自动生成数组

In [2]:

```
import numpy as np
np.arange(0,1,0.1)#指定开始值，终值和步长创建等差数列一维数组
```

Out[2]:

```
array([ 0. ,  0.1,  0.2,  0.3,  0.4,  0.5,  0.6,  0.7,  0.8,  0.9])
```

In [3]:

```
np.linspace(0,1,10)#linspace()通过指定开始值，终值和元素个数表示等差数列的一维数组,步长为1/9
```

Out[3]:

```
array([ 0.        ,  0.11111111,  0.22222222,  0.33333333,  0.4444444
4,
        0.55555556,  0.66666667,  0.77777778,  0.88888889,  1.
])
```

In [4]:

```
np.linspace(0,1,10,endpoint=False)#endpoint会改变等差数列的步长，步长为1/10
```

Out[4]:

```
array([ 0. ,  0.1,  0.2,  0.3,  0.4,  0.5,  0.6,  0.7,  0.8,  0.9])
```

In [5]:

```
np.linspace(0,1,10,endpoint=True)#endpoint为True时，包含终值，为false时不包含终值
```

Out[5]:

```
array([ 0.        ,  0.11111111,  0.22222222,  0.33333333,  0.4444444
4,
        0.55555556,  0.66666667,  0.77777778,  0.88888889,  1.
])
```

In [6]:

```
np.logspace(0,2,5)#logspace()创建等比数列。产生10^0到10^2，有5个元素的等比数列
```

Out[6]:

```
array([   1.        ,    3.16227766,   10.        ,   31.6227766 ,  10
0.        ])
```

In [12]:

```
np.logspace(0,1,12,base=2,endpoint=False)#base参数指定，默认为10
```

Out[12]:

```
array([ 1.        ,  1.05946309,  1.12246205,  1.18920712,  1.2599210
5,
        1.33483985,  1.41421356,  1.49830708,  1.58740105,  1.6817928
3,
        1.78179744,  1.88774863])
```

In [14]:

```
np.logspace(0,1,10,base=2,endpoint=True)
```

Out[14]:

```
array([ 1.        ,  1.08005974,  1.16652904,  1.25992105,  1.36079
,
        1.46973449,  1.58740105,  1.71448797,  1.85174942,  2.
])
```

In [15]:

```
np.logspace(0,1,10,base=2)
```

Out[15]:

```
array([ 1.        ,  1.08005974,  1.16652904,  1.25992105,  1.36079
,
        1.46973449,  1.58740105,  1.71448797,  1.85174942,  2.
])
```

In [16]:

```
np.empty((2,3),np.int)
```

Out[16]:

```
array([[46763120,        0,        0],
       [       0,        0,        0]])
```

In [17]:

```
np.zeros(4,np.int)#元素初始化为0
```

Out[17]:

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

In [19]:

```
np.ones(4,np.int)#元素初始化为1
```

Out[19]:

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

In [20]:

```
np.ones(5)
```

Out[20]:

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

In [21]:

```
np.full(4,np.pi)#full()将数组元素初始化为指定的值
```

Out[21]:

```
array([ 3.14159265,  3.14159265,  3.14159265,  3.14159265])
```

In [23]:

```
s="abcdefgh"
```

In [24]:

```
np.fromstring(s,dtype=np.int8)
```

Out[24]:

```
array([ 97,  98,  99, 100, 101, 102, 103, 104], dtype=int8)
```

In [28]:

```
print (98*256+97)
```

```
25185
```

In [29]:

```
np.fromstring(s,dtype=np.int16)
```

Out[29]:

```
array([25185, 25699, 26213, 26727], dtype=int16)
```

In [31]:

```
np.fromstring(s,dtype=np.float)
```

Out[31]:

```
array([  8.54088322e+194])
```

In [32]:

```
def func(i):
    return i%4+1
np.fromfunction(func,(10,))
```

Out[32]:

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

In [67]:

```
def func2(i,j):
    return(i+1)*(j+1)
np.fromfunction(func2,(9,9))
```

Out[67]:

```
array([[  1.,   2.,   3.,   4.,   5.,   6.,   7.,   8.,   9.],
       [  2.,   4.,   6.,   8.,  10.,  12.,  14.,  16.,  18.],
       [  3.,   6.,   9.,  12.,  15.,  18.,  21.,  24.,  27.],
       [  4.,   8.,  12.,  16.,  20.,  24.,  28.,  32.,  36.],
       [  5.,  10.,  15.,  20.,  25.,  30.,  35.,  40.,  45.],
       [  6.,  12.,  18.,  24.,  30.,  36.,  42.,  48.,  54.],
       [  7.,  14.,  21.,  28.,  35.,  42.,  49.,  56.,  63.],
       [  8.,  16.,  24.,  32.,  40.,  48.,  56.,  64.,  72.],
       [  9.,  18.,  27.,  36.,  45.,  54.,  63.,  72.,  81.]])
```

存取元素

In [3]:

```
import numpy as np
a=np.arange(10)
a
```

Out[3]:

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

In [4]:

```
a[5]
```

Out[4]:

```
5
```

In [6]:

```
a[3:5]#包括a[3],但不包括a[5]
```

Out[6]:

```
array([3, 4])
```

In [7]:

```
a[:4]
```

Out[7]:

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

In [8]:

```
a[:-1]#数组最后往前数
```

Out[8]:

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

In [9]:

```
a[:-5]
```

Out[9]:

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

In [10]:

```
a[1:-1:2]#第三个元素表示步长，隔两个元素取一个
```

Out[10]:

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

In [11]:

```
a[::-1]
```

Out[11]:

