■ 배열 연산 함수

● 배열의 요소값들을 이용하여 연산을 할 수 있게 해주는 함수

sum (합)	mean (평균)	std (표준편차)	exp (지수)
log (로그)	sqrt (제곱근)	max (최대값)	min (최소값)
sin (삼각함수)	cos (삼각함수)	tan (삼각함수)	abs / fabs (절대값)
ceil (올림)	floor (버림)	rint (반올림)	mod (나머지)
add (덧셈)	subtract (뺄셈)	multiply (곱셈)	divide (나눗셈)
power (제곱)	median, var, power, …		

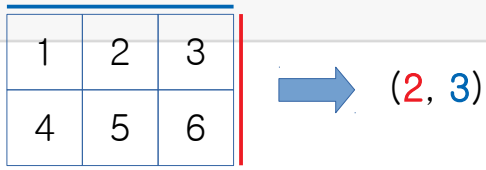
■ 합 (sum)

● 요소들의 합을 구해주는 함수

- 2차원 배열

```
matrix = np.arange(1, 7).reshape(2, 3)
matrix
```

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




```
matrix.sum()
```

```
21
```

```
matrix.sum(axis=0)    축의 방향을 지정하여 합 구하기 (행 ↓)
```

```
array([5, 7, 9])
```

```
matrix.sum(axis=1)    축의 방향을 지정하여 합 구하기 (열 →)
```

```
array([ 6, 15])
```

■ 합 (sum)

● 요소들의 합을 구해주는 함수

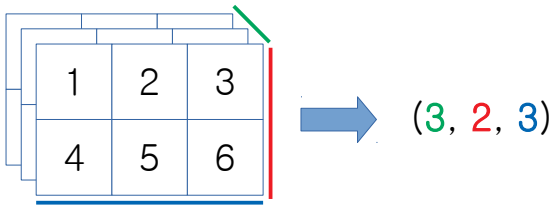
- 3차원 배열

```
tensor = np.arange(1, 19).reshape(3, 2, 3)
tensor
```

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

       [[ 7,  8,  9],
        [10, 11, 12]],

       [[13, 14, 15],
        [16, 17, 18]]])
```




```
tensor.sum()
```

```
171
```

```
tensor.sum(axis=0)
```

```
array([[21, 24, 27],
       [30, 33, 36]])
```

```
tensor.sum(axis=1)
```

```
array([[ 5,  7,  9],
       [17, 19, 21],
       [29, 31, 33]])
```

```
tensor.sum(axis=2)
```

```
array([[ 6, 15],
       [24, 33],
       [42, 51]])
```

■ 평균 (mean)

```
matrix.mean()
```

```
3.5
```

```
matrix.mean(axis=0), matrix.mean(axis=1)
```

```
(array([2.5, 3.5, 4.5]), array([2., 5.]))
```

■ 표준 편차 (std)

```
matrix.std()
```

```
1.707825127659933
```

```
matrix.std(axis=0), matrix.std(axis=1)
```

```
(array([1.5, 1.5, 1.5]), array([0.81649658, 0.81649658]))
```

■ 지수 (exp)

```
np.exp(matrix)
```

```
array([[  2.71828183,   7.3890561 ,  20.08553692],
       [ 54.59815003, 148.4131591 , 403.42879349]])
```

■ 로그 (log)

```
np.log(matrix)
```

```
array([[0.        , 0.69314718, 1.09861229],
       [1.38629436, 1.60943791, 1.79175947]])
```

■ 제곱근 (sqrt)

```
np.sqrt(matrix)
```

```
array([[1.        , 1.41421356, 1.73205081],
       [2.        , 2.23606798, 2.44948974]])
```

■ 삼각함수 (sin)

```
np.sin(matrix)
```

```
array([[ 0.84147098,  0.90929743,  0.14112001],
       [-0.7568025 , -0.95892427, -0.2794155 ]])
```

■ 삼각함수 (cos)

```
np.cos(matrix)
```

```
array([[ 0.54030231, -0.41614684, -0.9899925 ],
       [-0.65364362,  0.28366219,  0.96017029]])
```

■ 삼각함수 (tan)

