In [1]:
```
import numpy as np
```

DataTypes & Attributes

In [2]:
```
## Numpy's main datatype is ndarray
a1 = np.array([1,2,3])
```

In [3]:
```
type(a1)
```

Out[3]:
```
numpy.ndarray
```

In [4]:
```
a2 = np.array([[1,2,3],
               [4,5,6,]])

a3 = np.array([    [[1,2,3],[2,0,0]],
                   [[0,2,3],[2,0,2]],
                   [[0,2,3],[2,0,2]]       ])
```

In [5]:
```
a3.shape
```

Out[5]:
```
(3, 2, 3)
```

In [6]:
```
a1.shape
```

Out[6]:
```
(3,)
```

In [7]:
```
a2.shape
```

Out[7]:
```
(2, 3)
```

In [8]:
```
a3.shape
```

Out[8]:
```
(3, 2, 3)
```

In [9]:
```
a3.ndim
```

Out[9]:
```
3
```

In [10]:
```
a1.dtype , a2.dtype , a3.dtype
```

Out[10]:
```
(dtype('int32'), dtype('int32'), dtype('int32'))
```

In [11]:
```
a3.size
```

Out[11]:
```
18
```

In [12]:
```
type(a1), type(a2) , type(a3)
```

Out[12]:
```
(numpy.ndarray, numpy.ndarray, numpy.ndarray)
```

In [13]:
```
a2
```

Out[13]:
```
array([[1, 2, 3],
       [4, 5, 6]])
```

In [14]:
```
import pandas as pd

df = pd.DataFrame(a2)
df
```

Out[14]:

	0	1	2
0	1	2	3
1	4	5	6

creating Array

In [15]:
```
import numpy as np
```

In [16]:
```
ones = np.ones((2,3))
```

In [17]:
```
ones
```

Out[17]:
```
array([[1., 1., 1.],
       [1., 1., 1.]])
```

In [18]:
```
zeros = np.zeros((2,3))
```

In [19]:
```
zeros
```

Out[19]:
```
array([[0., 0., 0.],
       [0., 0., 0.]])
```

In [20]:
```
range_array = np.arange(0,10,2)
range_array
```

Out[20]:
```
array([0, 2, 4, 6, 8])
```

In [21]:
```
random_array = np.random.randint(0,10,size=(3,5))
```

In [22]:
```
random_array
```

Out[22]:
```
array([[4, 8, 1, 5, 8],
       [8, 1, 8, 4, 6],
       [7, 8, 0, 8, 5]])
```

In [23]:
```
random_array.shape
```

Out[23]:
```
(3, 5)
```

In [24]:
```
random_array_2 = np.random.random((5,3))
random_array_2
```

Out[24]:
```
array([[0.06029592, 0.19248014, 0.48402998],
       [0.35583366, 0.70733793, 0.78260088],
       [0.86365546, 0.46619734, 0.81310811],
       [0.9545508 , 0.5710412 , 0.60909644],
       [0.70309152, 0.46141875, 0.73121055]])
```

In [25]:
```
random_array_2.shape
```

Out[25]:
```
(5, 3)
```

In [26]:
```
random_array_3 = np.random.rand(5,3)
random_array_3
```

Out[26]:
```
array([[0.15744246, 0.04457178, 0.31675881],
       [0.64909918, 0.84724837, 0.62772998],
       [0.73575979, 0.36848744, 0.73025595],
       [0.75713127, 0.89081138, 0.12857797],
       [0.80797556, 0.72131409, 0.09689982]])
```

In [27]:
```
random_array_3.shape
```

Out[27]:
```
(5, 3)
```

In [28]:
```
np.random.seed(7)
random_array_5 = np.random.random((5,3))
random_array_5
```

Out[28]:
```
array([[0.07630829, 0.77991879, 0.43840923],
       [0.72346518, 0.97798951, 0.53849587],
       [0.50112046, 0.07205113, 0.26843898],
       [0.4998825 , 0.67923   , 0.80373904],
       [0.38094113, 0.06593635, 0.2881456 ]])
```

Viewing Arrays and Matrices

In [38]:
```
np.random.seed(7)
random_array
```

Out[38]:
```
array([[4, 8, 1, 5, 8],
       [8, 1, 8, 4, 6],
       [7, 8, 0, 8, 5]])
```

