Python

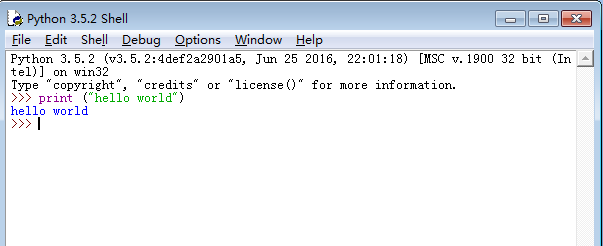
Python（英国发音：/ˈpaɪθən/ 美国发音：/ˈpaɪθɑːn/）, 是一种面向对象、解释型计算机程序设计语言，由Guido van Rossum于1989年发明，第一个公开发行版发行于1991年。 ---- 摘自 百度百科

本课程主要是讲基于python的selenium自动化测试，所以，关于python语言编程的部分，我们以够用为标准，以在工作中能用得上为导向进行学习，直接进入主题，以求达到快速入门，并能灵活运用的目的。

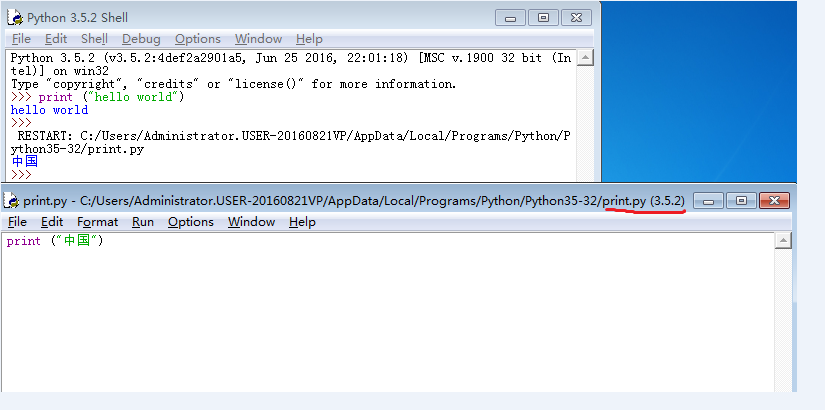
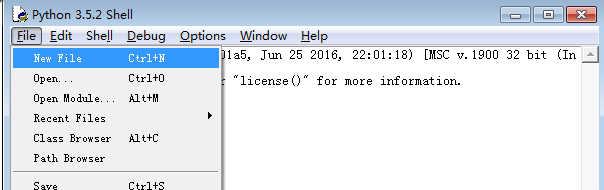
**1、输出**

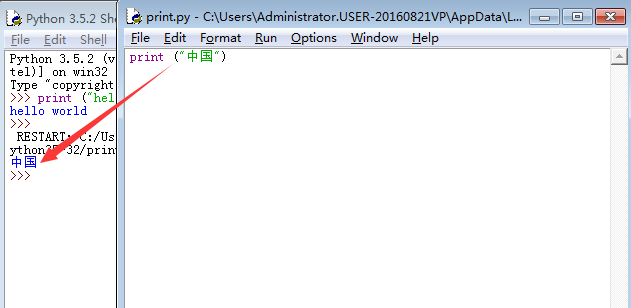
我们学习编程，都是从打印“Hello World！”开始，我们也从这里开始。

在python中，打印信息需要用到python的库函数：print()



通常，我们编程的时候，都要新建一个文件，并且保存成.py格式的文件，然后在文件里面编程。





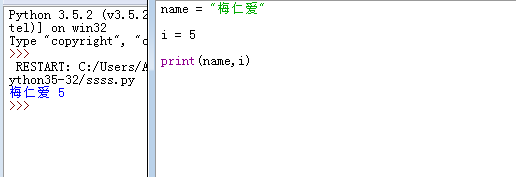
**2、变量**

在python中，定义一个变量很简单，直接写就好，不用像C语言或java那样先声明，如：

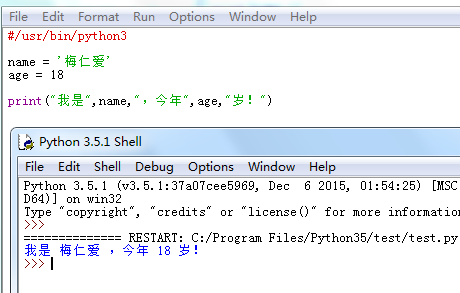
name = "梅仁爱"

