range（）函数

range(start,end,step)

开始，结束，步长　start的默认值是0 可缺省　step的默认值为1 也可缺省

前闭后开区间

```
for i in range(1,10,1):
    print(i)
```

```
1
2
3
4
5
6
7
8
9
```

```
for i in range(1,10,2):
    print(i)
```

```
1
3
5
7
9
```

```
for i in range(10,1,-1):
    print(i)
```

```
10
9
8
7
6
5
4
3
2
```

列表

```
a = [9,1,5,3,8]
b=a.copy()  #拷贝一个备份
```

```
a.append(10)  #append方法用来添加元素
```

```
print("a:{}".format(a))
print("b:{}".format(b))
```

```
a:[9, 1, 5, 3, 8, 10]
b:[9, 1, 5, 3, 8]
```

```
a.sort()   #.sort()进行排序  原地排序  会改变原有的数组
```

```
print("a:{}".format(a))
print("b:{}".format(b))
```

```
a:[1, 3, 5, 8, 9, 10]
b:[9, 1, 5, 3, 8]
```

字符串中的join方法

```
#join方法在N皇后中有涉及
temp =""
```

```
temp
```

```
''
```

```
str1="str"
```

```
str1
```

```
'str'
```

```
temp.join(str1)
```

```
'str'
```

```
temp
```

```
''
```

```
temp = temp.join(str1)
```

```
temp
```

```
'str'
```

类与对象

```
from typing import List
class Solution:
    def __init__(self):
        self.result = []    #定义最终的结果数组
        self.sumNow = 0     #当前所得的和
        self.path = []      #存放符合条件的叶子节点

    def backTracking(self,k:int,n:int,startIndex:int):
        if len(self.path) == k: #终止条件 遍历到叶子节点
            if self.sumNow == n:    #如果当前所得和等于输入的n 那么满足题述条件 加入到
result中
                self.result.append(self.path[:])    #加入到result中
            return
        for i in range(startIndex,10):  #横向遍历 从startIndex 遍历到9
            self.path.append(i)
            self.sumNow += i
            self.backTracking(k,n,i+1)  #再次纵向遍历
            self.path.pop() #回溯
            self.sumNow -= i #回溯

    def combinationSum3(self, k: int, n: int) -> List[List[int]]:
        # 回溯
        self.backTracking(k,n,1)
        return self.result
```