使用模块

2.7旧版教程

阅读: 177616

Python本身就内置了很多非常有用的模块，只要安装完毕，这些模块就可以立刻使用。

我们以内建的`sys`模块为例，编写一个`hello`的模块：

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-

' a test module '

__author__ = 'Michael Liao'

import sys

def test():
    args = sys.argv
    if len(args)==1:
        print('Hello, world!')
    elif len(args)==2:
        print('Hello, %s!' % args[1])
    else:
        print('Too many arguments!')

if __name__=='__main__':
    test()
```

第1行和第2行是标准注释，第1行注释可以让这个`hello.py`文件直接在Unix/Linux/Mac上运行，第2行注释表示.py文件本身使用标准UTF-8编码；

第4行是一个字符串，表示模块的文档注释，任何模块代码的第一个字符串都被视为模块的文档注释；

第6行使用`__author__`变量把作者写进去，这样当你公开源代码后别人就可以瞻仰你的大名；

以上就是Python模块的标准文件模板，当然也可以全部删掉不写，但是，按标准办事肯定没错。

后面开始就是真正的代码部分。

你可能注意到了，使用`sys`模块的第一步，就是导入该模块：

```
import sys
```

导入`sys`模块后，我们就有了变量`sys`指向该模块，利用`sys`这个变量，就可以访问`sys`模块的所有功能。

`sys`模块有一个`argv`变量，用list存储了命令行的所有参数。`argv`至少有一个元素，因为第一个参数永远是该.py文件的名称，例如：

运行`python3 hello.py`获得的`sys.argv`就是`['hello.py']`；

运行`python3 hello.py Michael`获得的`sys.argv`就是`['hello.py', 'Michael']`。

运行`python3 hello.py Michael`获得的`sys.argv`就是`['hello.py', 'Michael']`。

最后，注意到这两行代码：

```
if __name__=='__main__':
    test()
```

当我们在命令行运行`hello`模块文件时，Python解释器把一个特殊变量`__name__`置为`__main__`，而如果在其他地方导入该`hello`模块时，`if`判断将失败，因此，这种`if`测试可以让一个模块通过命令行运行时执行一些额外的代码，最常见的就是运行测试。

我们可以用命令行运行`hello.py`看看效果：

```
$ python3 hello.py
Hello, world!
$ python hello.py Michael
Hello, Michael!
```

如果启动Python交互环境，再导入`hello`模块：

```
$ python3
Python 3.4.3 (v3.4.3:9b73f1c3e601, Feb 23 2015, 02:52:03)
[GCC 4.2.1 (Apple Inc. build 5666) (dot 3)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> import hello
>>>
```

导入时，没有打印`Hello, word!`，因为没有执行`test()`函数。

调用`hello.test()`时，才能打印出`Hello, word!`：

```
>>> hello.test()
Hello, world!
```

作用域

在一个模块中，我们可能会定义很多函数和变量，但有的函数和变量我们希望给别人使用，有的函数和变量我们希望仅仅在模块内部使用。在Python中，是通过`_`前缀来实现的。

正常的函数和变量名是公开的（public），可以被直接引用，比如：`abc`，`x123`，`PI`等；

类似`__xxx__`这样的变量是特殊变量，可以被直接引用，但是有特殊用途，比如上面的`__author__`，`__name__`就是特殊变量，`hello`模块定义的文档注释也可以用特殊变量`__doc__`访问，我们自己的变量一般不要用这种变量名；

类似`_xxx`和`__xxx`这样的函数或变量就是非公开的（private），不应该被直接引用，比如`_abc`，`__abc`等；

之所以我们说，private函数和变量“不应该”被直接引用，而不是“不能”被直接引用，是因为Python并没有一种方法可以完全限制访问private函数或变量，但是，从编程习惯上不应该引用private函数或变量。

private函数或变量不应该被别人引用，那它们有什么用呢？请看例子：

```
def _private_1(name):
    return 'Hello, %s' % name
```

```
def _private_2(name):
    return 'Hi, %s' % name

def greeting(name):
    if len(name) > 3:
        return _private_1(name)
    else:
        return _private_2(name)
```

我们在模块里公开`greeting()`函数，而把内部逻辑用private函数隐藏起来了，这样，调用`greeting()`函数不用关心内部的private函数细节，这也是一种非常有用的代码封装和抽象的方法，即：

外部不需要引用的函数全部定义成private，只有外部需要引用的函数才定义为public。

感觉本站内容不错，读后有收获？

¥ 我要小额赞助，鼓励作者写出更好的教程

还可以分享给朋友

分享 buptwqp 等1人分享过

评论

如何在交互模式下使argv > 2？即打印出Hello Michael？？

西迁俺-煊_煊鑫 created at 11-28 22:58, Last updated at 11-28 22:58

```
>>> import hello
>>> hello.test()
Hello, world!
```

我想让他打印出 Hello, Michael!
要怎么输入命令呢？

View Full Discuss Reply This Topic

为什么不能调用hello模块？

西迁俺-煊_煊鑫 created at 11-28 22:47, Last updated at 11-28 22:52

```
>>> import hello
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
ImportError: No module named 'hello'
```

我代码和上面的一样的

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-

' a test module '

__author__ = 'Michael Liao'

import sys

def test():
    args = sys.argv
    if len(args)==1:
            print('Hello, world!')
    elif len(args)==2:
```