i=5

name和 i 就是个变量，"梅仁爱"，和 5 就是变量的值。输出变量的值时，可以在print后直接接变量名，如：

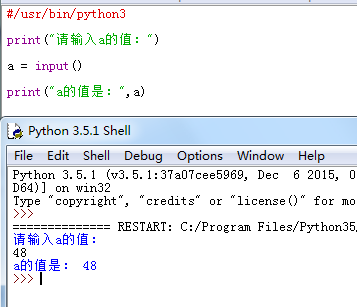


变量与其他字符串一起输出：



**3、输入**

一个程序要能实现人机交互功能，必须要能够接受从键盘上输入的数据。在python中，接收由键盘输入的数据的函数是 input()



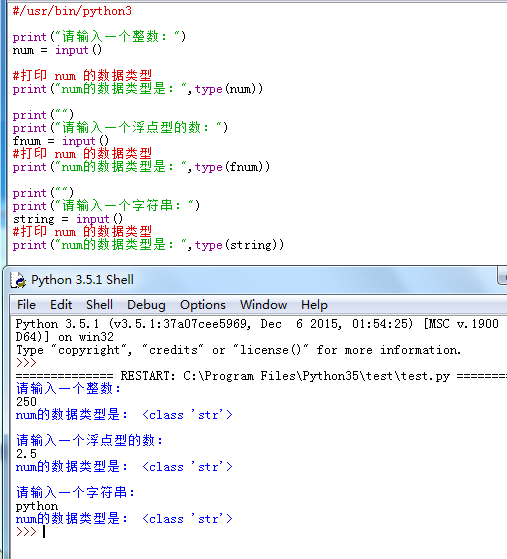
提示：input()函数接受用户从键盘输入的值，然后把值赋给变量a

**4、数据类型转换**

在python中，常见的数据类型有三种：整型（int），浮点型(float) -- 我们可以理解为小数，字符型(str)，列表，元祖，字典等。在这里，我们主要学习在符合特定情况下，可以相互转换的三种数据类型，整型、浮点型和字符型。

1）、查看数据类型: type(变量)

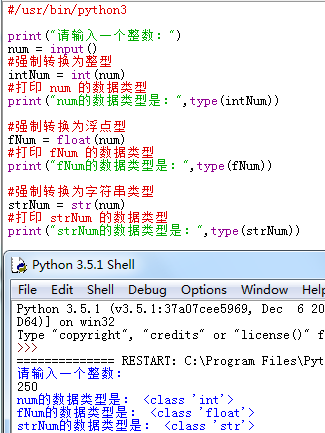
我们先来看看如何查看一个数据的数据类型。type()是查看数据类型的函数，现在我们查看一下input函数的返回类型



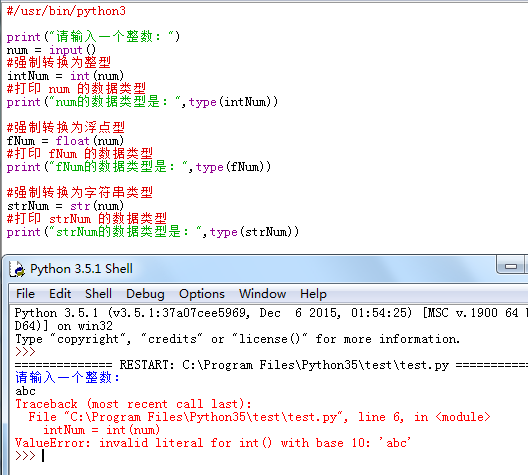
从上面的结果来看,不管我们输入哪种数据类型,input()函数默认将输入的数据转换为字符串类型。怎样才能将输入的整数换成为整数类型呢？这时，我们需要用到数据类型的强制转换。

