医療とAI・ビッグデータ応用
②MNISTの読み込みと加工

統合教育機構
須藤毅顕

医療とAI・ビッグデータ入門

・pythonの基本
・機械学習とは（アヤメのデータ）
・深層学習とは（肺のレントゲン画像）

体験してもらう（コピー＆ペースト）

医療とAI・ビッグデータ応用

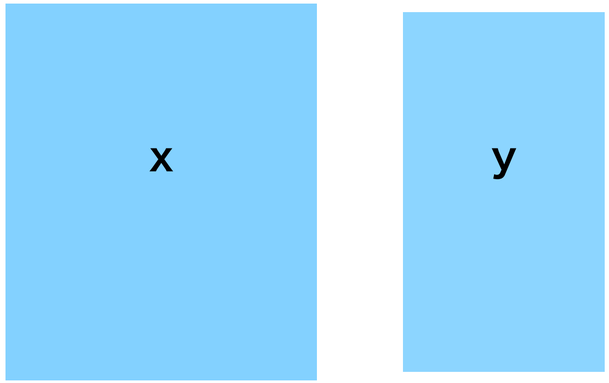
深層学習
・MLP（多層パーセプトロン）
・CNN（畳み込みニューラルネットワーク）

理解してもらう（自分でタイピング、グループ演習）

深層学習(教師あり機械学習)の復習

データを用意する

x(特徴量データ)　　**y(正解データ)**



深層学習(教師あり機械学習)の復習

データを用意する

x(特徴量データ)

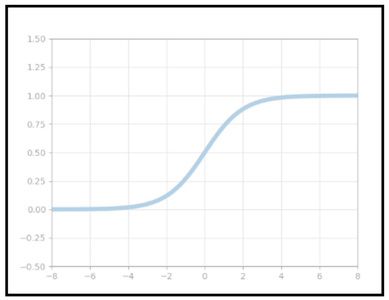
x

y(正解データ)

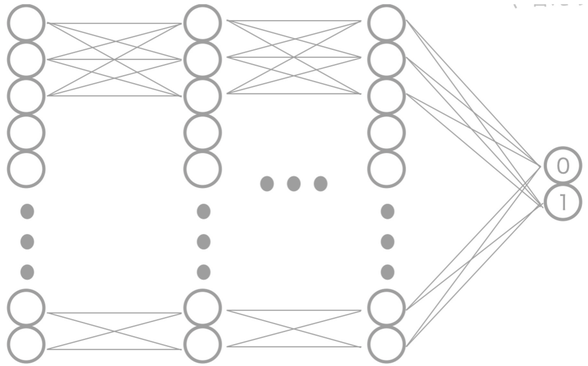
y

学習させる

ロジスティック回帰分析



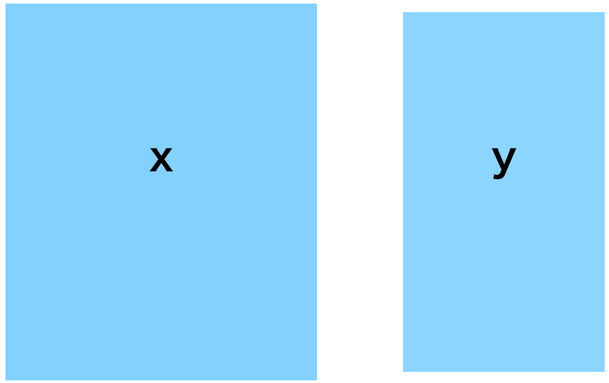
ニューラルネットワーク



深層学習(教師あり機械学習)の復習

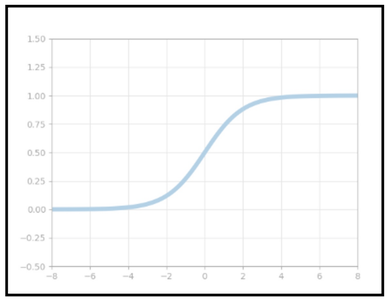
データを用意する

x(特徴量データ)　　**y(正解データ)**

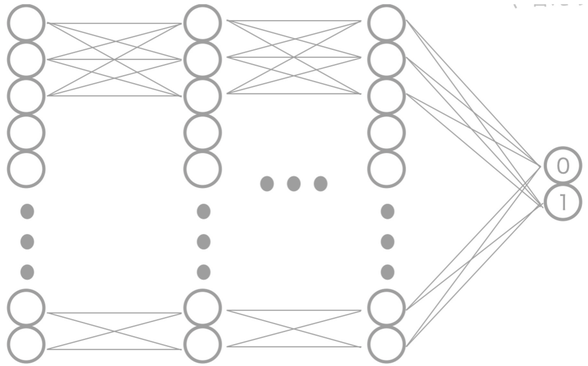


学習させる

ロジスティック回帰分析



ニューラルネットワーク



評価する
(分類、予測など)

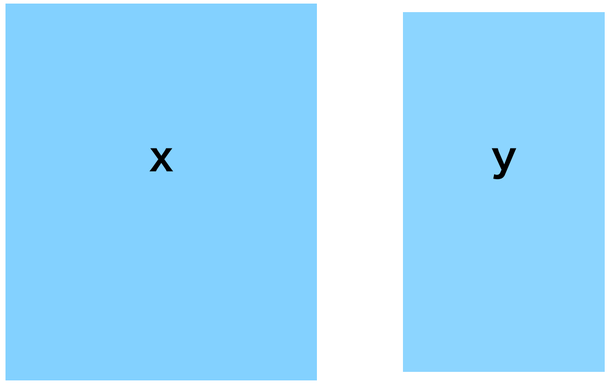
病気か否か

犬か猫か

深層学習(教師あり機械学習)の復習

データを用意する

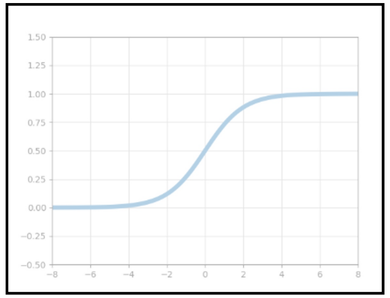
x(特徴量データ)　y(正解データ)



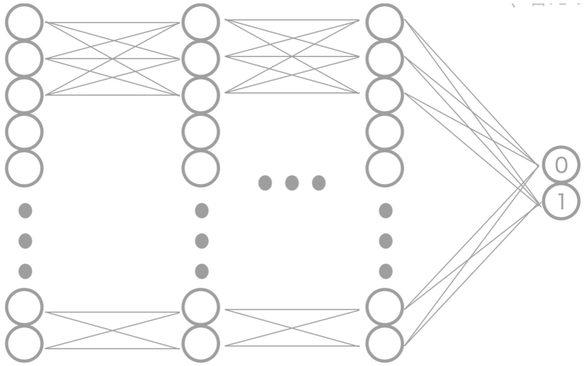
データを配列に整える

学習させる

ロジスティック回帰分析



ニューラルネットワーク



深層学習(教師あり機械学習)の復習

表データ(数値データ)

	X(特徴量データ)			*y*(正解データ)
	収縮時血圧	体重	年齢	糖尿病(有:*1*,無:*0*)
1	150	70	60	1
2	120	40	35	0
3	144	45	40	1
4	162	56	50	1
5	98	40	32	0
6	128	59	35	0
7	155	77	45	1

データを配列に整える

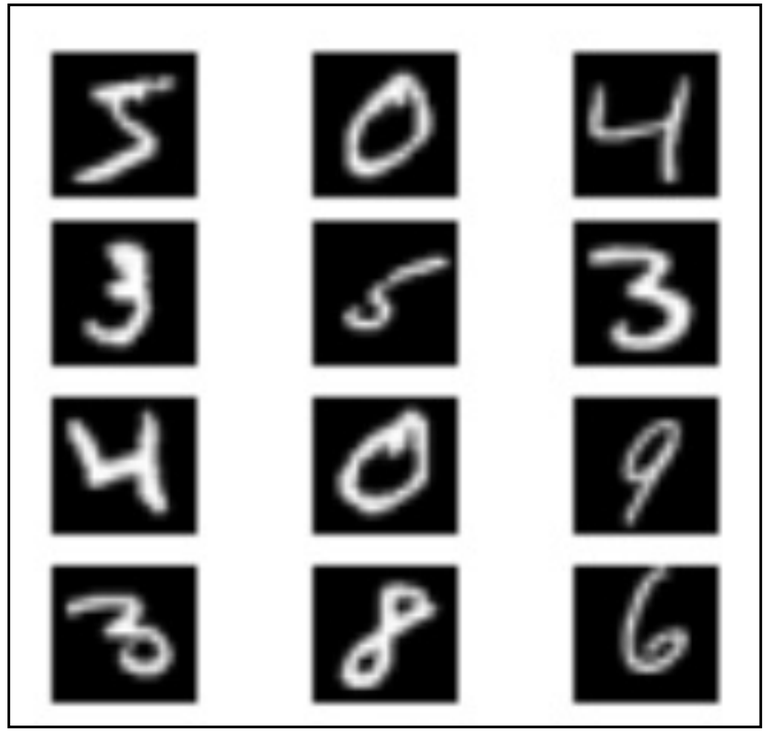
配列データ

x_train
```
[[150, 70, 60],
[120, 40, 35],
[144, 45, 40],
[162, 56, 50],
[98, 40, 32],
[128, 59, 35],
[155, 77, 45]]
```

y_train
```
[[1],
 [0],
 [1],
 [1],
 [0],
 [0],
 [1]]
```

画像を読み込んで学習出来るデータ(配列)にする

画像データ



配列データ

x_train
[[150, 70, 60],
[120, 40, 35],
[144, 45, 40],
[162, 56, 50],
[98, 40, 32],
[128, 59, 35],
[155, 77, 45]]

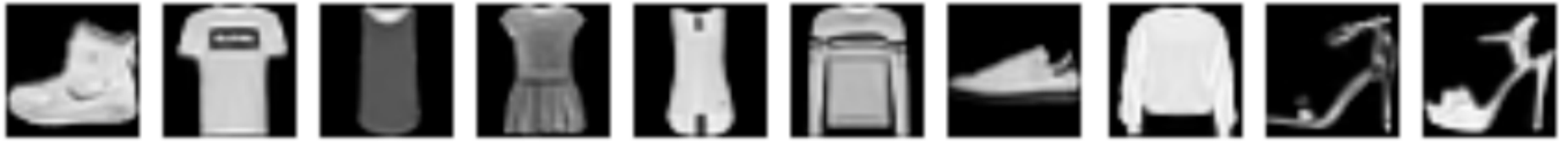
y_train
[[1],
[0],
[1],
[1],
[0],
[0],
[1]]

Pythonには機械学習を実践するために多くの画像セットが用意されている

MNIST：0~9の文字画像のデータ

FASHION-MNIST：白黒の洋服の画像データ

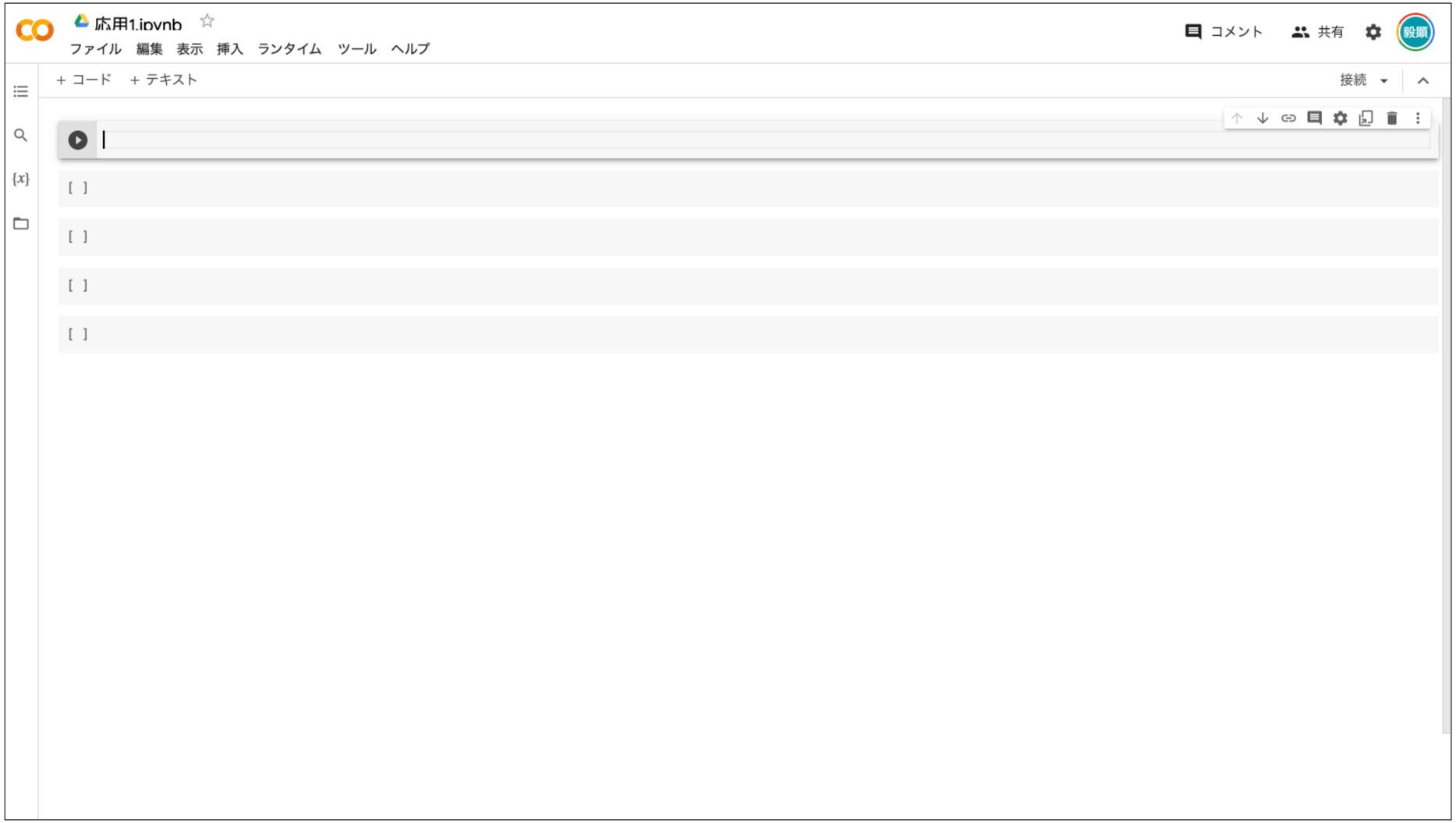
0 : T-shirt/top、1 : Trouser、2 : Pullover、3 : Dress、4 : Coat、5 : Sandal
6 : Shirt、7 : Sneaker、8 : Bag、9 : Ankle boot



