这里直接把三次作业分开三次输出

```python
import numpy as np

arr = np.random.randn(10,10)
print("1.元素服从标准正态分布：\n",arr)

arr=arr.astype(np.int8).T
print("2.数据类型转换并数组转置：\n",arr)

print("3.数据类型：",arr.dtype)

print("4.内存占用：",arr.size*arr.itemsize)

print("5.所有偶数存入一维数组：\n",arr.astype(np.int8)
[arr.astype(np.int8)%2==0])
```

```
1.元素服从标准正态分布：
 [[-1.71640705 -1.36375205  0.77709148  1.04075793 -0.52093443  0.21161555
  -0.11638943  0.28176402  0.37811038 -1.2718057 ]
 [ 0.18746565 -0.97311199 -2.09026807 -0.91119429  1.03519211  0.7303442
  -1.40079319  0.02218778  0.71517308  1.64133344]
 [-0.58830784  0.43125445  0.75946527 -1.27104082  0.57487741  0.82095592
   1.44739542 -1.25076367  0.31567356 -0.69484053]
 [ 0.89290329  1.02965755  0.89564722  0.53113981  0.30807379 -0.12903408
  -1.16215184  0.73628703  0.34884545  0.3414488 ]
 [ 0.91627137  0.40097505  0.97322833 -0.0569555   0.01927548 -0.30387167
  -1.3385821   0.54451452 -0.07343168  1.90051621]
 [-1.21277272  0.82106426 -0.95609444  1.44482761  0.35887744  0.35165501
  -0.33904677 -1.46480276  3.51241866 -0.38326833]
 [ 0.03304051  0.10397387  2.50108342  0.24948298 -1.17927873 -1.88231365
   0.45506379  1.41259587  1.33781204 -1.55848384]
 [ 0.85602009  0.22129632 -0.18355505 -0.54972469 -0.56829716 -0.25588674
  -0.52809431  0.57110728 -0.84124345  0.7791134 ]
 [ 1.47446956  1.98604258  1.57503125  0.55961043  0.10056909 -0.31016322
   0.76837805  0.1706068  -0.46985835  0.4484869 ]
 [ 1.01718764  0.44048646  0.39262661 -1.74245919 -1.0032485   1.3440442
  -0.97858516 -2.07665849  0.41512624 -0.46785088]]
2.数据类型转换并数组转置：
 [[-1  0  0  0  0 -1  0  0  1  1]
 [-1  0  0  1  0  0  0  0  1  0]
 [ 0 -2  0  0  0  0  2  0  1  0]
 [ 1  0 -1  0  0  1  0  0  0 -1]
 [ 0  1  0  0  0  0 -1  0  0 -1]
 [ 0  0  0  0  0  0 -1  0  0  1]
 [ 0 -1  1 -1 -1  0  0  0  0  0]
 [ 0  0 -1  0  0 -1  1  0  0 -2]
 [ 0  0  0  0  0  3  1  0  0  0]
 [-1  1  0  0  1  0 -1  0  0  0]]
3.数据类型： int8
4.内存占用： 100
5.所有偶数存入一维数组：
 [ 0  0  0  0  0  0  0  0  0  0  0  0  0  0 -2  0  0  0  0  2  0  0  0  0
  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
```

```
  0  0  0  0  0  0  0 -2  0  0  0  0  0  0  0  0  0  0  0  0  0]
```

```python
from numpy import mat

matr = mat(np.eye(5)*3)
print("1.主对角线为3的5*5矩阵matr:\n",matr)

matr1=matr[:,[1,0,2,3,4]]
print("2.交换第一列与第二列,构成matr1:\n",matr1)

matr2=matr[[0,2,1,3,4],:]
print("3.交换第二行与第三行,构成matr2:\n",matr2)

print("4.是否有元素不同：",len(np.argwhere(matr1!=matr2))!=0,"\n  不同元素的个数：",len(np.argwhere(matr1!=matr2)),"\n  不同元素的位置：\n",np.argwhere(matr1!=matr2))

print("5.matr1与matr2相乘:\n",matr1*matr2)

print("6.matr1与matr2对应元素相乘:\n",np.multiply(matr1,matr2))

print("7.matr1与matr2按行拼接:\n",np.hstack((matr1,matr2)))

```

```
1.主对角线为3的5*5矩阵matr:
 [[3. 0. 0. 0. 0.]
 [0. 3. 0. 0. 0.]
 [0. 0. 3. 0. 0.]
 [0. 0. 0. 3. 0.]
 [0. 0. 0. 0. 3.]]
2.交换第一列与第二列,构成matr1:
 [[0. 3. 0. 0. 0.]
 [3. 0. 0. 0. 0.]
 [0. 0. 3. 0. 0.]
