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

DataType & Attribute

In [2]:
```
#Numpy main datatype is ndarray
a1 = np.array([1,2,3,4])
a1
```
Out[2]:
```
array([1, 2, 3, 4])
```

In [3]:
```
type(a1)
```
Out[3]:
```
numpy.ndarray
```

In [4]:
```
a2 = np.array([[1,2.3,3.0,4.1],
              [4,5,6.5,4.0]])

a3 = np.array([[[2, 2, 3],
                [4, 5, 6],
                [7, 8, 9]],

               [[10, 11, 12],
                [13, 14, 15],
                [16, 17, 18]]])
```

In [5]:
```
a2
```
Out[5]:
```
array([[1. , 2.3, 3. , 4.1],
       [4. , 5. , 6.5, 4. ]])
```

In [6]:
```
a3
```
Out[6]:
```
array([[[ 2,  2,  3],
        [ 4,  5,  6],
        [ 7,  8,  9]],

       [[10, 11, 12],
        [13, 14, 15],
        [16, 17, 18]]])
```

In [7]:
```
a1.shape
```
Out[7]:
```
(4,)
```

In [8]:
```
a1
```
Out[8]:
```
array([1, 2, 3, 4])
```

In [9]:
```
a2.shape
```
Out[9]:
```
(2, 4)
```

In [10]:
```
a3.shape
```
Out[10]:
```
(2, 3, 3)
```

In [11]:
```
a1.ndim,a2.ndim, a3.ndim
```
Out[11]:
```
(1, 2, 3)
```

In [ ]:
```

```

a1.dtype, a2.dtype, a3.dtype

In [12]:
```
a1.size, a2.size, a3.size
```
Out[12]:
```
(4, 8, 18)
```

In [13]:
```
type(a1), type(a2),type(a3)
```
Out[13]:
```
(numpy.ndarray, numpy.ndarray, numpy.ndarray)
```

In [14]:
```
a2
```
Out[14]:
```
array([[1. , 2.3, 3. , 4.1],
       [4. , 5. , 6.5, 4. ]])
```

In [20]:
```
# create a dataFrame from numpy
import pandas as pd
df = pd.DataFrame(a2)
df
```
Out[20]:

	0	1	2	3
0	1.0	2.3	3.0	4.1
1	4.0	5.0	6.5	4.0

2. Creating numpy arrays

In [21]:
```
sample = np.array([1,2,3])
sample
```
Out[21]:
```
array([1, 2, 3])
```

In [22]:
```
sample.dtype
```
Out[22]:
```
dtype('int32')
```

In [23]:
```
sample.size
```
Out[23]:
```
3
```

In [24]:
```
names = np.ones((3,2))
names
```
Out[24]:
```
array([[1., 1.],
       [1., 1.],
       [1., 1.]])
```

In [25]:
```
type(names)
```
Out[25]:
```
numpy.ndarray
```

In [26]:
```
name = np.zeros((3,2))
name
```
Out[26]:
```
array([[0., 0.],
       [0., 0.],
       [0., 0.]])
```

In [27]:
```
number = np.arange(0,10,2)
number
```
Out[27]:
```
array([0, 2, 4, 6, 8])
```

In [28]:
```
numbers = np.random.randint(0, 10, size=(3, 5))
numbers
```
Out[28]:
```
array([[3, 5, 8, 7, 6],
       [6, 2, 7, 2, 8],
       [5, 0, 2, 0, 0]])
```

In [29]:
```
numbers.size
```
Out[29]:
```
15
```

In [30]:
```
numbers.shape
```
Out[30]:
```
(3, 5)
```

In [31]:
```
number_array = np.random.random((5,3))
number_array
```
Out[31]:
```
array([[0.17991133, 0.08734861, 0.94957471],
       [0.68626163, 0.88183037, 0.1733742 ],
       [0.84113894, 0.32177582, 0.67807146],
       [0.26515449, 0.17550891, 0.99123504],
       [0.24701084, 0.98168298, 0.59131612]])
```

In [32]:
```
number_array.shape
```
Out[32]:
```
(5, 3)
```

In [33]:
```
#pseudo-random number
#the seed() make the random.randint numbers dont change
np.random.seed(seed=0)
number_array_2 = np.random.randint(10,size=(5,3))
number_array_2
```
Out[33]:
```
array([[5, 0, 3],
       [3, 7, 9],
       [3, 5, 2],
       [4, 7, 6],
       [8, 8, 1]])
```