CIFAR10

0 : airplane、1 : automobile、2 : bird、3 : cat、4 : deer、5 : dog
6 : frog、7 : horse、8 : ship、9 : truck

colaboratoryを準備しよう

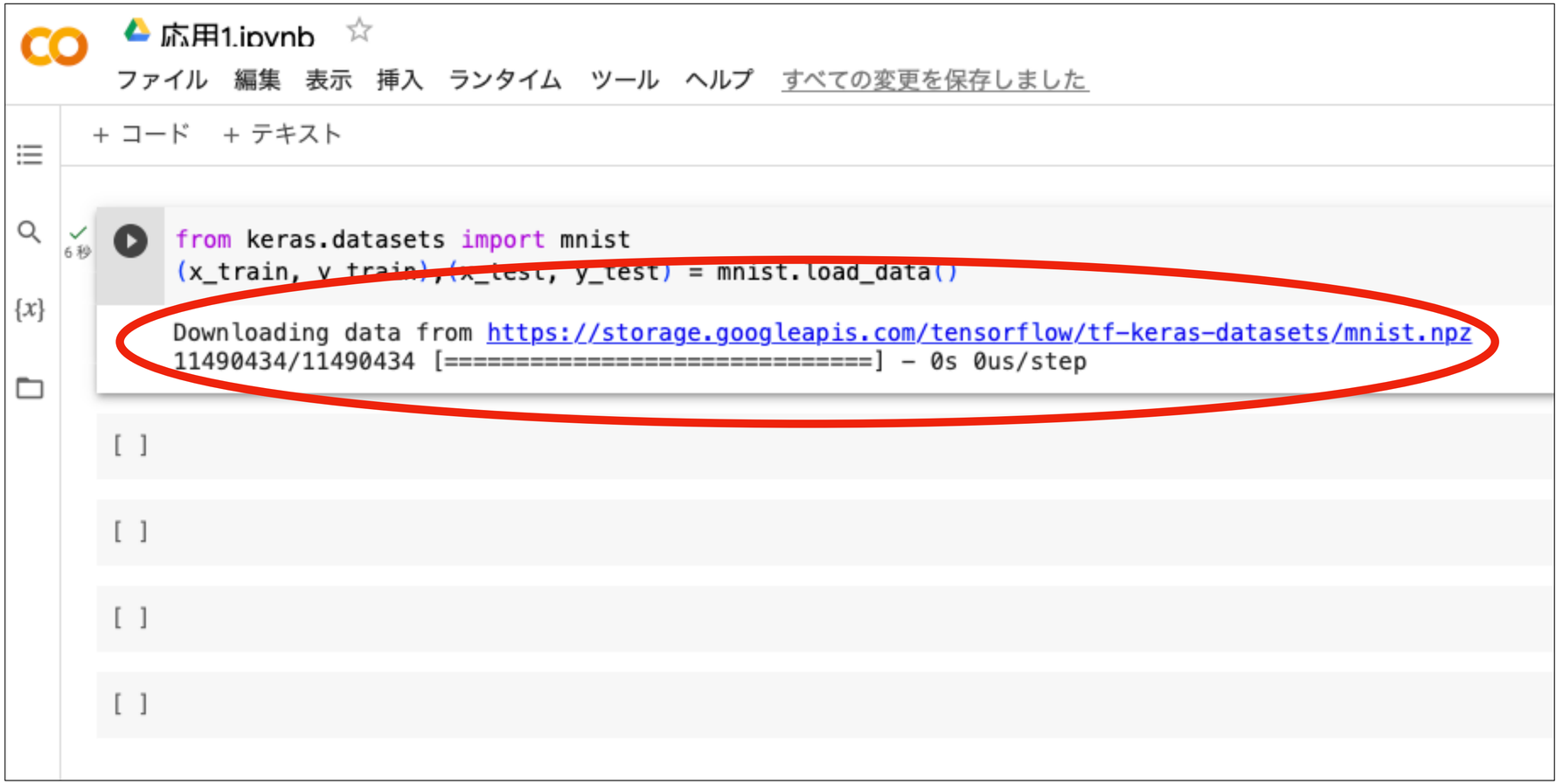


MNISTデータを扱ってみよう


```
from keras.datasets import mnist
(x_train, y_train), (x_test, y_test) = mnist.load_data()
```

この2行をセルに書き込んで実行してみよう

MNISTデータを扱ってみよう



実行するとmnistのデータがダウンロードされて
変数に代入されます

MNISTデータの理解

```
from keras.datasets import mnist
```

kerasというライブラリの中のdatasetsの中のmnistという関数を読み込む
mnist.〜〜でmnistの中の機能が使える

```
(x_train, y_train),(x_test, y_test) = mnist.load_data()
```

変数4つ = mnist.load_data()で、mnistの中に入っているデータが
①学習用データ、②学習用データの正解(ラベル)、③テスト用データ、④テスト用データの正解(ラベル)
の順に代入される

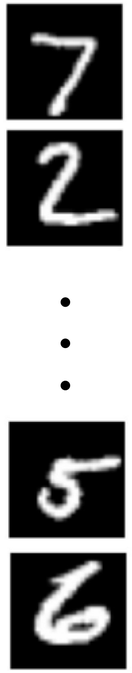
関数？変数？となった人はぜひ入門編のスライドを確認してください

(x_train, y_train),(x_test, y_test) = mnist.load_data()

60000個　60000個　10000個　10000個

mnistのdataを読み込む

5
0
4
1
⋮
6
8



7
2
⋮
5
6

x_train : 60000枚の画像の配列データ
y_train : 60000枚の正解の数字の配列データ
x_test : 10000枚の画像の配列データ
y_test : 10000枚の正解の数字の配列データ

読み込んだものはすでにnumpy配列になっている

```
print(x_train.shape)
print(y_train.shape)
print(x_test.shape)
print(y_test.shape)
```

変数.shape：配列の形を調べる(numpy配列)

```
(60000, 28, 28)
(60000,)
(10000, 28, 28)
(10000,)
```

28×28の文字画像が60000枚
60000枚の正解の数字
28×28の文字画像が10000枚
10000枚の正解の数字

```
import numpy as np
x = np.array([1,2,3,4])
print(x.shape)
print(x[0])
```

np.array(配列)
numpy配列を作成

numpy配列.shape
配列の形状を取得

```
import numpy as np
x = np.array([1,2,3,4])
print(x)          → [1 2 3 4]
print(x.shape)    → (4,)
print(x[0])       → 1
```

4つの要素からなる１次元配列
xの１つ目

x[0] x[1] x[2] x[3]

[1 , 2 , 3 , 4]

[ ]が1つなので1次元配列

```
x2 = np.array([[1,2,3],[4,5,6],[7,8,9]])
print(x2.shape)    →   (3,3)
print(x2[1])       →   [4 5 6]
print(x2[2][1])    →   8
```

3 × 3 の2次元配列

x2の2つ目

x2[2]の２つ目

[[1 , 2 , 3],[4 , 5 , 6],[7 , 8 , 9]]

[ ]の中に[ ]があるので2次元配列

```
x2 = np.array([[1,2,3],[4,5,6],[7,8,9])
print(x2.shape)     →   (3,3)
print(x2[1])        →   [4 5 6]
print(x2[2][1])     →   8
```

3 × 3 の2次元配列

x2の2つ目

x2[2]の２つ目

[[1 , 2 , 3] , [4 , 5 , 6] , [7 , 8 , 9]]

外側の[ ]だけに注目すると、コンマ(,)区切りで3つの要素が存在

[ ]の中に[ ]があるので2次元配列

```
x2 = np.array([[1,2,3],[4,5,6],[7,8,9])
print(x2.shape)      →   (3,3)
print(x2[1])         →   [4 5 6]
print(x2[2][1])      →   8
```

3 × 3 の2次元配列

x2の2つ目

x2[2]の２つ目

x2[0] x2[1] x2[2]

[[1 , 2 , 3] , [4 , 5 , 6] , [7 , 8 , 9]]

外側の[ ]だけに注目すると、コンマ(,)区切りで3つの要素が存在

[ ]の中に[ ]があるので2次元配列

```
x2 = np.array([[1,2,3],[4,5,6],[7,8,9])
print(x2.shape)     →   (3,3)
print(x2[1])        →   [4 5 6]
print(x2[2][1])     →   8
```

3 × 3 の2次元配列

x2の2つ目

x2[2]の２つ目

x2[0] x2[1] x2[2]

[[1 , 2 , 3],[4 , 5 , 6],[7 , 8 , 9]]

外側の[ ]だけに注目すると、コンマ(,)区切りで3つの要素が存在

[ ]の中に[ ]があるので2次元配列

```
x2 = np.array([[1,2,3],[4,5,6],[7,8,9])
print(x2.shape)   →   (3,3)
print(x2[1])      →   [4 5 6]
print(x2[2][1])   →   8
```

3 × 3 の2次元配列

x2の2つ目

x2[2]の２つ目

x2[0] x2[1] x2[2]

[[1 , 2 , 3],[4 , 5 , 6],[7 , 8 , 9]]

x2[2]の中身である[7,8,9]はコンマ区切りで要素が3つ

[ ]の中に[ ]があるので2次元配列

```
x2 = np.array([[1,2,3],[4,5,6],[7,8,9])
print(x2.shape)   →  (3,3)
print(x2[1])      →  [4 5 6]
print(x2[2][1])   →  8
```

3 × 3 の2次元配列

x2の2つ目

x2[2]の２つ目

x2[2]

x2[0] x2[1]

x2[2][0] x2[2][2]

[[1 , 2 , 3],[4 , 5 , 6],[7 , 8 , 9]]

x2[2][1]

x2[2]の中身である[7,8,9]はコンマ区切りで要素が3つ

[ ]の中に[ ]があるので2次元配列

```
x2 = np.array([[1,2,3],[4,5,6],[7,8,9])
print(x2.shape)   →  (3,3)
print(x2[1])      →  [4 5 6]
print(x2[2][1])   →  8
```

3 × 3 の2次元配列

x2の2つ目

x2[2]の２つ目

x2[2]

x2[0] x2[1] x2[2][0] x2[2][2]

[[1 , 2 , 3],[4 , 5 , 6],[7 , 8 , 9]]

x2[2][1]

x2[2]の中身である[7,8,9]はコンマ区切りで要素が3つ

[ ]の中に[ ]があるので2次元配列

```
x2 = np.array([[1,2,3],[4,5,6],[7,8,9])
print(x2.shape)          →   (3,3)
```