 [0. 0. 0. 3. 0.]
 [0. 0. 0. 0. 3.]]
3.交换第二行与第三行,构成matr2:
 [[3. 0. 0. 0. 0.]
 [0. 0. 3. 0. 0.]
 [0. 3. 0. 0. 0.]
 [0. 0. 0. 3. 0.]
 [0. 0. 0. 0. 3.]]
4.是否有元素不同： True
  不同元素的个数： 6
  不同元素的位置：
 [[0 0]
 [0 1]
 [1 0]
 [1 2]
 [2 1]
 [2 2]]
5.matr1与matr2相乘:
 [[0. 0. 9. 0. 0.]
 [9. 0. 0. 0. 0.]
```

```
 [0. 9. 0. 0. 0.]
 [0. 0. 0. 9. 0.]
 [0. 0. 0. 0. 9.]]
6.matr1与matr2对应元素相乘:
 [[0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0.]
 [0. 0. 0. 9. 0.]
 [0. 0. 0. 0. 9.]]
7.matr1与matr2按行拼接:
 [[0. 3. 0. 0. 0. 3. 0. 0. 0. 0.]
 [3. 0. 0. 0. 0. 0. 0. 3. 0. 0.]
 [0. 0. 3. 0. 0. 0. 3. 0. 0. 0.]
 [0. 0. 0. 3. 0. 0. 0. 0. 3. 0.]
 [0. 0. 0. 0. 3. 0. 0. 0. 0. 3.]]
```

```python
import collections

arr=np.random.randint(1,100,[6,6])
print("\n1.6行6列的二维数组arr:\n",arr)

print("\n2.每列的最大值:\n",np.max(arr,axis=0))

print("\n3.每行的最小值:\n",np.min(arr,axis=1))

print("\n4.每个元素的出现次数:\n ",collections.Counter(arr.flatten()))

print("\n5.第三行每个元素的排名:\n",np.sort(arr,axis=1)[2][:])

print("\n6.去除重复的行:\n",np.unique(arr, axis=0))

print("\n7.从第4行抽取3个元素:\n",np.random.choice(arr[3][:],3,False))

print("\n8.第5行中不含第1行数据的数据:\n",np.setdiff1d(arr[4][:],arr[4][arr[4]
[:]==arr[1][:]]))


arr2 , arr2[arr2<10]=arr.astype('float32') , float(np.nan)
print("\n9.小于10的元素修改为NaN,生成arr2:\n",arr2)


print("\n10.删除arr2中含有Nan的列:\n",arr2[:, ~np.isnan(arr2).any(axis=0)])
```

```
1.6行6列的二维数组arr:
 [[99 53 35 34 69 34]
 [50 29 58 79 57 78]
 [39 84 78 41 37 12]
 [93 64 96 17 63 71]
 [49 75 46 35 78 50]
 [39 21 72 70 11  9]]

2.每列的最大值:
 [99 84 96 79 78 78]
```

```
3.每行的最小值:
 [34 29 12 17 35  9]

4.每个元素的出现次数:
  Counter({78: 3, 35: 2, 34: 2, 50: 2, 39: 2, 99: 1, 53: 1, 69: 1, 29: 1, 58: 1, 79: 1, 57: 1, 84: 1, 41: 1, 37: 1, 12: 1, 93: 1, 64: 1, 96: 1, 17: 1, 63: 1, 71: 1, 49: 1, 75: 1, 46: 1, 21: 1, 72: 1, 70: 1, 11: 1, 9: 1})

5.第三行每个元素的排名:
 [12 37 39 41 78 84]

6.去除重复的行:
 [[39 21 72 70 11  9]
 [39 84 78 41 37 12]
 [49 75 46 35 78 50]
 [50 29 58 79 57 78]
 [93 64 96 17 63 71]
 [99 53 35 34 69 34]]

7.从第4行抽取3个元素:
 [63 96 93]

8.第5行中不含第1行数据的数据:
 [35 46 49 50 75 78]

9.小于10的元素修改为NaN,生成arr2:
 [[99. 53. 35. 34. 69. 34.]
 [50. 29. 58. 79. 57. 78.]
 [39. 84. 78. 41. 37. 12.]
 [93. 64. 96. 17. 63. 71.]
 [49. 75. 46. 35. 78. 50.]
 [39. 21. 72. 70. 11. nan]]

10.删除arr2中含有Nan的列:
 [[99. 53. 35. 34. 69.]
 [50. 29. 58. 79. 57.]
 [39. 84. 78. 41. 37.]
 [93. 64. 96. 17. 63.]
 [49. 75. 46. 35. 78.]
 [39. 21. 72. 70. 11.]]
```

```
1
```