模块: 函数 或者 变量的集合就是模块， python是由一系列的模块组成的，每个模块就是一个py为后缀的文件，同时模块也是一个命名空间，从而避免了变量名称冲突的问题

导入模块

import 语句

```
import module1[, module2[,... moduleN]
```

1：import a(文件名)
访问模块内的变量以及方法需要 使用 模块名.函数 的形式
2：import b(文件名) as a (别名)
使用 a.函数的 方式访问， 使用 引用 b 访问不了

from...import 语句

Python的from语句让你从模块中导入一个指定的部分到当前命名空间中，语法如下:

```
from modname import name1[, name2[, ... nameN]]
```

3：from a(模块) import b(函数),c(变量)
这相当于将 b，c 复制进 本文件，可以直接使用 b 和 c

深入模块

模块除了方法定义，还可以包括可执行的代码。这些代码一般用来初始化这个模块。这些代码只有在第一次被导入时才会被执行。
每个模块有各自独立的符号表，在模块内部为所有的函数当作全局符号表来使用。
所以，模块的作者可以放心大胆的在模块内部使用这些全局变量，而不用担心把其他用户的全局变量搞花。
从另一个方面，当你确实知道你在做什么的话，你也可以通过 modname.itemname 这样的表示法来访问模块内的函数。
模块是可以导入其他模块的。在一个模块（或者脚本，或者其他地方）的最前面使用 import 来导入一个模块，当然这只是一个惯例，而不是强制的。被导入的模块的名称将被放入当前操作的模块的符号表中。
还有一种导入的方法，可以使用 import 直接把模块内（函数，变量的）名称导入到当前操作模块。比如:

```
>>> from fibo import fib, fib2
>>> fib(500)
1 1 2 3 5 8 13 21 34 55 89 144 233 377
```

这种导入的方法不会把被导入的模块的名称放在当前的字符表中（所以在这个例子里面，fibo 这个名称是没有定义的）。

这还有一种方法，可以一次性的把模块中的所有（函数，变量）名称都导入到当前模块的字符表:

```
>>> from fibo import *
>>> fib(500)
1 1 2 3 5 8 13 21 34 55 89 144 233 377
```

这将把所有的名字都导入进来，但是那些由单一下划线（_）开头的名字不在此例。大多数情况，Python程序员不使用这种方法，因为引入的其它来源的命名，很可能覆盖了已有的定义。

一个模块被导入的时候，会运行，为了限制可以使用 __name__属性

说明： 每个模块都有一个__name__属性，当其值是'__main__'时，表明该模块自身在运行，否则是被引入。

dir() 函数

内置的函数 `dir()` 可以找到模块内定义的所有名称。以一个字符串列表的形式返回：

`dir(模块名)`

模块名为空 ，返回当前模块所定义的所有名称