In [39]:
```
np.unique(random_array)
```

Out[39]:
```
array([0, 1, 4, 5, 6, 7, 8])
```

In [40]:
```
a1
```

Out[40]:
```
array([1, 2, 3])
```

In [41]:
```
a2
```

Out[41]:
```
array([[1, 2, 3],
       [4, 5, 6]])
```

In [42]:
```
a3
```

Out[42]:
```
array([[[1, 2, 3],
        [2, 0, 0]],

       [[0, 2, 3],
        [2, 0, 2]],

       [[0, 2, 3],
        [2, 0, 2]]])
```

In [43]:
```
a1[0]
```

Out[43]:
```
1
```

In [45]:
```
a2[0]
```

Out[45]:
```
array([1, 2, 3])
```

In [46]:
```
a3[0]
```

Out[46]:
```
array([[1, 2, 3],
       [2, 0, 0]])
```

In [47]:
```
a2[0][1]
```

Out[47]: 2

In [50]:
```
a4 = np.random.randint(10 , size = (2,3,3,4))
```

In [51]:
```
a4
```

Out[51]:
```
array([[[[7, 5, 4, 9],
         [6, 8, 1, 5],
         [5, 8, 3, 7]],

        [[7, 9, 4, 7],
         [5, 9, 6, 2],
         [0, 5, 3, 0]],

        [[5, 7, 1, 8],
         [4, 9, 0, 2],
         [0, 7, 6, 2]]],


       [[[9, 9, 5, 1],
         [0, 0, 9, 1],
         [1, 5, 3, 2]],

        [[0, 4, 8, 7],
         [1, 4, 9, 3],
         [6, 7, 1, 0]],

        [[2, 7, 0, 0],
         [9, 8, 2, 0],
         [5, 5, 0, 9]]]])
```

Array Slicing

In [103]:
```
a4[ 1:, 1:2, 1:2 , :3 ]
```

Out[103]:
```
array([[[[1, 4, 9]]]])
```

In [102]:
```
a4[1][1][1][1]
```

Out[102]: 4

Manipulating & comparing arrays

Arithmetic

In [104]:
```
a1
```

Out[104]:
```
array([1, 2, 3])
```

In [108]:
```
ones = np.ones(3)
ones
```

Out[108]:
```
array([1., 1., 1.])
```

In [109]:
```
a1 + ones
```

Out[109]:
```
array([2., 3., 4.])
```

In [110]:
```
a1 - ones
```

Out[110]:
```
array([0., 1., 2.])
```

In [111]:
```
 a1 * ones
```

Out[111]:
```
array([1., 2., 3.])
```

In [112]:
```
a1 / ones
```

Out[112]:
```
array([1., 2., 3.])
```

In [117]:
```
arr = np.array([1.5, 3.2, .9])
```

In [118]:
```
# floor division removes the decimals (rounds down)
ones // arr
```

Out[118]:
```
array([0., 0., 1.])
```

In [119]:
```
## power
arr ** 3
```

Out[119]:
```
array([ 3.375, 32.768,  0.729])
```

In [121]:
```
np.square(arr)
```

Out[121]:
```
array([ 2.25, 10.24,  0.81])
```

In [122]:
```
np.add(a1 , ones)
```

Out[122]:
```
array([2., 3., 4.])
```

In [123]:
```
# remainders
a1 % 2
```

Out[123]:
```
array([1, 0, 1], dtype=int32)
```

In [124]:
```
np.exp(a1)
```

Out[124]:
```
array([ 2.71828183,  7.3890561 , 20.08553692])
```

In [125]:
```
np.log(a1)
```

Out[125]:
```
array([0.        , 0.69314718, 1.09861229])
```

In [126]:
```
a3
```

Out[126]:
```
array([[[1, 2, 3],
        [2, 0, 0]],

       [[0, 2, 3],
        [2, 0, 2]],

       [[0, 2, 3],
        [2, 0, 2]]])
```

In [127]:
```
a3 +a2
```

Out[127]:
```
array([[[2, 4, 6],
        [6, 5, 6]],

       [[1, 4, 6],
        [6, 5, 8]],

       [[1, 4, 6],
        [6, 5, 8]]])
```

