一个简单的函数

定义一组基矢量basis，空间中的点集是这组基矢量的线性组合

实现一个函数来寻找距离原点最近的前6组点，返回距离以及该距离对应的点的个数

```
import itertools

import numpy as np
import pandas as pd


def find_near_neighbor(basis):
    result = []
    values = np.arange(-3,4,1)
    for vector in itertools.product(*[values for i in range(basis.shape[0])]):
        vector = np.array(vector).reshape(1, -1)
        result.append(np.linalg.norm(vector @ basis))
    return pd.Series(result).value_counts().sort_index().iloc[:7]


sc = np.array([[1, 0, 0], [0, 1, 0], [0, 0, 1]])
find_near_neighbor(sc)
```

```
0.000000     1
1.000000     6
1.414214    12
1.732051     8
2.000000     6
2.236068    24
2.449490    24
dtype: int64
```

一些学习

1. itertools中的product可以实现多个可迭代结构的组合，类似于多维的np.meshgrid

```
for i, j, k in itertools.product([1, "a"], [2, "b"], [3, "c"]):
    print(i, j, k)
```

```
1 2 3
1 2 c
1 b 3
1 b c
a 2 3
```

```
a 2 c
a b 3
a b c
```

2. iterable for i in range(5) 和 *[iterable for i in range(5)]做参数时是不同的！

```
for i in itertools.product([1, 2] for i in range(2)):
    print(i)
print("------------")
for i in itertools.product(*[[1, 2] for i in range(2)]):
    print(i)
```

```
([1, 2],)
([1, 2],)
------------
(1, 1)
(1, 2)
(2, 1)
(2, 2)
```

3. 使用numpy做矩阵运算时一定要把一维的数组统一转换成二维的

```
print(np.array([1, 2]).T.shape)
print(np.array([1, 2]).reshape(1, -1).T.shape)
```

```
(2,)
(2, 1)
```

4. pd.Series()：

- 接收一个字典时，字典的key作为Series的index，字典的value作为对应index的value

```
pd.Series({"shen": [1, 2, 3]})
```

```
shen    [1, 2, 3]
dtype: object
```

- 接收一个列表时，只按axis=0解包一层，默认把它当成一维的列表（实际上列表也不存在维度的概念...只不过是列表中元素还是一个列表）

```
pd.Series([[1, 2, 3], [4, 5, 6]])
```

```
0    [1, 2, 3]
1    [4, 5, 6]
dtype: object
```

- 可以用name参数为Series的列命名（Series只有一个列）

```
pd.Series([1, 2, 3], name="attribute")
```

```
0    1
1    2
2    3
Name: attribute, dtype: int64
```

- Series的value_counts方法统计列中重复元素的个数，返回一个新的Series，新Series的index是原Series中不同的元素，data是该元素出现的个数

```
pd.Series([1, 1, 2, 3]).value_counts()
```

```
1    2
3    1
2    1
dtype: int64
```

- Series的sort_index方法按照index的值排序，sort_values方法按照data的值排序

```
pd.Series([1, 1, 2, 3]).value_counts().sort_index()
```

```
1    2
2    1
3    1
dtype: int64
```

- Series的rename方法改变Series的列标签（Series的name对应DataFrame的一个column的name）

```
pd.Series([1, 2, 3]).rename("attribute1")
```

```
0    1
1    2
2    3
Name: attribute1, dtype: int64
```