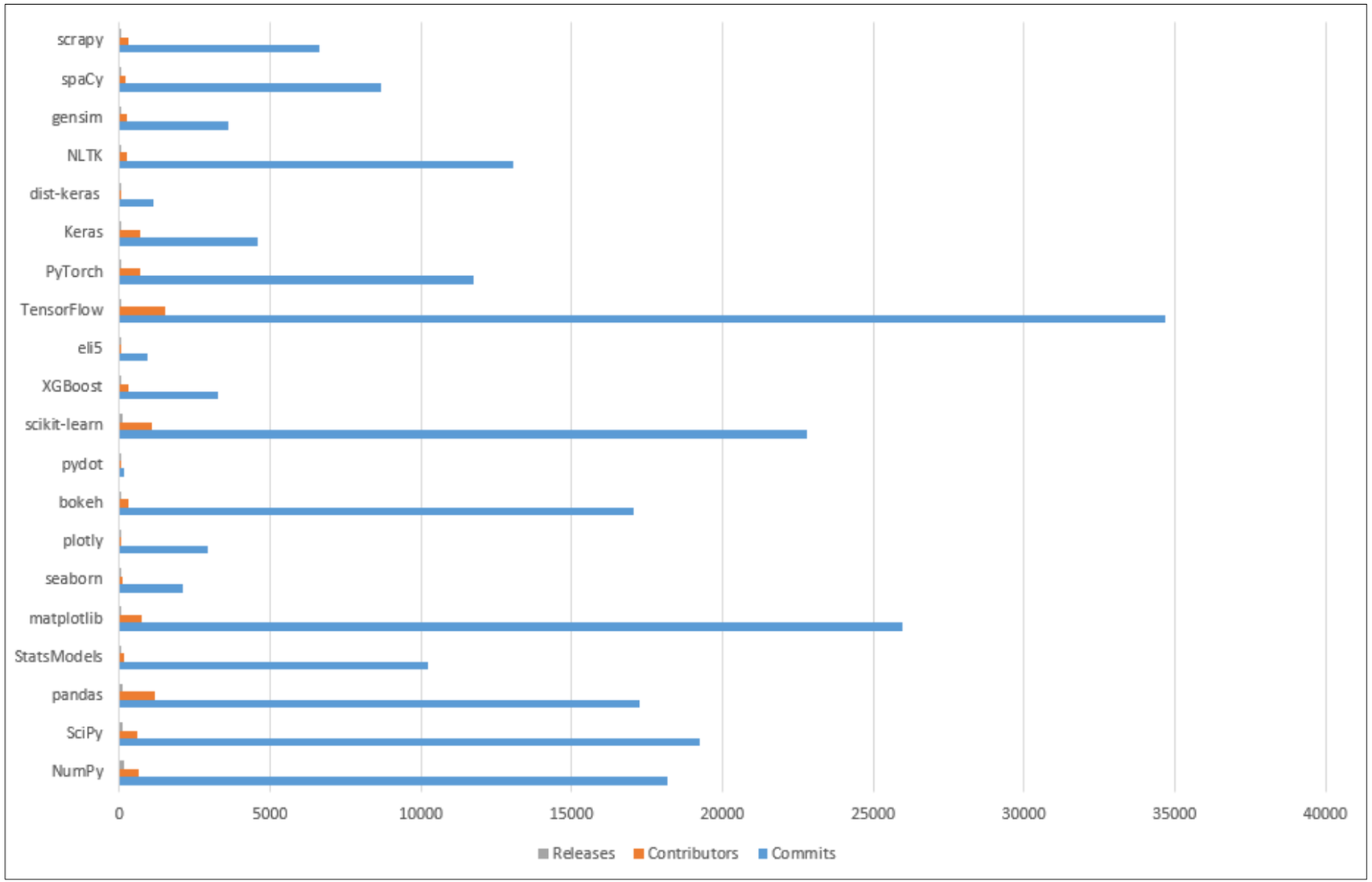
인공지능(AI)

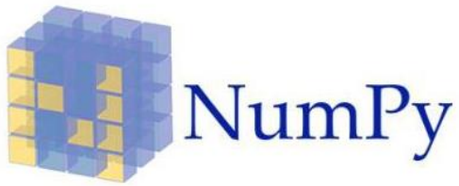
03주차.Numpy와 MNIST 데이터 셋에 대한 이해

비유 내용

Python library TOP 20 Github 활동 데이터



2018년 기준으로 Python library중 가장 많이 사용되는 20개의 library에 대한 Github 활동 데이터는 딥러닝 프레임워크 중 하나인 Tensorflow가 가장 많았고, 그래프 작성을 위한 matplotlib이 그 뒤를 이었으며, 고성능 수치계산을 위한 Numpy는 5위에 rank 되었다.



Numpy(Numerical Python)

C언어로 구현된 파이썬 라이브러리

고성능의 수치 계산을 하기 위해 만들어진 python의 핵심 라이브러리

Python 데이터 타입을 C의 int, float와 같은 데이터 포맷으로 변환 후 연산을 수행하여 속도 개선

* Python은 객체지향언어로 int, float와 같은 데이터 타입도 객체로 처리함으로써 다소 속도 저하

다차원배열을 효과적으로 다루기 위해 ndarray(N-Dimesional Array) 사용

* ndarray는 Tensorflow Tensor와 pandas DataFrame의 기반

벡터화 연산(vectorized operation)을 지원하여 반복문을 사용하지 않고도 배열의 모든 원소에 대해 반복 연산 가능

브로드캐스팅(broadcasting)을 지원하여 배열 간 연산 수행하는데 배열이 크기가 다르더라도 크기가 작은 배열을 자동으로 반복 확장하여 크기가 큰 배열에 맞추어 연산 수행

