求n的阶乘

```
def fac(n):
    if n == 1:
        return 1
    return n * fac(n-1)


def fac1(n, p = 1):
    if n == 1:
        return p
    p *= n
    print(p)
    fac1(n-1, p)
    return p


def fac2(n, p = None):
    if p is None:
        p = [1]
    if n == 1:
        return p[0]
    p[0] *= n
    print(p[0])
    fac2(n-1, p)
    return p


n = 10
print(fac(n))
print(fac1(n))
print(fac2(n))
```


fac函数性能最好，因为时间复杂度是相同的，fac最简单

将一个数逆序放入列表中，例如1234 => [4,3,2,1]

```
data = str(1234)

def revert(x):
    if x == -1:
        return ''
    return data[x] + reversal(x-1)

print(revert(len(data)-1))
```

```
def revert(n,lst=None):
    if lst is None:
        lst = []

    x,y = divmod(n, 10)
    lst.append(y)
    if x == 0:
        return lst
    return reverse(x, lst)

revert(12345)
```

```
num = 1234

def revert(num, target=[]):
    if num:
        target.append(num[len(num)-1]) # target.append(num[-1:])
        revert(num[:len(num)-1])
    return target

print(revert(str(num)))
```

解决猴子吃桃问题

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
```

```
def peach(days=10):
    if days == 1:
        return 1
    return (peach(days-1)+1)*2

print(peach())
```

注意这里必须是10，因为return (peach(days-1)+1)*2立即拿不到结果，必须通过再一次进入函数时判断是不是到了最后一天。
也就是当前使用的值是由下一次函数调用得到，所以要执行10次函数调用

换种方式表达

```
def peach(days=1):
    if days == 10:
        return 1
    return (peach(days+1)+1)*2

print(peach())
```