输入输出运行说明：

本程序的输入文件为 **quick.py** ，**quick.py** 文件中为需要解析的快排程序。（**quick.py** 文件和所有文件都需放在一个目录下）执行本程序则只需要运行 main.py 文件即可。输出则显示在控制台。

解析样例如下：

main.py quick_sort.py

```
def quick_sort(array, left, right){
    if(left >= right){
        return
    }
    low = left
    high = right
    key = array[low]
    while(left < right){
        while(left < right and array[right] > key){
            right -= 1
        }
        array[left] = array[right]
        while(left < right and array[left] <= key){
            left += 1
        }
        array[right] = array[left]
    }
    array[right] = key
    quick_sort(array, low, left - 1)
    quick_sort(array, left + 1, high)
}

a=[1,2,4,3,6,5,7,3]

quick_sort(a,0,len(a)-1)

print(a)
```

程序输出结果如下：

```
C:\Users\pc\AppData\Local\Programs\Python\Pytho
 + [PROGRAM]
   + [STATEMENTS]
     + [STATEMENTS]
       + [STATEMENTS]
         + [STATEMENTS]
           + [STATEMENT]
             + [FUNCTION]
               + def
               + quick_sort
               + (
               + [ARGS]
                 + [ARGS]
                   + [ARGS]
                     + array
                   + ,
                   + left
                 + ,
                 + right
               + )
               + {
               + [STATEMENTS]
                 + [STATEMENTS]
                   + [STATEMENTS]
                     + [STATEMENTS]
                       + [STATEMENTS]
                         + [STATEMENTS]
                           + [STATEMENTS]
                             + [STATEMENTS]
                               + [STATEMENT]
                                 + [IF]
                                   + a
                                   + )
                             + -
                             + [TERM]
                               + [FACTOR]
                                 + 1
                         + )
                     + [STATEMENT]
                       + [PRINT]
                         + print
                         + (
                         + a
                         + )
{'array': [1, 2, 4, 3, 6, 5, 7, 3], 'left': 0, 'right': -1}
{'array': [1, 2, 4, 3, 6, 5, 7, 3], 'left': 1, 'right': 0}
{'array': [1, 2, 3, 3, 4, 5, 7, 6], 'left': 2, 'right': 2}
{'array': [1, 2, 3, 3, 4, 5, 7, 6], 'left': 4, 'right': 3}
{'array': [1, 2, 3, 3, 4, 5, 7, 6], 'left': 3, 'right': 3, 'low': 2, 'high': 3, 'key': 3}
{'array': [1, 2, 3, 3, 4, 5, 7, 6], 'left': 5, 'right': 4}
{'array': [1, 2, 3, 3, 4, 5, 6, 7], 'left': 6, 'right': 6}
{'array': [1, 2, 3, 3, 4, 5, 6, 7], 'left': 8, 'right': 7}
{'array': [1, 2, 3, 3, 4, 5, 6, 7], 'left': 7, 'right': 7, 'low': 6, 'high': 7, 'key': 7}
{'array': [1, 2, 3, 3, 4, 5, 6, 7], 'left': 5, 'right': 5, 'low': 5, 'high': 7, 'key': 5}
{'array': [1, 2, 3, 3, 4, 5, 6, 7], 'left': 4, 'right': 4, 'low': 2, 'high': 7, 'key': 4}
{'array': [1, 2, 3, 3, 4, 5, 6, 7], 'left': 1, 'right': 1, 'low': 1, 'high': 7, 'key': 2}
{'array': [1, 2, 3, 3, 4, 5, 6, 7], 'left': 0, 'right': 0, 'low': 0, 'high': 7, 'key': 1}
[1, 2, 3, 3, 4, 5, 6, 7]
{'a': [1, 2, 3, 3, 4, 5, 6, 7]}

Process finished with exit code 0
```

设计思路：

其他部分设计和前几次文件并未有太多差别，添加了 return 和 def 一些逻辑，因此不加赘述。最主要的难点还是在于节点的问题。因为快排的核心是递归，所以程序主要是针对 return 语句的处理，为此，我为每一个 statement 语句设置了一个返回值，它可以为 None、BREAK、RETURN 或其他值；在 statements 循环执行时，当遇到一个 statement 的状态为 BREAK 或 RETURN 时，statements 立即停止执行，并将自己的状态也设置为 BREAK 或 RETURN；BREAK 是由当前结点的某个值为 while 或 for 的父节点对应处理、RETURN 则是由当前 function 的树根处理。

```
elif data == '[STATEMENTS]':
    for node in n:
        n.setvalue(self.translate(node))
        if n.getvalue() == BREAK or n.getvalue() == RETURN:
            break

elif data == '[STATEMENT]':
    if n[0].getdata() == 'BREAK':
        n.setvalue(BREAK)
    elif n[0].getdata() == '[RETURNSOMETHING]':
        n.setvalue(RETURN)
    else:
        n.setvalue(self.translate(n[0]))

elif data == '[RETURNSOMETHING]':
    if len(n) == 1:
        n.setvalue(RETURN)
    elif len(n) == 2:
        self.value = self.translate(n[1])
        n.setvalue(RETURN)
```