3 × 3 の2次元配列

```
print(x2)
```

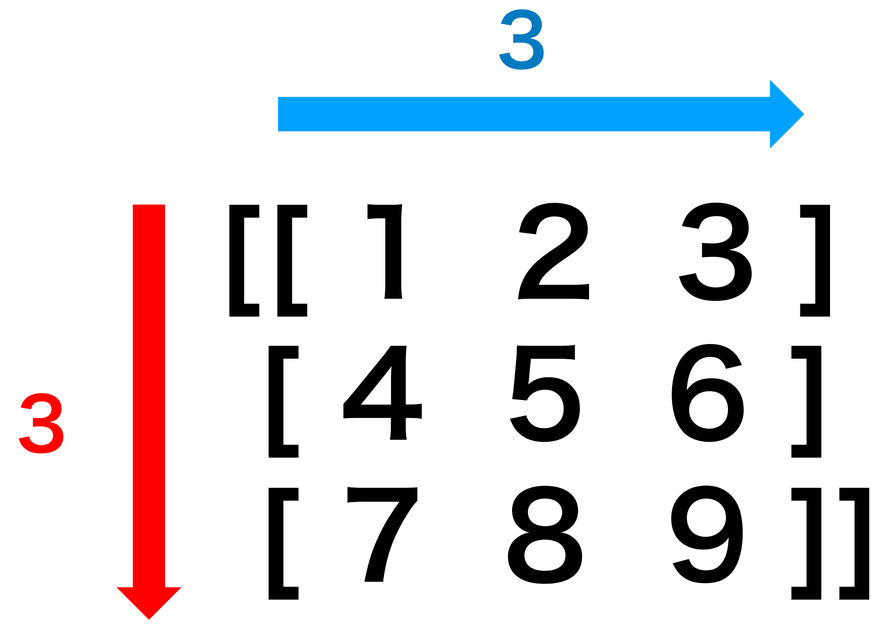
```
[[ 1  2  3 ]
 [ 4  5  6 ]
 [ 7  8  9 ]]
```

[ ]の中に[ ]があるので2次元配列

```
x2 = np.array([[1,2,3],[4,5,6],[7,8,9])
print(x2.shape)          →    (3,3)
```

3 × 3 の2次元配列

```
print(x2)
```



[ ]の中に[ ]があるので2次元配列

(2,3,2)は？

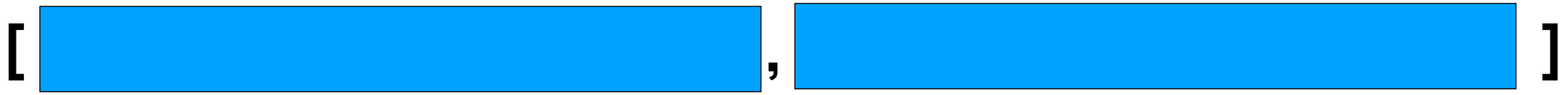
(2,3,2)は？

()の１つ目が2なので一番外側の[ ]の要素は2つ

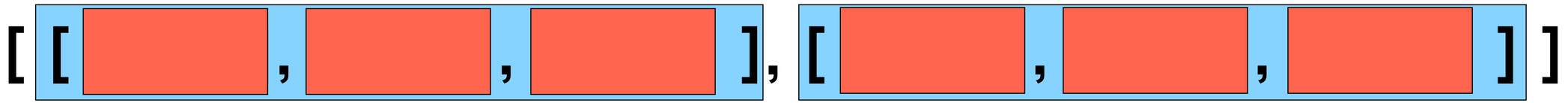
[ , ]

(2,3,2)は？

()の１つ目が2なので一番外側の[ ]の要素は2つ

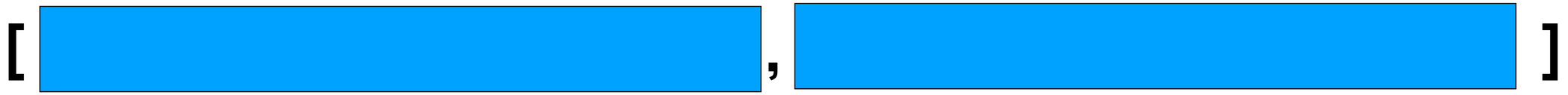


()の２つ目が3なので2つ目の[ ]の要素は３つ

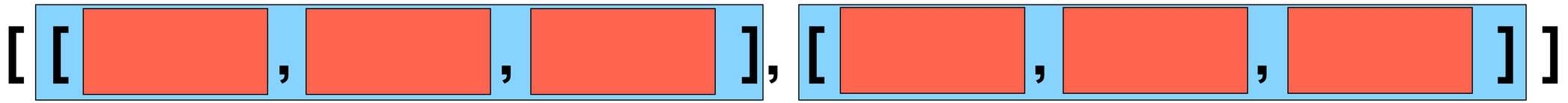


(2,3,2)は？

()の１つ目が2なので一番外側の[ ]の要素は2つ



()の２つ目が3なので2つ目の[ ]の要素は３つ



()の3つ目が2なので3つ目の[ ]の要素は2つ

(2,3,2)は？

```
x3 = np.arange(12)
print(x3.shape)
x3
```

```
(12,)
array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11])
```

```
x3 = x3.reshape(2,3,2)
print(x3.shape)
x3
```

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

       [[ 6,  7],
        [ 8,  9],
        [10, 11]]])
```

np.array(配列)
numpy配列を作成

np.arange(num)
0からnum未満の1次元配列を作成(連番)

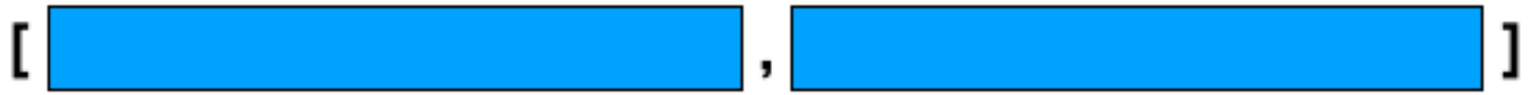
np配列.reshape(指定する形状)
np配列を指定する形状に変換

```
print(x3.shape)
x3
```

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

       [[ 6,  7],
        [ 8,  9],
        [10, 11]]])
```

()の１つ目が2なので一番外側の[ ]の要素は2つ

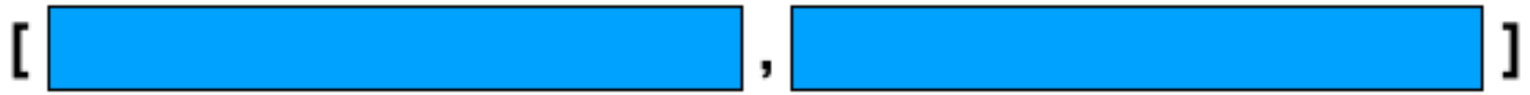


```
print(x3.shape)
x3
```

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

       [[ 6,  7],
        [ 8,  9],
        [10, 11]]])
```

()の１つ目が2なので一番外側の[ ]の要素は2つ



```
x3[0]
```

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

最初の[ ]の要素の1つ目

```
x3[1]
```

```
array([[ 6,  7],
       [ 8,  9],
       [10, 11]])
```

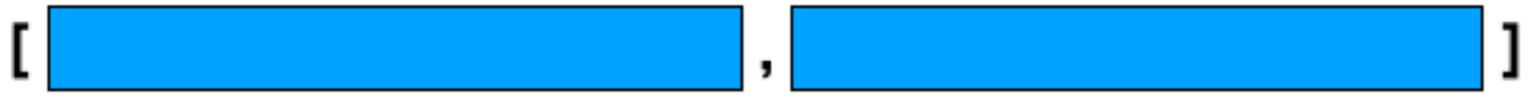
最初の[ ]の要素の２つ目

```
print(x3.shape)
x3
```

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

       [[ 6,  7],
        [ 8,  9],
        [10, 11]]])
```

()の１つ目が2なので一番外側の[ ]の要素は2つ



```
x3[0]
```

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

最初の[ ]の要素の1つ目

3×2の配列の1つ目

```
x3[1]
```

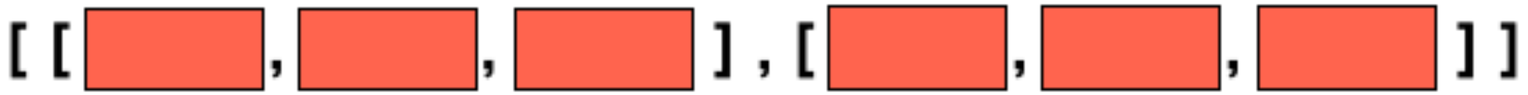
```
array([[ 6,  7],
       [ 8,  9],
       [10, 11]])
```

最初の[ ]の要素の２つ目

3×2の配列の２つ目

```
print(x3.shape)
x3
```

()の２つ目が3なので2つ目の[ ]の要素は３つ



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

       [[ 6,  7],
        [ 8,  9],
        [10, 11]]])
```

```
x3[0][0]
```

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

最初の[ ]の要素の1つ目
2つ目の[ ]の要素の1つ目

```
x3[1][2]
```

```
array([10, 11])
```

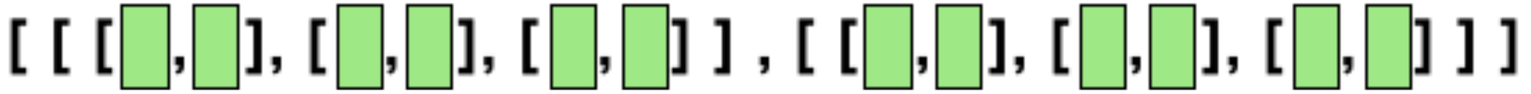
最初の[ ]の要素の２つ目
2つ目の[ ]の要素の３つ目

```
print(x3.shape)
x3
```

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

       [[ 6,  7],
        [ 8,  9],
        [10, 11]]])
```

()の3つ目が2なので3つ目の[ ]の要素は2つ



```
x3[0][0][0]
```

```
0
```

最初の[ ]の要素の1つ目
2つ目の[ ]の要素の1つ目
3つ目の[ ]の要素の1つ目

```
x3[1][2][1]
```

```
11
```

最初の[ ]の要素の２つ目
2つ目の[ ]の要素の３つ目
3つ目の[ ]の要素の2つ目

print(x_train.shape)
60000, 28, 28

print(x_train[0])は？

１つ目を取り出してみる

print(x_train[0])

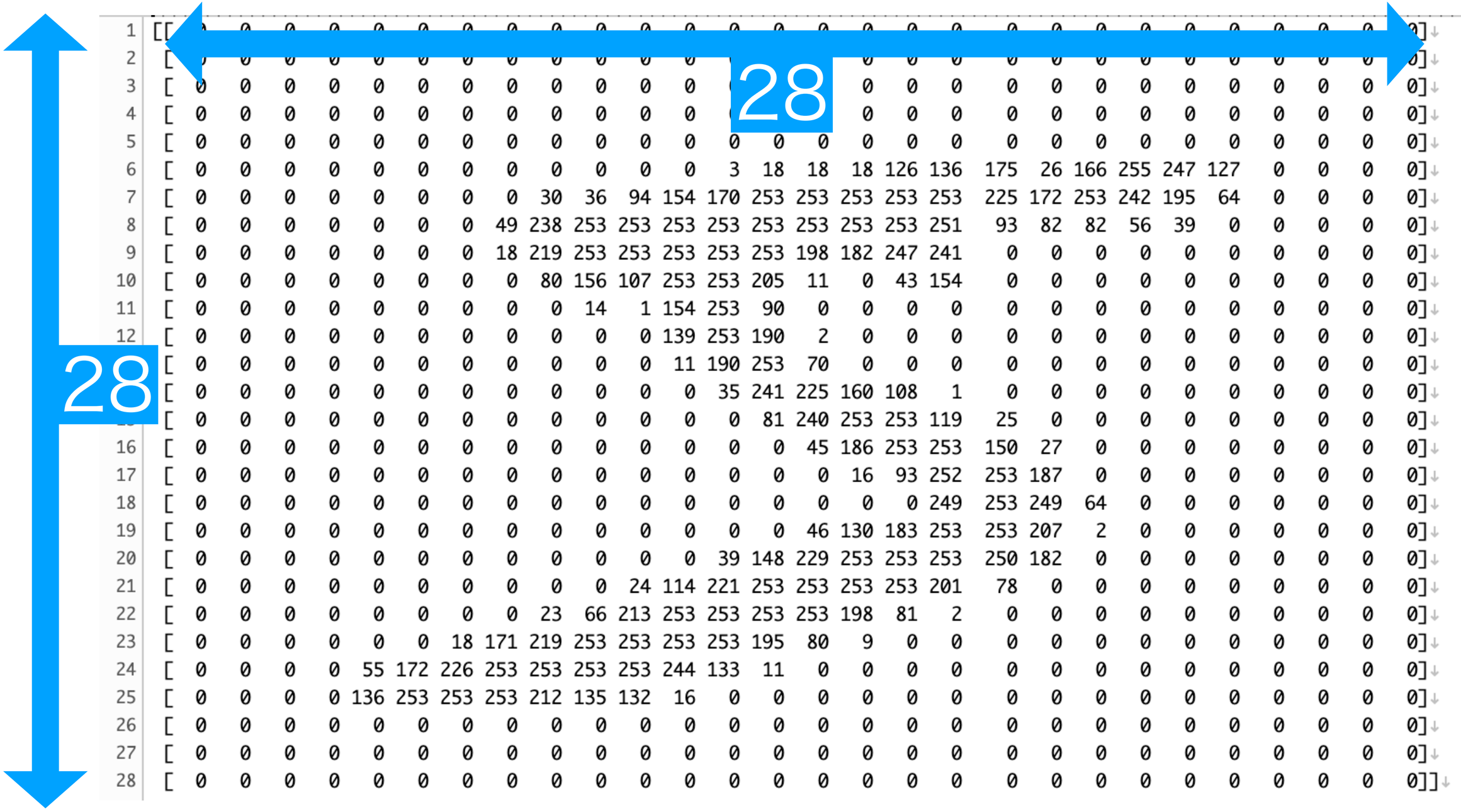
```
print(x_train[0])
```

```
[[  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   3  18  18  18 126 136
  175  26 166 255 247 127   0   0   0   0]
 [  0   0   0   0   0   0   0   0  30  36  94 154 170 253 253 253 253 253
  225 172 253 242 195  64   0   0   0   0]
 [  0   0   0   0   0   0   0  49 238 253 253 253 253 253 253 253 253 251
   93  82  82  56  39   0   0   0   0   0]
 [  0   0   0   0   0   0   0  18 219 253 253 253 253 253 198 182 247 241
    0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0  80 156 107 253 253 205  11   0  43 154
    0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0  14   1 154 253  90   0   0   0   0
    0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0 139 253 190   2   0   0   0
    0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0  11 190 253  70   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  35 241 225 160 108   1
    0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0  81 240 253 253 119
   25   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0  45 186 253 253
  150  27   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0  16  93 252
  253 187   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0 249
  253 249  64   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0  46 130 183 253
  253 207   2   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0  39 148 229 253 253 253
  250 182   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0  24 114 221 253 253 253 253 201
   78   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0  23  66 213 253 253 253 253 198  81   2
    0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0  18 171 219 253 253 253 253 195  80   9   0   0
    0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0  55 172 226 253 253 253 253 244 133  11   0   0   0   0
    0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0 136 253 253 253 212 135 132  16   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]]
```