3. viewing arrays and matrices

In [34]:
```
np.unique(number_array_2)
```
Out[34]:
```
array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])
```

In [35]:
```
a3
```
Out[35]:
```
array([[[ 2,  2,  3],
        [ 4,  5,  6],
        [ 7,  8,  9]],

       [[10, 11, 12],
        [13, 14, 15],
        [16, 17, 18]]])
```

In [36]:
```
a3[:2, :2, :2]
```
Out[36]:
```
array([[[ 2,  2],
        [ 4,  5]],

       [[10, 11],
        [13, 14]]])
```

In [37]:
```
a3.shape
```
Out[37]:
```
(2, 3, 3)
```

4.MANIPULATING AND COMPARING ARRAYS

In [38]:
```
a1
```
Out[38]:
```
array([1, 2, 3, 4])
```

In [39]:
```
ones = np.ones(4)
ones
```
Out[39]:
```
array([1., 1., 1., 1.])
```

In [40]:
```
 a1 + ones
```
Out[40]:
```
array([2., 3., 4., 5.])
```

In [41]:
```
a1 - ones
```
Out[41]:
```
array([0., 1., 2., 3.])
```

In [42]:
```
a1 * ones
```
Out[42]:
```
array([1., 2., 3., 4.])
```

In [43]:
```
a2
```
Out[43]:
```
array([[1. , 2.3, 3. , 4.1],
       [4. , 5. , 6.5, 4. ]])
```

In [44]:
```
a1 * a2
```
Out[44]:
```
array([[ 1. ,  4.6,  9. , 16.4],
       [ 4. , 10. , 19.5, 16. ]])
```

In [45]:
```
np.square(a2)
```
Out[45]:
```
array([[ 1.  ,  5.29,  9.  , 16.81],
       [16.  , 25.  , 42.25, 16.  ]])
```

In [46]:
```
np.add(a1, ones)
```
Out[46]:
```
array([2., 3., 4., 5.])
```

In [47]:
```
a1 % 2
```
Out[47]:
```
array([1, 0, 1, 0], dtype=int32)
```

In [48]:
```
1 / 2
```
Out[48]:
```
0.5
```

Aggregation

In [49]:
```
listy_list = [1, 2, 3]
type(listy_list)
```
Out[49]:
```
list
```

In [50]:
```
sum(listy_list)
```
Out[50]:
```
6
```

In [51]:
```
massive_array = np.random.random(100000)
massive_array.size
```
Out[51]:
```
100000
```

In [52]:
```
massive_array[:10]
```
Out[52]:
```
array([0.79172504, 0.52889492, 0.56804456, 0.92559664, 0.07103606,
       0.0871293 , 0.0202184 , 0.83261985, 0.77815675, 0.87001215])
```

In [53]:
```
%timeit sum(massive_array) #python's sum()
%timeit np.sum(massive_array) # Numpy's np.sum()
```
```
9.36 ms ± 159 µs per loop (mean ± std. dev. of 7 runs, 100 loops each)
57.4 µs ± 2.49 µs per loop (mean ± std. dev. of 7 runs, 10,000 loops each)
```

In [54]:
```
a2
```
Out[54]:
```
array([[1. , 2.3, 3. , 4.1],
       [4. , 5. , 6.5, 4. ]])
```

In [55]:
```
np.mean(a2)
```
Out[55]:
```
3.7375
```

In [56]:
```
np.var(a2)
```
Out[56]:
```
2.4498437500000003
```

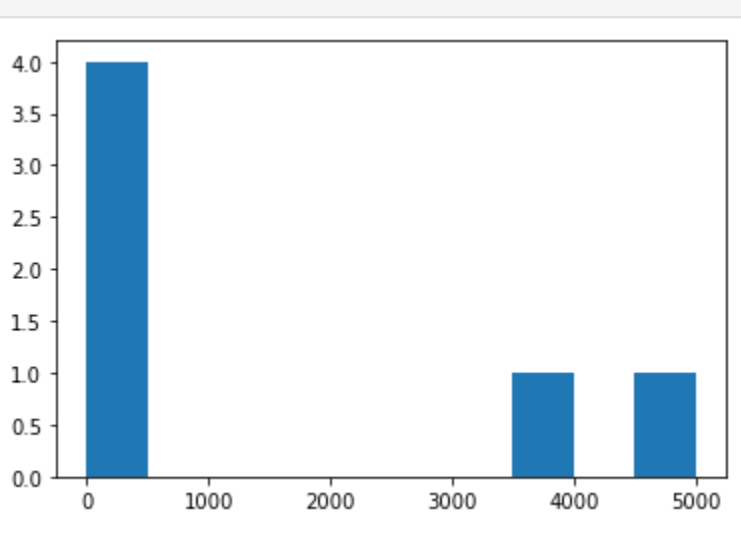
In [57]:
```
# demo of and var
high_array = np.array([1, 100, 200, 300, 4000, 5000])
low_array = np.array([2, 4, 6, 8, 10])
```