다차원배열을 효과적으로 다루기 위해 ndarray(N-Dimesional Array) 객체 사용

numpy 설치 : pip install numpy(Anaconda 기본 설치)

numpy 사용 : import numpy as np

ndarray 생성

```
nd1 = np.array([1, 2, 3, 4])

nd2 = np.array([[1, 2, 3], [4, 5, 6]])

nd3 = np.array([[[1, 2, 3], [4, 5, 6], [7, 8, 9], [10, 11, 12]],
                [[13, 14, 15], [16, 17, 18], [19, 20, 21], [22, 23, 24]]])
```

ndarray 확인

```
type(nd1)
```

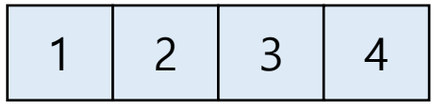
```
numpy.ndarray
```

ndarray 객체는 차원(ndim), 자료형(dtype), 모양(shape), 크기(size) 등의 멤버 변수 포함

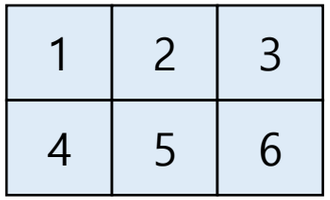
ndarray의 차원은 1D, 2D, 3D, 4D... ND 등이며, 일반적으로 1D ~ 4D 까지 많이 사용

ndarray.ndim 변수를 이용하여 ndarray의 차원 확인 가능

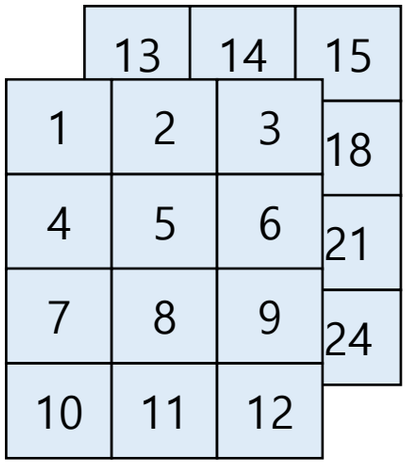
1차원(1D)



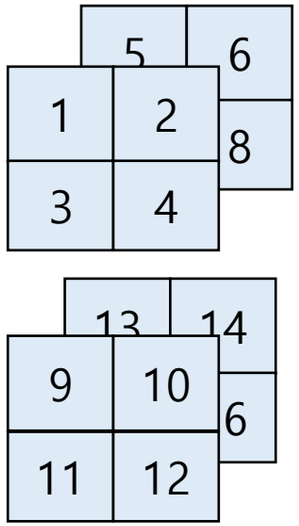
2차원(2D)



3차원(3D)



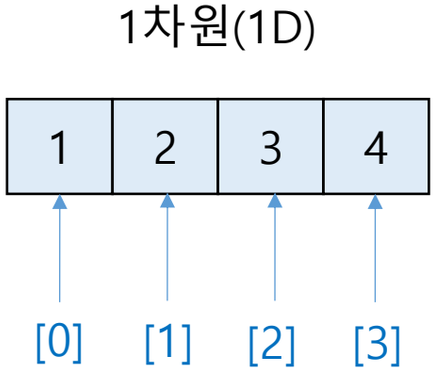
4차원(4D)



...

N차원(ND)

1차원(1D) ndarray




```
nd1 = np.array([1, 2, 3, 4])
```

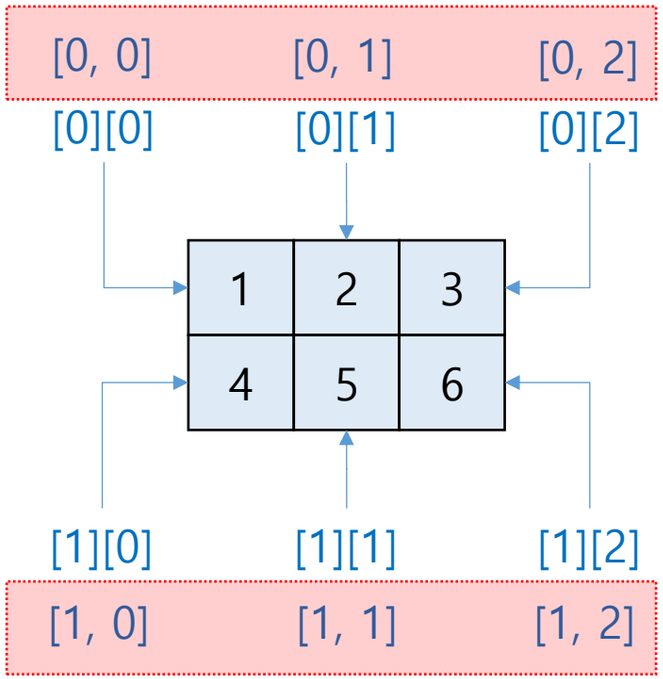
```
print(nd1)
```

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

```
nd1.ndim
```

```
1
```

2차원(2D) ndarray




```
nd2 = np.array([[1, 2, 3], [4, 5, 6]])
```

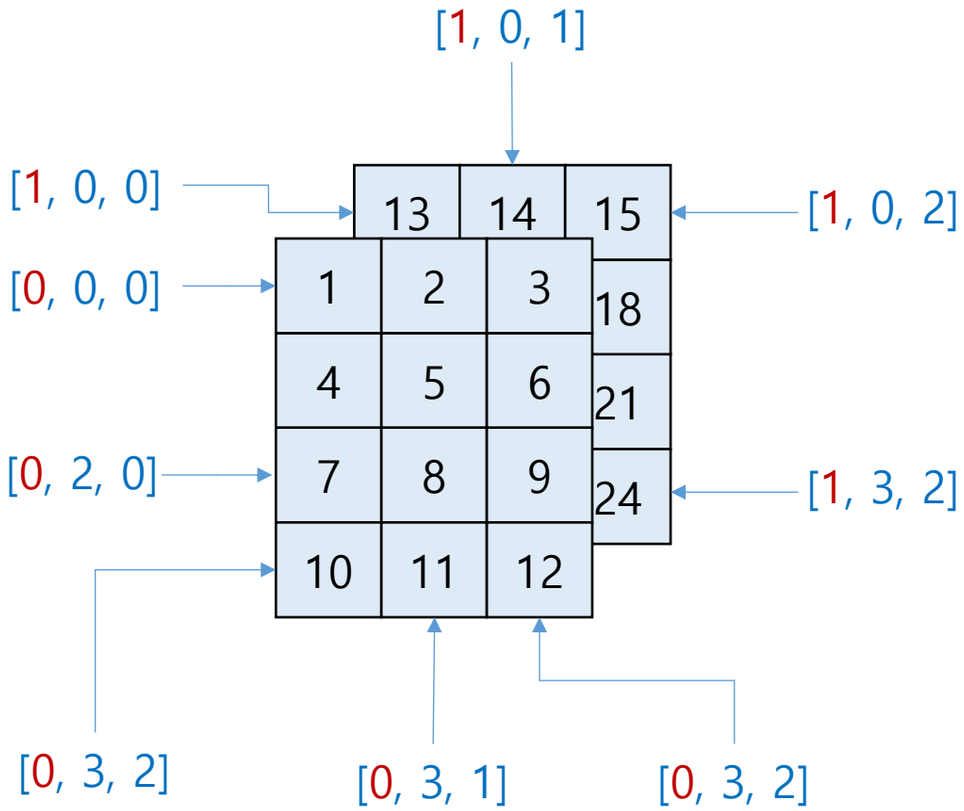
```
print(nd2)
```

```
[[1 2 3]
 [4 5 6]]
```

```
nd2.ndim
```

```
2
```