テキストエディタで開いたところ

```
[[  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   3  18  18  18 126 136 175  26 166 255 247 127   0   0   0   0]
 [  0   0   0   0   0   0   0   0  30  36  94 154 170 253 253 253 253 253 225 172 253 242 195  64   0   0   0   0]
 [  0   0   0   0   0   0   0  49 238 253 253 253 253 253 253 253 253 251  93  82  82  56  39   0   0   0   0   0]
 [  0   0   0   0   0   0   0  18 219 253 253 253 253 253 198 182 247 241   0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0  80 156 107 253 253 205  11   0  43 154   0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0  14   1 154 253  90   0   0   0   0   0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0 139 253 190   2   0   0   0   0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0  11 190 253  70   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  35 241 225 160 108   1   0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0  81 240 253 253 119  25   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0  45 186 253 253 150  27   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0  16  93 252 253 187   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0 249 253 249  64   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0  46 130 183 253 253 207   2   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0  39 148 229 253 253 253 250 182   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0  24 114 221 253 253 253 253 201  78   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0  23  66 213 253 253 253 253 198  81   2   0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0  18 171 219 253 253 253 253 195  80   9   0   0   0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0  55 172 226 253 253 253 253 244 133  11   0   0   0   0   0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0 136 253 253 253 212 135 132  16   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]]
```

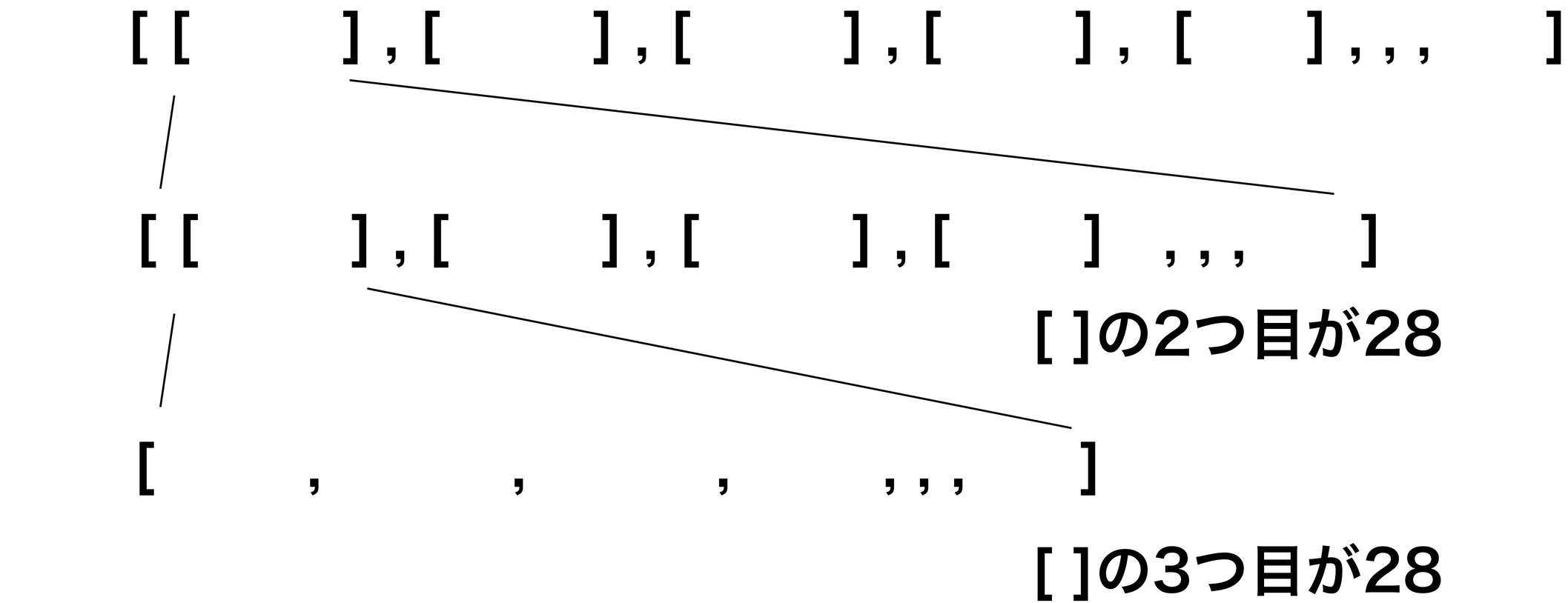
テキストエディタで開いたところ




```
[[  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   3  18  18  18 126 136 175  26 166 255 247 127   0   0   0   0]
 [  0   0   0   0   0   0   0   0  30  36  94 154 170 253 253 253 253 253 225 172 253 242 195  64   0   0   0   0]
 [  0   0   0   0   0   0   0  49 238 253 253 253 253 253 253 253 253 251  93  82  82  56  39   0   0   0   0   0]
 [  0   0   0   0   0   0   0  18 219 253 253 253 253 253 198 182 247 241   0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0  80 156 107 253 253 205  11   0  43 154   0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0  14   1 154 253  90   0   0   0   0   0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0 139 253 190   2   0   0   0   0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0  11 190 253  70   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  35 241 225 160 108   1   0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0  81 240 253 253 119  25   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0  45 186 253 253 150  27   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0  16  93 252 253 187   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0 249 253 249  64   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0  46 130 183 253 253 207   2   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0  39 148 229 253 253 253 250 182   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0  24 114 221 253 253 253 253 201  78   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0  23  66 213 253 253 253 253 198  81   2   0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0  18 171 219 253 253 253 253 195  80   9   0   0   0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0  55 172 226 253 253 253 253 244 133  11   0   0   0   0   0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0 136 253 253 253 212 135 132  16   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]]
```

print(x_train.shape)
60000, 28, 28

[ ]の1つ目が60000



```
x3.shape
```

```
(2, 3, 2)
```

```
x_train.shape
```

```
(60000, 28, 28)
```

```
x3.shape
```

```
(2, 3, 2)
```

```
x3[0]
```

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

1つ目の
3×2の配列

```
x3[1]
```

```
array([[ 6,  7],
       [ 8,  9],
       [10, 11]])
```

2つ目の
3×2の配列

```
x_train.shape
```

```
(60000, 28, 28)
```

```
x3.shape
```

```
(2, 3, 2)
```

```
x3[0]
```

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

1つ目の
3×2の配列

```
x3[1]
```

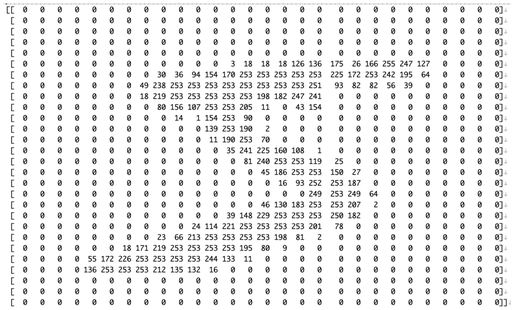
```
array([[ 6,  7],
       [ 8,  9],
       [10, 11]])
```

2つ目の
3×2の配列

```
x_train.shape
```

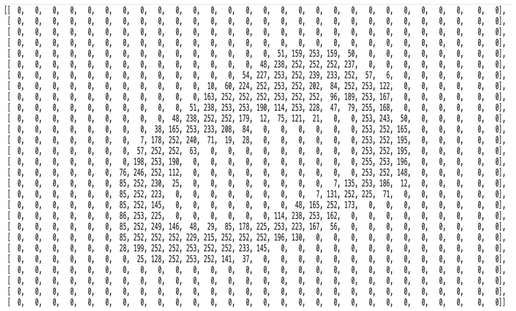
```
(60000, 28, 28)
```

```
x_train[0]
```



1つ目の
28×28の配列

```
x_train[1]
```



2つ目の
28×28の配列

:

60000個目の
28×28の配列

```
print(x_train[0][7])
```

```
[  0   0   0   0   0   0   0  49 238 253 253 253 253 253 253 253 253 251
  93  82  82  56  39   0   0   0   0   0]
```

```
1  [[  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]
2   [  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]
3   [  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]
4   [  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]
5   [  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]
6   [  0   0   0   0   0   0   0   0   0   0   0   0   3  18  18  18 126 136 175  26 166 255 247 127   0   0   0   0]
7   [  0   0   0   0   0   0   0   0  30  36  94 154 170 253 253 253 253 253 225 172 253 242 195  64   0   0   0   0]
8   [  0   0   0   0   0   0   0  49 238 253 253 253 253 253 253 253 253 251  93  82  82  56  39   0   0   0   0   0]
9   [  0   0   0   0   0   0   0  18 219 253 253 253 253 253 198 182 247 241   0   0   0   0   0   0   0   0   0   0]
10  [  0   0   0   0   0   0   0   0  80 156 107 253 253 205  11   0  43 154   0   0   0   0   0   0   0   0   0   0]
11  [  0   0   0   0   0   0   0   0   0  14   1 154 253  90   0   0   0   0   0   0   0   0   0   0   0   0   0   0]
12  [  0   0   0   0   0   0   0   0   0   0   0 139 253 190   2   0   0   0   0   0   0   0   0   0   0   0   0   0]
13  [  0   0   0
14  [  0   0   0
15  [  0   0   0                                                                                                  0   0   0   0   0]
16  [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0  45 186 253 253 150  27   0   0   0   0   0   0   0]
17  [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0  16  93 252 253 187   0   0   0   0   0   0   0]
18  [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0 249 253 249  64   0   0   0   0   0   0]
19  [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0  46 130 183 253 253 207   2   0   0   0   0   0]
20  [  0   0   0   0   0   0   0   0   0   0   0   0   0   0  39 148 229 253 253 253 250 182   0   0   0   0   0   0]
21  [  0   0   0   0   0   0   0   0   0   0   0   0  24 114 221 253 253 253 253 201  78   0   0   0   0   0   0   0]
22  [  0   0   0   0   0   0   0   0   0   0  23  66 213 253 253 253 253 198  81   2   0   0   0   0   0   0   0   0]
23  [  0   0   0   0   0   0   0  18 171 219 253 253 253 253 195  80   9   0   0   0   0   0   0   0   0   0   0   0]
24  [  0   0   0   0  55 172 226 253 253 253 253 244 133  11   0   0   0   0   0   0   0   0   0   0   0   0   0   0]
25  [  0   0   0   0 136 253 253 253 212 135 132  16   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0]
26  [  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]
27  [  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]
28  [  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始まりなので[0][7]は1枚目の8行目