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

In [12]:

```
a[6:2:-2]
```

Out[12]:

```
array([6, 4])
```

In [14]:

```
a[2:5]=100,200,300
```

In [15]:

```
a
```

Out[15]:

```
array([  0,   1, 100, 200, 300,   5,   6,   7,   8,   9])
```

In [16]:

```
b=a[2:4]
```

In [17]:

```
b
```

Out[17]:

```
array([100, 200])
```

In [18]:

```
b[0]=10#通过切片获得一个新数组是原来数组的一个视图，共享一个数据储存空间
a
```

Out[18]:

```
array([  0,   1,  10, 200, 300,   5,   6,   7,   8,   9])
```

In [19]:

```
x=np.arange(10,1,-1)
x
```

Out[19]:

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

In [20]:

```
a=x[[3,3,1,8]]
```

In [21]:

```
a
```

Out[21]:

```
array([7, 7, 9, 2])
```

In [22]:

```
b=x[[1,3,-6,-2]]#使用列表作为下标，数组不和原始数组共享，x与b不共享内存
b
```

Out[22]:

```
array([9, 7, 7, 3])
```

In [24]:

```
a[3]=100
a
```

Out[24]:

```
array([  7,   7,   9, 100])
```

In [25]:

```
x
```

Out[25]:

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

In [27]:

```
c=x[1:2]#共享内存
c
```

Out[27]:

```
array([9])
```

In [29]:

```
x=np.arange(10,1,-1)
x
```

Out[29]:

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

In [32]:

```
x[np.array([3,3,1,8])]
```

Out[32]:

```
array([7, 7, 9, 2])
```

In [33]:

```
x[np.array([[1,1,2],[3,3,3]])]
```

Out[33]:

```
array([[9, 9, 8],
       [7, 7, 7]])
```

In [34]:

```
x[[1,2,2,2,2,2]].reshape(2,3)
```

Out[34]:

```
array([[9, 8, 8],
       [8, 8, 8]])
```

In [42]:

```
x=np.arange(1,10,1)
x
```

Out[42]:

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

In [43]:

```
x[[2,2,2,2,4,4,4,4]].reshape(4,2)
```

Out[43]:

```
array([[3, 3],
       [3, 3],
       [5, 5],
       [5, 5]])
```

In [44]:

```
x[1:5].reshape(2,2)
```

Out[44]:

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

In [45]:

```
x=np.arange(5,0,-1)
```

In [46]:

```
x
```

Out[46]:

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

In [50]:

```
x[np.array([True,True,False,False,True])]#使用布尔数组作为下表存取x中的元素时，将获得数组x
```

Out[50]:

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

In [51]:

```
x[[True,True,True,False,False]]#布尔数组，布尔列表都可以
```

Out[51]:

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

In [52]:

```
x[[True]]
```

```
---------------------------------------------------------------------------
IndexError                                Traceback (most recent call
 last)
<ipython-input-52-ac0216ffb3fc> in <module>()
----> 1 x[[True]]

IndexError: boolean index did not match indexed array along dimension
 0; dimension is 5 but corresponding boolean dimension is 1
```

In [54]:

```
x[np.array([True,False,True,True])]#布尔数组长度不够
```

```
---------------------------------------------------------------------------
IndexError                                Traceback (most recent call
 last)
<ipython-input-54-254d618bdbe5> in <module>()
----> 1 x[np.array([True,False,True,True])]

IndexError: boolean index did not match indexed array along dimension
 0; dimension is 5 but corresponding boolean dimension is 4
```

In [56]:

```
x[np.array([True,True,False,True,False])]=11,11,11#true对应元素改变
```

In [57]:

```
x
```

Out[57]:

```
array([11, 11,  3, 11,  1])
```

In [58]:

```
x=np.random.randint(0,10,6)#产生一个长度为6，元素为0到9的随机数组
x
```

Out[58]:

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

In [59]:

```
x>5
```

Out[59]:

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

In [60]:

```
x[x>5]
```

Out[60]:

```
array([7, 8, 6])
```

In [64]:

```
x=np.arange(1,10,1)
```

In [65]:

```
x
```

Out[65]:

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

In [66]:

```
x>6
```

Out[66]:

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

多维数组

In [68]:

```
a=np.arange(0,60,10).reshape(-1,1)+np.arange(0,6)
a
```

Out[68]:

```
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]])
```

In [69]:

```
a[0,3:5]
```

Out[69]:

```
array([3, 4])
```

In [70]:

```
a[2,2:4]
```

Out[70]:

```
array([22, 23])
```

In [71]:

```
a[4:,4:]
```

Out[71]:

```
array([[44, 45],
       [54, 55]])
```

In [72]:

```
a[:,2]
```

Out[72]:

```
array([ 2, 12, 22, 32, 42, 52])
```

In [73]:

```
a[2::2,::2]
```

Out[73]:

```
array([[20, 22, 24],
       [40, 42, 44]])
```

In [74]:

```
b=a[0,3:5]#ab共享数据
b
```

Out[74]:

```
array([3, 4])
```

In [75]:

```
b[0]=-b[0]
```

In [76]:

```
b
```

Out[76]:

```
array([-3,  4])
```

In [77]:

```
a
```

Out[77]:

```
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]])
```

In [78]:

```
idx=slice(None,None,2),slice(2,None)#a[idx]与a[::2,2:]相同
```

In [79]:

```
a[idx]
```

Out[79]:

```
array([[ 2, -3,  4,  5],
       [22, 23, 24, 25],
       [42, 43, 44, 45]])
```

In [80]:

```
a[idx][idx]
```

Out[80]:

```
array([[ 4,  5],
       [44, 45]])
```