numpy 练习题

numpy 的array操作

1.导入numpy库

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

2.建立一个一维数组 a 初始化为[4,5,6], (1)输出a 的类型（type）(2)输出a 的各维度的大小（shape）(3)输出 a的第一个元素（值为4）

In [2]:
```
a = np.array([4,5,6])
print(type(a))
print(a.shape)
print(a[0])
```

```
<class 'numpy.ndarray'>
(3,)
4
```

3.建立一个二维数组 b,初始化为 [[4, 5, 6],[1, 2, 3]] (1)输出各维度的大小（shape）(2)输出 b(0,0)， b(0,1),b(1,1) 这三个元素（对应值分别为4,5,2）

In [3]:
```
#二维数组写法
b = np.array([[4,5,6],[1,2,3]])
print(b.shape)
print(b[0][0],b[0][1],b[1][1])
```

```
(2, 3)
4 5 2
```

4. (1)建立一个全0矩阵 a, 大小为 3x3; 类型为整型（提示: dtype = int）(2)建立一个全1矩阵b,大小为4x5; (3)建立一个单位矩阵c ,大小为4x4; (4)生成一个随机数矩阵d,大小为 3x2.

In [4]:
```
a = np.zeros((3,3),dtype=int)
b = np.ones((4,5))
c = np.eye(4)
#随机
d = np.random.random((3,2))
print(a,'\n',b,'\n',c,'\n',d)
```

```
[[0 0 0]
 [0 0 0]
 [0 0 0]]
 [[1. 1. 1. 1. 1.]
 [1. 1. 1. 1. 1.]
 [1. 1. 1. 1. 1.]
 [1. 1. 1. 1. 1.]]
 [[1. 0. 0. 0.]
 [0. 1. 0. 0.]
 [0. 0. 1. 0.]
 [0. 0. 0. 1.]]
 [[0.05728998 0.33450443]
 [0.12143767 0.68020134]
 [0.65187068 0.81334452]]
```

5. 建立一个数组 a,(值为[[1, 2, 3, 4], [5, 6, 7, 8], [9, 10, 11, 12]]) ,(1)打印a; (2)输出 下标为(2,3),(0,0) 这两个数组元素的值

In [5]:
```
a = np.array([[1,2,3,4],[5,6,7,8],[9,10,11,12]])
print(a)
print(a[2,3],a[0,0])
```

```
[[ 1  2  3  4]
 [ 5  6  7  8]
 [ 9 10 11 12]]
12 1
```

6.把上一题的 a数组的 0到1行 2到3列，放到b里面去，（此处不需要从新建立a,直接调用即可）(1),输出b;(2) 输出b 的（0,0）这个元素的值

In [6]:
```
b = a[0:2,2:]
print(b)
print(b[0,0])
```

```
[[3 4]
 [7 8]]
3
```

7. 把第5题中数组a的最后两行所有元素放到 c中，（提示： a[1:2, :]）(1)输出 c ; (2) 输出 c 中第一行的最后一个元素（提示，使用 -1 表示最后一个元素）

In [7]:
```
c = a[1:,:]
print(c)
print(c[1,-1])
```

```
[[ 5  6  7  8]
 [ 9 10 11 12]]
12
```

8.建立数组a,初始化a为[[1, 2], [3, 4], [5, 6]]，输出 （0,0） （1,1） （2,0）这三个元素（提示： 使用 print(a[[0, 1, 2], [0, 1, 0]]) ）

In [8]:
```
a=np.array([[1, 2], [3, 4], [5, 6]])
#第一个[]里面是行标，第二个[]里面是列标
print(a[[0,1,2],[0,1,0]])
```

```
[1 4 5]
```

9.建立矩阵a ,初始化为[[1, 2, 3], [4, 5, 6], [7, 8, 9], [10, 11, 12]]，输出(0,0),(1,2),(2,0),(3,1) (提示使用 b = np.array([0, 2, 0, 1]) print(a[np.arange(4), b]))

In [9]:
```
a = np.array([[1, 2, 3], [4, 5, 6], [7, 8, 9], [10, 11, 12]])
b = np.array([0,2,0,1])
#np.arange(st,ed,step)
print(a[np.arange(4),b])
```

```
[ 1  6  7 11]
```

10.对9 中输出的那四个元素，每个都加上10，然后重新输出矩阵a.(提示: a[np.arange(4), b] += 10)

In [10]:
```
a[np.arange(4),b]+=10
print(a)
```

```
[[11  2  3]
 [ 4  5 16]
 [17  8  9]
 [10 21 12]]
```

array 的数学运算

11. 执行 x = np.array([1, 2])，然后输出 x 的数据类型

In [11]:
```
x = np.array([1,2])
print(x.dtype)
```

```
int32
```

12.执行 x = np.array([1.0, 2.0])， 然后输出 x 的数据类类型

In [12]:
```
x = np.array([1.0, 2.0])
print(x.dtype)
```

```
float64
```

13.执行 x = np.array([[1, 2], [3, 4]], dtype=np.float64)， y = np.array([[5, 6], [7, 8]], dtype=np.float64)，然后输出 x+y ,和 np.add(x,y)

