Quick Intro

Python 数据类型 - List

In [1]:
```
a = [1, 2, 3]
print(type(a))            # Prints "<class 'list'>"
print(len(a))            # Prints "(3,)"
print(a[0], a[1], a[2])   # Prints "1 2 3"
a[0] = 5                  # Change an element of the array
print(a)                  # Prints "[5, 2, 3]"
```

```
<class 'list'>
3
1 2 3
[5, 2, 3]
```

Numpy 数据类型 - ndarray

In [2]:
```
import numpy as np

a = np.array([1, 2, 3])   # Create a 1d array
print(type(a))            # Prints "<class 'numpy.ndarray'>"
print(a.shape)            # Prints "(3,)"
print(a[0], a[1], a[2])   # Prints "1 2 3"
a[0] = 5                  # Change an element of the array
print(a)                  # Prints "[5, 2, 3]"

b = np.array([[1,2,3],[4,5,6]])    # Create a 2d array
print(b.shape)                     # Prints "(2, 3)"
print(b[0, 0], b[0, 1], b[1, 0])   # Prints "1 2 4"
```

```
<class 'numpy.ndarray'>
(3,)
1 2 3
[5 2 3]
(2, 3)
1 2 4
```

创建数组

手动创建数组

1D 1维数组

In [3]:
```
a = np.array([0, 1, 2, 3])
print(a)
print(type(a))
print(len(a))
print(a.ndim)
print(a.shape)
```

```
[0 1 2 3]
<class 'numpy.ndarray'>
4
1
(4,)
```

nD, n>1 多维数组

In [4]:
```
b = np.array([[0, 1, 2], [3, 4, 5]])    # 2 x 3 array

print(b)
print(type(b))
print(len(b)) # returns the size of the first dimension
print(b.ndim)
print(b.shape)
```

```
[[0 1 2]
 [3 4 5]]
<class 'numpy.ndarray'>
2
2
(2, 3)
```

In [5]:
```
c = np.array([[[1], [2]], [[3], [4]]])
print(c)
print(c.shape)
```

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

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

使用函数创建数组

等间距的数组

In [6]:
```
a = np.arange(10) # 0 .. n-1  (!)
a
```

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

In [7]:
```
b = np.arange(1, 9, 2) # start, end (exclusive), step
b
```

Out[7]:
```
array([1, 3, 5, 7])
```

指定元素个数的数组

In [8]:
```
c = np.linspace(0, 1, 6)   # start, end, num-points
c
```

Out[8]:
```
array([ 0. ,  0.2,  0.4,  0.6,  0.8,  1. ])
```

In [9]:
```
d = np.linspace(0, 1, 5, endpoint=False)
d
```

Out[9]:
```
array([ 0. ,  0.2,  0.4,  0.6,  0.8])
```

获得帮助

In [10]:
```
np.linspace?
```

In [11]:
```
a = np.ones((3, 3))  # reminder: (3, 3) is a tuple
a
```

Out[11]:
```
array([[ 1.,  1.,  1.],
       [ 1.,  1.,  1.],
       [ 1.,  1.,  1.]])
```

In [12]:
```
b = np.zeros((2, 2))
b
```

Out[12]:
```
array([[ 0.,  0.],
       [ 0.,  0.]])
```

In [13]:
```
c = np.eye(3)
c
```

Out[13]:
```
array([[ 1.,  0.,  0.],
       [ 0.,  1.,  0.],
       [ 0.,  0.,  1.]])
```

In [14]:
```
d = np.diag(np.array([1, 2, 3, 4]))
d
```

Out[14]:
```
array([[1, 0, 0, 0],
       [0, 2, 0, 0],
       [0, 0, 3, 0],
       [0, 0, 0, 4]])
```

