## 局部变量

局部变量是只能在函数或代码段内使用的变量。

局部变量的作用范围示意图

文件2不能访问文件1的局部变量

文件2

文件1

局部变量只能被fun（）访问

局部变量不能被fun2访问

局部变量

Fun（）

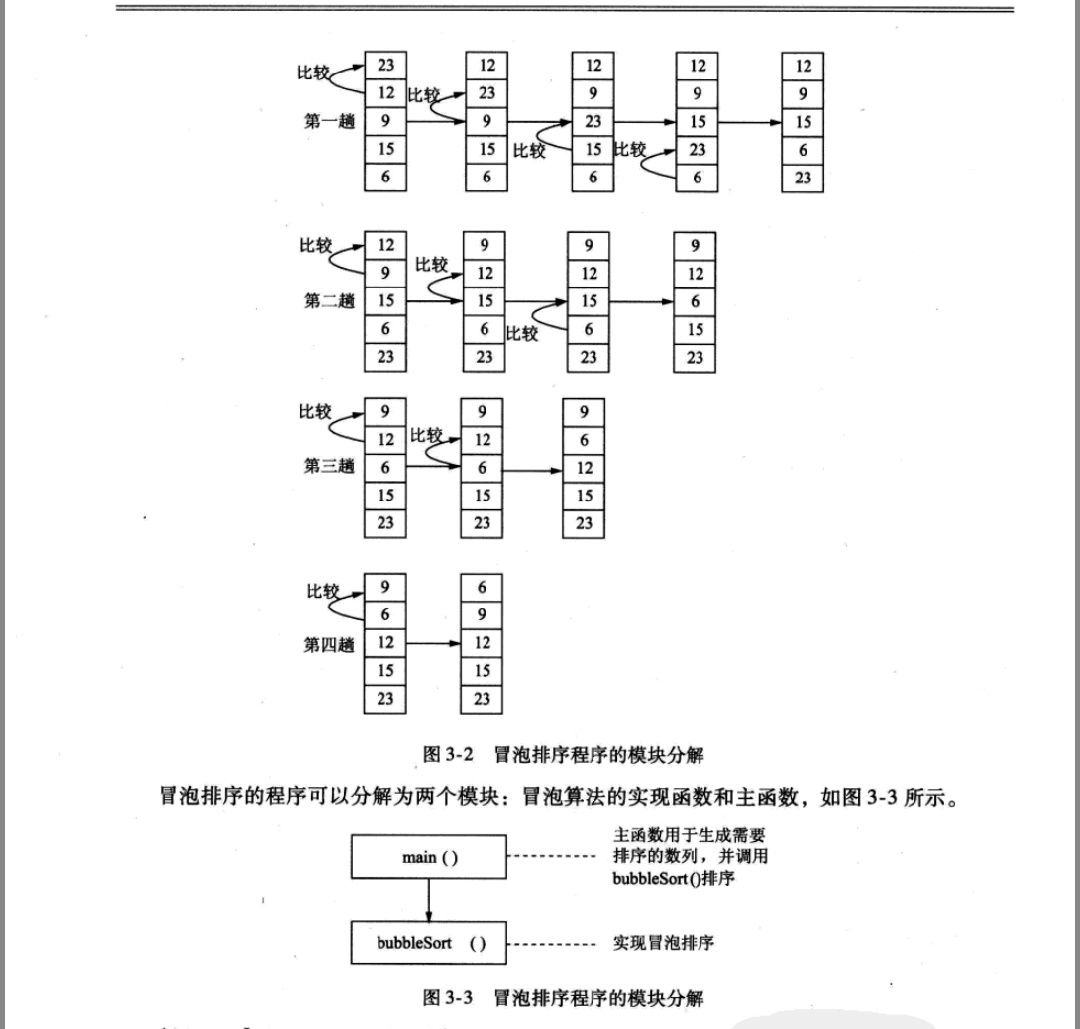
Fun2()

（）

## 全局变量

全局变量是能够被不同的函数，类或文件共享的变量，在函数之外定义的变量都可以称为全局变量。函数内部使用gloabal来引用全局变量。

## 冒泡排序



代码实现：

def bubbleSort(numbers):

for j in xrange(len(numbers) – 1,-1,-1):

for I in xrange(j):

if numbers[i] > numbers[i+1]:

numbers[i],numbers[i+1] = numbers[i+1],numbers[i]

print numbers

def main():

numbers = [23,12,9,15,6]

bubbleSort(numbers)

if \_\_name\_\_ == “\_\_main\_\_”:

main()

## 列表的常用方法

|  |  |
| --- | --- |
| 方法名 | 说明 |
| append(object) | 在列表的末尾添加一个对象object |
| insert(index,object) | 在指定的索引index处插入一个对象object |
| remove(value) | 删除列表中首次出现的value值 |
| pop([index]) | 删除索引index指定的值，如果index不指定，则删除列表中最后一个元素 |
| extend(iterable) | 将iterable指定的元素添加到列表的末尾 |
| index(value,[start,[stop]] | 返回value现在出现在列表中的索引 |
| sort(cmp=None,key=None,reverse=False) | 列表的排序 |
| reverse() | 列表的反转 |

### 用列表实现堆栈和队列

堆栈和队列是数据结构中常用的数据结构，使用列表的append和pop方法可以模拟这两种数据结构。

#### 堆栈的实现

堆栈：符合后进先出的原则。

堆栈的实现方法是：调用append()可以把一个元素添加到堆栈的顶部，调用pop()可以把堆栈中的最后一个元素取出。

列表实现堆栈原理图

append(“orange”)

依次向下取值

pop()

Banana

Apple

Orange

Grape

#### 队列的实现

队列符合先进先出原则。

队列也可以通过调用append和pop方法实现，但是pop方法的使用方式不同，使用pop（0）取出队列最前面的元素。

列表实现队列的原理图

append(“orange”)

Banana

Apple

Orange

Grape

pop()

## 字典

由键-值对组成的集合。

字典的常用方法

|  |  |
| --- | --- |
| 方法名 | 说 明 |
| items() | 返回（key，value）元组组成的列表 |
| iteritems() | 返回指向字典的遍历器 |
| setdefault(k[,d]) | 创建新的元素并设置默认值 |
| pop(k[,d]) | 删除索引k对应的value值，并返回该值 |
| get(k[,d]) | 返回索引k对应的value值 |
| keys() | 返回字典中key的列表 |
| values() | 返回字典中value的列表 |
| update(E) | 把字典E中数据扩展到原字典中 |
| copy() | 复制一个字典中所有的数据 |