第5章 模　块

5.1 模块的概念

我们来考虑如下几种场景：

(1) 编写一个 Python 程序，如果程序比较简单，则可以把代码放到一个 Python 文件中。但如果程序功能比较多，可能需要多个 Python 文件来组织源代码。而这些文件之间的代码肯定有关联，例如一个文件中的 Python 代码调用另一个 Python 文件中定义的函数，怎么能做到呢？

(2) 编写程序，肯定不会所有的东西都自己写，我们肯定会用到 Python 提供的一些标准库，那么该怎么使用呢？

(3) 我们可能想编写一个公共代码，或从外部找到一个第三方的公共代码，如何放到整个 Python 系统中？如何被自己编写的代码使用？

上面这些场景，都是在编写程序时常见的问题，而这些问题，Python 是通过模块和包的机制来解决的。

简单地说，一个模块就是一个 Python 文件，一个包包含一组模块。下面将详细说明 Python 中模块和包的概念。

5.1.1 命名空间

Python 支持面向对象编程，遵守一切皆对象的原则，所以想要好好地理解模块，一定要先理解命名空间的概念。所谓命名空间，是指标识符的可见范围。对于 Python 而言，常见的命名空间主要有以下几种：

1. Build-in Namespace（内建命名空间）

内建命名空间是任何模块均可访问的命名空间，它存放着内置的函数和异常。内建命名空间在 Python 解释器启动时创建，会一直保留，不被删除。

2. Global Namespace（全局命名空间）

每个模块拥有它自己的命名空间，叫作全局命名空间，它记录了模块的变量，包括函数、类、其他导入的模块、模块级的变量和常量。模块的全局命名空间在模块定义被读入时创建，通常模块命名空间也会一直保存到解释器退出。

3. Local Namespace（局部命名空间）

每个函数都有着自己的命名空间，叫作局部命名空间，它记录了函数的变量，包括函数的参数和局部定义的变量。当函数被调用时创建一个局部命名空间，当函数返回结果或抛

出异常时，被删除。每一个递归调用的函数都拥有自己的命名空间。

有了命名空间的概念，可以有效地解决函数或者是变量重名的问题。当一行代码要使用变量 x 的值时，Python 会到所有可用的名字空间去查找变量，按照如下顺序：

(1) 局部命名空间：特指当前函数或类的方法。如果函数定义了一个局部变量 x，或一个参数 x，Python 将使用它，然后停止搜索。

(2) 全局命名空间：特指当前的模块。如果模块定义了一个名为 x 的变量、函数或类，Python 将使用它然后停止搜索。

(3) 内置命名空间：对每个模块都是全局的。作为最后的尝试，Python 将假设 x 是内置函数或变量。

(4) 如果 Python 在这些名字空间找不到 x，它将放弃查找并引发一个 NameError 异常，如 NameError：name 'x' is not defined。

不同的命名空间中允许出现相同的函数名或者是变量名。它们彼此之间不会相互影响，例如在 Namespace A 和 Namespace B 中同时有一个名为 var 的变量，对 A.var 赋值并不会改变 B.var 的值。

5.1.2 模块

Python 中的一个模块对应的就是一个.py 文件。其中定义的所有函数或者是变量都属于这个模块。这个模块对于所有函数而言就相当于一个全局的命名空间。而每个函数又都有自己局部的命名空间。如图 5.1 所示，Graph.py 就是一个名字叫 Graph 的模块。

使用模块最大的好处是大大提高了代码的可维护性。其次，编写代码不必从零开始。当一个模块编写完毕，就可以被其他地方引用。我们在编写程序的时候，也经常引用其他模块，包括 Python 内置的模块和来自第三方的模块，如图 5.2 所示。

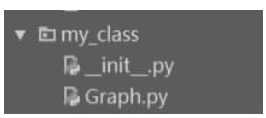

图 5.1 模块

```
import datetime


from my_class.Graph import Graph
```

图 5.2 导入模块

如图 5.2 所示，datatime 是 Python 内置模块，我们用 import 语句导入模块后，就有了变量 datetime 指向该模块，利用 datetime 这个变量，就可以访问 datetime 模块的所有功能。而 Graph 模块此时则作为已定义模块在另一模块中的引用。

使用模块还可以避免函数名和变量名冲突。相同名字的函数和变量完全可以分别存在不同的模块中，因此，我们自己在编写模块时，不必考虑名字会与其他模块冲突。但是也要注意，尽量不要与内置函数名字冲突。

