模块

IlleniumDillon

2024 年 6 月 4 日

退出 Python 解释器后，再次进入时，之前在 Python 解释器中定义的函数和变量就丢失了。因此，编写较长程序时，最好用文本编辑器代替解释器，执行文件中的输入内容，这就是编写脚本。随着程序越来越长，为了方便维护，最好把脚本拆分成多个文件。编写脚本还一个好处，不同程序调用同一个函数时，不用把函数定义复制到各个程序。

为实现这些需求，Python 把各种定义存入一个文件，在脚本或解释器的交互式实例中使用。这个文件就是模块；模块中的定义可以导入到其他模块或主模块（在顶层和计算器模式下，执行脚本中可访问的变量集）。

1 模块

模块包含可执行语句及函数定义。这些语句用于初始化模块，且仅在 import 语句第一次遇到模块名时执行。文件作为脚本运行时，也会执行这些语句。

每个模块都有自己的私有命名空间，它会被用作模块中定义的所有函数的全局命名空间。因此，模块作者可以在模块内使用全局变量而不必担心与用户的全局变量发生意外冲突。另一方面，如果你知道要怎么做就可以通过与引用模块函数一样的标记法 modname.itemname 来访问一个模块的全局变量。

模块可以导入其他模块。根据惯例可以将所有 import 语句都放在模块（或者也可以说是脚本）的开头但这并非强制要求。如果被放置于一个模块的最高层级，则被导入的模块名称会被添加到该模块的全局命名空间。

还有一种 import 语句的变化形式可以将来自某个模块的名称直接导入到导入方模块的命名空间中。

```
from fibo import fib, fib2
```

这条语句不会将所导入的模块的名称引入到局部命名空间中（因此在本示例中，fibo 将是未定义的名称）。

还有一种变化形式可以导入模块中的所有名称：

```
from fibo import *
```

这种方式会导入所有不以下划线（_）开头的名称。大多数情况下，不要用这个功能，这种方式向解释器导入了一批未知的名称，可能会覆盖已经定义的名称。

模块名后使用 as 时，直接把 as 后的名称与导入模块绑定。

1.1 以脚本方式执行模块

可以用以下方式运行 Python 模块：

$$python fibo.py < arguments >$$

这项操作会在模块的全局命名空间内定义一个名为 __name__ 的变量，并将其设为 __main__。因此，可以通过检查这个变量来确定模块是以脚本方式执行还是被导入。如果一个模块被导入，__name__ 的值为模块名，如果模块被执行， __name__ 的值为 __main__。

将下列代码添加到模块的末尾：

```
if __name__ == "__main__":
    import sys
    fib(int(sys.argv[1]))
```

这个文件既能被用作脚本，又能被用作一个可供导入的模块，因为解析命令行参数的那两行代码只有在模块作为“main”文件执行时才会运行，当这个模块被导入到其它模块时，那两行代码不运行。

1.2 模块搜索路径

当一个模块名被一个 import 语句使用，Python 会在以下位置搜索：

1. 当前目录。
2. 如果模块不在当前目录，Python 会搜索在 shell 变量 PYTHONPATH 中的目录。
3. 如果没有找到，Python 会搜索 Python 安装时的默认路径。

sys.path 变量包含当前目录， PYTHONPATH 和由 Python 安装决定的默认目录, 初始化后，Python 程序可以更改 sys.path。

```
import sys
sys.path.append('/ufs/guido/lib/python')
```

1.3 dir()函数

内置函数 dir() 用于按模块名搜索模块定义的名称。返回一个字符串列表, 包含模块定义的所有类型、变量和函数。

2 包

导入包时，Python 搜索 sys.path 里的目录，查找包的子目录。需要有 __init__.py 文件才能让 Python 将包含该文件的目录当作包来处理,在最简单的情况下，__init__.py 可以只是一个空文件，但它也可以执行包的初始化代码或设置变量。

使用 from package import item 时，item 可以是包的子模块（或子包），也可以是包中定义的函数、类或变量等其他名称。import 语句首先测试包中是否定义了 item；如果未在包中定义，则假定 item 是模块，并尝试加载。如果找不到 item，则触发 ImportError 异常。

相反，使用 import item.subitem.subsubitem 句法时，除最后一项外，每个 item 都必须是包；最后一项可以是模块或包，但不能是上一项中定义的类、函数或变量。

2.1 从包中导入*

当使用 from package import * 时，如果包的 __init__.py 文件定义了一个名为 __all__ 的列表，那么在执行 from package import * 时就会导入这个列表中的所有名称。

2.2 包的相对导入

包的相对导入中，用句点表示相对导入的基准。一个单独的句点表示当前目录；两个句点表示父目录。

注意，相对导入基于当前模块名。因为主模块名永远是 "__main__" ，所以如果计划将一个模块用作 Python 应用程序的主模块，那么该模块内的导入语句必须始终使用绝对导入。

2.3 多个目录中的包

包还支持一个特殊属性 __path__ 。在包的 __init__.py 中的代码被执行前，该属性被初始化为一个只含一项的列表，该项是一个字符串，是 __init__.py 所在目录的名称。可以修改此变量；这样做会改变在此包中搜索模块和子包的方式。