```python
print(x_train[0][7][7])
```

```
49
```

```
[[  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   3  18  18  18 126 136 175  26 166 255 247 127   0   0   0   0]
 [  0   0   0   0   0   0   0   0  30  36  94 154 170 253 253 253 253 253 225 172 253 242 195  64   0   0   0   0]
 [  0   0   0   0   0   0   0  49 238 253 253 253 253 253 253 253 253 251  93  82  82  56  39   0   0   0   0   0]
 [  0   0   0   0   0   0   0  18 219 253 253 253 253 253 198 182 247 241   0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0  80 156 107 253 253 205  11   0  43 154   0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0  14   1 154 253  90   0   0   0   0   0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0 139 253 190   2   0   0   0   0   0   0   0   0   0   0   0   0   0]
 [ [illegible]                                                                                                0   0]
 [ [illegible]                                                                                                0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0  81 240 253 253 119  25   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0  45 186 253 253 150  27   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0  16  93 252 253 187   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0 249 253 249  64   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0  46 130 183 253 253 207   2   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0  39 148 229 253 253 253 250 182   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0  24 114 221 253 253 253 253 201  78   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0  23  66 213 253 253 253 253 198  81   2   0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0  18 171 219 253 253 253 253 195  80   9   0   0   0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0  55 172 226 253 253 253 253 244 133  11   0   0   0   0   0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0 136 253 253 253 212 135 132  16   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始まりなので[0][7][7]は1枚目の8行目の8列目

```
[6] print(y_train)

[5 0 4 ... 5 6 8]
```

```
print(y_train[0])

5
```

y_trainには60000枚の数字(=正解)
y_train[0]が1枚目の数字(=5)

```
                                     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   3  18  18  18 126 136 175  26 166 255 247 127   0   0   0   0]
 [  0   0   0   0   0   0   0   0  30  36  94 154 170 253 253 253 253 253 225 172 253 242 195  64   0   0   0   0]
 [  0   0   0   0   0   0   0  49 238 253 253 253 253 253 253 253 253 251  93  82  82  56  39   0   0   0   0   0]
 [  0   0   0   0   0   0   0  18 219 253 253 253 253 253 198 182 247 241   0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0  80 156 107 253 253 205  11   0  43 154   0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0  14   1 154 253  90   0   0   0   0   0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0 139 253 190   2   0   0   0   0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0  11 190 253  70   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  35 241 225 160 108   1   0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0  81 240 253 253 119  25   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0  45 186 253 253 150  27   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0  16  93 252 253 187   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0 249 253 249  64   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0  46 130 183 253 253 207   2   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0  39 148 229 253 253 253 250 182   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0  24 114 221 253 253 253 253 201  78   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0  23  66 213 253 253 253 253 198  81   2   0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0  18 171 219 253 253 253 253 195  80   9   0   0   0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0  55 172 226 253 253 253 253 244 133  11   0   0   0   0   0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0 136 253 253 253 212 135 132  16   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]]
```

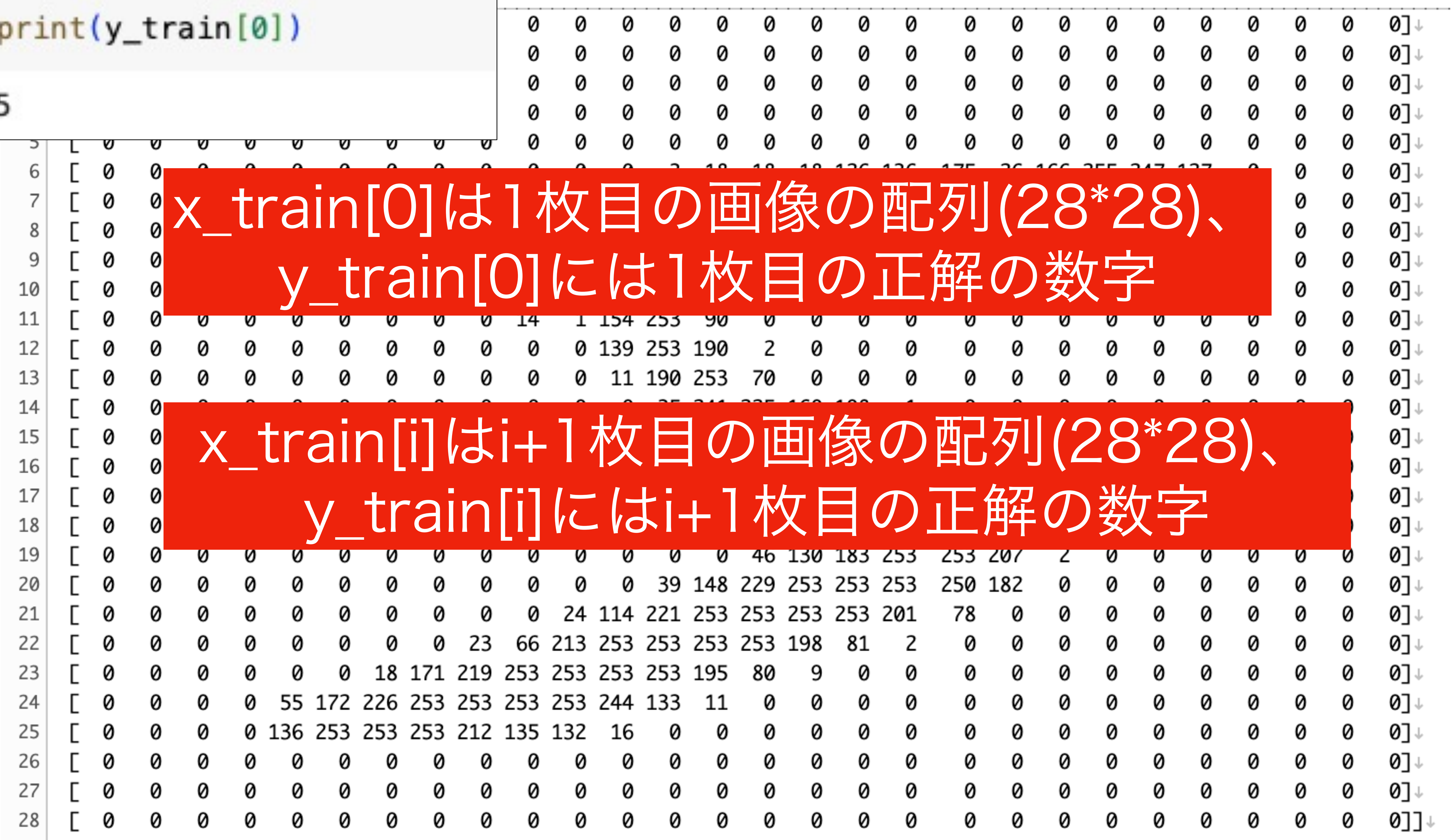
```
[6] print(y_train)

[5 0 4 ... 5 6 8]

print(y_train[0])

5
```

y_trainには60000枚の数字(=正解)
y_train[0]が1枚目の数字(=5)

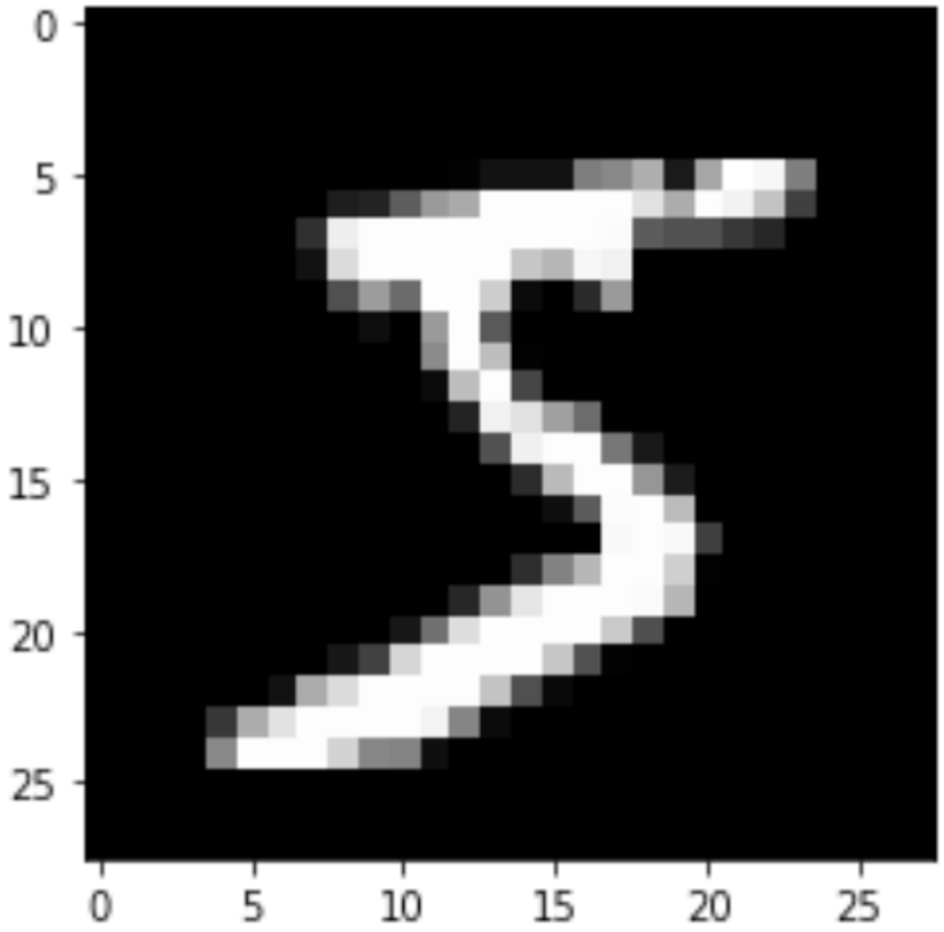


x_train[0]は1枚目の画像の配列(28*28)、
y_train[0]には1枚目の正解の数字

x_train[i]はi+1枚目の画像の配列(28*28)、
y_train[i]にはi+1枚目の正解の数字

画像を描画する

```
import matplotlib.pyplot as plt
plt.imshow(x_train[0], 'gray')
plt.show()
```



matplotlib(描画ライブラリ)

plt.imshow(画像もしくは配列, ‘color_mode’)

‘gray’で白黒を指定

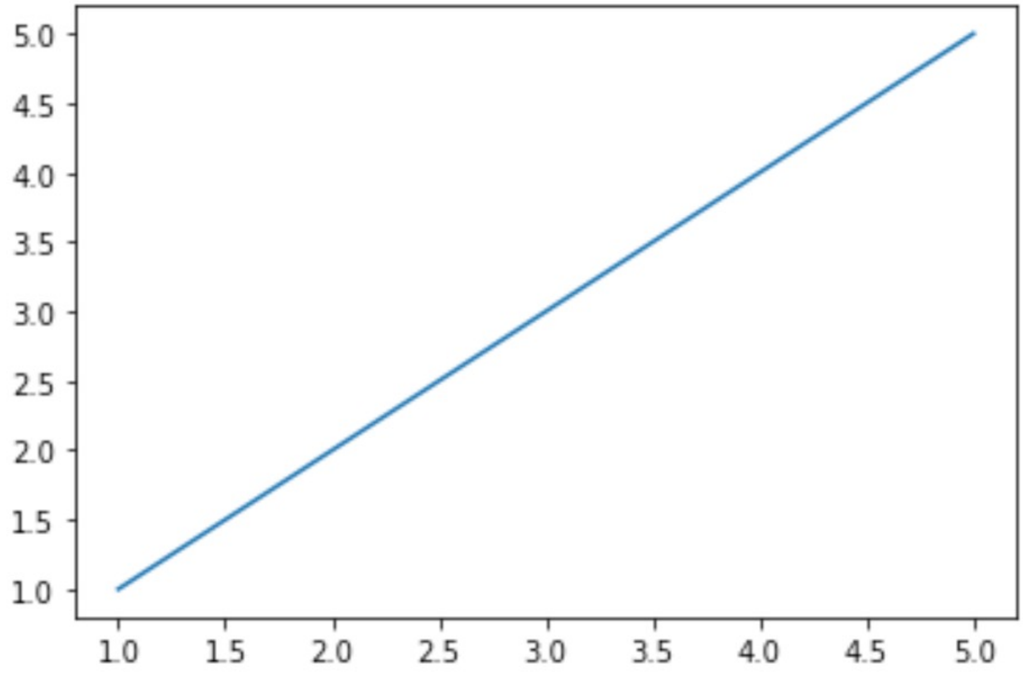
plt.show()で表示

１つ目を取り出してみる print(x_train[0])