In [58]:
```
np.var(high_array ), np.var(low_array)
```
Out[58]:
```
(4296133.472222221, 8.0)
```

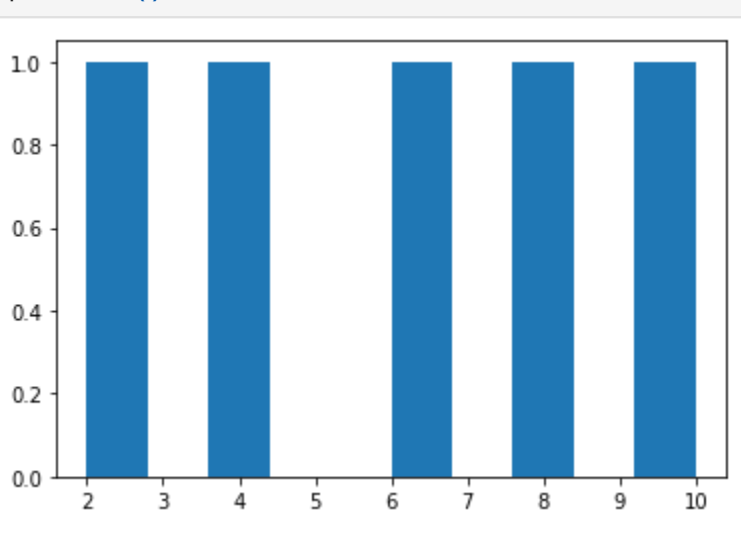
In [59]:
```
np.std(high_array ), np.std(low_array )
```
Out[59]:
```
(2072.711623024829, 2.8284271247461903)
```

In [60]:
```
np.mean(high_array ), np.mean(low_array )
```
Out[60]:
```
(1600.1666666666667, 6.0)
```

In [61]:
```
%matplotlib inline
import matplotlib.pyplot as plt
plt.hist(high_array)
plt.show()
```



In [62]:
```
plt.hist(low_array)
plt.show()
```



Reshape and transpose

In [63]:
```
a2
```
Out[63]:
```
array([[1. , 2.3, 3. , 4.1],
       [4. , 5. , 6.5, 4. ]])
```

In [64]:
```
a2.shape
```
Out[64]:
```
(2, 4)
```

In [65]:
```
 a3
```
Out[65]:
```
array([[[ 2,  2,  3],
        [ 4,  5,  6],
        [ 7,  8,  9]],

       [[10, 11, 12],
        [13, 14, 15],
        [16, 17, 18]]])
```

In [66]:
```
a3.shape
```
Out[66]:
```
(2, 3, 3)
```

In [67]:
```
a2.reshape(2, 4, 1)
```
Out[67]:
```
array([[[1. ],
        [2.3],
        [3. ],
        [4.1]],

       [[4. ],
        [5. ],
        [6.5],
        [4. ]]])
```

In [68]:
```
a2_reshape = a2.reshape(2, 4, 1)
a2_reshape.shape
```
Out[68]:
```
(2, 4, 1)
```

In [69]:
```
 a3.shape
```
Out[69]:
```
(2, 3, 3)
```

In [70]:
```
a4 = np.array([[[1. ],
        [2.3],
        [3. ],
        [4.1]],

       [[4. ],
        [5. ],
        [6.5],
        [4. ]]])
```

In [71]:
```
a4
```
Out[71]:
```
array([[[1. ],
        [2.3],
        [3. ],
        [4.1]],

       [[4. ],
        [5. ],
        [6.5],
        [4. ]]])
```

In [72]:
```
a4.shape
```
Out[72]:
```
(2, 4, 1)
```

In [73]:
```
 a4 * a2_reshape
```
Out[73]:
```
array([[[ 1.  ],
        [ 5.29],
        [ 9.  ],
        [16.81]],

       [[16.  ],
        [25.  ],
        [42.25],
        [16.  ]]])
```

In [74]:
```
# Traanspose = switches the axis'
a2.T
```
Out[74]:
```
array([[1. , 4. ],
       [2.3, 5. ],
       [3. , 6.5],
       [4.1, 4. ]])
```

