Python中的模块

在Python中有一个概念叫做模块（module）。

说的通俗点：模块就好比是工具包，要想使用这个工具包中的工具(就好比函数)，就需要导入这个模块

比如我们经常使用工具 random，就是一个模块。使用 import random 导入工具之后，就可以使用 random 的函数。

导入模块

<1> 导入模块有五种方式

- import 模块名
- from 模块名 import 功能名
- from 模块名 import *
- import 模块名 as 别名
- from 模块名 import 功能名 as 别名

下面来挨个的看一下。

<2>import

在Python中用关键字 `import` 来引入某个模块，比如要引入系统模块 math，就可以在文件最开始的地方用 `import math` 来引入。

语法:

```
import 模块1,模块2,...  # 导入方式

模块名.函数名()  # 使用模块里的函数
```

- 想一想:

> 为什么必须加上模块名调用呢?

- 答:

> 因为可能存在这样一种情况：在多个模块中含有相同名称的函数，此时如果只是通过函数名来调用，解释器无法知道到底要调用哪个函数。所以如果像上述这样引入模块的时候，调用函数必须加上模块名

示例：

```
import math

#这样才能正确输出结果
print math.sqrt(2)

#这样会报错
print(sqrt(2))
```

<3>from…import

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，此时可以用下面方法实现：

```
from 模块名 import 函数名1,函数名2....
```

不仅可以引入函数，还可以引入一些全局变量、类等

- 注意:

> 通过这种方式引入的时候，调用函数时只能给出函数名，不能给出模块名，但是当两个模块中含有相同名称函数的时候，后面一次引入会覆盖前一次引入。也就是说假如模块A中有函数function( )，在模块B中也有函数function( )，如果引入A中的function在先、B中的function在后，那么当调用function函数的时候，是去执行模块B中的function函数。

例如，要导入模块fib的fibonacci函数，使用如下语句：

```
from fib import fibonacci
```

注意

- 不会把整个fib模块导入到当前的命名空间中，它只会将fib里的fibonacci单个函数引入

<4>from … import *

把一个模块的所有内容全都导入到当前的命名空间也是可行的，只需使用如下声明：

```
from modname import *
```

注意

- 这提供了一个简单的方法来导入一个模块中的所有项目。然而这种声明不该被过多地使用。

<5> as 别名

```
In [1]: import time as tt  # 导入模块时设置别名为 tt

In [2]: time.sleep(1)
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-2-07a34f5b1e42> in <module>()
----> 1 time.sleep(1)

NameError: name 'time' is not defined

In [3]:

In [3]: tt.sleep(1)  # 使用别名才能调用方法

In [4]:

In [4]: from time import sleep as sp  # 导入方法时设置别名

In [5]: sleep(1)
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-5-82e5c2913b44> in <module>()
----> 1 sleep(1)

NameError: name 'sleep' is not defined

In [6]:

In [6]: sp(1)  # 使用别名才能调用方法

In [7]:
```