In [129]:
```
random_array_ = np.random.random((5,4))
random_array_
```

Out[129]:
```
array([[0.18257051, 0.69602194, 0.52673506, 0.19849106],
       [0.65046402, 0.50977649, 0.68903717, 0.57342801],
       [0.77011832, 0.39304825, 0.95858995, 0.97805187],
       [0.20688114, 0.4760013 , 0.79016346, 0.12793637],
       [0.82699424, 0.23743592, 0.94056563, 0.82065819]])
```

In [130]:
```
a2 + random_array_
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-130-ed48fa13b84c> in <module>
----> 1 a2 + random_array_

ValueError: operands could not be broadcast together with shapes (2,3) (5,4)
```

Aggregation

Aggregation = Performing the same operation on a number of things

In [132]:
```
# Create a massive Numpy array
massive_array = np.random.random(100000)
```

In [134]:
```
# time comparision between Python's method and Numpy's methods
%timeit sum(massive_array) #Python sum
%timeit np.sum(massive_array) #Numpy sum
```

```
38.4 ms ± 1.43 ms per loop (mean ± std. dev. of 7 runs, 10 loops each)
150 µs ± 6.19 µs per loop (mean ± std. dev. of 7 runs, 10000 loops each)
```

In [135]:
```
a2
```

Out[135]:
```
array([[1, 2, 3],
       [4, 5, 6]])
```

In [136]:
```
np.mean(a2)
```

Out[136]:
```
3.5
```

In [137]:
```
np.max(a2)
```

Out[137]:
```
6
```

In [138]:
```
np.min(a2)
```

Out[138]:
```
1
```

In [139]:
```
# standard deviation = a measure of how spread out a group of numbers is from the mean
# standard deviation is actually just the square root of the variance
np.std(a2)
```

Out[139]:
```
1.707825127659933
```

In [142]:
```
# Variance = measure of the average degree to which each number is different to the mean
# higher variance = wider range of numbers
# Lower variance = lower range of numbers

np.var(a2)
```

Out[142]:
```
2.9166666666666665
```

In [141]:
```
# standard deviation equals square root of variance
np.sqrt(np.var(a2))
```

Out[141]:
```
1.707825127659933
```

In [143]:
```
# Demo of std and var
high_var_array = np.array([1, 100, 200, 300, 4000, 5000])
low_var_array = np.array([2, 4, 6, 8, 10])
```

In [144]:
```
np.var(high_var_array) , np.var(low_var_array)
```

Out[144]: (4296133.472222221, 8.0)

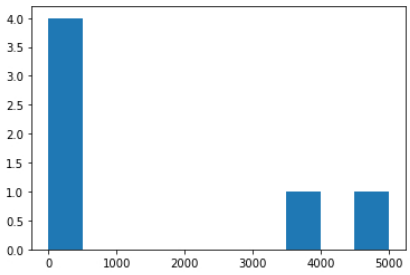
In [145]:
```
np.std(high_var_array) , np.std(low_var_array)
```

Out[145]: (2072.711623024829, 2.8284271247461903)

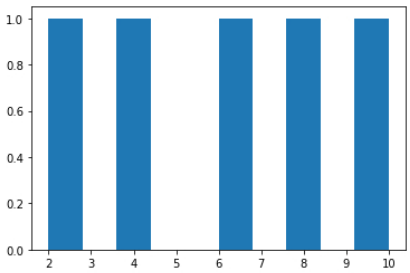
In [146]:
```
np.mean(high_var_array) , np.mean(low_var_array)
```

Out[146]: (1600.1666666666667, 6.0)

In [148]:
```
%matplotlib inline
import matplotlib.pyplot as plt
plt.hist(high_var_array)
plt.show()
```



In [149]:
```
plt.hist(low_var_array)
plt.show()
```



Reshape and Transpose

In [150]:
```
a2
```

Out[150]:
```
array([[1, 2, 3],
       [4, 5, 6]])
```

In [153]:
```
a4
```

Out[153]:
```
array([[[[7, 5, 4, 9],
         [6, 8, 1, 5],
         [5, 8, 3, 7]],

        [[7, 9, 4, 7],
         [5, 9, 6, 2],
         [0, 5, 3, 0]],

        [[5, 7, 1, 8],
         [4, 9, 0, 2],
         [0, 7, 6, 2]]],


       [[[9, 9, 5, 1],
         [0, 0, 9, 1],
         [1, 5, 3, 2]],

        [[0, 4, 8, 7],
         [1, 4, 9, 3],
         [6, 7, 1, 0]],

        [[2, 7, 0, 0],
         [9, 8, 2, 0],
         [5, 5, 0, 9]]]])
```