Read More

西迁俺-煊_煊鑫
Created at 11-28 22:52, Last updated at 11-28 22:52

解决了，我把

```
#!/usr/bin/env python3
```

修改成了

```
#!/usr/bin/env python
```

因为我在命令模式下是通过 python xxx.py 调用的。

View Full Discuss Reply This Topic

还是不懂name==main的作用

繁星与夏末之夜 created at 10-27 20:10, Last updated at 11-18 17:57

看了教程以及评论区还是对if **name**=='**main**'的作用表示不解，直接不用这一行，在模块底部写上test()能实现同样效果，既不影响import，又能在命令行直接运行模块的时候显示结果

Young先森森
Created at 11-18 17:57, Last updated at 11-18 17:57

方便调试的时候用到

View Full Discuss Reply This Topic

__name__ = '__main__' 的作用

我会掏鸟蛋啊 created at 11-8 17:27, Last updated at 11-8 17:27

没明白的朋友可以看看这篇博文，看了之后瞬间懂了哦也~(●´ω｀●)φ
http://www.jb51.net/article/51892.htm

View Full Discuss Reply This Topic

导入模块与声明函数

繁星与夏末之夜 created at 10-27 20:33, Last updated at 11-6 14:31

import hello：相当于加载了一个对象吧，可以调用对象hello里的任何函数，形式为hello.xxx,比如hello.test()
from hello import test：个人感觉就相当于声明了test()函数，之后可以直接用test()函数了，如果只是导入了import hello,使用test()时必须加上hello.test(),纯个人理解，如有谬误还请指出，免得误人子弟了

王李_WL
Created at 11-6 14:31, Last updated at 11-6 14:31

看你下面有几个回复，这里和你解释一下。
在建立的模块中，用
if **name** == **main**:
    模块名称()
有两个作用。

第一个，在别的地方引用此模块时，输出的name是此模块的名称，而不是**main**这个缺省值。可以清楚的知道到底是直接运行的该模块，还是引用的该模块。

第二个，在里面可以加入一些代码。比如
if **name** == **main**:
    模块名称()
    print('正常运行')
在别的地方引用此模块时，‘正常运行’这条输出语句不会执行。
而如果我们要排查此模块有没有问题时，不需要在dos命令行用这个文件。直接打开就可以看到此模块中定义的输出语句和‘正常运行’。

View Full Discuss Reply This Topic

这样理解就好了~~~

AmadeuSTom created at 10-29 17:41, Last updated at 10-29 17:41

在此插入代码

编写一个hello模块

```
import sys

def test():
    args = sys.argv
    if len(args)==1:
        print('Hello, world!')
    elif len(args)==2:
        print('Hello, %s!' % args[1])
    else:
        print('Too many arguments!')

if name=='hello':
    test()
```

Read More

View Full Discuss Reply This Topic

__name__=‘main'的问题

冶舒悦 created at 9-26 21:14, Last updated at 10-27 20:05

一开始我看待这两行代码的时候也一头雾水，一定要先了解name函数的意义。这个脚本被执行的时候，**name** 值就是 **main** ，才会执行 main()函数。如果这个脚本是被 import 的话，**name**的值不一样。main()函数就不会被调用。这个句子用来写既能直接运行，又能给其他python程序import，提供库调用的脚本

在此插入代码

!/usr/bin/env python

* coding: utf-8 *

print **name**

运行结果就是——main——

如果这个脚本被import，则name就不再是main。所以在交互式环境下import hello 就不会有有任何反应，除非调用，即hello.test()

LSJOP
Created at 10-25 20:03, Last updated at 10-25 20:03

本來還有點亂，看了你的解釋就明白了，謝謝

繁星与夏末之夜
Created at 10-27 20:05, Last updated at 10-27 20:05

但是不用if语句，直接在模块底部加上test()，既不妨碍import，在命令行直接运行hello.py的时候，也同样可以直接显示结果，感觉if **name** =='**main**'还是有点累赘的样子

View Full Discuss Reply This Topic

get it

xfe1234 created at 10-25 13:24, Last updated at 10-25 13:24

模块是对象，并且所有的模块都有一个内置属性 name。
如果 import 一个模块，那么模块name 的值通常为模块文件名，不带路径或者文件扩展名。
如在交互环境中运行：先要导入包名,再输入**package.hello.test()**
如果在命令行中直接运行模块，在这 种情况下, name 的值将是一个特别缺省"main"。
如：在**dos**中直接定位该文件所在路径,然后运行**py hello.py**

View Full Discuss Reply This Topic

为什么我运行的结果老是这样的

期货小虾兵 created at 10-12 9:37, Last updated at 10-25 12:51

为什么我运行的结果老是这样的

```
Python 3.5.2 (v3.5.2:4def2a2901a5, Jun 25 2016, 22:18:55) [MSC v.1900 64 bit (AMD64)] on win32
>>> import hello
>>> hello.test()
Too many arguments!
```

xfe1234
Created at 10-25 12:51, Last updated at 10-25 12:51

你用的不是dos吧?

View Full Discuss | Reply This Topic

怎么setter多个值

nuomo999 created at 10-7 12:39, Last updated at 10-7 12:42

```
#!/usr/bin/env python
#coding:utf-8

class Student(object):
    def __init__(self,name):
        self.name=name
        self.m=None
        self.n=None
    @property
    def money(self):
        return self.m,self.n
    @money.setter
    def money(self,*args):
        self.m=args[0]
        self.n=args[1]
s=Student('xiaohua')
s.money=*('20','30')
print s.money
```

这么写报错

nuomo999
Created at 10-7 12:42, Last updated at 10-7 12:42

抱歉发错位置了

View Full Discuss | Reply This Topic

发表评论

Sign In to Make a Comment