3차원(3D) ndarray




```
nd3 = np.array([[[1, 2, 3], [4, 5, 6], [7, 8, 9], [10, 11, 12]],
                [[13, 14, 15], [16, 17, 18], [19, 20, 21], [22, 23, 24]]])
```

```
print(nd3)
```

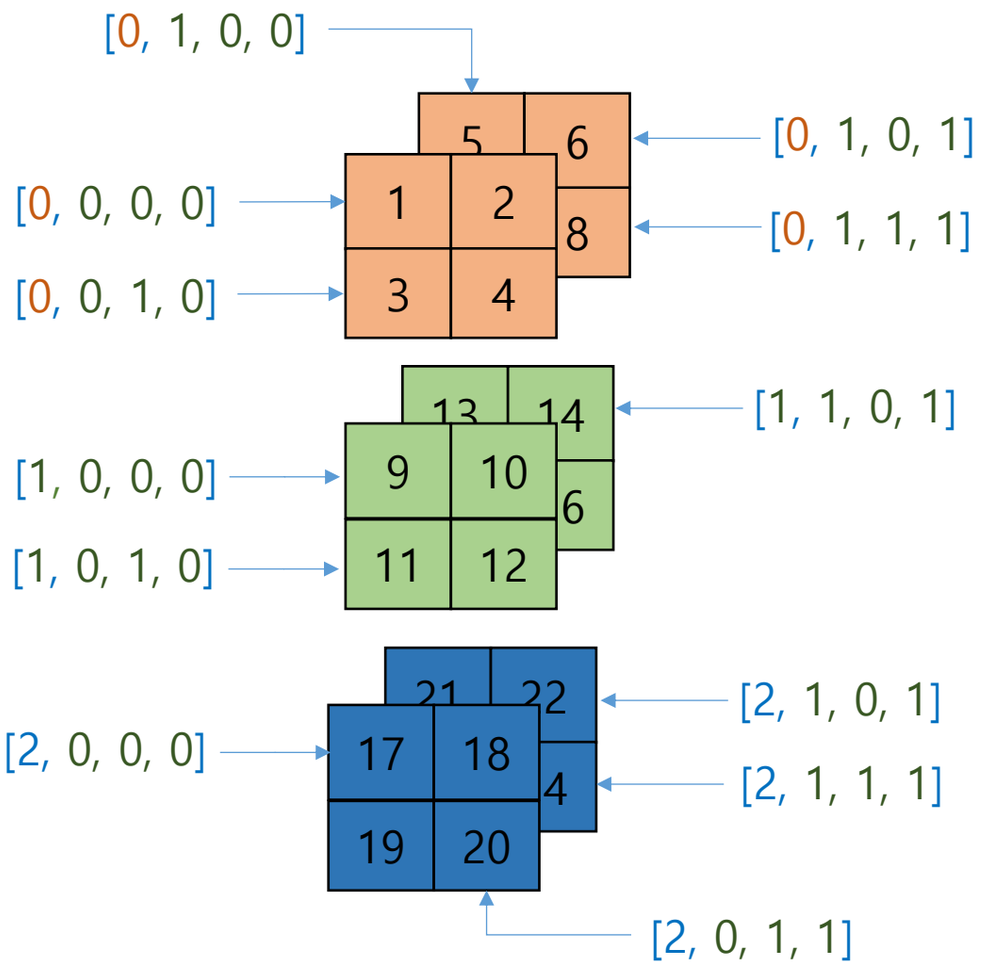
```
[[[ 1  2  3]
  [ 4  5  6]
  [ 7  8  9]
  [10 11 12]]

 [[13 14 15]
  [16 17 18]
  [19 20 21]
  [22 23 24]]]
```

```
nd3.ndim
```

```
3
```

4차원(4D) ndarray




```
nd4 = np.array([[[[1, 2], [3, 4]], [[5, 6], [7, 8]]],
                [[[9, 10], [11, 12]], [[13, 14], [15, 16]]],
                [[[17, 18], [19, 20]], [[21, 22], [23, 24]]]])
```

```
print(nd4)
```

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

  [[ 5  6]
   [ 7  8]]]


 [[[ 9 10]
   [11 12]]

  [[13 14]
   [15 16]]]


 [[[17 18]
   [19 20]]

  [[21 22]
   [23 23]]]]
```

```
nd4.ndim
```

```
4
```

numpy의 ndarray는 하나의 ndarray 객체 당 하나의 자료형만 지정 가능

numpy 자료형의 종류

구분		dtype	설명
숫자형	Boolean	bool	Boolean(True, False)
	integer	int8	8bit 정수형 8bit
		int16	16bit 정수형
		int32	32bit 정수형 32bit
		int64	64bit 정수형
	unsigned integer	uint8	8bit 양의 정수형
		uint16	16bit 양의 정수형
		uint32	32bit 양의 정수형
		uint64	64bit 양의 정수형
	floating-point	float16	16bit 부동 소수형
		float32	32bit 부동 소수형
		float64	64bit 부동 소수형
	complex floating-point	complex64	64bit 복소수형(실수 + 허수)
		complex128	128bit 복소수형(실수 + 허수)
날짜형	datetime	datetime64	64bit 날짜시간형
	timedelata	timedelata64	64bit 날짜차이형
문자형	string	string_	byte 문자열형
	Unicode	str_	유니코드 문자열형
객체형	object	object	객체형

ndarray 자료형 할당