格式：数据类型(变量)

例如：

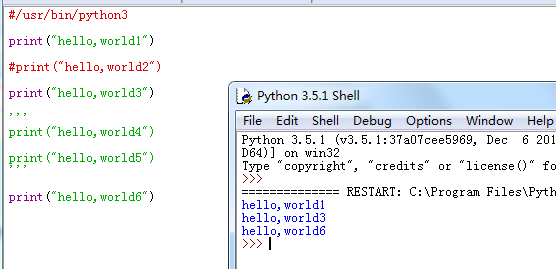


注意：字母是无法转换为整型和浮点型的数据类型的。



**5、注释**

在python中，注释代码，可以在代码前加上符号 # ，如果要注释多行，就要用三个单引号括住 '''，如以下：

****

被注释了的代码是不会不运行的。

**切记：单引号只能是三个，不能多也不能少。**

练习: 从键盘输入 a 和 b的值，并输出他们的和。(提示：需要把从键盘输入的数据转换为整型 或 浮点型)

**6、if判断语句**

**if判断语句的语法:**

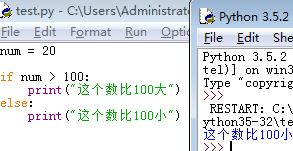
if 条件:

语句1

else:

语句2

当if后面的条件成立，则执行语句1；否则，执行语句2。



**if .. elif .. else 语句的语法：**

if 条件1:

语句1

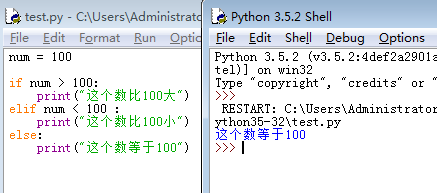
elif 条件2:

语句2

else:

语句3

当条件1成立，则执行语句1，然后跳出if判断；当条件2满足，则执行语句2，然后跳出if判断；如果条件1和条件2都不满足，则执行语句3。



**练习：**

1. 从键盘输入一个数,判断该数大于100还是小于100。

**2)**从键盘输入 y 或 n，如果是y，则打印：“you answer right”；如果输入n，则打印：" you answer wrong"；否则，打印：“input wrong”

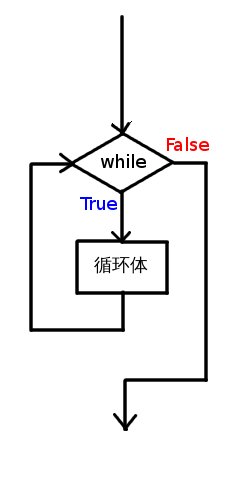
**7、while循环语句**

同if一样，while也是一种控制流语句，另外它也被称作循环语句。

**语法为**：

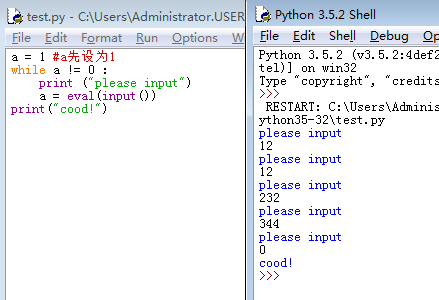
while 条件:

　循环执行的语句



程序执行到while处，“当”条件为True时，就去执行while内部的代码，“当”条件为False时，就跳过。

同if一样，注意冒号，注意缩进。



练习：

1、接着上面那个小游戏，如果输入不是y或n，让程序一直运行下去，直到输入正确的答案。

2、输出1到100

**8、for循环语句**

接下来，跟大家讲解下另外一种常用的循环语句，for循环。

语法：

for ... in ...

