2 递归

2.1 何谓递归

- **递归**是解决问题的一种方法，它将问题不断地分成更小的子问题，直到子问题可以用普通的方法解决。
- 通常情况下，递归会使用一个不停调用自己的函数。

2.1.1 计算一列数之和

- 循环求和函数

```
def listsum(numList):
    theSum = 0
    for i in numList:
        theSum += i
    return theSum
```

```
>>> listsum([1, 3, 5, 7, 9])
25
```

- 将问题从求一列数之和重新定义成求数字对之和。
- 数字列表`numList`的综合等于列表中的第一个元素（`numList[0]`）加上其余元素（`numList[1:]`）之和。
- $listSum(numList) = first(numList) + listSum(rest(numList))$
- $first(numList)$返回列表中的第一个元素，$rest(numList)$则返回其余元素。

```
def listsum(numList):
    if len(numList) == 1:
        return numList[0]
    else:
        return numList[0] + listsum(numList[1:])
```

```
>>> listsum([1, 3, 5, 7, 9])
25
```

2.1.2 递归三原则

1. 递归算法必须有**基本情感**；
2. 递归算法必须改变其状态并向基本情况靠近；
3. 递归算法必须递归地调用自己。

2.1.3 将整数转换成任意进制的字符串

1. 将原来的整数分成一系列仅有单数位的数；
2. 通过查表将单数位的数转换成字符串；
3. 连接得到的字符串，从而形成结果。

```
def toStr(n, base):
    convertString = '0123456789ABCDEF'
    if n < base:
        return convertString[n]
    else:
        return toStr(n//base, base) + convertString[n%base]
```

```
>>> toStr(769, 10)
'769'
>>> toStr(10, 2)
'1010'
```

2.2 复杂的递归问题（汉诺塔）

借助一根中间柱子，将高度为`height`的一叠盘子从起点柱子移到终点柱子：

1. 借助终点柱子，将高度为`height-1`的一叠盘子移到中间柱子；
2. 将最后一个盘子移到终点柱子；
3. 借助起点柱子，将高度为`height-1`的一叠盘子从中间柱子移到终点柱子。

```
def moveTower(height, fromPole, withPole, toPole):
    if height >= 1:
        moveTower(height-1, fromPole, toPole, withPole)
        moveDisk(height, fromPole, toPole)
        moveTower(height-1, withPole, fromPole, toPole)

def moveDisk(disk, fromPole, toPole):
    print(f'moving disk{disk} from {fromPole} to {toPole}')
```

```
>>> moveTower(1, '#1', '#2', '#3')
moving disk1 from #1 to #3
>>> moveTower(2, '#1', '#2', '#3')
moving disk1 from #1 to #2
moving disk2 from #1 to #3
moving disk1 from #2 to #3
>>> moveTower(3, '#1', '#2', '#3')
moving disk1 from #1 to #3
moving disk2 from #1 to #2
moving disk1 from #3 to #2
moving disk3 from #1 to #3
moving disk1 from #2 to #1
moving disk2 from #2 to #3
moving disk1 from #1 to #3
```