函数的递归

- 递归 : 函数直接或者间接调用函数本身，称之为递归

```
# 4!   fun(4)
#      ->  return fun(3) * 4
#                  ->  return fun(2) * 3
#                              -> return fun(1) * 2
#                                         return 1
def fun(n):
    if n == 1:
        return 1
    return fun(n-1) * n

num = int(input())
ans = fun(num)
print(ans)
```

- 问题本身应该具有如下特性:
  - 问题本身规模可以化小，而小规模问题（子问题）的解可以可以求出大规模问题的解
  - 问题应该具有极限，极小问题（不需要复杂运算就可以求到的）
- 如何写一个递归函数:
  - 确定递归公式：大问题 和 小问题 的关系
    - 例如 n! = (n-1)! * n
  - 确定问题极限，极小问题是什么
    - 例如 1! = 1
  - 在递归函数的编码时，应先判断极小问题，再进行递归

```
# 1 1 2 3 5 8 13 21 ...

#  年份      1  2  3  4  5  6 ...
#  大兔子    0  1  1  2  3  5
#  小兔子    1  0  1  1  2  3
#            1  1  2  3  5  8  ....
#  第n年的兔子数量
#    =  第n年大兔子的数量 + 第n年小兔子的数量
#    =  第n-1年的兔子数量 + 第n-1年大兔子的数量
#    =  第n-1年的兔子数量 + 第n-2年的兔子数量

def fun(n):
    if n == 1 or n == 2:
        return 1
    return fun(n-1) + fun(n-2)

def fun1(n):
    n1, n2, nn = 1, 1, 0
    for i in range(3, n+1):
```

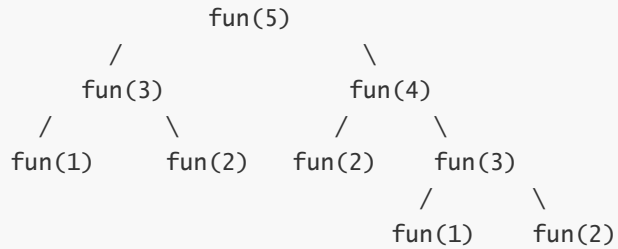
```
        nn = n1 + n2
        n2 = n1
        n1 = nn
    return nn

def fun2(n):
    ll = [0] * 100
    ll[1], ll[2] = 1, 1
    for i in range(3, n+1):
        ll[i] = ll[i-1] + ll[i-2]
    return ll[n]

def fun3(n):
    ll = [1, 1]
    for i in range(3, n+1):
        ll[i%2] = ll[(i-1)%2] + ll[(i-2)%2]
    return ll[n%2]

num = int(input())

print(fun(num))
print(fun1(num))
print(fun2(num))
print(fun3(num))
```



- n阶楼梯，杜浩裕要爬完这些楼梯，他因为腿短，每次要么迈1阶，要么迈2阶，问杜浩裕爬完n阶楼梯，总共有多少种不同的爬楼方式。

- 内存 1M， 以字编址，每个字2字节，数据线条数是多少?
  - 1M = 1024KB = 1024 *1024 B = 2^20B
  - 字的数量 = 2^20 / 2 = 2 ^19