In [ ]:
```

```

常见数据类型

In [15]:
```
a = np.array([1, 2, 3])
a.dtype
```

Out[15]:
```
dtype('int32')
```

In [16]:
```
b = np.array([1., 2., 3.])
b.dtype
```

Out[16]:
```
dtype('float64')
```

指定使用某种数据类型

In [17]:
```
c = np.array([1, 2, 3], dtype=float)
c.dtype
```

Out[17]:
```
dtype('float64')
```

默认数据类型为**float**

In [18]:
```
a = np.ones((3, 3))
a.dtype
```

Out[18]:
```
dtype('float64')
```

复数, **j**而不是**i**是工程上的惯例

In [19]:
```
d = np.array([1+2j, 3+4j, 5+6*1j])
d.dtype
```

Out[19]:
```
dtype('complex128')
```

布尔

In [20]:
```
e = np.array([True, False, False, True])
e.dtype
```

Out[20]:
```
dtype('bool')
```

字符串

In [21]:
```
f = np.array(['Bonjour', 'Hello', 'Hallo',])
f.dtype     # <--- strings containing max. 7 letters
```

Out[21]:
```
dtype('<U7')
```

数组的索引与切割

索引

In [22]:
```
a = np.arange(10)
a, a[0], a[2], a[-1]
```

Out[22]:
```
(array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9]), 0, 2, 9)
```

In [23]:
```
a[::-1]
```

Out[23]:
```
array([9, 8, 7, 6, 5, 4, 3, 2, 1, 0])
```

In [24]:
```
a = np.diag(np.arange(3))
a
```

Out[24]:
```
array([[0, 0, 0],
       [0, 1, 0],
       [0, 0, 2]])
```

In [25]:
```
a[1, 1]
```

Out[25]:
```
1
```

In [26]:
```
a[2, 1] = 10 # third line, second column
a
```

Out[26]:
```
array([[ 0,  0,  0],
       [ 0,  1,  0],
       [ 0, 10,  2]])
```

In [27]:
```
a[1]
```

Out[27]:
```
array([0, 1, 0])
```

切割

In [28]:
```
a = np.arange(10)
a

a[2:9:3] # [start:end:step]
```

Out[28]:
```
array([2, 5, 8])
```

和**python**一样, 最后的索引是不包含在切割出的数组中的

In [29]:
```
a[:4]
```

Out[29]:
```
array([0, 1, 2, 3])
```

by default, start is 0, end is the last and step is 1

In [30]:
```
a[1:3], a[::2], a[3:]
```

Out[30]:
```
(array([1, 2]), array([0, 2, 4, 6, 8]), array([3, 4, 5, 6, 7, 8, 9]))
```

复杂操作

In [31]:
```
a = np.arange(10)
a[5:] = 10
a

b = np.arange(5)
a[5:] = b[::-1]
a
```

Out[31]:
```
array([0, 1, 2, 3, 4, 4, 3, 2, 1, 0])
```

图示

In [32]:
```
np.arange(6) + np.arange(0, 51, 10)[:, np.newaxis]
```

Out[32]:
```
array([[ 0,  1,  2,  3,  4,  5],
       [10, 11, 12, 13, 14, 15],
       [20, 21, 22, 23, 24, 25],
       [30, 31, 32, 33, 34, 35],
       [40, 41, 42, 43, 44, 45],
       [50, 51, 52, 53, 54, 55]])
```

A small illustrated summary of NumPy indexing and slicing…

Fancy Indexing

Boolean Masking

In [33]:
```
np.random.seed(3)
a = np.random.randint(0, 21, 15)
a
```

Out[33]:
```
array([10,  3,  8,  0, 19, 10, 11,  9, 10,  6,  0, 20, 12,  7, 14])
```

In [34]:
```
a % 3 == 0
```

Out[34]:
```
array([False,  True, False,  True, False, False, False,  True, False,
        True,  True, False,  True, False, False], dtype=bool)
```

In [35]:
```
mask = (a % 3 == 0)
a[mask] # or,  a[a%3==0], extract a sub-array with the mask
```

Out[35]:
```
array([ 3,  0,  9,  6,  0, 12])
```

使用数组索引数组

In [36]:
```
a = np.arange(0, 100, 10)
a
```

Out[36]:
```
array([ 0, 10, 20, 30, 40, 50, 60, 70, 80, 90])
```

In [37]:
```
a[[2, 3, 2, 4, 2]]
```

Out[37]:
```
array([20, 30, 20, 40, 20])
```

In [38]:
```
a[[9, 7]] = -100
a
```

Out[38]:
```
array([   0,   10,   20,   30,   40,   50,   60, -100,   80, -100])
```

Fancy Indexing

In [ ]:
```
```