```
nd1 = np.array([1.0, 2.1, 3.0, 4.2]) # float64로 default 할당
nd2 = np.array([[1, 2, 3], [4, 5, 6]]) # int32로 default 할당
nd2 = np.array([[1, 2, 3], [4, 5, 6]], dtype = np.float32)
nd2 = np.array([[1, 2, 3], [4, 5, 6]], dtype = 'float64')
```

ndarray 자료형 확인

```
print(nd1.dtype)
```

ndarray 자료형 형변환

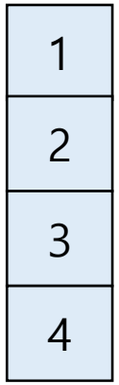
```
nd1 = nd1.astype(np.int32)
```

머신러닝/딥러닝에서 입력데이터의 shape를 맞추고 확인하는 과정은 매우 중요함

ndarray.shape 변수를 이용하여 shape 확인 가능하며 결과는 python 튜플 형태로 반환

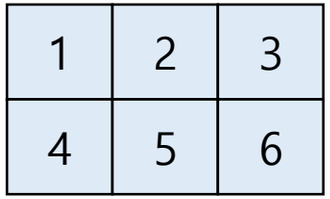
1차원(1D)

shape : (4,)



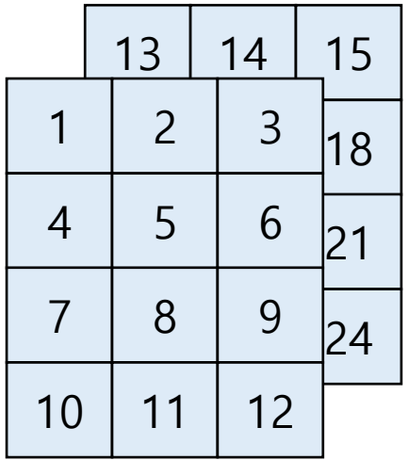
2차원(2D)

shape : (2, 3)



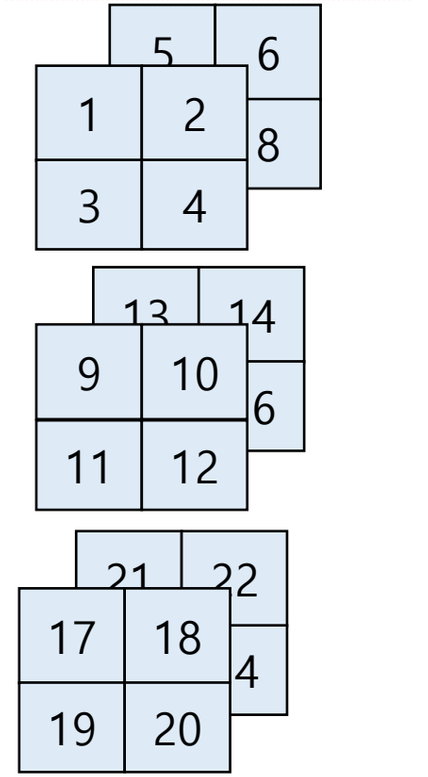
3차원(3D)

shape : (2, 4, 3)



4차원(4D)

shape : (3, 2, 2, 2)



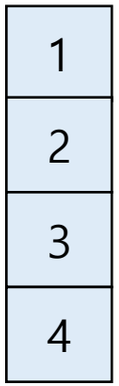
ndarray의 전체 크기(size)이며, ndarray.size 변수를 이용하여 size 확인 가능

ndarray의 size는 모든 element 갯수이며, shape의 모든 값들을 곱한 값

1차원(1D)

4X1

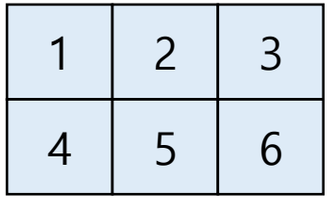
shape : (4,), size : 4



2차원(2D)

2X3

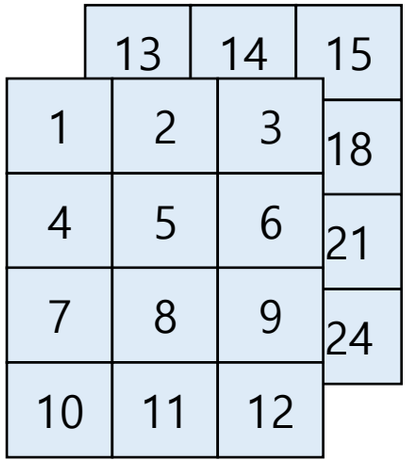
shape : (2, 3), size : 6



3차원(3D)

2X4X3

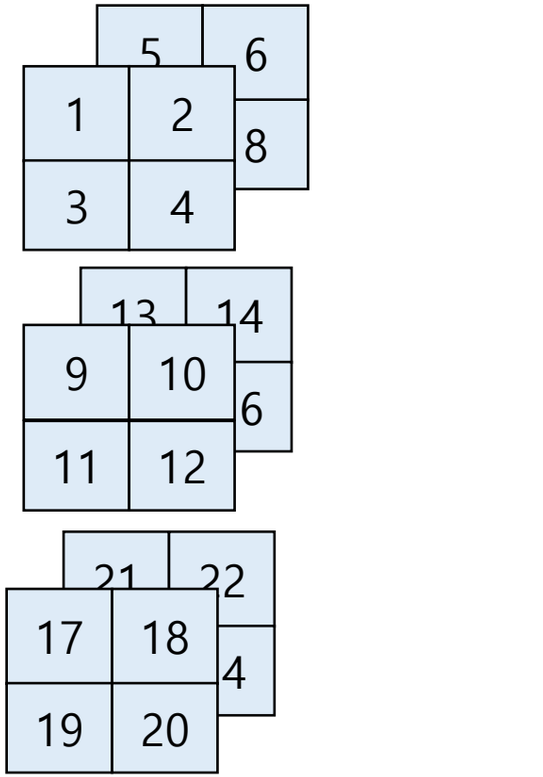
shape : (2, 4, 3), size : 24



4차원(4D)

3X2X2X2

shape : (3, 2, 2, 2), size : 24



입력데이터의 shape이 불일치한 경우, 이를 맞추기 의한 shape의 변환 과정 필요

ndarray.reshape 변수를 이용하여 shape의 변환이 가능하며, 파라미터는 python tuple 형태

동일 차원내 변환 및 이종 차원간 변환 가능하며, ndarray의 size가 동일해야 reshape이 가능

1차원(1D)

shape : (24,)

2차원(2D)

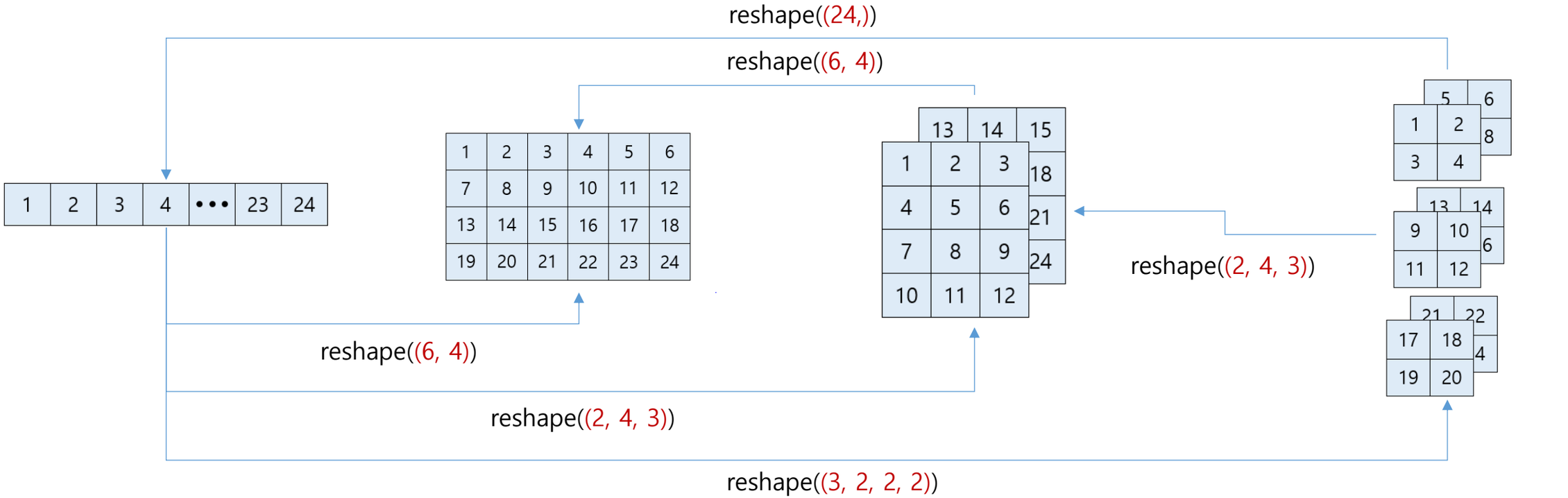
shape : (6, 4)

3차원(3D)

shape : (2, 4, 3)

4차원(4D)

shape : (3, 2, 2, 2)



np.zeros(shape) : shape의 모양대로 0으로 채운 ndarray 생성