同while一样，for循环可以用来重复做一件事情。在某些场景下，它比while更好用。比如，输出1到100。我们用while来做，需要有一个值来记录已经做了多少次，还需要在while后面判断是不是到了100。

如果用for循环，则可以这么写：

for i in range(1, 101):

print(i)

解释一下，range(1, 101)表示从1开始，到101为止（不包括101），取其中所有的整数。for i in range(1, 101)就是说，把这些数，依次赋值给变量i。相当于一个一个循环过去，第一次i = 1，第二次i = 2，……，直到i = 100。每一次循环，i的值都会自增1。当i = 101时跳出循环。所以，当你需要一个循环10次的循环，你就只需要写：

for i in range(1, 11)

或者

for i in range(0, 10)

区别在于前者i是从1到10，后者i是从0到9。当然，你也可以不用i这个变量名。

比如一个循环n次的循环：

for count in range(0, n)

for循环的本质是对一个序列中的元素进行递归。

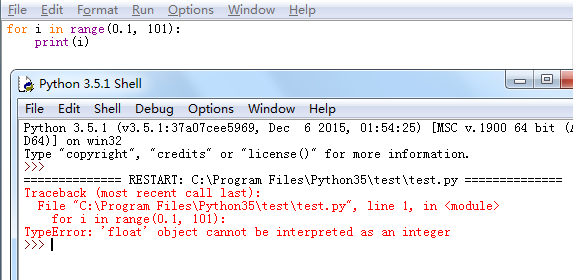
for i in range(a, b)

从a循环至b-1

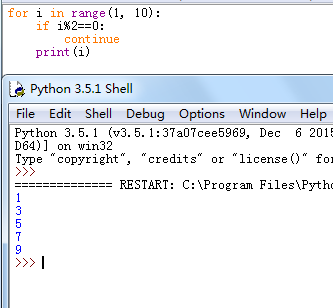
附：

range(start，end，scan)

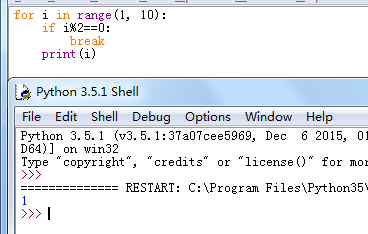
range()函数，start表示开始位置，end表示结束位置，scan表示每一次循环的步长。注意：range()函数括号里面的值，只能是整数，不能为小数



补充：break（跳出循环），continue（结束本次循环）



解释：这里的%，是指求余数。程序的意思是，当i除以2的余数等于0时，退出本次循环，程序不再往下执行，进入下一次循环继续执行。



解释：程序的意思是，当i除以2的余数等于0时，退出循环，程序不再执行循环里的所有内容。

练习：

1、计算： 1 + 2 + ... + 100 的和。

2、求：1/1 + 1/3 + 1/5 + ... + 1/999 的和。

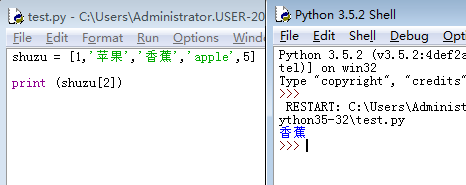
**9、列表**

什么是列表呢？列表，在C语言或java里面叫数组，把它看作是一个类型的所有数据的一个集合。但是在python里则不一样，列表可以存放各种类型的数据，也就是说，可以把列表理解为很多种数据的集合。比如，一个购物清单里面，会有很多种商品，我们可以把这个购物清单理解为是一个列表。大家看看一下下面这个例子：

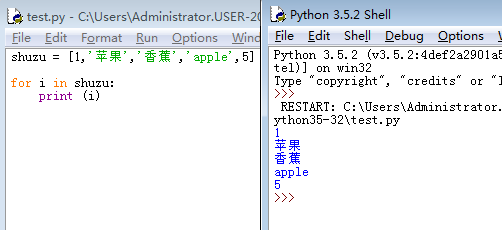
shuzu = [1,'苹果','香蕉','apple',5]

