快速理解 Python 命名空间

命名空间是Python中的基础概念，对于构建和组织你的代码（尤其如果你有足够大的项目）非常有帮助。然而，如果你是编程新手，或者是其他语言的程序员，可能会很难理解Python的命名空间。我会在这篇文章中解释Python命名空间，使之容易理解。

什么是一个命名/名字/变量名?

在理解命名空间之前，你必须明白Python中的一切都是以命名来表达意义的。 一个Python命名大致类似于其他几乎任何语言的变量，但有一些附加功能。首先，因为Python的动态特性，你可以为任何一个物体（对象）声明一个命名。你当然可以将值给予命名。

```
a = 12
b = 'B'
c = [1, 2, 3, 4]
```

但你也可以给予一个函数命名：

```
def func():
    print 'This is a function'

f = func
```

现在，你想使用func函数时，你可以使用f()。你也可以将一个命名重用于其他的不同的对象。

```
var = 12
var = "This is a string now"
var = [2, 4, 6, 8]
```

那么，如果你在这三个赋值之间使用var这个命名，你会得到不同的对象。在Python中，一切

都是对象。数字，字符串，函数，类都是对象。只有通过一个命名（变量名），才能摄取一个对象。

模组与命名空间相辅相成

当命名变多时，一个命名空间显然不够。Python教程一般都说，命名空间是名字到对象的映射。你可以把命名空间想象成一个你可以定义的所有名字的列表。每个名字对应了一个对象。

要了解命名空间，你必须对Python模组有一定的了解。一个Python模组仅仅是一个包含Python代码的文件。这些代码可以是类、函数、一些变量或者任何形式。每个模块都有自己的全局命名空间。所以在同一个模组中，你不能有两个名字相同的对象，因为它们共享同一个模组的命名空间（除非它们嵌套，我们将在后面来讲到）。

然而，每个命名空间是完全孤立的。所以同一个变量名可以存在于不同的模组中。你可以有一个名为Integer的模组和一个名为FloatingPoint的模组，他们都可以有一个名为add的函数。一旦你在脚本中import模组，你可以通过模组名作为前缀访问模组中的命名：FloatingPoint.add（）和Integer.add（）。

当你运行一个Python脚本时，解释器将其视为一个享有自己独立命名空间的__main__模组。Python的自带函数在__builtin__模组中。

导入Import

如果你不导入一个模组并使用，那么该模组是没用的。有许多方法可以导入模组，每个方法对命名空间会有不同的影响。

方法一：import 模组名

这是最简单的import方式。这样你可以使用模组名作为前缀来访问该模组的命名空间。这意味着，你的程序里的命名可以与该模块重名。当你导入大量模组的时候，这种方式可以让你知道每个名字属于哪个模组。

方法二：from 模组名 import 模块名

这种方法把一个名字从一个模组的命名空间内导入。这样你可以直接使用这个名字。缺点是，你的程序中就不能使用同样的命名来表示其他的对象。

方法三：from 模组名 import *

将某模组命名空间内的所有名字导入。一般来说不是一个好主意，因为它导致'命名空间污染“。因为有可能很多名字都被该模组使用了。

import对于所有对象都适用，不管是类，函数，还是其他的。import一开始可能会有点让人迷惑，但是稍作练习后会变得很清晰的。

作用域Scope

作用域是一个可以不使用前缀而使用某个命名的区域。作用域提供了模组内的孤立。一个函数有作用域，一模组有作用域，Python builtin有作用域。作用域是互相嵌套的。在一个作用域内不能使用另一个作用域内的命名。

命名空间被从里到外地搜索，这意味着，你可以在一个作用域里面使用模组中的全局名字（全局作用域包含了本地作用域）。当然，你可以使用global关键字来强制使用全局名字。

原创： Shrutarshi Basu 优达学城Udacity

文/ Shrutarshi Basu

康奈尔大学 PhD，程序员，技术博客作者

译者 王选哲