```
np.zeros((4, 3))
```

```
array([[0., 0., 0.],
       [0., 0., 0.],
       [0., 0., 0.],
       [0., 0., 0.]])
```

np.ones(shape) : shape의 모양대로 1으로 채운 ndarray 생성

```
np.ones((3, 4))
```

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

np.arange(start, stop, step) : python의 range함수와 동일

```
np.arange(12)
```

```
array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11])
```

```
np.arange(1, 10, 2)
```

```
array([1, 3, 5, 7, 9])
```

```
np.arange(0, 10, 2)
```

```
array([0, 2, 4, 6, 8])
```

```
np.arange(12).reshape(4, 3)
```

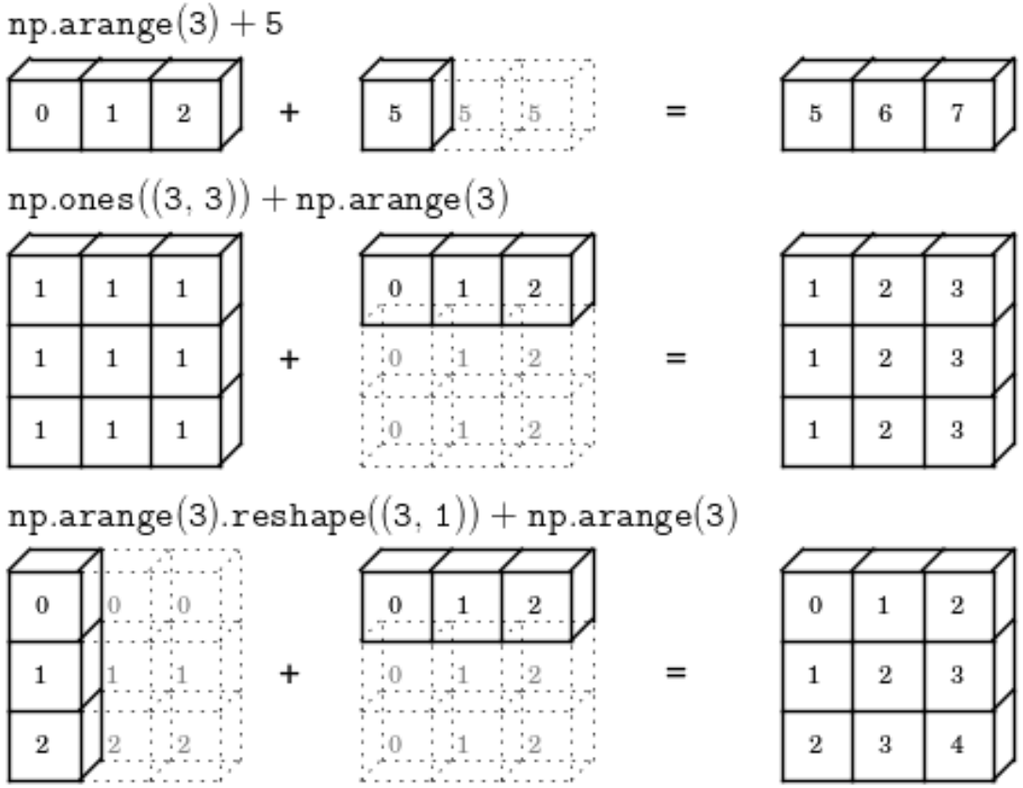
```
array([[ 0,  1,  2],
       [ 3,  4,  5],
       [ 6,  7,  8],
       [ 9, 10, 11]])
```

ndarray는 벡터화 연산(vectorized operation)을 지원하여 별도의 반복문(for loop)을 사용하지 않고 element-wise(같은 위치의 값 끼리)한 반복 연산 가능

1	2	+	2	2	=	3	4	1	2	-	2	2	=	-1	0
3	4		2	2		5	6	3	4		2	2		1	2
5	6		2	2		7	8	5	6		2	2		3	4

1	2	*	2	2	=	2	4	1	2	/	2	2	=	0.5	1
3	4		2	2		6	8	3	4		2	2		1.5	2
5	6		2	2		10	12	5	6		2	2		2.5	3

ndarray의 브로드캐스팅(broadcasting)은 연산이 가능한 형태로 자동 reshape한 후 반복된 값으로 자동으로 할당하여 연산을 수행



출처 : http://www.astroml.org/book_figures/appendix/fig_broadcast_visual.html

numpy.ipynb 실습