这就定义了一个列表，列表里面包含5个元素，分别是：1,'苹果', '香蕉','apple',5 。这5个元素，都是用[]括号括起来。[]括号里面的每一个元素用逗号（,）隔开。在python里面，就是用这样的一个方式定义一个列表。

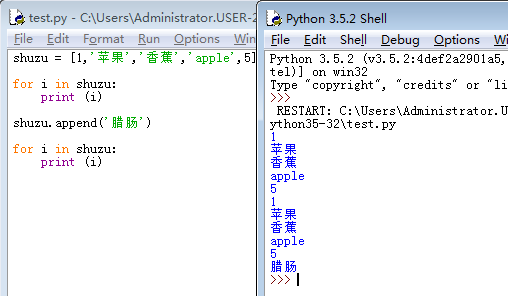
怎么使用列表的值呢？这时，我们就要用到列表的下标方式去访问具体的值了。列表的第一个值的下标是0，第二个是1，以此类推。也就是说，访问列表的下标，是从0开始的。比如：shuzu[1]，它的值就是“苹果”。



怎样把列表的所有值的遍历出来呢？这时，我们可以使用for循环语句。如下：



如果我定义的列表，写漏了某个元素，怎么办呢？这时，我们可以使用python的内置函数append()向数组追加元素。



修改列表中的某一个元素，只需要直接给那个元素赋值就可以了：

shuzu[0] = 123

删除列表中的某一个元素，要用到del：

del shuzu[0]

**练习：**

定义一个列表，通过键盘向列表输入N个整数，再求他们的和

**10、函数**

什么是函数呢？数学上的函数，是指给定一个输入，就会有唯一输出的一种对应关系。编程语言里的函数跟这个意思差不多，但也有不同。就是你编写了一些语句，为了方便使用这些语句，把这些语句组合在一起，给它起一个名字。使用的时候只要调用这个名字，就可以实现语句组的功能了。它有可能需要输入，有可能会返回输出。

举一个现实中的场景：我们去餐厅吃饭，跟服务员点了菜，过了一会儿，服务员把做好的菜端上来。餐厅的厨房就可以看作是一个函数，我们点的菜单，就是给这个函数的参数；厨师在厨房里做菜的过程就是这个函数的执行过程；做好的菜是返回结果，返回到我们的餐桌上。

我们之前已经用到过python里内建的函数，比如input和range。以range(1,10)为例，range是这个函数的名称，后面括号里的1和10是range需要的参数。它有返回结果，就是一个从1到9的序列。再来看input()，括号里面没有，表示我们没有给参数。函数执行过程中，需要我们从控制台输入一个值。函数的返回结果就是我们输入的内容。

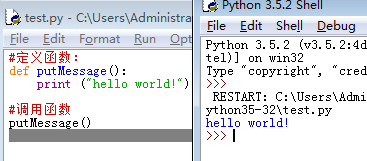
要使用函数，首先要定义好函数。函数分两种，一种是无参函数，一种是有参函数。

**无参函数定义的语法如下**：

def 函数名():

函数体

例：定义一个无参函数，并打印“Hello,world!”



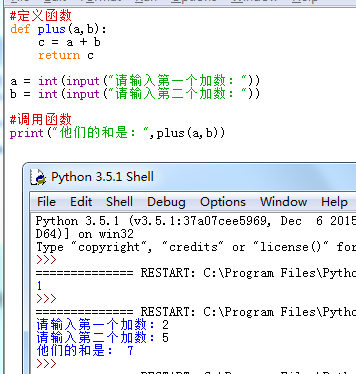
无参函数，在工作中的实际意义不大，因为，不能按我们的要求来做事。

**有参函数定义的语法如下：**

def 函数名(参数列表):