Python 支持以下几种导入方法：

```
import 模块   #导入模块
from 包/模块 import 模块/方法/对象 #导入包或模块下的模块或方法或对象等
import 模块 as 别名 #导入模块并重命名
from 包/模块 import 模块/方法/对象 as 别名
#导入包或模块下的模块或方法或对象等并重命名
```

5.1.3 包

包是多个模块的集合，也就是多个.py 文件的集合。我们可以用如下方式来创建一个包：

(1) 新建一个文件夹。

(2) 在该文件夹下新建一个空的__init__.py 文件(必须存在，内容可以为空)。

(3) 在该文件夹下新建 py 文件。

包提供了一种很好的管理模块的方式，可以有效地减少模块的命名冲突，不同包的模块命名可以相同，如图 5.3 所示。

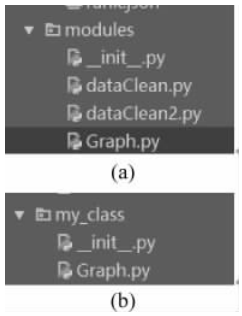

图 5.3 包的文件结构

包 modules 和包 my_class 中都有名为 Graph 的模块，但两者的内容是完全不同的，引用模块时需要带上包名，如 import my_class.Graph。

5.2 模块内置属性

在每一个模块中，都有一些内置属性，这些属性不需要手动声明或者赋值，可以直接访问。

例如，__name__为当前模块名。如果是直接运行该模块，其值为__main__；如果通过导入运行，属性值就是模块名。因此可以通过__name__属性判断该模块是直接运行还是被导入运行的，对于一些不需要在导入运行时执行的，就需要添加__name__=="__main__"判断，如下所示：

```
if __name__ == '__main__':
  print("不是 import 的")
else:
  print("是 import 的")
if __name__ == '__main__':
    print("不是 import 的")
else:
    print("是 import 的")运行结果为:
if __name__ == '__main__':
    print("不是 import 的")
else:
    print("是 import 的")
```

__file__也是 python 模块的一个内置属性，__file__用来获得模块所在的路径。__file__的返回值根据调用模块的方式不同，得到的结果可能不同。使用绝对路径调用模块时__file__将返回绝对路径，使用相对路径调用__file__时，会得到相对路径。因此，想要正确地得到绝对路径，我们需要使用 os.path.realpath(__file__)。

5.3 第三方模块安装方法

在 Python 中安装第三方模块，是通过包管理工具 pip 来完成的。如果正在使用 Windows 操作系统，在命令提示符窗口下尝试运行 pip，如果 Windows 提示未找到命令，可以重新运行安装程序添加 pip，然后再试着在命令提示符窗口下运行 pip，若仍然提示未找到命令，可以试着在命令行提示符窗口输入“python-m pip install-upgrade pip”更新 pip。

在 pip 更新完成以后，我们尝试安装第三方模块 numpy，输入命令“pip install numpy”。Numpy 是用于科学计算的 Python 库；一般来说，第三方库都会在 Python 官方的 pypi.python.org 网站注册，要安装一个第三方库，必须先知道该库的名称，可以在官网或者 pypi 上搜索。

安装第三方模块后，我们可以像使用自带模块一样使用它，例如：

```
>>> import numpy
>>> import sys
>>> a = numpy.arange(0,100,100)
>>> print a
```

习 题 5

一、选择题

1. 数据类型转化中，转为字符串的函数“str()”存在于(　　)命名空间中。

A. 内置命名空间　　B. 全局命名空间　　C. 局部命名空间　　D. 包命名空间

2. 当我们使用“import datetime as dt”导入 Python 的 datetime 模块时，我们下文需要使用(　　)来引用模块。

A. datetime　　B. date　　C. as　　D. dt

3. 通过(　　)属性可以判断当前环境是主环境还是被导入的。

A. __item__　　B. __main__　　C. __init__　　D. __name__

二、填空题

1. Python 中命名空间有_____，_____，_____。

2. 如果脚本不是被导入的，其__name__属性的值为_____。

3. __file__属性_____(是/否)总是输出绝对路径。

三、论述题

查阅资料，了解 Python 有哪些常用的自带模块、第三方模块。

四、编程题

编写一个自己的模块 mymath，该模块中需要有函数 add(a,b)、sub(a,b)、mul(a,b)、div(a,b)实现两个数的加减乘除并返回结果。