快速排序

怎么排

通过一趟排序把数据分成两个部分，其中一部分的所有数据都比另一部分的所有数据小，在分别对两部分进行快速排序。

啊哈算法

code

1.递归

从前有个山，山里有个庙，庙里有个老和尚，老和尚给小和尚讲故事：从前有个山，山里有个庙，庙里有个老和尚，老和尚给小和尚讲故事：。。。。。

```
def f(x):
    if x > 0:
       return x + f(x - 1)
    else:
        return 0

print(f(3))
```

利用递归计算1到100的和

2.想一想我们的程序中有没有递归?

对一个列表进行排序，将基准数归位，将大列表分成两个小列表，分别对小列表做同样的操作。直到序列只剩下一个元素，哨兵一开始就相遇

区别仅仅是列表的起始结束位置发生了变化。

所以我们只需要设计一个函数，让他递归(通过改变开始结束的位置)，就能完成排序。

分析一遍程序如何递归的解决这个问题!

3.分析这个递归函数应该怎么写，可以以最初的列表为例去分析

传参：list, left, right 。 每次递归的时候 这三个值都是不同的

首先，哨兵i,j归初始位

哨兵j 找比基准数小的

哨兵i找比基准数大的

找到之后判断 i是否等于j， 等于的话哨兵相遇，探测结束，否则，交换继续找。

探测结束后基准数归位，分别递归基准数左边的列表和基准数右边的列表。。。。

一直递归下去。

4.首先要判断哨兵i,j 的位置，决定探测是否结束。没结束的话就要一直执行到 i == j， 哨兵相遇，

结束的话就该递归基准数左边的和右边的子列表了。

1,5,8,9,46,7

```
def quicksort(l, left, right):
    递归出口 if left > right
                return
    哨兵赋初值 i = left j = right
    while 哨兵未相遇:  i!=j
        j哨兵找小于基准数的数
        while l[j] >= l[left]#找小于基准数的j
            j-=1
        i哨兵找大于基准数的数

        if 哨兵没有相遇:  i != j
            交换
     将基准数归位 L[i] or L[j] 与基准数交换
递归左边 quicksort(l,XXX,yyy)
递归右边 quicksort(l,ppp,qqq)

l = [516546546546]
quicksort(l, 0, len(l)-1)
```

```
################################################
def quicksort(l, left, right):
    if left > right:
        return
    i = left #哨兵
    j = right
```

```
    while i!=j:
        while l[j] >= l[left] and i < j: #在有效范围内找大于基准数的
            j-=1
        while l[i] <= l[left] and i < j:
            i+=1
        if i < j: #哨兵还没碰面
            l[i], l[j] = l[j], l[i]
    l[left], l[i] = l[i], l[left]


    quicksort(l, left, i -1)
    quicksort(l, i+1, right)

list1 = [6, 1, 2, 7, 9, 3, 4, 5, 10, 8]
quicksort(list1, 0, len(list1)-1)
print(list1)
```