函数体

例子：设计一个函数，并实现求两个数的和。



**练习：**

定义一个函数，比较两个数(a,b)的大小，并输出他们的结果，如：a>b，a<b，a=b

**11、类**

面向对象技术简介：

**类(Class):** 用来描述具有相同的属性和方法的对象的集合。它定义了该集合中每个对象所共有的属性和方法。

**类变量：**类变量，就是在类里面定义的变量，在整个实例化的对象中是公用的。类变量定义在类中且在函数体之外。类变量通常不作为实例变量使用。

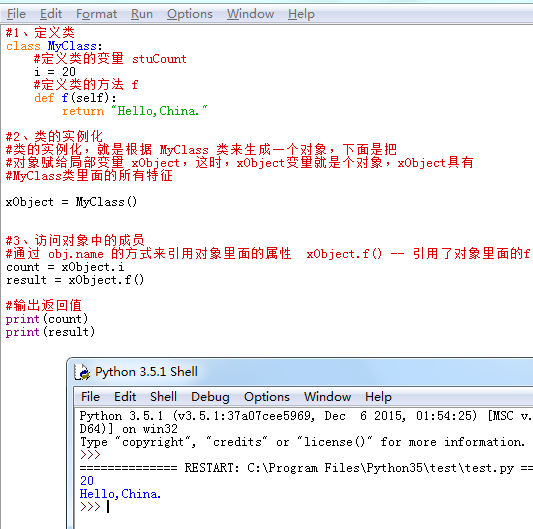
**方法：**类中定义的函数，在类里面，我们把函数叫做方法。

**对象：**通过类定义的数据结构实例。对象包括两个数据成员（类变量和实例变量）和方法。

---面向对象的脚本编写方式，极大地实现的代码的复用性。

(以上是官方的解释，是不是很抽象，抽象得要吐血身亡了？其实，我们可以这么理解类：类就是一个概念，比如：汽车，帅哥，美女，仅仅只是一个概念。汽车有汽车的特征，帅哥也有帅哥的特征，那么，具有什么样的特征才叫美女呢？比如，“胳膊比你大腿还粗，犀利哥式的发型，声如洪钟，走起路来地动山摇。”哈哈，各花入各眼，你喜欢就好。这些“美女”的特征，我们可以理解为是美女这个类的属性。类变量和方法是类里面的两个属性。对象就是根据这些属性制造出来的一个实体，比如，你同桌，如果符合帅哥或者美女的特征，那么，他/她就是帅哥或者美女这个类中的一个实实在在的，看得见，摸得着的对象)。

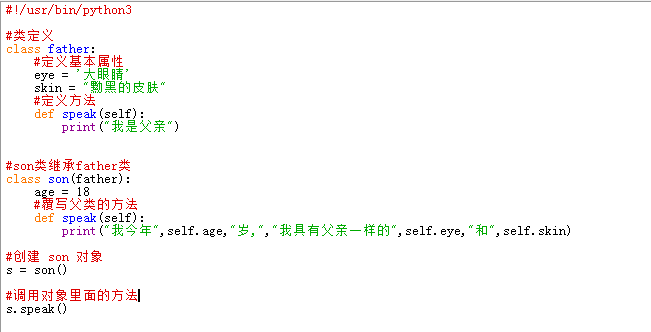
**类定义、实例化和对象成员的访问**



在定义 f()方法是，我们在括号里面也了个self, 类的方法与普通的函数只有一个特别的区别——它们必须有一个额外的**第一个参数名称**, 按照惯例它的名称是 self，在调用这个方法时，无需向这个参数传递值。

**继承**

Python 同样支持类的继承，如果一种面向对象的语言不支持继承，类就没有什么意义。继承，可以产生各种特点的子类，也就能产生不一样的对象。帅哥千千万，但是没有哪两个是一模一样的,原因就是,帅哥们都继承了帅哥应该有的特征,而又具备自己的特色。



在定义son类的时候，继承了father类，使得son类具有的father类的所有属性，因此，可以在son类中调用father类的eye和skin属性。

关于类，对象，继承，还有更复杂的写法，本课程只是为我们后面编写selenium自动化测试脚本服务，所以，不一一展开探讨。有兴趣，想把python学得更深入些的同学，可以查看相关的资料。