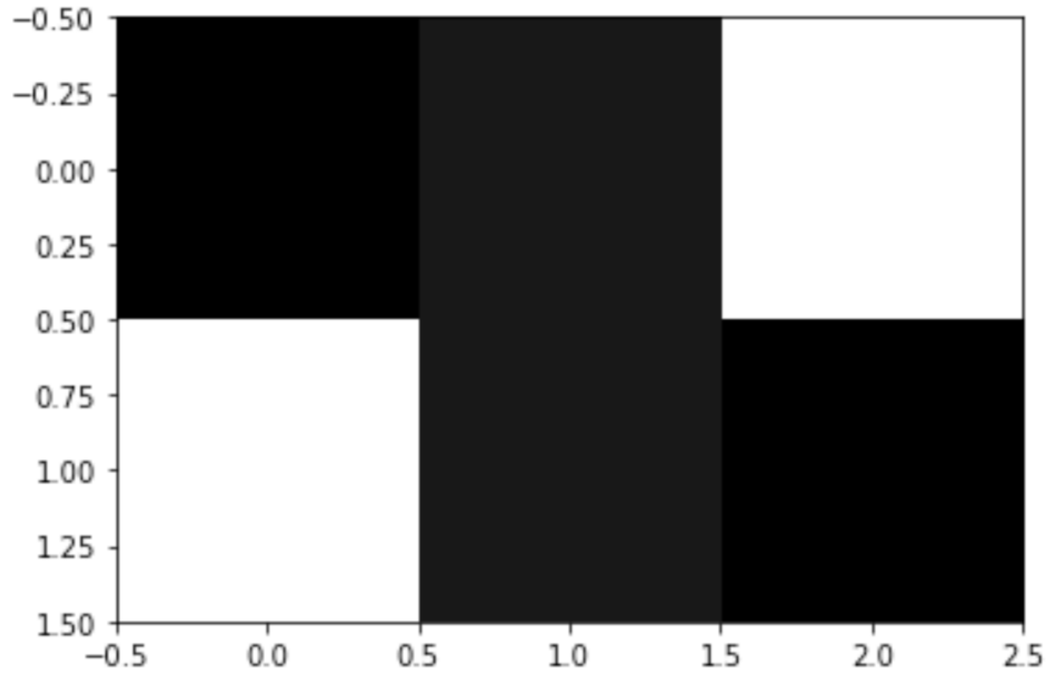
```
[[  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   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   3  18  18  18 126 136 175  26 166 255 247 127   0   0   0   0]
 [  0   0   0   0   0   0   0   0  30  36  94 154 170 253 253 253 253 253 225 172 253 242 195  64   0   0   0   0]
 [  0   0   0   0   0   0   0  49 238 253 253 253 253 253 253 253 253 251  93  82  82  56  39   0   0   0   0   0]
 [  0   0   0   0   0   0   0  18 219 253 253 253 253 253 198 182 247 241   0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0  80 156 107 253 253 205  11   0  43 154   0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0  14   1 154 253  90   0   0   0   0   0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0 139 253 190   2   0   0   0   0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0  11 190 253  70   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  35 241 225 160 108   1   0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0  81 240 253 253 119  25   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0  45 186 253 253 150  27   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0  16  93 252 253 187   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0 249 253 249  64   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0  46 130 183 253 253 207   2   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0  39 148 229 253 253 253 250 182   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0  24 114 221 253 253 253 253 201  78   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0  23  66 213 253 253 253 253 198  81   2   0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0  18 171 219 253 253 253 253 195  80   9   0   0   0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0  55 172 226 253 253 253 253 244 133  11   0   0   0   0   0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0 136 253 253 253 212 135 132  16   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]]
```

plt.plot(x,y)でxとyの値を直線で結ぶ
plt.imshow(x)でxの画像データもしくは配列を描画する
白黒(gray)を指示した場合、数字が大きいほど白い(0~255)

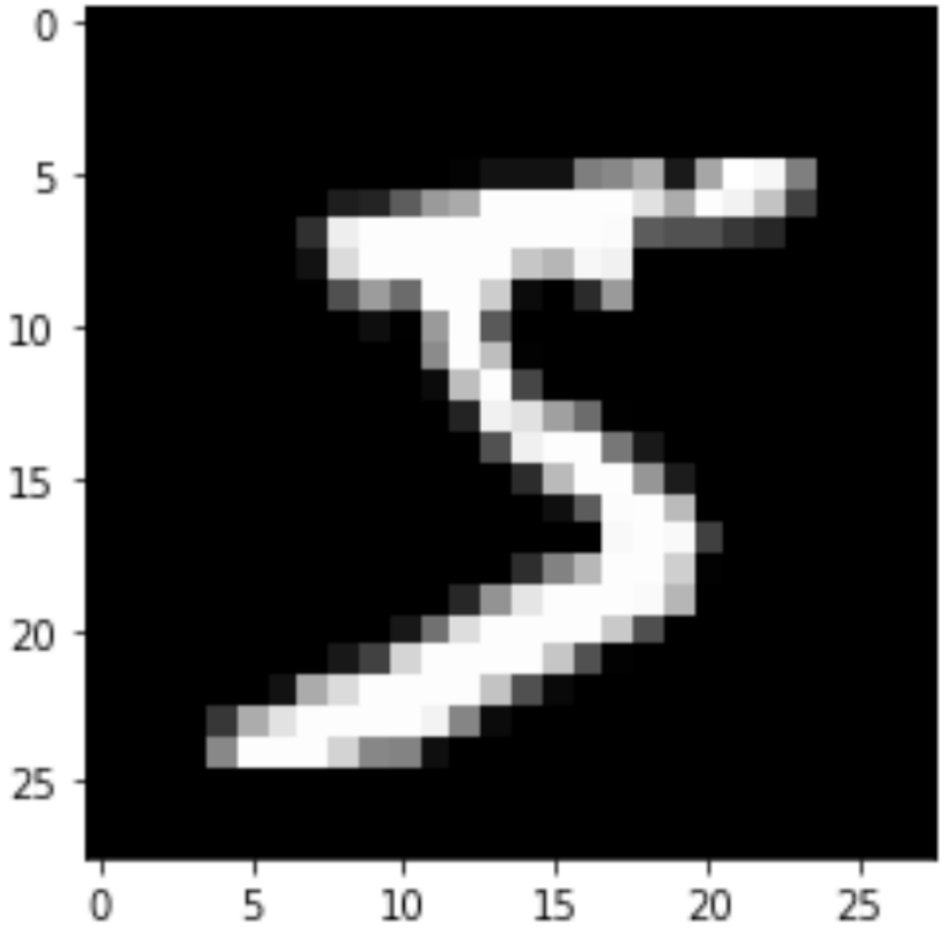
```
x = [1,2,3,4,5]
y = [1,2,3,4,5]
plt.plot(x,y)
plt.show()
```




```
import numpy as np
x = np.array([[1,10,100],[100,10,1]])
plt.imshow(x,'gray')
plt.show()
```



```
import matplotlib.pyplot as plt
plt.imshow(x_train[0], 'gray')
plt.show()
```



画像を描画する

matplotlib(描画ライブラリ)

‘gray’で白黒を指定

```
print(y_train[0])
```

```
5
```

数字の５らしい

課題

- WebClassにある”kadai2.ipynb”をやってみましょう
- 実行したら”学籍番号_名前_2.ipynb”という名前で保存して提出して下さい。

締め切りは2週間後の8/3の23:59です。
締め切りを過ぎた課題は受け取らないので注意して下さい

ipynbのファイルの開き方は次ページ参照

医療とAI・ビッグデータ応用
③MNISTの読み込みと前処理

統合教育機構
須藤毅顕

この授業では何をしていたか？

深層学習で画像分類をしたい

深層学習(教師あり機械学習)の復習

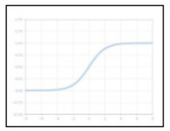
データを用意する

x(特徴量データ)　y(正解データ)

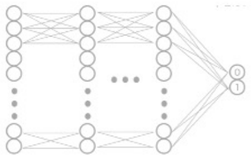
x　y

学習させる

ロジスティック回帰分析



ニューラルネットワーク



評価する
(分類、予測など)

病気か否か

犬か猫か

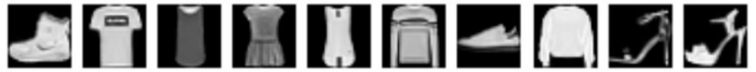
Pythonには機械学習を実践するために多くの画像セットが用意されている

MNIST：0~9の文字画像のデータ

0　1　2　3　4　5　6　7　8　9

FASHION-MNIST：白黒の洋服の画像データ

0 : T-shirt/top、1 : Trouser、2 : Pullover、3 : Dress、4 : Coat、5 : Sandal
6 : Shirt、7 : Sneaker、8 : Bag、9 : Ankle boot

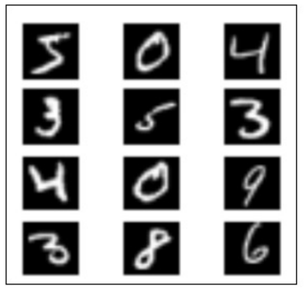


CIFAR10

0 : airplane、1 : automobile、2 : bird、3 : cat、4 : deer、5 : dog
6 : frog、7 : horse、8 : ship、9 : truck

画像を読み込んで学習出来るデータ(配列)にする

画像データ



配列データ

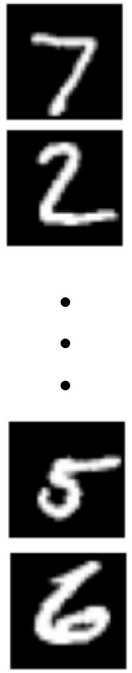
```
x_train                    y_train
[[150, 70, 60],            [[1],
[120, 40, 35],              [0],
[144, 45, 40],              [1],
[162, 56, 50],              [1],
[98, 40, 32],               [0],
[128, 59, 35],              [0],
[155, 77, 45]]              [1]]
```

画像を配列の数値データにして学習させたい

```
from keras.datasets import mnist
(x_train, y_train),(x_test, y_test) = mnist.load_data()
```

60000個　60000個　10000個　10000個　mnistのdataを読み込む

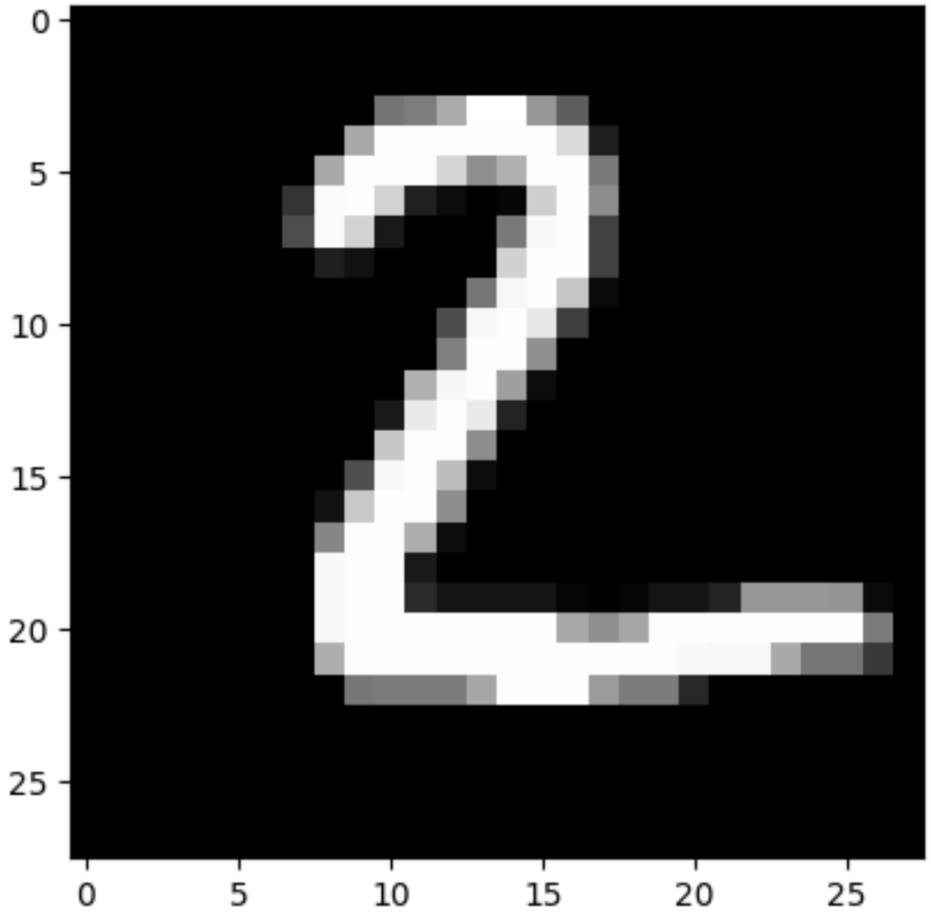
5
0
4
1
⋮
6
8



7
2
⋮
5
6

x_train : 60000枚の画像の配列データ
y_train : 60000枚の正解の数字の配列データ
x_test : 10000枚の画像の配列データ
y_test : 10000枚の正解の数字の配列データ

```
import matplotlib.pyplot as plt
plt.imshow(x_test[1],'gray')
plt.show()
```



画像を描画する

matplotlib(描画ライブラリ)

‘gray’で白黒を指定

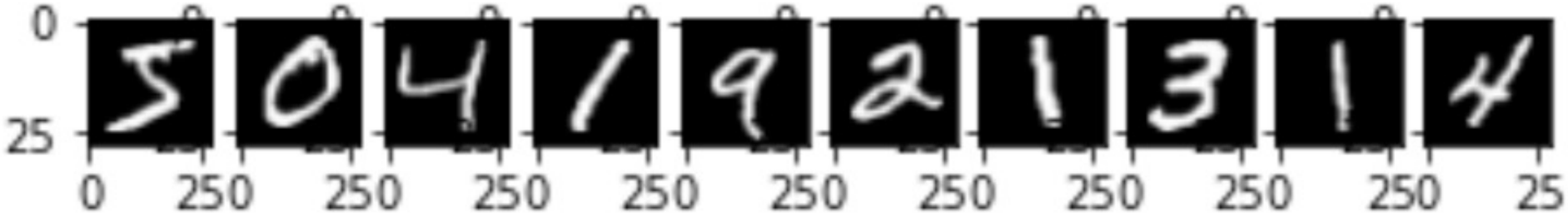
```
print(y_test[1])
```

```
2
```

数字の2らしい

10個並べてみる

```
for i in range(10):
    plt.subplot(1,10,i+1)
    plt.imshow(x_train[i], 'gray')
plt.show()
```



for文とrange関数

```
for i in range(1,10,2):
  print(i)
```

```
1
3
5
7
9
```

```
for i in range(5):
  print(i)
```

```
0
1
2
3
4
```