In [13]:
```
x = np.array([[1,2],[3,4]],dtype=np.float64)
y = np.array([[5,6],[7,8]],dtype=np.float64)
print(x+y)
print(np.add(x,y))
```

```
[[ 6.  8.]
 [10. 12.]]
[[ 6.  8.]
 [10. 12.]]
```

14. 利用 13题目中的x,y 输出 x-y 和 np.subtract(x,y)

In [14]:
```
print(x-y)
print(np.subtract(x,y))
```

```
[[-4. -4.]
 [-4. -4.]]
[[-4. -4.]
 [-4. -4.]]
```

15. 利用13题目中的x，y 输出 x*y ,和 np.multiply(x, y) 还有 np.dot(x,y),比较差异。然后自己换一个不是方阵的试试。

In [15]:
```
print(x*y)
print(np.multiply(x,y))
#矩阵乘法
print(np.dot(x,y))
```

```
[[ 5. 12.]
 [21. 32.]]
[[ 5. 12.]
 [21. 32.]]
[[19. 22.]
 [43. 50.]]
```

16. 利用13题目中的x,y,输出 x / y .(提示： 使用函数 np.divide())

In [16]:
```
print(x/y)
print(np.divide(x,y))
```

```
[[0.2        0.33333333]
 [0.42857143 0.5       ]]
[[0.2        0.33333333]
 [0.42857143 0.5       ]]
```

17. 利用13题目中的x,输出 x的 开方。(提示： 使用函数 np.sqrt())

In [17]:
```
print(np.sqrt(x))
```

```
[[1.         1.41421356]
 [1.73205081 2.        ]]
```

18.利用13题目中的x,y ,执行 print(x.dot(y)) 和 print(np.dot(x,y))

In [18]:
```
print(x.dot(y))
print(np.dot(x,y))
```

```
[[19. 22.]
 [43. 50.]]
[[19. 22.]
 [43. 50.]]
```

19.利用13题目中的 x,进行求和。提示：输出三种求和 (1)print(np.sum(x)): (2)print(np.sum(x， axis =0)); (3)print(np.sum(x,axis = 1))

In [19]:
```
print(np.sum(x))
#按列求和
print(np.sum(x,axis=0))
#按行求和
print(np.sum(x,axis=1))
```

```
10.0
[4. 6.]
[3. 7.]
```

20.利用13题目中的 x,进行求平均数（提示：输出三种平均数(1)print(np.mean(x)) (2)print(np.mean(x,axis = 0))(3) print(np.mean(x,axis =1)))

In [20]:
```
print(np.mean(x))
print(np.mean(x,axis=0))
print(np.mean(x,axis=1))
```

```
2.5
[2. 3.]
[1.5 3.5]
```

21.利用13题目中的x，对x 进行矩阵转置，然后输出转置后的结果，（提示： x.T 表示对 x 的转置）

In [21]:
```
print(x.T)
```

```
[[1. 3.]
 [2. 4.]]
```

22.利用13题目中的x,求e的指数（提示： 函数 np.exp()）

In [22]:
```
print(np.exp(x))
```

```
[[ 2.71828183  7.3890561 ]
 [20.08553692 54.59815003]]
```

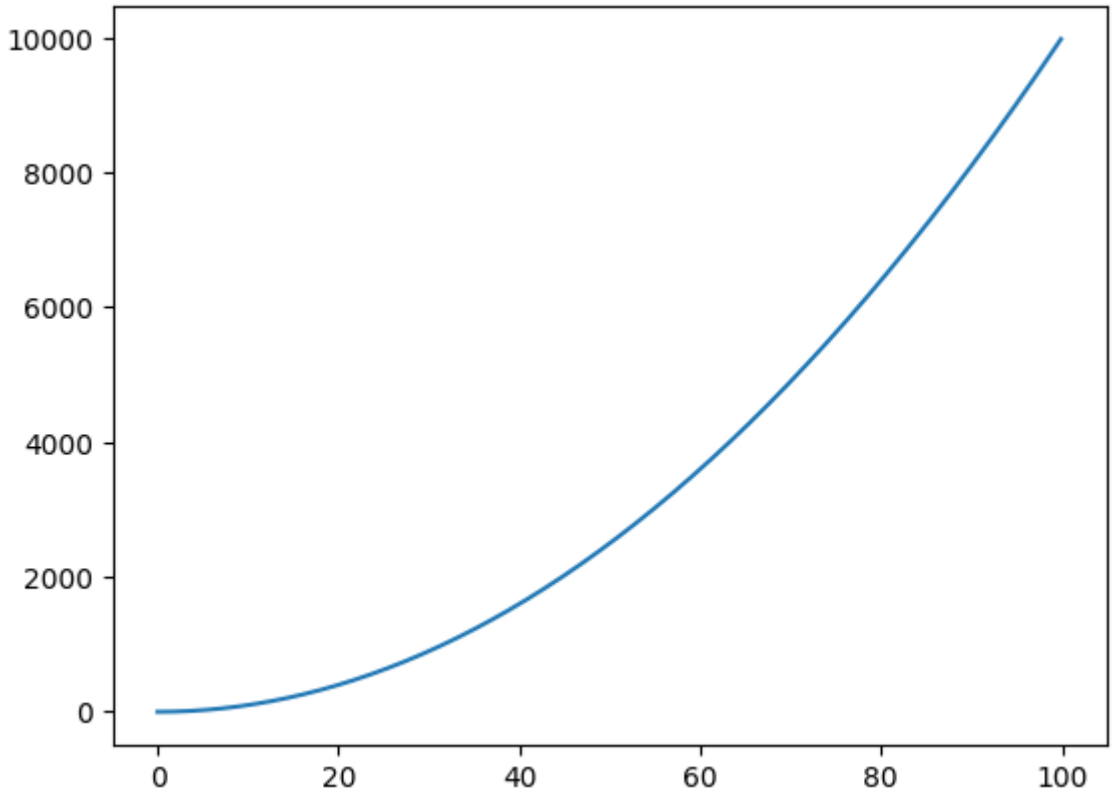
23.利用13题目中的 x,求值最大的下标（提示(1)print(np.argmax(x)) ,(2) print(np.argmax(x, axis =0))(3)print(np.argmax(x),axis =1))

