1 内容

1. numpy
2. pandas
3. matplotlib

2 numpy

数组跟列表，列表可以存储任意类型的数据，而数组只能存储一种类型数据

In [1]:

```
import array
```

In [3]:

```
a = array.array('i', range(20))
```

In [4]:

```
a[1] = 10
```

In [5]:

```
a
```

Out[5]:

```
array('i', [0, 10, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19])
```

In [6]:

```
# 数据类型必须统一
a[1] = 'a'
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-6-162bacf31dfa> in <module>
----> 1 a[1] = 'a'

TypeError: an integer is required (got type str)
```

In [8]:

```
import numpy as np
```

3 从原有列表转换为数组

In [9]:

```
a_list = list(range(10))
b = np.array(a_list)
type(b)
```

Out[9]:

```
numpy.ndarray
```

4 生成数组

In [10]:

```
a = np.zeros(10, dtype = int)
print(type(a))
# 查看数组类型
a.dtype
```

```
<class 'numpy.ndarray'>
```

Out[10]:

```
dtype('int32')
```

In [11]:

```
a = np.zeros((4, 4), dtype = int)
print(type(a))
# 查看数组类型
print(a.dtype)
a
```

```
<class 'numpy.ndarray'>
int32
```

Out[11]:

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

In [12]:

```
np.ones((4, 4), dtype = float)
```

Out[12]:

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

In [13]:

```
np.full((3, 3), 3.14)
```

Out[13]:

```
array([[3.14, 3.14, 3.14],
       [3.14, 3.14, 3.14],
       [3.14, 3.14, 3.14]])
```

In [14]:

```
np.zeros_like(a)
```

Out[14]:

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

In [16]:

```
np.full_like(a, 3.14, dtype = float)
```

Out[16]:

```
array([[3.14, 3.14, 3.14, 3.14],
       [3.14, 3.14, 3.14, 3.14],
       [3.14, 3.14, 3.14, 3.14],
       [3.14, 3.14, 3.14, 3.14]])
```

5 random

In [22]:

```
# 生成随机数
import random
print(random.randint(5,10))
print(random.random())
```

```
8
0.8410368140834427
```

In [25]:

```
# 生成3 * 3的随机数
r = np.random.random((3,3))
print(type(r))
```

```
<class 'numpy.ndarray'>
```

In [26]:

```
# 经常用到
# 生成5 * 5的多维数组（元素为0 - 10之间的随机整数）
np.random.randint(0, 10, (5, 5))
```

Out[26]:

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

6 范围取值

In [30]:

```
# 取0 - 10之间步长为2的整数
list(range(0, 10, 2))
```

Out[30]:

```
[0, 2, 4, 6, 8]
```

In [31]:

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

...

In [32]:

```
# 生成0 - 3之间步长相同的100个数
np.linspace(0, 3, 100)
```

Out[32]:

```
array([0.        , 0.03030303, 0.06060606, 0.09090909, 0.12121212,
       0.15151515, 0.18181818, 0.21212121, 0.24242424, 0.27272727,
       0.3030303 , 0.33333333, 0.36363636, 0.39393939, 0.42424242,
       0.45454545, 0.48484848, 0.51515152, 0.54545455, 0.57575758,
       0.60606061, 0.63636364, 0.66666667, 0.6969697 , 0.72727273,
       0.75757576, 0.78787879, 0.81818182, 0.84848485, 0.87878788,
       0.90909091, 0.93939394, 0.96969697, 1.        , 1.03030303,
       1.06060606, 1.09090909, 1.12121212, 1.15151515, 1.18181818,
       1.21212121, 1.24242424, 1.27272727, 1.3030303 , 1.33333333,
       1.36363636, 1.39393939, 1.42424242, 1.45454545, 1.48484848,
       1.51515152, 1.54545455, 1.57575758, 1.60606061, 1.63636364,
       1.66666667, 1.6969697 , 1.72727273, 1.75757576, 1.78787879,
       1.81818182, 1.84848485, 1.87878788, 1.90909091, 1.93939394,
       1.96969697, 2.        , 2.03030303, 2.06060606, 2.09090909,
       2.12121212, 2.15151515, 2.18181818, 2.21212121, 2.24242424,
       2.27272727, 2.3030303 , 2.33333333, 2.36363636, 2.39393939,
       2.42424242, 2.45454545, 2.48484848, 2.51515152, 2.54545455,
       2.57575758, 2.60606061, 2.63636364, 2.66666667, 2.6969697 ,
       2.72727273, 2.75757576, 2.78787879, 2.81818182, 2.84848485,
       2.87878788, 2.90909091, 2.93939394, 2.96969697, 3.        ])
```

In [33]:

```
# 生成n维单位矩阵
np.eye(5)
```

Out[33]:

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

访问数组中的元素

In [49]:

```
# 嵌套列表的元素访问
var = [[1, 2, 3], [3, 4, 5], [5, 6, 7]]
var[0][2]
```

Out[49]:

```
3
```

In [54]:

```
a = np.full((3, 3), 9.9, dtype = float)
a
```

Out[54]:

```
array([[9.9, 9.9, 9.9],
       [9.9, 9.9, 9.9],
       [9.9, 9.9, 9.9]])
```