=

```
for i in range(0,5,1):
  print(i)
```

```
0
1
2
3
4
```

for (変数) in range(A,B,C):
　　(処理内容)

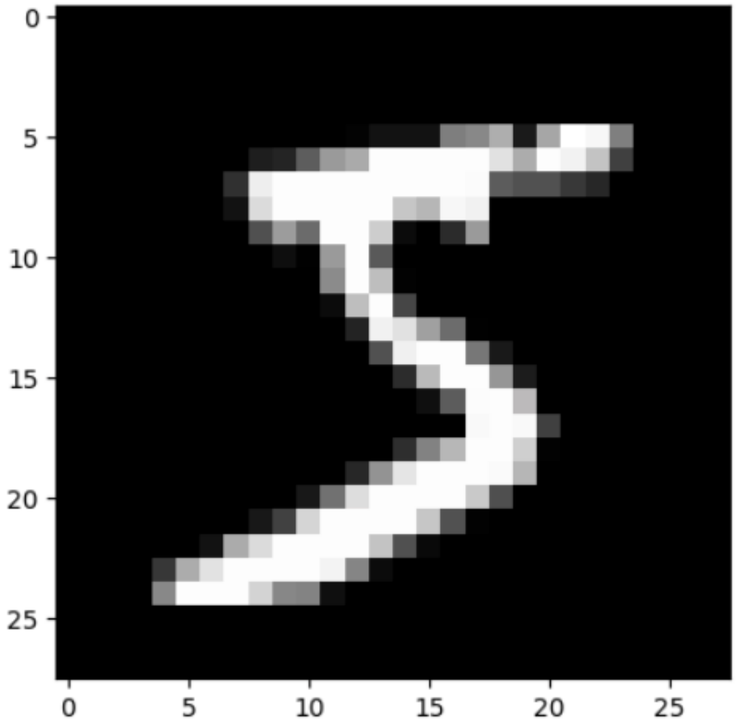
A(以上)からB(未満)でC刻みに変数に代入

この例だと1から1つ飛ばしで9まで
iは順に1,3,5,7,9が代入されて処理

range(B)のように数字1つにすると
開始のAを0、刻み幅Cを1の設定で省略出来る

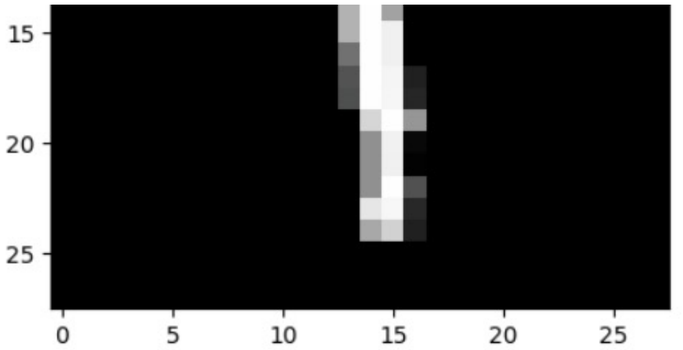
for文

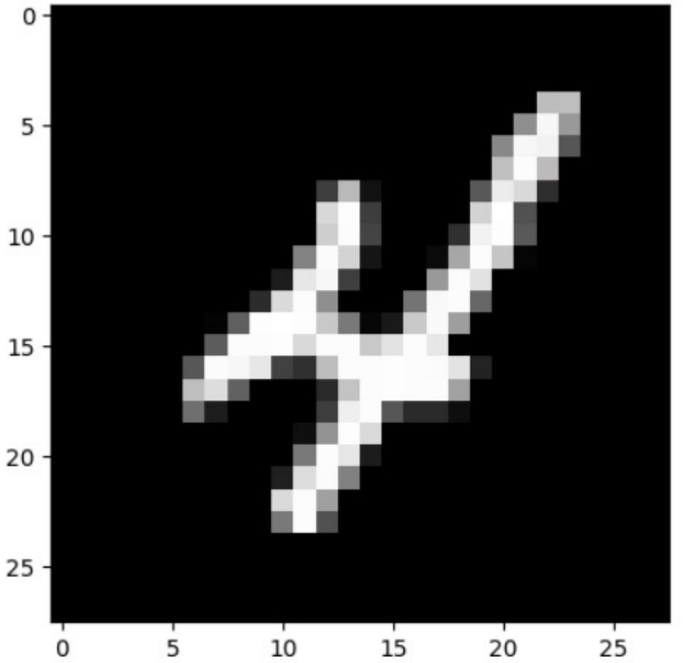
```
for i in range(10):
  plt.imshow(x_train[i],'gray')
  plt.show()
```



:

:

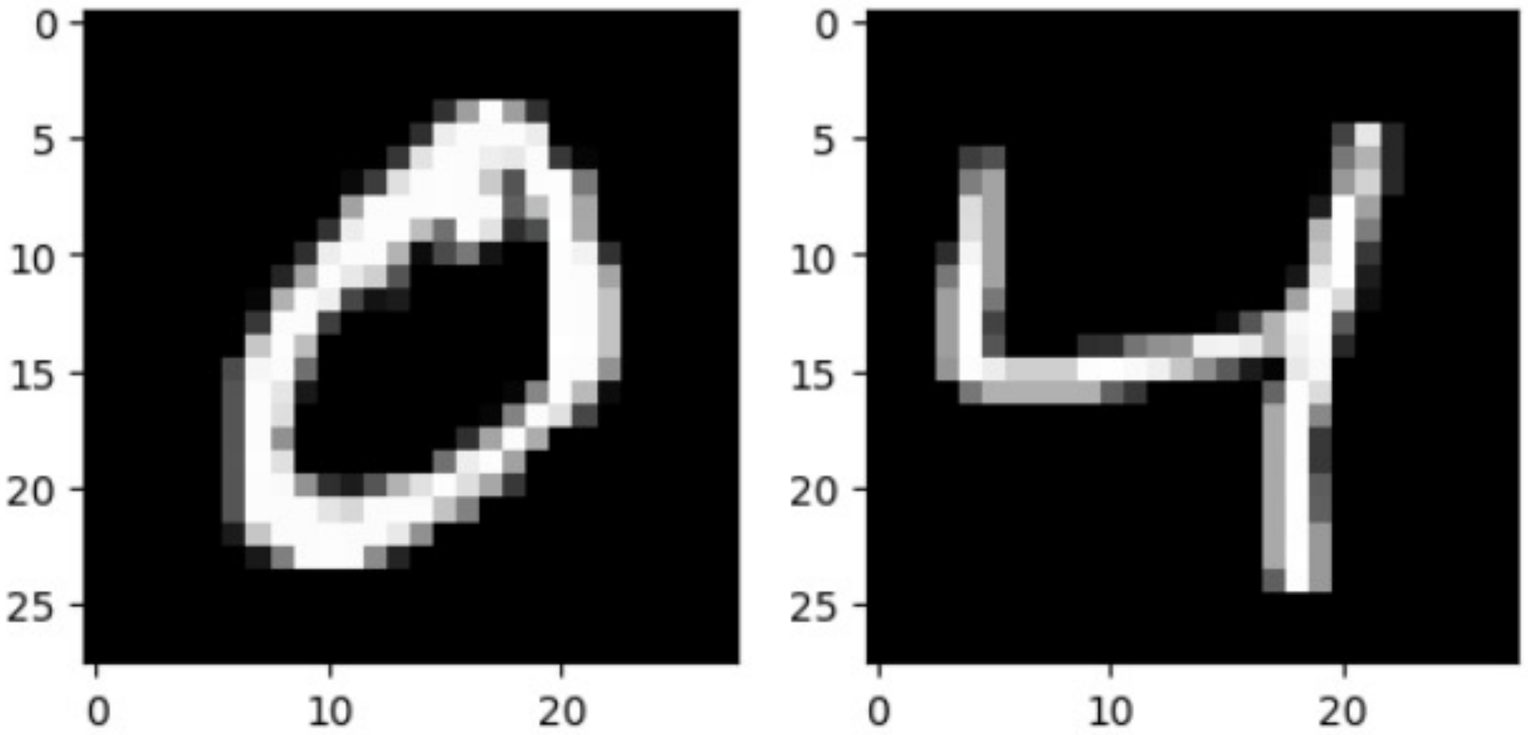




plt.subplotは図を並べる plt.subplot(縦,横,左上から何番目か)

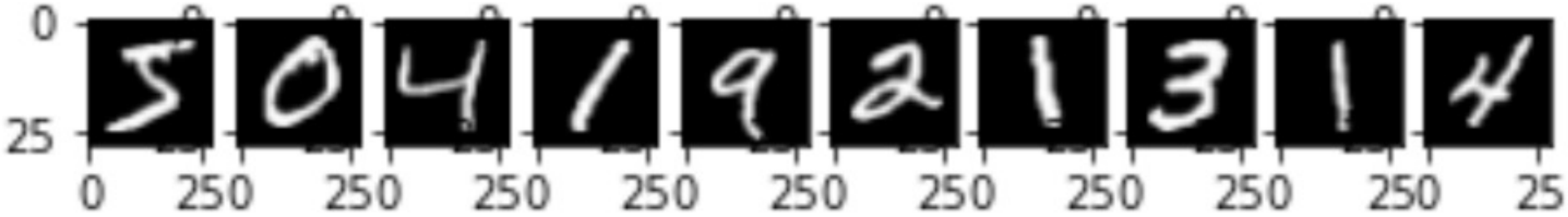
```
[8]  plt.subplot(1,2,1)          ←縦1、横2に並べる内の1つ目の宣言
     plt.imshow(x_train[1],'gray')
     plt.subplot(1,2,2)          ←縦1、横2に並べる内の2つ目の宣言
     plt.imshow(x_train[2],'gray')

     plt.show()
```



10個並べてみる

```
for i in range(10):
    plt.subplot(1,10,i+1)
    plt.imshow(x_train[i], 'gray')
plt.show()
```



for文

```
for i in range(10):
    plt.subplot(1,10,i+1)
    plt.imshow(x_train[i], 'gray')
plt.show()
```

plt.subplot(1,10,i+1)
plt.imshow(x_train[i], ‘gray’)

縦に１つ、横に10個、図を書く。
i=0のなので(1,10,1)で１番左の図を指定する

x_train[0]

for文

```
for i in range(10):
    plt.subplot(1,10,i+1)
    plt.imshow(x_train[i], 'gray')
plt.show()
```

plt.subplot(1,10,i+1)
plt.imshow(x_train[i], ‘gray’)

次がi=1
(1,10,2)で左から2つ目の図を指定する
plt.imshow(x_train[1], ‘gray’)

x_train[1]

for文

```
for i in range(10):
    plt.subplot(1,10,i+1)
    plt.imshow(x_train[i], 'gray')
plt.show()
```

plt.subplot(1,10,i+1)
plt.imshow(x_train[i], ‘gray’)

最後はi=9
(1,10,10)で左から10個目の図を指定する
plt.imshow(x_train[1], ‘gray’)

x_train[9]

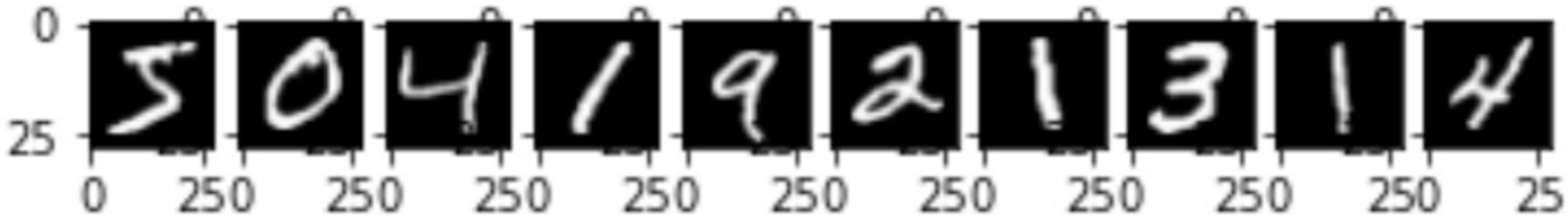
for文

```
for i in range(10):
    plt.subplot(1,10,i+1)
    plt.imshow(x_train[i], 'gray')
plt.show()
```

plt.subplot(1,10,i+1)
plt.imshow(x_train[i], ‘gray’)

最後はi=9
(1,10,10)で左から10個目の図を指定する
plt.imshow(x_train[1], ‘gray’)

x_train[9]



plt.subplot(2,5,i+1)にすると縦２、横５の図になる

```
for i in range(10):
    plt.subplot(2,5,i+1)
    plt.imshow(x_train[i], 'gray')
plt.show()
```