```
np.tan(matrix)
```

```
array([[ 1.55740772, -2.18503986, -0.14254654],
       [ 1.15782128, -3.38051501, -0.29100619]])
```

■ 배열 합치기 (vstack / hstack / concatenate)

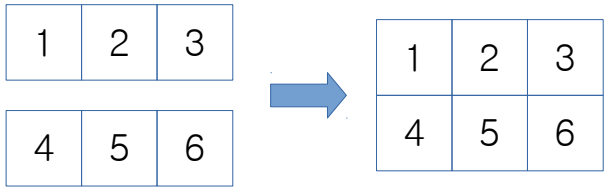
● 배열을 합쳐주는 함수

– vstack (수직)

```
matrix1 = np.array([1, 2, 3])
matrix2 = np.array([4, 5, 6])
```

```
np.vstack((matrix1, matrix2))
```

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

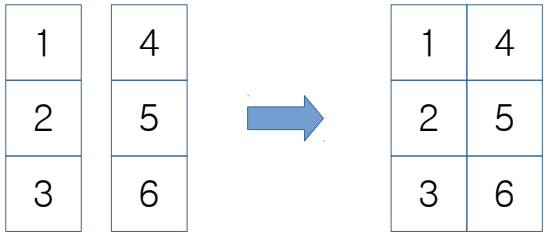


– hstack (수평)

```
matrix1 = np.array([1, 2, 3]).reshape(3, 1)
matrix2 = np.array([4, 5, 6]).reshape(3, 1)
```

```
np.hstack((matrix1, matrix2))
```

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



■ 배열 합치기 (vstack / hstack / concatenate)

● 배열을 합쳐주는 함수

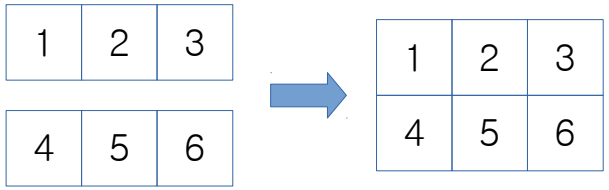
– concatenate (수직 / 수평)

- axis 0

```
matrix3 = np.array([1, 2, 3])
matrix4 = np.array([4, 5, 6])
```

```
np.concatenate((matrix3, matrix4), axis=0)
```

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

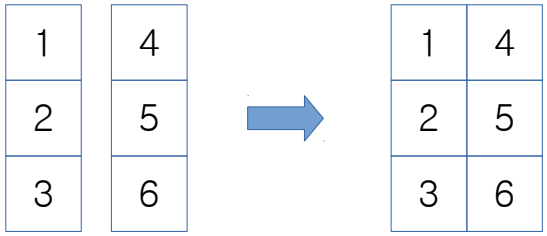


- axis 1

```
matrix3 = np.array([1, 2, 3]).reshape(3, 1)
matrix4 = np.array([4, 5, 6]).reshape(3, 1)
```

```
np.concatenate((matrix3, matrix4), axis=1)
```

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



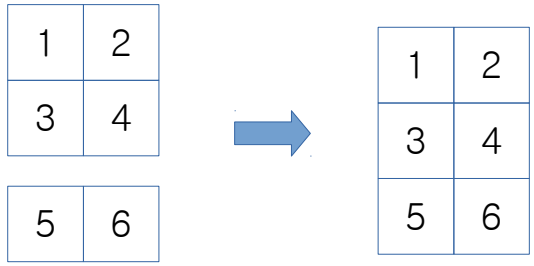
■ 배열 합치기 (vstack / hstack / concatenate)

● 배열을 합쳐주는 함수

– concatenate (수직 / 수평)

• transpose (행 / 열 바꾸기)

```
matrix5 = np.array([[1, 2], [3, 4]])
matrix6 = np.array([[5, 6]])
```




```
np.concatenate((matrix5, matrix6), axis=0)
```

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

```
np.concatenate((matrix5, matrix6.T), axis=1)
```


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

```
np.concatenate((matrix5, matrix6.transpose()), axis=1)
```

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