In [155]:
```

```

Out[155]: array([1.5, 3.2, 0.9])

In [157]:
```
np.shape(a2)
```

Out[157]: (2, 3)

In [162]:
```
arr2 = np.random.randint(0,10,size=(2,3,3))
```

In [163]:
```
arr2 * a2
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-163-864d1d25ae0c> in <module>
----> 1 arr2 * a2

ValueError: operands could not be broadcast together with shapes (2,3,3) (2,3)
```

In [166]:
```
a2_reshape = a2.reshape(2,3,1)
```

In [167]:
```
arr2 * a2_reshape
```

Out[167]:
```
array([[[ 9,  7,  0],
        [ 4,  8,  8],
        [21,  6, 24]],

       [[12,  8, 20],
        [15, 15, 15],
        [42,  0,  6]]])
```

In [169]:
```
a2
```

Out[169]:
```
array([[1, 2, 3],
       [4, 5, 6]])
```

In [168]:
```
a2.T
```

Out[168]:
```
array([[1, 4],
       [2, 5],
       [3, 6]])
```

Dot product

In [171]:
```
np.random.seed(0)
matrix1 = np.random.randint(10, size = (5,3))
matrix2= np.random.randint(10, size = (5,3))
```

In [175]:
```
#Dot Product
np.dot(matrix1, matrix2)
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-175-bbff5df69907> in <module>
      1 #Dot Product
----> 2 np.dot(matrix1, matrix2)

<__array_function__ internals> in dot(*args, **kwargs)

ValueError: shapes (5,3) and (5,3) not aligned: 3 (dim 1) != 5 (dim 0)
```

In [181]:
```
#changing axis using transpose
matrix2_T = matrix2.T
```

In [182]:
```
np.dot(matrix1 , matrix2_T)
```

Out[182]:
```
array([[ 51,  55,  72,  20,  15],
       [130,  76, 164,  33,  44],
       [ 67,  39,  85,  27,  34],
       [115,  69, 146,  37,  47],
       [111,  77, 145,  56,  64]])
```

dot product example (nut butter sales)

In [207]:
```
np.random.seed(0)
# Number of jars sold
sales_amounts = np.random.randint(20, size=(5,3))
```

In [209]:
```
# Create weekly_sales DataFrame
weekly_sales = pd.DataFrame(sales_amounts, index=["Mon","Tues","Wed","thurs","Fri"],
                            columns = ["Almond Butter","Peanut Butter","Cashew Butter"])
weekly_sales
```

Out[209]:

	Almond Butter	Peanut Butter	Cashew Butter
Mon	12	15	0
Tues	3	3	7
Wed	9	19	18
thurs	4	6	12
Fri	1	6	7

In [210]:
```
# Create prices array
prices = np.array([10,8,12])
prices
```

Out[210]:
```
array([10,  8, 12])
```

In [191]:
```
# Create butter_prices DataFrame
butter_prices = pd.DataFrame(prices.reshape(1,3), index=["Prices"],
                             columns = ["Almond Butter","Peanut Butter","Cashew Butter"])

butter_prices
```

Out[191]:

	Almond Butter	Peanut Butter	Cashew Butter
Prices	10	8	12

In [192]:
```
prices
```

Out[192]:
```
array([10,  8, 12])
```

In [193]:
```
total_sales = prices.dot(sales_amounts)
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-193-e4058314ac09> in <module>
----> 1 total_sales = prices.dot(sales_amounts)

ValueError: shapes (3,) and (5,3) not aligned: 3 (dim 0) != 5 (dim 0)
```

In [194]:
```
# Shape aren't aligned, let's transpose
total_sales = prices.dot(sales_amounts.T)
```

In [195]:
```
total_sales
```

Out[195]:
```
array([240, 138, 458, 232, 142])
```

In [197]:
```
# create daily_sales
daily_sales = butter_prices.dot(weekly_sales.T)
```

In [198]:
```
daily_sales
```

Out[198]:

	Mon	Tues	Wed	thurs	Fri
Prices	240	138	458	232	142

In [199]:
```
weekly_sales
```

Out[199]:

	Almond Butter	Peanut Butter	Cashew Butter
Mon	12	15	0
Tues	3	3	7
Wed	9	19	18
thurs	4	6	12
Fri	1	6	7