正解も10個並べてみる

```
print(y_train[0:10])
```

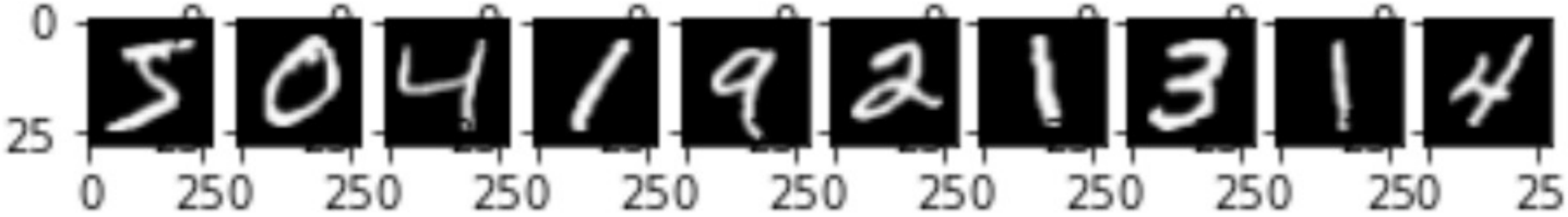
```
[5 0 4 1 9 2 1 3 1 4]
```

配列は[始まりの数字：終わりの数字]で中身(要素)を取り出せる

[0:10]で0から9番目まで！

x_trainとy_trainが特徴量と正解の関係になっている（図でも確認）

```
for i in range(10):
    plt.subplot(1,10,i+1)
    plt.imshow(x_train[i], 'gray')
plt.show()
```




```
print(y_train[0:10])
```

```
[5 0 4 1 9 2 1 3 1 4]
```

深層学習前のデータの整理

x_train（特徴量）

- 画像の２次元の配列を１次元にする
- 正規化する

y_train（正解)

- one-hot encoding

深層学習前のデータの整理

x_train（特徴量）

・画像の２次元の配列を１次元にする

まだ入力しなくていいです

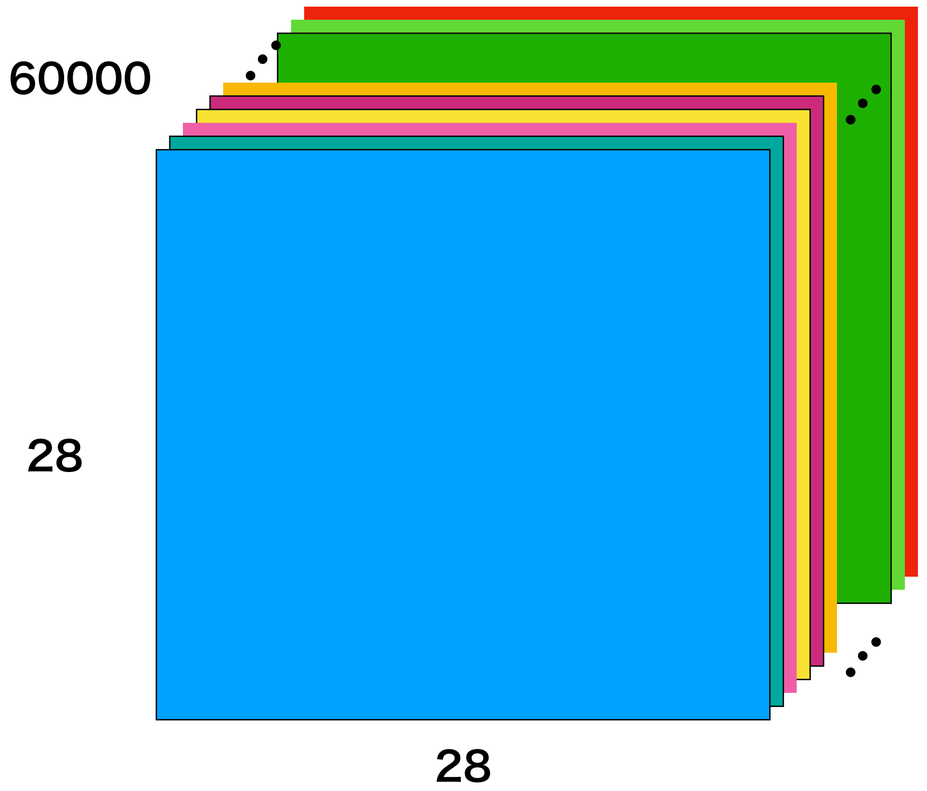
```
x_train = x_train.reshape((x_train.shape[0],784))
x_test = x_test.reshape((x_test.shape[0],784))
print(x_train.shape)
print(x_test.shape)
```

```
(60000, 784)
(10000, 784)
```

x_trainのshapeは？

print(x_train.shape)
(60000, 28, 28)

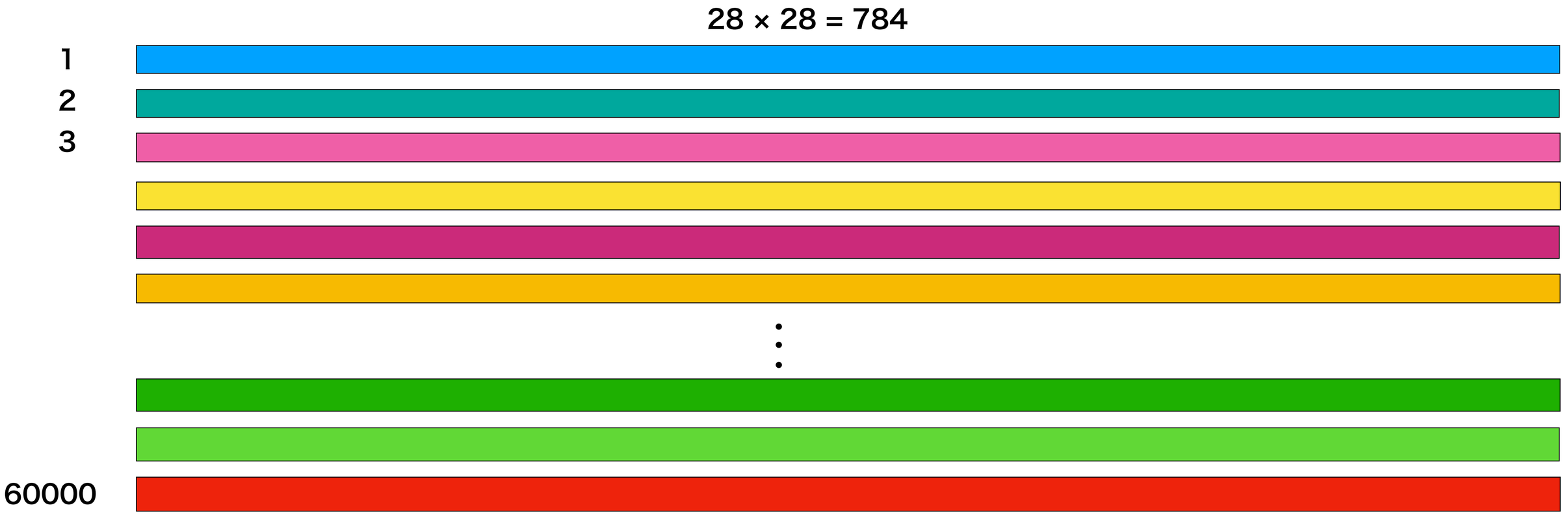
x_train[0]のshapeは？



print(x_train[0].shape)は1枚目の画像の配列なので(28,28)となる

画像の２次元配列を１次元配列にしたい

(60000, 28, 28) → (60000, 28×28)



画像の２次元配列を１次元配列にしたい

reshape()で配列の形状を変える

```
a = np.array([1,2,3,4,5,6,7,8,])
print(a)
```

```
[1 2 3 4 5 6 7 8]
```

aを(2,4)に変える

```
a = a.reshape(2,4)
print(a)
```

```
[[1 2 3 4]
[5 6 7 8]]
```

画像の２次元配列を１次元配列にしたい

reshape()で配列の形状を変える

```
a = np.array([1,2,3,4,5,6,7,8,])
print(a)
[1 2 3 4 5 6 7 8]
```

aを(2,4)に変える

```
a = a.reshape(2,4)
print(a)

    [[1 2 3 4]
    [5 6 7 8]]
```

aを(2,2,2)に変える

```
a = a.reshape(2,2,2)
print(a)

    [[[1 2]
      [3 4]]
     [[5 6]
      [ 7 8]]]
```

要素の合計が合っていれば
どの形にも変えられる

画像の２次元配列を１次元配列にしたい

reshape()で配列の形状を変える

```
a = np.array([1,2,3,4,5,6,7,8,])
print(a)
```

[1 2 3 4 5 6 7 8]

aを(2,4)に変える

```
a = a.reshape(2,4)
print(a)
```

[[1 2 3 4]

[5 6 7 8]]

```
print(x_train.shape)
```

(60000, 28, 28)

```
x_train = x_train.reshape(60000,784)
x_test = x_test.reshape(10000,784)
```

```
x_train = x_train.reshape((x_train.shape[0],784))
x_test = x_test.reshape((x_test.shape[0],784))
print(x_train.shape)
print(x_test.shape)
```

```
(60000, 784)
(10000, 784)
```

x_train.shape[0]は(60000, 28, 28)の1つ目なので60000

深層学習前のデータの整理

x_train（特徴量）

・正規化する

```
x_train = x_train / 255
x_test = x_test / 255
```

配列の数字は0~255のいずれか
全てを255で割って0~1の間に変換する

numpy配列は四則演算が
それぞれの要素に行なわれます

```
a = a.reshape(2,2,2)
print(a)
print(a.shape)
```

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

 [[5 6]
  [7 8]]]
(2, 2, 2)
```

```
a = a / 10
print(a)
```

```
[[[0.1 0.2]
  [0.3 0.4]]

 [[0.5 0.6]
  [0.7 0.8]]]
```

(x_train = x_train / 255 は省略して x_train /= 255 と書くことも出来ます)

深層学習前のデータの整理

y_train（正解)

・one-hot encoding

正解は全て0から９のいずれか

これを全て０と１だけで表現するための方法(理由は後述)

one-hot encoding

0
1
2
3
4
5
6
7
8
9

1, 0, 0, 0, 0, 0, 0, 0, 0 , 0
0, 1, 0, 0, 0, 0, 0, 0, 0 , 0
0, 0, 1, 0, 0, 0, 0, 0, 0 , 0
0, 0, 0, 1, 0, 0, 0, 0, 0 , 0
0, 0, 0, 0, 1, 0, 0, 0, 0 , 0
0, 0, 0, 0, 0, 1, 0, 0, 0 , 0
0, 0, 0, 0, 0, 0, 1, 0, 0 , 0
0, 0, 0, 0, 0, 0, 0, 1, 0 , 0
0, 0, 0, 0, 0, 0, 0, 0, 1 , 0
0, 0, 0, 0, 0, 0, 0, 0, 0 , 1

0と1だけで0から9の数字を表現する

one-hot encoding

to_categorical()関数を使う

```
[17] from keras.utils import to_categorical
     y_train = to_categorical(y_train,10)
     y_test = to_categorical(y_test,10)
```

```
[18] print(y_train.shape)
     print(y_test.shape)

     (60000, 10)
     (10000, 10)
```

```
print(y_train[0:10])

[[0. 0. 0. 0. 0. 1. 0. 0. 0. 0.]
 [1. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 1. 0. 0. 0. 0. 0.]
 [0. 1. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 1.]
 [0. 0. 1. 0. 0. 0. 0. 0. 0. 0.]
 [0. 1. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 1. 0. 0. 0. 0. 0. 0.]
 [0. 1. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 1. 0. 0. 0. 0. 0.]]
```

to_categorical(変えたい配列,正解の数)

```
print(y_train[0:10])

[5 0 4 1 9 2 1 3 1 4]
```

深層学習前のデータの整理

x_train（特徴量）

・画像の２次元の配列を１次元にする
・正規化する

y_train（正解)

・one-hot encoding

次回もこのファイルを使用します

「Driveにコピーを保存」

```
from keras.datasets import mnist
(x_train,y_train),(x_test,y_test) = mnist.load_data()
```

```
import matplotlib.pyplot as plt
plt.imshow(x_test[1],'gray')
plt.show()
```

```
print(y_test[1])
```

```
x_train = x_train.reshape(x_train.shape[0],784)
x_test = x_test.reshape(x_test.shape[0],784)
print(x_train.shape)
print(x_test.shape)
```

```
x_train = x_train / 255
x_test = x_test / 255
```

```
from keras.utils import to_categorical
y_train = to_categorical(y_train,10)
y_test = to_categorical(y_test,10)
```

課題

- WebClassにある”kadai3.ipynb”をやってみましょう
- 実行したら”学籍番号_名前_3.ipynb”という名前で保存して提出して下さい。

締め切りは2週間後の11/9の23:59です。
締め切りを過ぎた課題は受け取らないので注意して下さい