In [23]:
```
print(np.argmax(x))
print(np.argmax(x,axis=0))
print(np.argmax(x,axis=1))
```

```
3
[1 1]
[1 1]
```

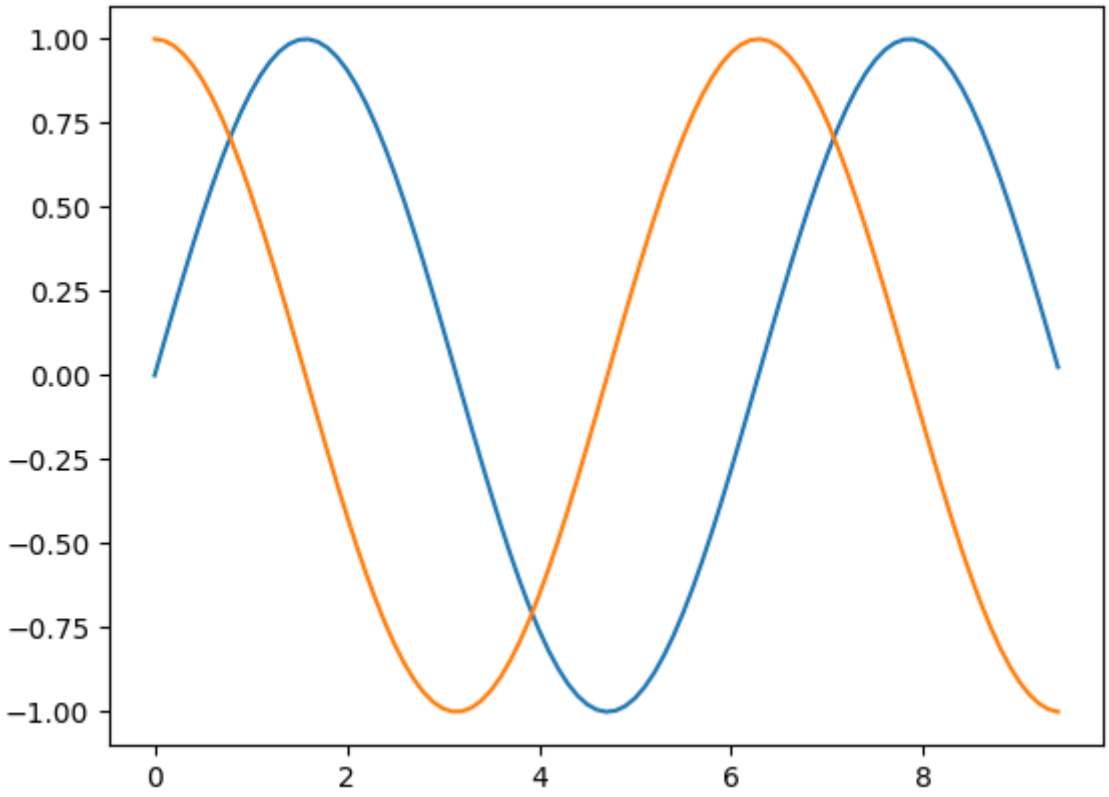
24,画图，y=x*x 其中 x = np.arange(0, 100, 0.1) （提示这里用到 matplotlib.pyplot 库）

In [24]:
```
import matplotlib.pyplot as pt
x = np.arange(0,100,0.1)
y = x*x
pt.plot(x,y)
pt.show()
```



25.画图。画正弦函数和余弦函数， x = np.arange(0, 3 * np.pi, 0.1)(提示：这里用到 np.sin() np.cos() 函数和 matplotlib.pyplot 库)

In [25]:
```
x = np.arange(0,3*np.pi,0.1)
y1 = np.sin(x)
y2 = np.cos(x)
pt.plot(x,y1)
pt.plot(x,y2)
pt.show()
```


1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
0
2
4
6
8