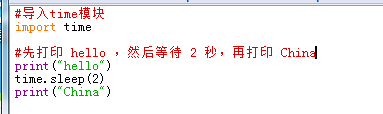
附加：变量分为全局变量和局部变量，有兴趣的同学可以课后查阅相关资料。

**12、模块**

模块，在C语言中成为“头文件”。python自带了功能丰富的标准库，另外还有数量庞大的各种第三方库。使用这些功能的基本方法就是使用模块，通过模块，可以重用别的程序中的代码。

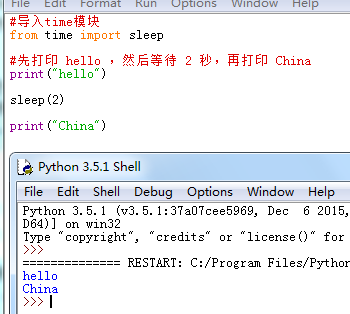
**导入标准库里面的模块：**

模块可以理解为是一个包含了函数和变量的py文件。在你的程序中引入了某个模块，就可以使用其中的函数和变量。比如要引用模块math，就可以在文件最开始的地方用import math来引入。在调用math模块中的函数时，必须这样引用：模块名.函数名



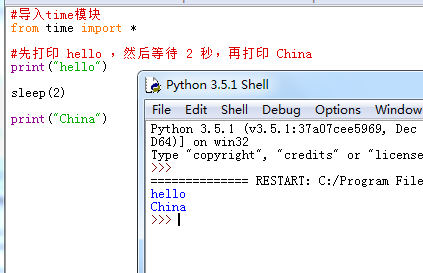
有时候我们只需要用到模块中的某个函数，只需要引入该函数即可，此时可以通过语句：

　　from 模块名 import 函数名1,函数名2....



如果想把一个模块里面的所有方法都导进来，可以这样写：

from 模块名 import \*

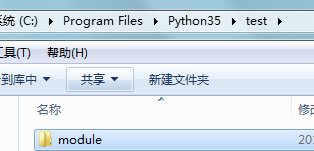


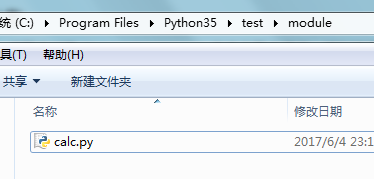
对于刚接触python的同学来说，很难确定哪个方法是哪个模块下的，不知道什么时候要用什么方法。这是正常的，我们网上找了一份《Python标准库》中文版.pdf》，大家可以参考下，或者查看 <http://www.pythondoc.com/pythontutorial3/stdlib.html#tut-os-interface> 这上面的内容，这里列举是python标准库的所有内容。

**自定义模块：**

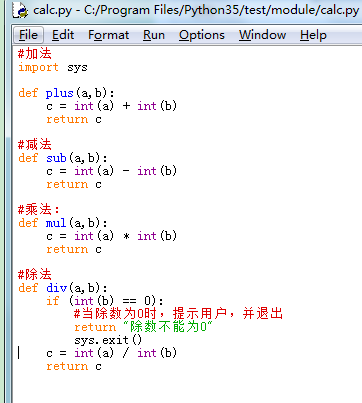
自定义模块的意思是，模块里面的内容要自己写，然后在需要使用模块里面的方法时，再把模块导进来用。自定义模块，很多同学学起来都是一脸懵逼，大家先看完下面的例子，看完后再跟着步骤来做，做完后在回头想想，就有醍醐灌顶的觉悟了。