```
In [3]: import numpy as np

In [90]: np.__version__
Out[90]: '1.15.4'
```

1D ndarray

```
In [5]: nd1 = np.array([1, 2,
```

```
In [23]: print(nd1)
[1 2 3 4]
```

```
In [15]: nd1.ndim
Out[15]: 1
```

2D ndarray

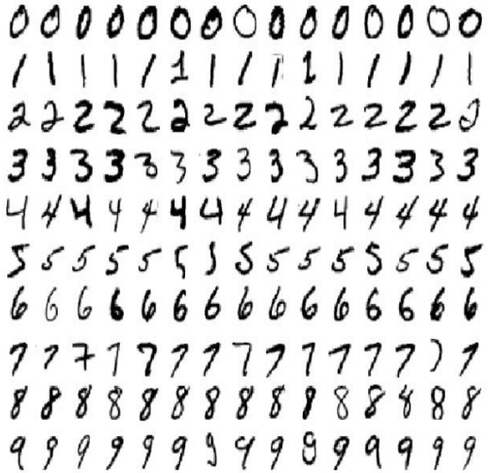
```
In [137]: nd2 = np.array([[1, 2, 3], [4, 5, 6]])
```

```
In [24]: print(nd2)
[[1 2 3]
 [4 5 6]]
```

```
In [20]: nd2.ndim
Out[20]: 2
```

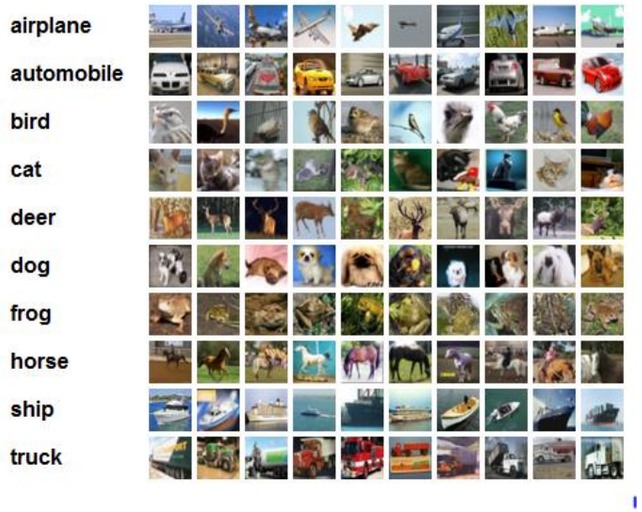
딥러닝 학습을 위해 많이 사용되는 데이터 셋으로 MNIST, Fashion MNIST, CIFAR-10, CIFAR-100 데이터 셋이 있음

MNIST 데이터 셋

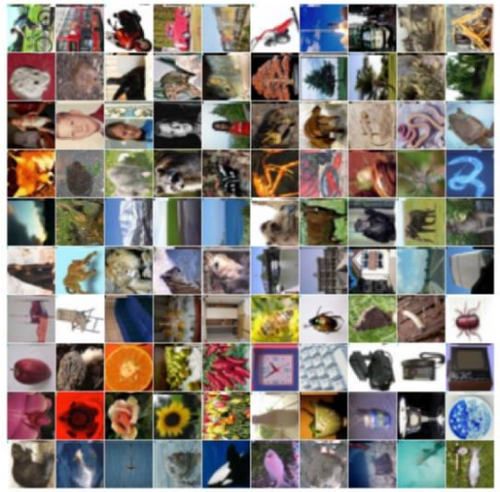


Fashion MNIST 데이터 셋

CIFAR-10 데이터 셋



CIFAR-100 데이터 셋



MNIST 데이터 셋은 인공지능 연구의 권위자 LeCun교수가 만든 데이터 셋(http://yann.lecun.com/exdb/mnist/)

컴퓨터 비전(vision) 분야에서 머신러닝/딥러닝 학습 및 예측을 위해 많이 사용

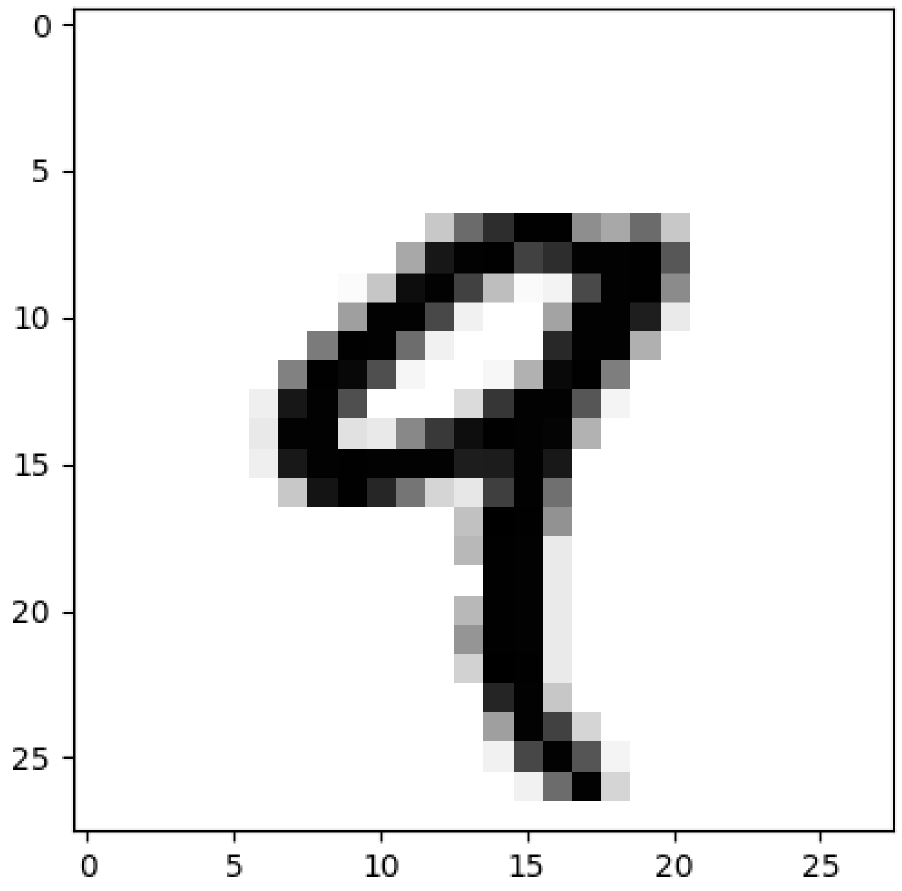
(tensorflow, keras 등 대부분의 프레임워크에서 기본 함수로 제공)

0에서 9까지의 숫자로 고정 크기 이미지 28 x 28 픽셀, 흑백 이미지

ndarray type의 mnist data 예시

