# 作用域

### Python作用域基础

python变量名在第一次赋值的时候已经创建，并且必须经过赋值后才能使用。

* 一个在def内定义的变量名能够被def内的代码使用。不能在函数外部引用这样的变量名。
* d ef内的变量名与def外的变量名并不冲突

#### 作用域法则

函数定义了本地作用域，而模块定义的是全局作用域。

* 内嵌的模块是全局作用域。
* 全局作用域的作用范围仅限于单个文件。
* 每次对函数的调用都创建了一个新的本地作用域。
* 赋值的变量名除非声明为全局变量或非本地变量，否则均为本地变量
* 所有其他的变量名都可以归纳为本地、全局或者内置的。

修改一个对象并不是对一个名称赋值。

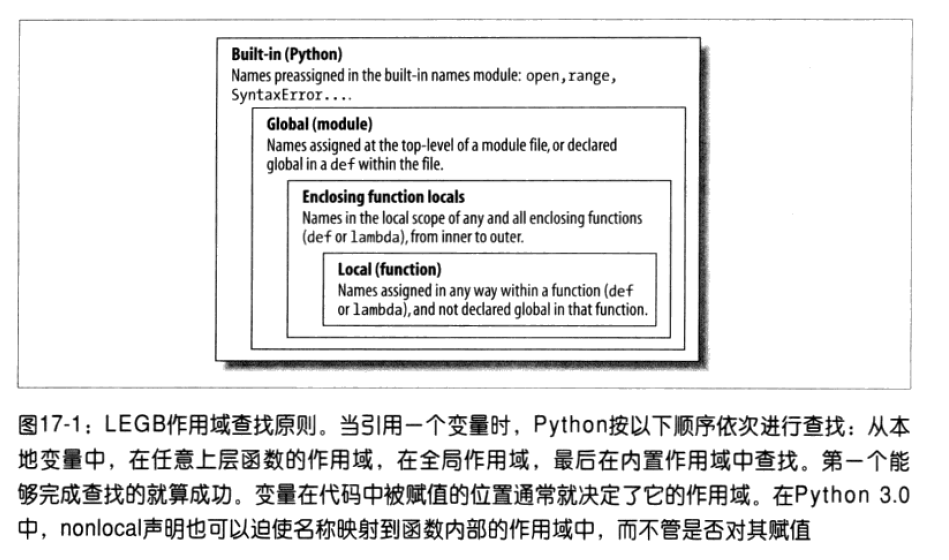
交互命令模式输入的代码也遵从这些规则。交互模式运行代码实际上真的输入到一个叫做\_\_main\_\_的内置模块中。

修改一个对象并不是对一个名称赋值。

#### 变量名解析：LEGB原则

对于一个def语句：

* 变量名引用分为三个作用域进行查找：首先是本地，之后是函数内，之后全局，最后是内置。
* 在默认情况下，变量名赋值会创建或者改变本地变量。
* 全局声明和非本地声明将赋值的变量名映射到模块文件内部的做用域。



#### 作用域实例

#### 内置作用域

内置作用域是一个名为\_\_builtin\_\_的内置模块，但是必须要import \_\_builtin\_\_之后才能使用内置作用域，因为变量名builtin本身没有预先内置。

### Global语句

* 全局变量是位于模块文件内部的顶层的变量名
* 全局变量如果是在函数内被赋值的话，必须经过声明
* 全局变量名在函数内部不经过声明也可以引用

#### 最小化全局变量

#### 最小化文件间的修改

在文件中进行通讯最好的方法就是通过调用函数，传递参数，然后得到其返回值。

#### 其他访问全局变量的方法

全局变量与模块的属性是等效的。

### 作用域和嵌套函数

#### 嵌套作用域的细节

* 一个引用（X）首先在本地（函数内）作用域查找变量名X；之后会在代码的语法上嵌套了的函数的本地作用域，从内到外查找；之后查找当前的全局作用域（模块文件）；最后再内置作用域内（模块\_\_buildin\_\_）.全局声明将会直接从全局（模块文件）作用域进行搜索。
* 在默认情况下，一个赋值（X=value）创建或改变了变量X的当前作用域。

#### 嵌套作用域示例

**工厂函数：**闭合（closure）——一个能够记住嵌套作用域的变量值的函数。

**使用默认参数来保留嵌套作用域的状态**

**嵌套作用域和lambda**

**作用域与带有循环变量的默认参数相比较**

**任意作用域嵌套**

### nonlocal语句

Nonlocal应用于一个嵌套的函数的作用域中的一个名称，而不是所有def之外的全局模块作用域；而且在声明nonlocal名称的时候，它必须已经存在于该嵌套函数的作用域内。

#### nonlocal基础

当执行nonlocal时，nonlocal中列出的名称必须在一个嵌套的def中提前定义过。

nonlocal语句的主要作用是允许嵌套的作用域中的名称被修改，而不只是被引用。

在python2.6中，对嵌套的def的作用域名称的引用是允许的，但不能对其赋值。

#### nonlocal应用

边界情况：

* 和global语句不同，当执行一条nonlocal语句时，nonlocal名称必须已经在一个嵌套的def作用域中赋值过。
* nonlocal限制作用域查找仅为嵌套的def，nonlocal不会在嵌套的模块的全局作用域或所在的def之外的内置作用域内查找，即便已经有了这些作用域。

#### 为什么使用nonlocal？

与全局共享状态

使用类的状态

使用函数属性的状态