, 0, 6]]
```

Out[17]:

```
array([[16, 17, 18, 19],
       [12, 13, 14, 15],
       [ 0,  1,  2,  3],
       [24, 25, 26, 27]])
```

In [18]:

```
arr = np.arange(32).reshape((8, 4))
arr
```

Out[18]:

```
array([[ 0,  1,  2,  3],
       [ 4,  5,  6,  7],
       [ 8,  9, 10, 11],
       [12, 13, 14, 15],
       [16, 17, 18, 19],
       [20, 21, 22, 23],
       [24, 25, 26, 27],
       [28, 29, 30, 31]])
```

In [19]:

```
arr[[-3, -5, -7]]
```

Out[19]:

```
array([[20, 21, 22, 23],
       [12, 13, 14, 15],
       [ 4,  5,  6,  7]])
```

1번 축 팬시 인덱싱

In [20]:

```
arr = np.arange(32).reshape((8, 4))
arr
```

Out[20]:

```
array([[ 0,  1,  2,  3],
       [ 4,  5,  6,  7],
       [ 8,  9, 10, 11],
       [12, 13, 14, 15],
       [16, 17, 18, 19],
       [20, 21, 22, 23],
       [24, 25, 26, 27],
       [28, 29, 30, 31]])
```

In [21]:

```
arr[:3, [0, 3, 1]]
```

Out[21]:

```
array([[ 0,  3,  1],
       [ 4,  7,  5],
       [ 8, 11,  9]])
```

- 축별 팬시 인덱싱을 연속으로 실행 가능

In [22]:

```
arr = np.arange(32).reshape((8, 4))
arr
```

Out[22]:

```
array([[ 0,  1,  2,  3],
       [ 4,  5,  6,  7],
       [ 8,  9, 10, 11],
       [12, 13, 14, 15],
       [16, 17, 18, 19],
       [20, 21, 22, 23],
       [24, 25, 26, 27],
       [28, 29, 30, 31]])
```

In [23]:

```
arr[[1, 5, 7, 2]][:, [0, 3, 1]]
```

Out[23]:

```
array([[ 4,  7,  5],
       [20, 23, 21],
       [28, 31, 29],
       [ 8, 11,  9]])
```

축과 팬시 인덱싱

- `(1, 0), (5, 3), (7, 2), (2, 2)` 좌표에 위치한 항목 추출
- 축별로 항목을 모아놓은 두 개의 인덱스 어레이를 사용

In [24]:

```
arr[[1, 5, 7, 2], [0, 3, 1, 2]]
```

Out[24]:

```
array([ 4, 23, 29, 10])
```

팬시 인덱싱 활용 예제

0	1	2	3	4	5
10	11	12	13	14	15
20	21	22	23	24	25
30	31	32	33	34	35
40	41	42	43	44	45
50	51	52	53	54	55

In [25]:

```
arr = np.arange(36).reshape(6, 6) + np.arange(0, 21, 4).reshape(6, 1)  
arr
```

Out[25]:

```
array([[ 0,  1,  2,  3,  4,  5],  
       [10, 11, 12, 13, 14, 15],  
       [20, 21, 22, 23, 24, 25],  
       [30, 31, 32, 33, 34, 35],  
       [40, 41, 42, 43, 44, 45],  
       [50, 51, 52, 53, 54, 55]])
```

- 초록색 1차원 어레이: 0번 축과 1번 축의 팬시 인덱싱 조합

0	1	2	3	4	5
10	11	12	13	14	15
20	21	22	23	24	25
30	31	32	33	34	35
40	41	42	43	44	45
50	51	52	53	54	55

In [26]:

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

Out[26]:

```
array([ 1, 12, 23, 34, 45])
```

- 빨강색 1차원 어레이: 0번 축 팬시 인덱싱과 1번 축 인덱싱

0	1	2	3	4	5
10	11	12	13	14	15
20	21	22	23	24	25
30	31	32	33	34	35
40	41	42	43	44	45
50	51	52	53	54	55

In [27]:

```
arr[[0, 2, 5], 2]
```

Out[27]:

```
array([ 2, 22, 52])
```

- 파랑색 2차원 어레이: 0번 축 슬라이싱과 1번 축 팬시 인덱싱

0	1	2	3	4	5
10	11	12	13	14	15
20	21	22	23	24	25
30	31	32	33	34	35
40	41	42	43	44	45
50	51	52	53	54	55

In [28]:

```
arr[3:, [0, 2, 5]]
```

Out[28]:

```
array([[30, 32, 35],  
       [40, 42, 45],  
       [50, 52, 55]])
```

3차원 어레이 팬시 인덱싱

In [29]:

```
arr = np.arange(32).reshape((4, 2, 4))
arr
```

Out[29]:

```
array([[[ 0,  1,  2,  3],
       [ 4,  5,  6,  7]],

       [[ 8,  9, 10, 11],
       [12, 13, 14, 15]],

       [[16, 17, 18, 19],
       [20, 21, 22, 23]],

       [[24, 25, 26, 27],
       [28, 29, 30, 31]])
```

In [30]:

```
arr[[1, 2], [0, 1], [2, 3]]
```

Out[30]:

```
array([10, 23])
```

In [31]:

```
arr[[1, 2], [0, 1]]
```

Out[31]:

```
array([[ 8,  9, 10, 11],
       [20, 21, 22, 23]])
```

11.5 어레이 쪼개기

np.split() 함수

- np.split() 함수의 인자
 - 첫째 인자: 쪼개기 대상 어레이
 - 둘째 인자: 하나의 정수 또는 여러 인덱스들의 리스트
 - 셋째 인자: 쪼개기 대상 축을 지정하는 axis=0 키워드 인자. 기본값은 0.
- 둘째 인자가 정수인 경우: 지정된 축의 크기의 약수이어야 하며, 지정된 인자만큼 균등하게 쪼개는 용도로 사용됨.
- 둘째 인덱스들의 리스트인 경우: 지정된 축의 인덱스의 구분하는 구분점으로 활용

- 예제: 행의 인덱스를 0, 1, 2-3 세 개의 구간으로 쪼개기

In [32]:

```
arr = np.random.randn(4, 3)
arr
```

Out[32]:

```
array([[-0.76883635, -0.23003072,  0.74505627],
       [ 1.97611078, -1.24412333, -0.62641691],
       [-0.80376609, -2.41908317, -0.92379202],
       [-1.02387576,  1.12397796, -0.13191423]])
```

In [33]:

```
np.split(arr, [1, 2]) # np.split(arr, [1, 2], axis=0)
```

Out[33]:

```
[array([[ -0.76883635, -0.23003072,  0.74505627]]),
 array([[ 1.97611078, -1.24412333, -0.62641691]]),
 array([[[-0.80376609, -2.41908317, -0.92379202],
        [-1.02387576,  1.12397796, -0.13191423]]])
```

- 열의 인덱스를 0-1, 2 두 개의 구간으로 쪼개기

In [34]:

```
np.split(arr, [2], axis=1)
```

Out[34]:

```
[array([[ -0.76883635, -0.23003072],
       [ 1.97611078, -1.24412333],
       [-0.80376609, -2.41908317],
       [-1.02387576,  1.12397796]]),
 array([[ 0.74505627],
       [-0.62641691],
       [-0.92379202],
       [-0.13191423]])]
```

np.vsplit() 함수

- `np.vsplit(arr, z) := np.split(arr, z, axis=0)`

In [35]:

```
np.vsplit(arr, [1, 2])
```

Out[35]:

```
[array([[ -0.76883635, -0.23003072,  0.74505627]]),  
 array([[ 1.97611078, -1.24412333, -0.62641691]]),  
 array([[[-0.80376609, -2.41908317, -0.92379202],  
        [-1.02387576,  1.12397796, -0.13191423]])]
```

np.hsplit() 함수

- `np.hsplit(arr, z) := np.split(arr, z, axis=1)`

In [36]:

```
np.hsplit(arr, [2])
```

Out[36]:

```
[array([[ -0.76883635, -0.23003072],
       [ 1.97611078, -1.24412333],
       [-0.80376609, -2.41908317],
       [-1.02387576,  1.12397796]]),
 array([[ 0.74505627],
       [-0.62641691],
       [-0.92379202],
       [-0.13191423]])]
```