```
[[  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0  55 148 210 253 253 113  87 148  55   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0  87 232 252 253 189 210 252 252 253 168   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   4  57 242 252 190  65   5  12 182 252 253 116   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0  96 252 252 183  14   0   0  92 252 252 225  21   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0 132 253 252 146  14   0   0   0 215 252 252  79   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0 126 253 247 176   9   0   0   8  78 245 253 129   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0  16 232 252 176   0   0   0  36 201 252 252 169  11   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0  22 252 252  30  22 119 197 241 253 252 251  77   0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0  16 231 252 253 252 252 252 226 227 252 231   0   0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0  55 235 253 217 138  42  24 192 252 143   0   0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0  62 255 253 109   0   0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0  71 253 252  21   0   0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0 253 252  21   0   0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0  71 253 252  21   0   0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0 106 253 252  21   0   0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0  45 255 253  21   0   0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0 218 252  56   0   0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0  96 252 189  42   0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0  14 184 252 170  11   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0  14 147 252  42   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0]]
```

ndarray의 이미지 표현



MNIST 데이터 셋은 훈련을 위한 training 데이터 셋과 예측 및 평가를 위한 test 데이터 셋으로 구성

training 데이터 셋은 train_images과 train_labels의 pair로 구성
test 데이터 셋은 test_images과 test_labels의 pair로 구성

train_images : 60,000개의 training data 이미지
train_labels : 60,000개의 training data 에 대한 label

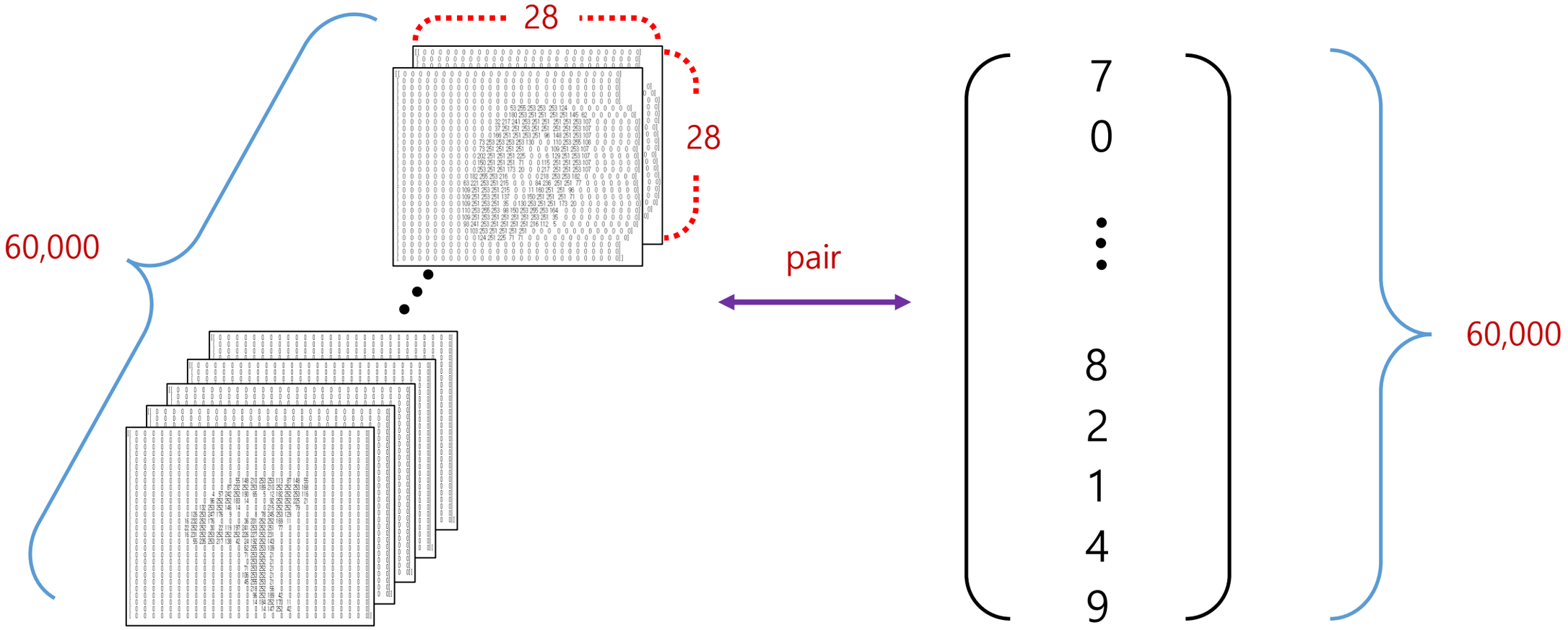
test_images : 10,000개의 training data 이미지
test_labels : 10,000개의 training data 에 대한 label

MNIST training_images

shape : (60000, 28, 28)

MNIST training_labels

shape : (60000,)

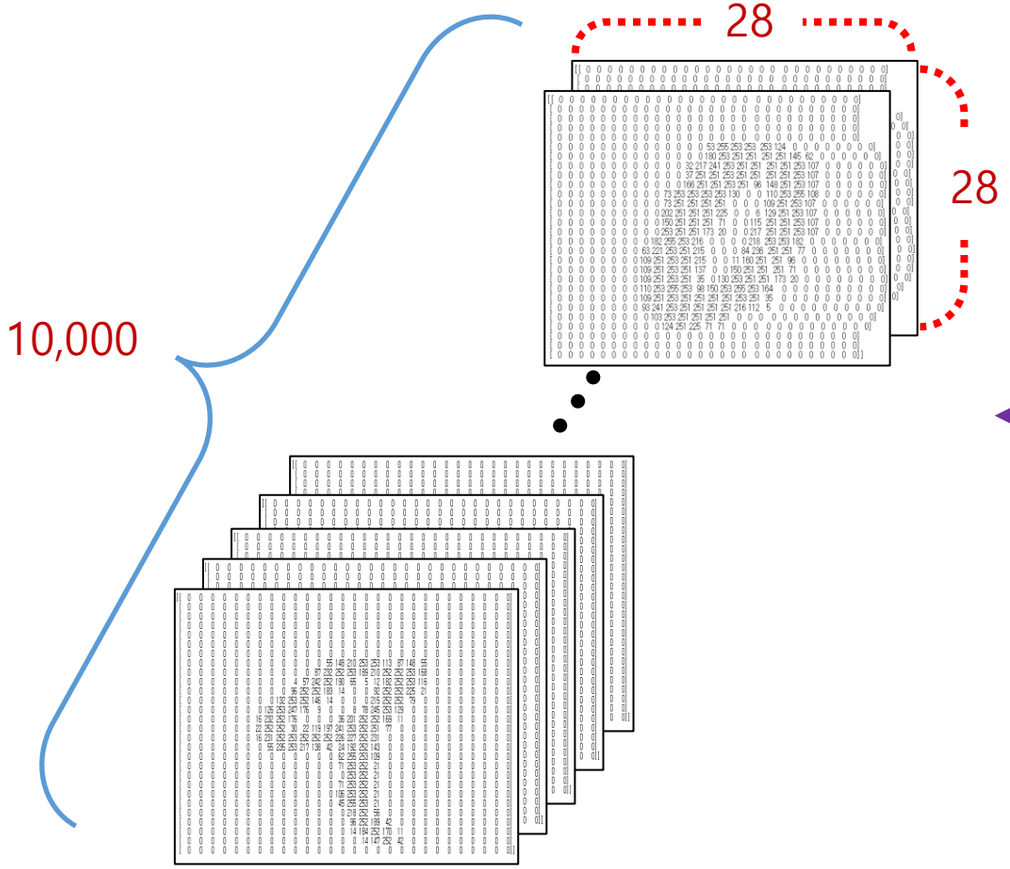


MNIST test_images

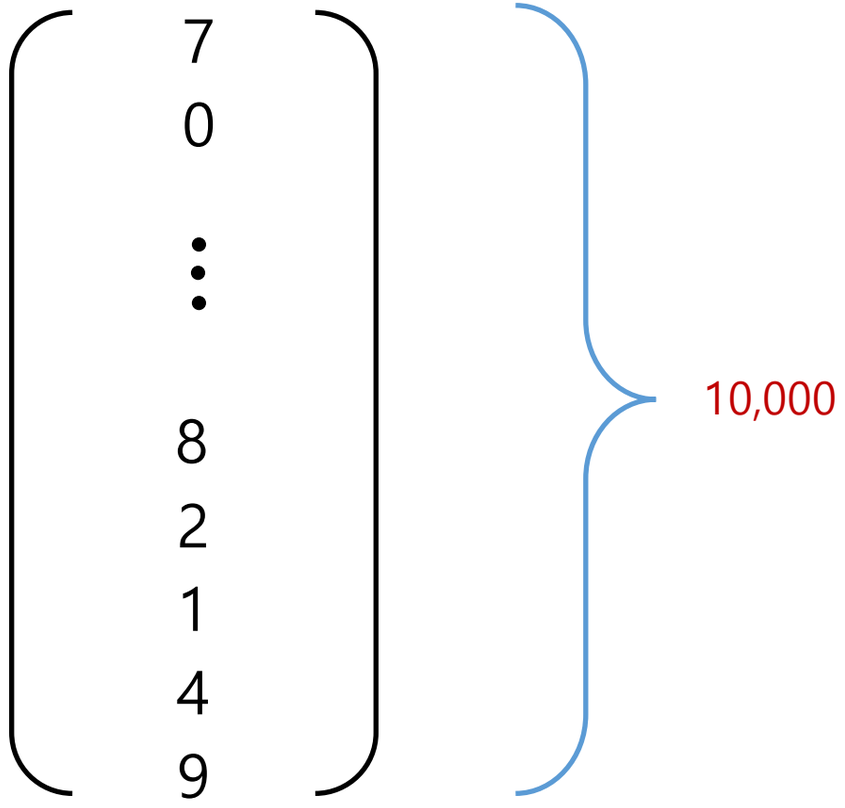
shape : (10000, 28, 28)

MNIST test_labels

shape : (10000,)



pair



Keras를 이용한 MNIST 데이터 셋 로드

