

课程： 模块和包

目标

- 了解模块
- 导入模块
- 制作模块
- `__all__`
- 包的使用方法

一. 模块

Python 模块(Module)，是一个 Python 文件，以 .py 结尾，包含了 Python 对象定义和Python语句。

模块能定义函数，类和变量，模块里也能包含可执行的代码。

1.1. 导入模块

1.1.1 导入模块的方式

- `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`

1.1.2 导入方式详解

1.1.2.1 import

- 语法

```
1  # 1. 导入模块
2  import 模块名
3  import 模块名1, 模块名2...
4
5  # 2. 调用功能
6  模块名.功能名()
```

- 体验

```
1 import math
2 print(math.sqrt(9)) # 3.0
```

1.1.2.2 from..import..

- 语法

```
1 from 模块名 import 功能1, 功能2, 功能3...
```

- 体验

```
1 from math import sqrt
2 print(sqrt(9))
```

1.1.2.3 from .. import *

- 语法

```
1 from 模块名 import *
```

- 体验

```
1 from math import *
2 print(sqrt(9))
```

1.1.2.4 as定义别名

- 语法

```
1 # 模块定义别名
2 import 模块名 as 别名
3
4 # 功能定义别名
5 from 模块名 import 功能 as 别名
```

- 体验

```
1 # 模块别名
2 import time as tt
3
4 tt.sleep(2)
5 print('hello')
6
7 # 功能别名
8 from time import sleep as sl
9 sl(2)
10 print('hello')
```

1.2. 制作模块

在Python中，每个Python文件都可以作为一个模块，模块的名字就是文件的名字。也就是说自定义模块名必须要符合标识符命名规则。

1.2.1 定义模块

新建一个Python文件，命名为 `my_module1.py`，并定义 `testA` 函数。

```
1 def testA(a, b):
2     print(a + b)
```

1.2.2 测试模块

在实际开中，当一个开发人员编写完一个模块后，为了让模块能够在项目中达到想要的效果，这个开发人员会自行在py文件中添加一些测试信息，例如，在 `my_module1.py` 文件中添加测试代码。

```
1 def testA(a, b):
2     print(a + b)
3
4
5 testA(1, 1)
```

此时，无论是当前文件，还是其他已经导入了该模块的文件，在运行的时候都会自动执行 `testA` 函数的调用。

解决办法如下：

```
1 def testA(a, b):
2     print(a + b)
3
4 # 只在当前文件中调用该函数，其他导入的文件内不符合该条件，则不执行testA函数调用
5 if __name__ == '__main__':
6     testA(1, 1)
```

1.2.3 调用模块

```
1 import my_module1
2 my_module1.testA(1, 1)
```

1.2.4 注意事项

如果使用 `from .. import ..` 或 `from .. import *` 导入多个模块的时候，且模块内有同名功能。当调用这个同名功能的时候，调用到的是后面导入的模块的功能。

- 体验

```
1 # 模块1代码
2 def my_test(a, b):
3     print(a + b)
4
5 # 模块2代码
6 def my_test(a, b):
7     print(a - b)
8
9 # 导入模块和调用功能代码
10 from my_module1 import my_test
11 from my_module2 import my_test
12
13 # my_test函数是模块2中的函数
14 my_test(1, 1)
```

1.3. 模块定位顺序

当导入一个模块，Python解析器对模块位置的搜索顺序是：

1. 当前目录
2. 如果不在当前目录，Python则搜索在shell变量PYTHONPATH下的每个目录。
3. 如果都找不到，Python会察看默认路径。UNIX下，默认路径一般为/usr/local/lib/python/

模块搜索路径存储在system模块的sys.path变量中。变量里包含当前目录，PYTHONPATH和由安装过程决定的默认目录。

- 注意
 - 自己的文件名不要和已有模块名重复，否则导致模块功能无法使用
 - 使用from 模块名 import 功能的时候，如果功能名字重复，调用到的是最后定义或导入的功能。

1.4. __all__

如果一个模块文件中有__all__变量，当使用from xxx import *导入时，只能导入这个列表中的元素。

- my_module1模块代码

```
1  __all__ = ['testA']
2
3
4  def testA():
5      print('testA')
6
7
8  def testB():
9      print('testB')
```

- 导入模块的文件代码

```
1  from my_module1 import *
2  testA()
3  testB()
```

```
C:\Users\黑马程序员\AppData\Local\Programs\Python\Python37\python.exe
testA
Traceback (most recent call last):
  File "C:/Users/黑马程序员/Desktop/code/模块.py", line 5, in <module>
    testB()
NameError: name 'testB' is not defined

Process finished with exit code 1
```

二. 包

包将有联系的模块组织在一起，即放到同一个文件夹下，并且在这个文件夹创建一个名字为__init__.py文件，那么这个文件夹就称之为包。

2.1 制作包

[New] — [Python Package] — 输入包名 — [OK] — 新建功能模块(有联系的模块)。

注意：新建包后，包内部会自动创建 `__init__.py` 文件，这个文件控制着包的导入行为。

2.1.1 快速体验

1. 新建包 mypackage
2. 新建包内模块：my_module1 和 my_module2
3. 模块内代码如下

```
1 # my_module1
2 print(1)
3
4
5 def info_print1():
6     print('my_module1')
```

```
1 # my_module2
2 print(2)
3
4
5 def info_print2():
6     print('my_module2')
```

2.2 导入包

2.2.1 方法一

```
1 import 包名.模块名
2
3 包名.模块名.目标
```

2.2.1.1 体验

```
1 import my_package.my_module1
2
3 my_package.my_module1.info_print1()
```

2.2.2 方法二

注意：必须在 `__init__.py` 文件中添加 `__all__ = []`，控制允许导入的模块列表。

```
1 from 包名 import *
2 模块名.目标
```

2.2.2.1 体验

```
1 from my_package import *
2
3 my_module1.info_print1()
```

三. 总结

- 导入模块方法

```
1 import 模块名
2
3 from 模块名 import 目标
4
5 from 模块名 import *
```

- 导入包

```
1 import 包名.模块名
2
3 from 包名 import *
```

- `__all__ = []`：允许导入的模块或功能列表