In [201]:
```
weekly_sales["Total ($)"] = daily_sales.T
```

In [202]:
```
weekly_sales
```

Out[202]:

	Almond Butter	Peanut Butter	Cashew Butter	Total ($)
Mon	12	15	0	240
Tues	3	3	7	138
Wed	9	19	18	458
thurs	4	6	12	232
Fri	1	6	7	142

In [206]:
```
weekly_sales.to_csv("Our Sales")
```

<----------dot product example (nut butter sales) Ended---------->

comparison operators

In [211]:
```
a1
```

Out[211]:
```
array([1, 2, 3])
```

In [212]:
```
 a2
```

Out[212]:
```
array([[1, 2, 3],
       [4, 5, 6]])
```

In [213]:
```
a1 > a2
```

Out[213]:
```
array([[False, False, False],
       [False, False, False]])
```

In [221]:
```
boolen_array = a1 >= a2
boolen_array , type(boolen_array), boolen_array.dtype
```

Out[221]:
```
(array([[ True,  True,  True],
        [False, False, False]]),
 numpy.ndarray,
 dtype('bool'))
```

In [222]:
```
a1 > 5
```

Out[222]:
```
array([False, False, False])
```

In [223]:
```
a1 == a2
```

Out[223]:
```
array([[ True,  True,  True],
       [False, False, False]])
```

Sorting Arrays

In [224]:
```
random_array
```

Out[224]:
```
array([[4, 8, 1, 5, 8],
       [8, 1, 8, 4, 6],
       [7, 8, 0, 8, 5]])
```

In [225]:
```
np.sort(random_array)
```

Out[225]:
```
array([[1, 4, 5, 8, 8],
       [1, 4, 6, 8, 8],
       [0, 5, 7, 8, 8]])
```

In [226]:
```
 random_array
```

Out[226]:
```
array([[4, 8, 1, 5, 8],
       [8, 1, 8, 4, 6],
       [7, 8, 0, 8, 5]])
```

In [227]:
```
np.argsort(random_array)
```

Out[227]:
```
array([[2, 0, 3, 1, 4],
       [1, 3, 4, 0, 2],
       [2, 4, 0, 1, 3]], dtype=int64)
```

In [228]:
```
a1
```

Out[228]:
```
array([1, 2, 3])
```

In [229]:
```
np.argmin(a1)
```

Out[229]:
```
0
```

In [230]:
```
a1
```

Out[230]:
```
array([1, 2, 3])
```

In [231]:
```
np.argmax(a1)
```

Out[231]:
```
2
```

In [232]:
```
random_array
```

Out[232]:
```
array([[4, 8, 1, 5, 8],
       [8, 1, 8, 4, 6],
       [7, 8, 0, 8, 5]])
```