In [52]:

```
a[0]
```

Out[52]:

```
array([9.9, 9.9, 9.9])
```

In [53]:

```
a[0][0]
```

Out[53]:

```
9.9
```

In [55]:

```
#数组中元素访问
a = np.array(var)
a
```

Out[55]:

```
array([[1, 2, 3],
       [3, 4, 5],
       [5, 6, 7]])
```

In [56]:

```
a[-1][0]
```

Out[56]:

```
5
```

In [58]:

```
# 两种访问方式等价
a[0, 0], a[0][0]
```

Out[58]:

```
(1, 1)
```

In [62]:

```
# 数组切片
a[:3, :2]
```

Out[62]:

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

In [63]:

```
# 不等价
a[:2][:2]
```

Out[63]:

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

In [72]:

```
# 维度
print(a.ndim)
# shape
print(a.shape)
# size
print(a.size)
# dtype
print(a.dtype)
# a.itemsize 每个数字占的字节数
print(a.itemsize)
# nbytes
print(a.nbytes)
```

```
2
(3, 3)
9
int32
4
36
```

7 运算

In [74]:

```
a = np.array(list(range(10)))
a
```

Out[74]:

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

In [77]:

```
print(a + 10)
print(a - 10)
print(a * 10)
```

```
[10 11 12 13 14 15 16 17 18 19]
[-10  -9  -8  -7  -6  -5  -4  -3  -2  -1]
[ 0 10 20 30 40 50 60 70 80 90]
```

In [78]:

```
a = np.full((3, 3), 1.0, dtype = float)
a + 10  # 等价于np.add(a, 10)
```

Out[78]:

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

Operator	Equivalent ufunc	Description
+	np.add	Addition(e.g., 1 + 1 = 2)

Operator	Equivalent ufunc	Description
-	np.subtract	Subtraction(e.g., 3 - 2 = 1)
-	np.negative	Unary negation(e.g., -2)
*	np.multiply	Mulutiplication(e.g., 2 * 3 = 6)
/	np.divide	Dividsion(e.g., 3 / 2 = 1.5)
//	np.floor_divide	Floor division(e.g., 3 // 2 = 1)
**	np.power	Exponentiation(e.g., 2 ** 3 = 8)
%	np.mod	Mudulus/remainder(e.g., 9 % 4 = 1)

In [79]:

```
np.add(a,10)
```

Out[79]:

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

In [84]:

```
a = np.linspace(0, np.pi, 5)
b = np.sin(a)
b
```

Out[84]:

```
array([0.00000000e+00, 7.07106781e-01, 1.00000000e+00, 7.07106781e-01,
       1.22464680e-16])
```

8 统计类型

In [101]:

```
# 求和
print(sum([1, 2, 3, 4, 5, 6]))

# 数组一维求和
a = np.full(10, 2.3)
print(sum(a))

# 数组多维求和
a = np.array([[1, 2], [2, 3]])
print(sum(a))

# np.sum求和
np.sum(a)
np.sum(a, axis = 1)

np.max(a, axis = 0)
```

```
21
23.000000000000004
[3 5]
```

Out[101]:

```
array([2, 3])
```

In [104]:

```
n = np.random.random(1000)
```

9 notebook 使用小技巧

In [105]:

```
# np.sum 执行效率高
%timeit sum(n)
```

```
339 µs ± 8.18 µs per loop (mean ± std. dev. of 7 runs, 1000 loops each)
```

In [106]:

```
%timeit np.sum(n)
```

```
7.41 µs ± 177 ns per loop (mean ± std. dev. of 7 runs, 100000 loops each)
```

10 比较

In [110]:

```
a = np.array(range(10))
a
```

Out[110]:

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

In [109]:

```
a > 3
```

Out[109]:

```
array([False, False, False, False,  True,  True,  True,  True,  True,
        True])
```

In [111]:

```
a != 3
```

Out[111]:

```
array([ True,  True,  True, False,  True,  True,  True,  True,  True,
        True])
```

In [113]:

```
a == a
```

Out[113]:

```
array([ True,  True,  True,  True,  True,  True,  True,  True,  True,
        True])
```

In [114]:

```
np.all(a > -1)
```

Out[114]:

```
True
```

In [115]:

```
np.any(a > -a)
```

Out[115]:

```
True
```

11 变形

In [118]:

```
a = np.full((2, 10), 1, dtype = float)
a
```

Out[118]:

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

In [120]:

```
a.reshape(10,2)
a.reshape(4, 5)
```

Out[120]:

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

12 排序

In [138]:

```
l = [
    [1, 2, 3],
    [45, 6, 34],
    [28, 3, 68]
]
a = np.array(l)
a
```

Out[138]:

```
array([[ 1,  2,  3],
       [45,  6, 34],
       [28,  3, 68]])
```

In [139]:

```
np.sort(a)
```

Out[139]:

```
array([[ 1,  2,  3],
       [ 6, 34, 45],
       [ 3, 28, 68]])
```

In [144]:

```
a.sort(axis = 1)
a
```

Out[144]:

```
array([[ 1,  2,  3],
       [ 3, 28, 34],
       [ 6, 45, 68]])
```

13 拼接

In [158]:

```
a = np.array([1, 2, 3])
b = np.array([[0, 2, 4], [1, 3, 5]])
np.concatenate([b, b, b], axis = 0)
```

Out[158]:

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

In [159]:

```
np.concatenate([b, b, b], axis = 1)
```

Out[159]:

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