第一步：新建一个文件夹，把自定义的模块(calc.py)都放这个文件夹里面。自定义的模块，其实就是一个.py的文件：

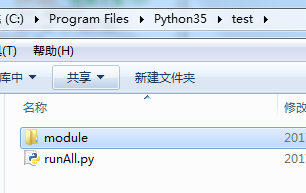


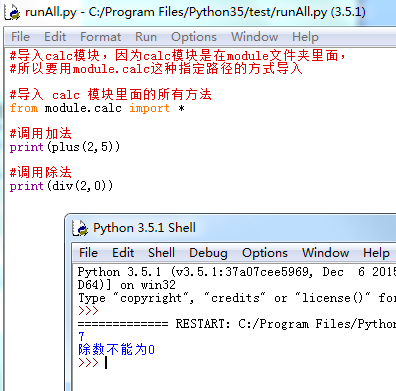


第二步：在calc.py模块文件里面输入以下内容



第三步：新建一个runAll.py文件，并导入自定义的calc模块使用





**13、异常**

Python用异常对象（exceptionobject）来表示异常情况。遇到错误后，会引发异常。如果异常对象并未被处理或捕捉，程序就会用所谓的回溯（Traceback，一种错误信息）终止执行。

在实际的脚本开发中，有时候程序并不会像我们设计它时那样工作，它也有“生病”的时候，那么我们可以通过异常处理机制，有预见性地获得这些病症，并开出药方。比如，对一个正常人，大冬天的洗冷水澡，那么就有可能感冒，我们可以事先在洗冷水澡时准备好感冒药，假如真感冒了，就立刻吃药。

比如我们写了一句：

print (int('0.5'))

运行后程序得到错误提示：

print int('0.5')

ValueError: invalid literal for int() with base 10: '0.5'

意思是，在test.py这个文件，第1行，print int('0.5')这里，你拿了一个不是10进制能够表示的字符，我没法把它转成int值。

上面的错误可以避免，但在实际的应用中，有很多错误是开发者无法控制的，例如用户输入了一个不合规定的值，或者需要打开的文件不存在。这些情况被称作“异常”，一个好的程序需要能处理可能发生的异常，避免程序因此而中断。

我们把上述程序加入异常机制

try :

print (int('0.5'))

print ("no excpet")

except :

print ("err")

print ("end")

在python中，可以使用try...except语句来处理异常。做法是，把可能引发异常的语句放在try-块中，把处理异常的语句放在except-块中。

当程序在try内部引发异常时，会跳过try中剩下的代码，直接跳转到except中的语句处理异常。于是输出了“err.”。如果print (int('0.5'))正常，而不会去执行except中的语句。

但无论如何，整个程序不会中断，最后的“end”都会被输出。

在try...except语句中，try中引发的异常就像是扔出了一只飞盘，而except就是一只灵敏的狗，总能准确地接住飞盘。

**面试题练习：**

1. 从键盘向列表中输入n个整数，并按由小到大进行排序。(即冒泡排序)

提示：

冒泡排序（BubbleSort）的基本概念是：依次比较相邻的两个数，将小数放在前面，大数放在后面。即首先比较第1个和第2个数，将小数放前，大数放后。然后比较第2个数和第3个数，将小数放前，大数放后，如此继续，直至比较最后两个数，将小数放前，大数放后。重复以上过程，仍从第一对数开始比较（因为可能由于第2个数和第3个数的交换，使得第1个数不再大于第2个数），将小数放前，大数放后，一直比较到最小数前的一对相邻数，将小数放前，大数放后，第二趟结束，在倒数第二个数中得到一个新的最小数。如此下去，直至最终完成排序。

　　由于在排序过程中总是小数往前放，大数往后放，相当于气泡往上升，所以称作冒泡排序。

假定有下标为0～n的n+1个数的序列，要求按升序排列，实现的步骤如下：

(1)从第0个元素开始与后一个比较，如果比后大两元素交换，依次比较到第n个元素，最终将最大的数换入第n个元素中，a(n)不动；

(2)重复(1) ，依次比较到第n-1个元素，最终将最大的数换入第n-1个元素中，a(n-1)不动；

(3)重复(1) ，依次比较到第n-2个元素，最终将最大的数换入第n-2个元素中，a(n-2)不动。

1. 用python写出乘法口诀表