1、如何垂直叠加两个数组（三种方法）：

```
a = np.arange(10).reshape(2,-1)
b = np.repeat(1, 10).reshape(2,-1)

# Answers
# Method 1:
np.concatenate([a, b], axis=0)

# Method 2:
np.vstack([a, b])

# Method 3:
np.r_[a, b]
# > array([[0, 1, 2, 3, 4],
# >        [5, 6, 7, 8, 9],
# >        [1, 1, 1, 1, 1],
# >        [1, 1, 1, 1, 1]])
```

2、如何水平叠加两个数组（三种方法）：

```
a = np.arange(10).reshape(2,-1)
b = np.repeat(1, 10).reshape(2,-1)

# Answers
# Method 1:
np.concatenate([a, b], axis=1)

# Method 2:
np.hstack([a, b])

# Method 3:
np.c_[a, b]
# > array([[0, 1, 2, 3, 4, 1, 1, 1, 1, 1],
# >        [5, 6, 7, 8, 9, 1, 1, 1, 1, 1]])
```

3、找两个数组的公共元素：

问题：获取数组a和数组b之间的公共项。

给定：

```
a = np.array([1,2,3,2,3,4,3,4,5,6])
b = np.array([7,2,10,2,7,4,9,4,9,8])
```

期望的输出：

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

答案：

```
a = np.array([1,2,3,2,3,4,3,4,5,6])
b = np.array([7,2,10,2,7,4,9,4,9,8])
np.intersect1d(a,b)
# > array([2, 4])
```

4、从a数组中删除和数组b公共的元素：

```
a = np.array([1,2,3,4,5])
b = np.array([5,6,7,8,9])
```

期望的输出：

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

答案：

```
a = np.array([1,2,3,4,5])
b = np.array([5,6,7,8,9])

# From 'a' remove all of 'b'
np.setdiff1d(a,b)
# > array([1, 2, 3, 4])
```

5、获取a和b元素匹配的位置：

```
a = np.array([1,2,3,2,3,4,3,4,5,6])
b = np.array([7,2,10,2,7,4,9,4,9,8])
```

期望的输出：

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

答案：

```
a = np.array([1,2,3,2,3,4,3,4,5,6])
b = np.array([7,2,10,2,7,4,9,4,9,8])

np.where(a == b)
# > (array([1, 3, 5, 7]),)
```

6、从numpy数组中提取给定范围的数字（三种方法，第三中最简单）：

```
a = np.arange(15)

# Method 1
index = np.where((a >= 5) & (a <= 10))
a[index]

# Method 2:
index = np.where(np.logical_and(a>=5, a<=10))
a[index]
# > (array([6, 9, 10]),)

# Method 3: (thanks loganzk!)
a[(a >= 5) & (a <= 10)]
```

7、用函数来处理数组来比较两个数组中对应元素的大小然后返回一个数组：

```
def maxx(x, y):
    """Get the maximum of two items"""
    if x >= y:
        return x
    else:
        return y

pair_max = np.vectorize(maxx, otypes=[float])

a = np.array([5, 7, 9, 8, 6, 4, 5])
b = np.array([6, 3, 4, 8, 9, 7, 1])

pair_max(a, b)
# > array([ 6.,  7.,  9.,  8.,  9.,  7.,  5.])
```

8、交换二维数组中的两行和两列：

```
# Input
arr = np.arange(9).reshape(3,3)
arr

# Solution
arr[:, [1,0,2]]
# > array([[1, 0, 2],
# >        [4, 3, 5],
# >        [7, 6, 8]])
```

```
# Input
arr = np.arange(9).reshape(3,3)

# Solution
arr[[1,0,2], :]
# > array([[3, 4, 5],
# >        [0, 1, 2],
# >        [6, 7, 8]])
```

9、反转二维数组的行：

```
# Input
arr = np.arange(9).reshape(3,3)
```

```
# Solution
arr[::-1]
array([[6, 7, 8],
       [3, 4, 5],
       [0, 1, 2]])
```

10、反转二维数组的列：

```
# Input
arr = np.arange(9).reshape(3,3)

# Solution
arr[:, ::-1]
# > array([[2, 1, 0],
# >        [5, 4, 3],
# >        [8, 7, 6]])
```

11、创建两个数之间的浮动的二维随机数(两种方法，第二中方法好用)：

```
# Input
arr = np.arange(9).reshape(3,3)

# Solution Method 1:
rand_arr = np.random.randint(low=5, high=10, size=(5,3)) + np.random.random
# print(rand_arr)

# Solution Method 2:
rand_arr = np.random.uniform(5,10, size=(5,3))
print(rand_arr)
# > [[ 8.50061025  9.10531502  6.85867783]
# >  [ 9.76262069  9.87717411  7.13466701]
# >  [ 7.48966403  8.33409158  6.16808631]
# >  [ 7.75010551  9.94535696  5.27373226]
# >  [ 8.0850361   5.56165518  7.31244004]]
```

如何只打印数组中小数点后三位:

答案:

```
# Input
rand_arr = np.random.random((5,3))

# Create the random array
rand_arr = np.random.random([5,3])

# Limit to 3 decimal places
np.set_printoptions(precision=3)
rand_arr[:4]
# > array([[ 0.443,  0.109,  0.97 ],
# >        [ 0.388,  0.447,  0.191],
# >        [ 0.891,  0.474,  0.212],
# >        [ 0.609,  0.518,  0.403]])
```

12、限制数组中打印的元素数:

```
a = np.arange(15)
# > array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 1
```

期望的输出：

```
# > array([ 0,  1,  2, ..., 12, 13, 14])
```

答案：

```
np.set_printoptions(threshold=6)
a = np.arange(15)
a
# > array([ 0,  1,  2, ..., 12, 13, 14])
```

13、打印完整的数组而不被截断：

```
# Input
np.set_printoptions(threshold=6)
a = np.arange(15)

# Solution
np.set_printoptions(threshold=np.nan)
a
# > array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14])
```

14、将数组转换为平面一维数组（两种方法，第二种更好用）：

第一步：

```
arr1 = np.arange(3)
arr2 = np.arange(3,7)
arr3 = np.arange(7,10)

array_of_arrays = np.array([arr1, arr2, arr3])
array_of_arrays
# > array([array([0, 1, 2]), array([3, 4, 5, 6]), array([7, 8, 9])], dtype=object)
```

第二步：

```
arr1 = np.arange(3)
arr2 = np.arange(3,7)
arr3 = np.arange(7,10)

array_of_arrays = np.array([arr1, arr2, arr3])
print('array_of_arrays: ', array_of_arrays)

# Solution 1
arr_2d = np.array([a for arr in array_of_arrays for a in arr])

# Solution 2:
arr_2d = np.concatenate(array_of_arrays)
print(arr_2d)
# > array_of_arrays:  [array([0, 1, 2]) array([3, 4, 5, 6]) array([7, 8, 9])]
# > [0 1 2 3 4 5 6 7 8 9]
```

15、numpy线性代数的一些算法：

https://docs.scipy.org/doc/numpy-1.13.0/reference/routines.dual.html#linear-algebra

https://docs.scipy.org/doc/scipy-1.2.1/reference/linalg.html

https://blog.csdn.net/xtingjie/article/details/71937687

16、