Dot product

In [75]:
```
a2.T
```
Out[75]:
```
array([[1. , 4. ],
       [2.3, 5. ],
       [3. , 6.5],
       [4.1, 4. ]])
```

In [76]:
```
np.random.seed(0)
mat1 = np.random.randint(10, size=(5, 3))
mat2 = np.random.randint(10, size=(5, 3))
mat1
```
Out[76]:
```
array([[5, 0, 3],
       [3, 7, 9],
       [3, 5, 2],
       [4, 7, 6],
       [8, 8, 1]])
```

In [77]:
```
mat2
```
Out[77]:
```
array([[6, 7, 7],
       [8, 1, 5],
       [9, 8, 9],
       [4, 3, 0],
       [3, 5, 0]])
```

In [78]:
```
mat1.shape, mat2.shape
```
Out[78]:
```
((5, 3), (5, 3))
```

In [79]:
```
# element-wise multiplication
mat1 * mat2
```
Out[79]:
```
array([[30,  0, 21],
       [24,  7, 45],
       [27, 40, 18],
       [16, 21,  0],
       [24, 40,  0]])
```

In [80]:
```
mat1.T
```
Out[80]:
```
array([[5, 3, 3, 4, 8],
       [0, 7, 5, 7, 8],
       [3, 9, 2, 6, 1]])
```

In [81]:
```
mat1.T.shape, mat2.T.shape
```
Out[81]:
```
((3, 5), (3, 5))
```

In [82]:
```
mat3 = np.dot(mat1, mat2.T)
mat3
```
Out[82]:
```
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]])
```

In [83]:
```
mat3.shape
```
Out[83]:
```
(5, 5)
```

In [84]:
```
mat2.T ,mat1.T
```
Out[84]:
```
(array([[6, 8, 9, 4, 3],
        [7, 1, 8, 3, 5],
        [7, 5, 9, 0, 0]]),
 array([[5, 3, 3, 4, 8],
        [0, 7, 5, 7, 8],
        [3, 9, 2, 6, 1]]))
```

In [85]:
```
mat2.T * mat1.T
```
Out[85]:
```
array([[30, 24, 27, 16, 24],
       [ 0,  7, 40, 21, 40],
       [21, 45, 18,  0,  0]])
```

comparison operators

In [86]:
```
a1
```
Out[86]:
```
array([1, 2, 3, 4])
```

In [87]:
```
a2
```
Out[87]:
```
array([[1. , 2.3, 3. , 4.1],
       [4. , 5. , 6.5, 4. ]])
```

In [88]:
```
a1 >= a2
```
Out[88]:
```
array([[ True, False,  True, False],
       [False, False, False,  True]])
```

In [89]:
```
bool_array = a1 >= a2
bool_array
type(bool_array), bool_array.dtype
```
Out[89]:
```
(numpy.ndarray, dtype('bool'))
```

In [90]:
```
a1==4
```
Out[90]:
```
array([False, False, False,  True])
```

In [91]:
```
a1==a1
```
Out[91]:
```
array([ True,  True,  True,  True])
```

In [92]:
```
a2==a1
```
Out[92]:
```
array([[ True, False,  True, False],
       [False, False, False,  True]])
```

sorting array

In [93]:
```
random_array = np.random.randint(10, size=(3, 5))
random_array
```
Out[93]:
```
array([[2, 3, 8, 1, 3],
       [3, 3, 7, 0, 1],
       [9, 9, 0, 4, 7]])
```

In [94]:
```
random_array.shape
```
Out[94]:
```
(3, 5)
```

In [95]:
```
np.sort(random_array)
```
Out[95]:
```
array([[1, 2, 3, 3, 8],
       [0, 1, 3, 3, 7],
       [0, 4, 7, 9, 9]])
```

In [96]:
```
random_array
```
Out[96]:
```
array([[2, 3, 8, 1, 3],
       [3, 3, 7, 0, 1],
       [9, 9, 0, 4, 7]])
```

In [97]:
```
np.argsort(random_array)
```
Out[97]:
```
array([[3, 0, 1, 4, 2],
       [3, 4, 0, 1, 2],
       [2, 3, 4, 0, 1]], dtype=int64)
```

In [98]:
```
a1
```
Out[98]:
```
array([1, 2, 3, 4])
```

In [99]:
```
np.argsort(a1)
```
Out[99]:
```
array([0, 1, 2, 3], dtype=int64)
```