In [228]:
```
a1
```

Out[228]:
```
array([1, 2, 3])
```

In [229]:
```
np.argmin(a1)
```

Out[229]:
```
0
```

In [230]:
```
a1
```

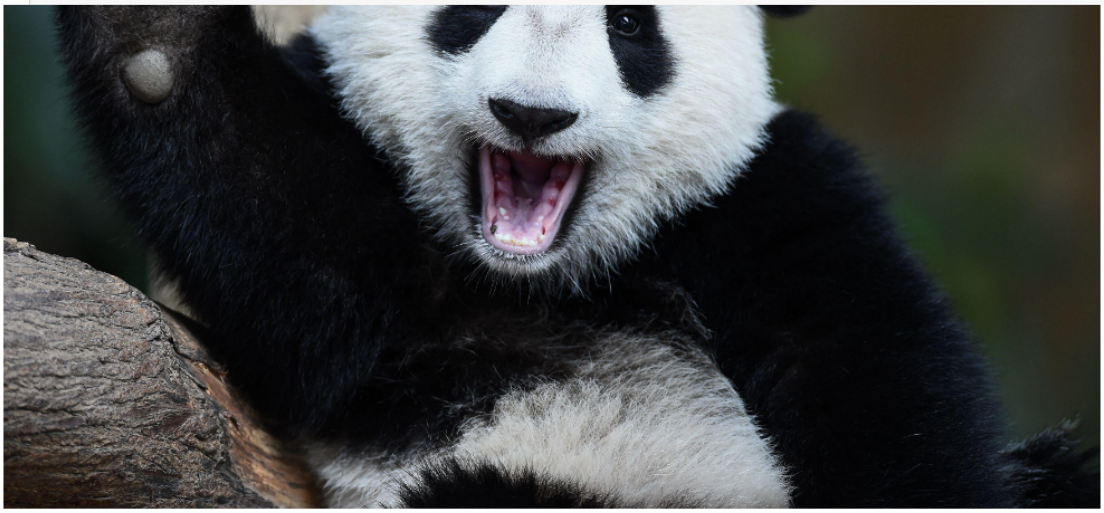
Out[230]:
```
array([1, 2, 3])
```

In [231]:
```
np.argmax(a1)
```

Out[231]:
```
2
```

In [232]:
```
random_array
```

Out[232]:
```
array([[4, 8, 1, 5, 8],
       [8, 1, 8, 4, 6],
       [7, 8, 0, 8, 5]])
```



In [239]:
```
# Turn an image into a Numpy array
```

```
from matplotlib.image import imread

panda = imread("images/panda.png")
type(panda)
```

Out[239]:
```
numpy.ndarray
```

In [241]:
```
panda.size, panda.shape, panda.ndim
```

Out[241]:
```
(24465000, (2330, 3500, 3), 3)
```

In [ ]:
```

```

In [244]:
```
car = imread("images/car-photo.png")
car.T
```

Out[244]:
```
array([[[0.5019608 , 0.44313726, 0.39607844, ..., 0.47058824,
         0.4745098 , 0.4745098 ],
        [0.3372549 , 0.3137255 , 0.31764707, ..., 0.4392157 ,
         0.47058824, 0.49411765],
        [0.20392157, 0.2       , 0.28627452, ..., 0.48235294,
         0.4117647 , 0.49019608],
        ...,
        [0.64705884, 0.5058824 , 0.44705883, ..., 0.6156863 ,
         0.54901963, 0.5294118 ],
        [0.59607846, 0.49803922, 0.45882353, ..., 0.61960787,
         0.65882355, 0.5529412 ],
        [0.44705883, 0.4       , 0.4509804 , ..., 0.5921569 ,
         0.60784316, 0.6156863 ]],

       [[0.50980395, 0.43529412, 0.38039216, ..., 0.57254905,
         0.5803922 , 0.57254905],
        [0.34509805, 0.31764707, 0.3137255 , ..., 0.53333336,
         0.58431375, 0.6       ],
        [0.21568628, 0.21176471, 0.29411766, ..., 0.5803922 ,
         0.5019608 , 0.5921569 ],
        ...,
        [0.7058824 , 0.5372549 , 0.45882353, ..., 0.7529412 ,
         0.6784314 , 0.63529414],
        [0.63529414, 0.52156866, 0.48235294, ..., 0.7607843 ,
         0.8117647 , 0.67058825],
        [0.47058824, 0.42745098, 0.49019608, ..., 0.73333335,
         0.74509805, 0.73333335]],

       [[0.4862745 , 0.40392157, 0.34117648, ..., 0.6313726 ,
         0.6392157 , 0.627451  ],
        [0.30588236, 0.27450982, 0.27450982, ..., 0.5882353 ,
         0.63529414, 0.65882355],
        [0.14901961, 0.14117648, 0.24705882, ..., 0.6392157 ,
         0.5529412 , 0.64705884],
        ...,
        [0.54901963, 0.4117647 , 0.32156864, ..., 0.827451  ,
         0.74509805, 0.69803923],
        [0.45882353, 0.39607844, 0.3529412 , ..., 0.83137256,
         0.8862745 , 0.7372549 ],
        [0.3372549 , 0.34117648, 0.38039216, ..., 0.7921569 ,
         0.8117647 , 0.8       ]],

       [[1.        , 1.        , 1.        , ..., 1.        ,
         1.        , 1.        ],
        [1.        , 1.        , 1.        , ..., 1.        ,
         1.        , 1.        ],
        [1.        , 1.        , 1.        , ..., 1.        ,
         1.        , 1.        ],
        ...,
        [1.        , 1.        , 1.        , ..., 1.        ,
         1.        , 1.        ],
        [1.        , 1.        , 1.        , ..., 1.        ,
         1.        , 1.        ],
        [1.        , 1.        , 1.        , ..., 1.        ,
         1.        , 1.        ]]], dtype=float32)
```

In [ ]:
```

```