## ■ 배열 요소 비교 (all / any)

### ● 모든 요소 또는 일부 요소의 조건 만족 여부 확인

```
a = np.arange(1, 11)
a
```

```
array([ 1,  2,  3,  4,  5,  6,  7,  8,  9, 10])
```

```
np.all(a >= 1), np.all(a > 1)
```

```
(True, False)
```

```
np.any(a > 10), np.any(a > 5)
```

```
(False, True)
```

```
a > 5
```

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

## ■ 배열 요소 비교 (all / any)

### ● 모든 요소 또는 일부 요소의 조건 만족 여부 확인

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

```
a == b
```

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

```
(a == b).all(), (a == b).any()
```

```
(False, True)
```

## ■ 배열 요소 비교 (logical_and / logical_or / logical_not)

### ● 조건에 따른 논리 연산

```
a = np.array([1, 2, 3])
np.logical_and(a > 1, a < 3)
```

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

```
a = np.array([1, 2, 3])
np.logical_or(a > 1, a < 3)
```

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

```
a = np.array([False, False, False])
np.logical_not(a)
```

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

## ■ 배열 요소 비교 (where)

### ● 조건에 따른 논리 연산

```
a = np.arange(1, 11)
a
```

```
array([ 1,  2,  3,  4,  5,  6,  7,  8,  9, 10])
```

```
np.where(a > 5)
```

```
(array([5, 6, 7, 8, 9], dtype=int64),)
```

```
np.where(a > 5, True, False)
```

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

## ■ 최대값 / 최소값 (argmax / argmin)

### ● 배열 내의 최대값 / 최소값의 index 반환

```
a = np.arange(1, 13).reshape(3, 4)
a
```

```
array([[ 1,  2,  3,  4],
       [ 5,  6,  7,  8],
       [ 9, 10, 11, 12]])
```

```
np.argmax(a), np.argmin(a)
```

```
(11, 0)
```

```
np.argmax(a, axis=0), np.argmin(a, axis=0)
```

```
(array([2, 2, 2, 2], dtype=int64), array([0, 0, 0, 0], dtype=int64))
```

## ■ 최대값 / 최소값 (argmax / argmin)

### ● 배열 내의 최대값 / 최소값의 index 반환

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

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

```
np.argmax(b, axis=1)
```

```
array([3, 2, 1], dtype=int64)
```

```
np.argmin(b, axis=1)
```

```
array([0, 3, 2], dtype=int64)
```

## ■ boolean index

- ● 특정 조건에 따라 값을 배열 형태로 추출 가능

```
a = np.array([1, 2, 3, 4, 5])
a > 3
```

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

```
a[[False, False, False, True, True]]
```

```
array([4, 5])
```

```
a[a > 3]
```

```
array([4, 5])
```

## ■ boolean index

- 특정 조건에 따라 값을 배열 형태로 추출 가능

```
b = np.random.uniform(1, 11, 10)  # 균등분포 랜덤 생성
b
```

```
array([9.50983127, 4.43029517, 4.60024648, 2.50360452, 8.81840722,
       7.84602316, 8.75459294, 3.41832847, 4.87391863, 6.04625995])
```

```
condition = b > 6
condition
```

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

```
b[condition]
```

```
array([9.50983127, 8.81840722, 7.84602316, 8.75459294, 6.04625995])
```

```
b[condition].astype(np.int8)  # 배열 요소의 타입 변경
```

```
array([9, 8, 7, 8, 6], dtype=int8)
```

## ■ fancy index

### ● index 값을 가지는 배열을 이용하여 값을 추출

```
a = np.arange(1, 11)  # 실제 값 배열
a
```

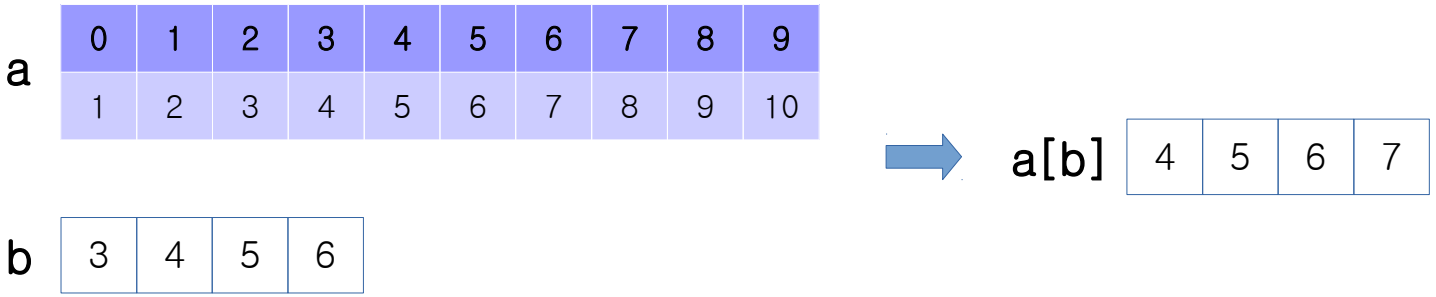
```
array([ 1,  2,  3,  4,  5,  6,  7,  8,  9, 10])
```

```
b = np.array([3, 4, 5, 6])  # index 값을 가지는 배열 생성
b
```

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

```
a[b]  # fancy index
```

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



## ■ fancy index

### ● index 값을 가지는 배열을 이용하여 값을 추출

```
a = np.arange(1, 11)  # 실제 값 배열
a
```

```
array([ 1,  2,  3,  4,  5,  6,  7,  8,  9, 10])
```

```
b = np.array([3, 4, 5, 6])  # index 값을 가지는 배열 생성
b
```

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

```
a.take(b)
```

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