```
import keras
from keras.datasets import mnist
(train_images, train_labels), (test_images, test_labels) = mnist.load_data()
```

mnist.load_data()

https://s3.amazonaws.com/img-datasets/mnist.npz 에서 MNIST 데이터셋 다운로드

C:₩Users₩사용자계정₩.keras₩datasets₩mnist.npz 로 저장

최초 한번만 수행되며,
이후에는 로컬 PC의
mnist.npz 파일을 읽어서 로드

mnist.ipynb 실습

In [1]:
```
import keras
from keras.datasets import mnist
```

```
C:₩ProgramData₩Anaconda3₩lib₩site-packages₩h5py₩__init__.py:34: FutureWarning: Conversion of the second argument of issubdtype from `float`
to `np.floating` is deprecated. In future, it will be treated as `np.float64 == np.dtype(float).type`.
  from ._conv import register_converters as _register_converters
Using TensorFlow backend.
```

In [2]:
```
import matplotlib.pyplot as plt
```

In [3]:
```
(train_images, train_labels
```

In [7]:
```
type(train_images)
```

Out [7]:
```
numpy.ndarray
```

In [8]:
```
digit = train_images[4]
digit
```

...

In [10]:
```
plt.imshow(digit, cmap=plt.cm.binary)
plt.show()
```