作业参考

递归的解法，一般来说有2种：

1. 一种如同数学公式
2. 一种类似循环，相当于循环的改版，将循环迭代，变成了函数调用压栈

1、n的阶乘

```python
def factorial(n):
    s = 1
    for i in range(2, n+1):
        s *= i
    return s

def factorial(n):
    s = 1
    for i in range(n, 1, -1):
        s *= i
    return s
```

```python
# 循环 变 递归。三元表达式自己实现
def factorial2(n, p=1): # 需要提供一个形参传入上一次
    # 循环中的循环体
    if n == 1: # 边界条件。1就不用乘了，直接返回
        return p # 直到现在的乘积

    # 进行下一次计算，使用函数调用实现
    return factorial2(n-1, p * n)
```

```python
# 阶乘公式。三元表达式自己实现
def factorial1(n):
    if n < 2:
        return 1
    return factorial1(n-1) * n
```

2、猴子吃桃问题

猴子第一天摘下若干个桃子，当即吃了一半，还不过瘾，又多吃了一个。第二天早上又将剩下的桃子吃掉一半，又多吃了一个。以后每天早上都吃了前一天剩下的一半零一个。到第10天早上想吃时，只剩下一个桃子了。求第一天共摘多少个桃子

```
思路：
假设猴子摘了x个桃

d1  x //2 - 1
d2  d1//2 - 1
d3  d2//2 - 1
...
d9  d8//2 - 1
d10 1
```

```python
# 循环实现
peach = 1
days = 9

for i in range(days):
    peach = 2 * (peach + 1)

print(peach)

# 改版的递归实现
def fn(days=9, peach=1):
    peach = 2 * (peach + 1)
    if days == 1: # 保证计算过days次，9天就是9次
        return peach
    return fn(days-1, peach)

print(fn())
```

```python
def peach(days=10): # 虽然是10，但是计算了9次，最后一次是为了等于1触底反弹的
    if days == 1:
        return 1
    return 2 * (peach(days-1) + 1)

print(peach())
```