**[python中的代码对象](http://www.cnblogs.com/btchenguang/archive/2012/08/29/2662571.html)**

可调用的对象是python执行环境中最重要的部分，python语句，赋值，表达式，模块等，这些

对象只是构成可执行代码块的拼图的很少的一部分，而这些代码块被称为代码对象。

每个可调用的对象的核心都是代码对象。一般来说，代码对象可以作为函数或者方法调用的一部分来

执行，也可以用exec语句或者是内建函数eval()来执行。从整体上来看，一个python模块的代码对象，

是构成该模块的全部代码。

如果要执行python代码，那么该代码必须先要转换成字节编译的代码，这才是真正的代码对象。然而，

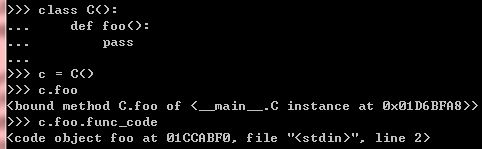
它们不包含任何关于它们执行环境的信息，这便是可调用物存在的原因，它被用来包装一个代码对象并

提供额外的信息。

udf.func\_code就是代码对象，用户定义函数

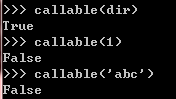


函数对象是代码对象的包装，而方法则是函数对象的包装。



python提供了很多内置函数来支持可调用/执行对象，其中exec用来执行代码对象，complie用于生成代码对象。

* callable(obj) 判断obj是否可调用。是的话，返回True；否则False



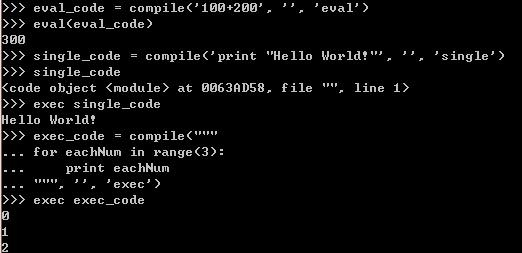
* compile(string, file, type) 从type类型中创建代码对象;file是代码存放的地方，通常设为''

最后的type有三种类型：

'eval' 可求值的表达式（和eval()一起用）

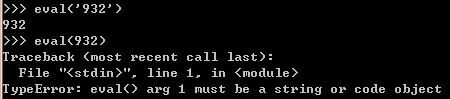
'single' 单一可执行语句（和exec一起用）

'exec' 可执行语句组（和exec一起用）



* eval(obj, globals=globals(), locals=locals())

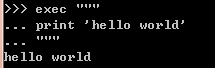
对obj进行求值，obj是已编译为代码对象的表达式，或是一个字符串表达式；全局和局部变量是可选



使用整数，反而报错，只接受字符串和代码对象。

* exec obj

执行代码对象或者是字符串，也可以是一个文件对象（已经打开的有效的python脚本）



还可以支持脚本文件对象

>>>f = open('try.py')

>>>exec f

#显示执行

>>>exec f #再次调用没反应，这是因为文件指针已经指向文件的末尾

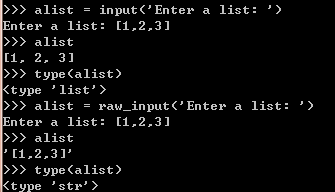
>>> f.tell() #返回当前文件指针的位置

>>> f.seek(0) #重新回到文件头

>>> exec f #又执行成功

>>> f.close() #记得关闭文件

* input(promp='') 等同于eval(raw\_input(promt=''))



可以看出input会对输入进行eval，而raw\_input只是单纯的字符串

利用exec来控制代码测试

#！/usr/bin/env python  
  
**def** foo():  
    **return** **True**  
  
**def** bar():  
    'bar() does not do much'  
    **return**   
  
foo.\_\_doc\_\_ = 'foo() does not do much'  
#把测试代码用字符串表示   
foo.tester = """  
if foo():  
    print 'PASSED'  
else:  
    print 'FAILED'  
"""  
  
# dir()取得当前命名空间的对象列表  
**for** eachAttr **in** **dir**():  
    obj = **eval**(eachAttr)  
    # 控制只运行函数类型   
    **if** **isinstance**(obj, **type**(foo)):  
        **if** **hasattr**(obj, '\_\_doc\_\_'):  
            **print** '\nFunction "%s" has a doc string :\n\t%s' % (eachAttr, obj.\_\_doc\_\_)  
        **if** **hasattr**(obj, 'tester'):  
            **print** 'Function "%s" has a tester... executin' % eachAttr  
            **exec** obj.tester  
        **else**:  
            **print** 'Function "%s" has no tester... skipping' % eachAttr  
    **else**:  
         **print** '"%s" is not a function' % eachAttr

