

## 1. 模块的概念

在 Python 中,一个.py 文件就是一个模块,其中模块名称就是文件名称

使用模块可以提高代码的维护性,如果把所有的功能都放到一个.py 文件中的话,以后维护起来特别麻烦,我们可以按照功能分成不同的模块进行管理

使用模块可以提高代码的复用性,对于 python 中的模块,我们编写完成以后,在别的.py 文件中需要用到某个功能,只需要把对应的模块导入到当前的 python 文件中即可

使用模块可以避免函数名,变量名,类名的冲突.在不同的模块中,可以有相同的函数名称和变量名称

使用包(目录)来组织模块的管理方法,可以避免模块名相同的问题

比如:user.py 文件的模块名称为 `user`,dateutil.py 文件的模块名称为 `dateutil`

模块名和包名的命名规则:基本上使用小写字母,如果是多个单词也是全部小写

## 2. 模块的引入

如果需要应用到别人的一个功能或者模块,我们需要先引入对应的模块才可以使用上

方式一:

引入: `import 模块名`

使用: `模块名.函数名()`

main.py

```
# 导入模块
import random

# 调用模块中的方法
num = random.randint(1, 10)
print(num)
```

方式二:

引入: `from 模块名 import 函数名 1,函数名 2....` 如果想引入一个模块下的所有的功能,我们可以使用 `from 模块名 import *`

使用: `函数名()`

```
# 方式二: 使用 from 模块名 import 函数名
from log import test1

# 直接通过函数名称调用
test1()
```

两种方式的区别:

方式一可以使用到一个模块下面所有的函数,变量,类等

方式二只能使用导入的某个特定的函数,变量,类等,如果需要导入多个,则多个之间使用逗号“,” 隔开,如果需要导入所有的,可以使用 `from log import *`

## 3. 模块制作

### 3.1. 定义自己的模块

在 python 总,每个 python 文件都是一个模块,模块的名字就是文件的名字,直接创建一个 py 文件,在文件中定义相关的函数即可,注意在定义函数,变量名称的时候,尽量不要和内置的函数,变量名称一样

```
def add(*args):
    total = 0
    for value in args:
        total += value
    return total
```

### 3.2. 调用自己定义的模块




```
import mathutil

total = mathutil.add(1, 2, 3, 4)
print(total)
```

```
from mathutil import add

total = add(1, 2, 3)
print(total)
```

在导入一个自定义模块的 python 文件的时候,其实就是执行对应的 python 文件,而且在第一次导入的时候,为了提高运行效率,会把模块的 python 文件编译一个对应的文件

 log.cpython-36.pyc	2018/4/25 0:10	Compiled Pytho...	1 KB
 mathutil.cpython-36.pyc	2018/4/25 0:10	Compiled Pytho...	1 KB
 util.cpython-36.pyc	2018/4/25 0:10	Compiled Pytho...	1 KB

适用于C解释器运行的程序

### 3.3. 对自己定义的模块进行功能测试

```
def add(*args):
    total = 0
    for value in args:
```

```
        total += value
    return total

if __name__ == '__main__':
    value = add(1, 2)
    print(value)
```

### 3.4. \_\_all\_\_属性

在模块中的\_\_all\_\_属性通常放入文件中的第一行,其值是一个列表对象,主要是显示 import 模块 from \* 导入的函数和变量,只有在\_\_all\_\_ 列表中指定的才会被 import 模块 from \* 导入进去

```
__all__ = ["add", "test1"]

def add(*args):
    total = 0
    for value in args:
        total += value
    return total

def test1():
    print("...mathutil...test1")

def test2():
    print("...mathutil...test2")
#测试代码
if __name__ == '__main__':
    test1()
    test2()
    print(add(1, 2, 3))
```

调用代码

```
from mathutil import *

test1()
# 找不到 test2 函数,因为不在__all__ 这个属性中定义
# test2()
```

## 4. 包的管理

当我们有很多模块的时候,我们可以分门别类的相同的模块存档到一个目录中去,方便我们对模块的维护,

而且,在不同的目录下面可以有相同的模块名,也解决了模块名重复的问题.这样的目录我们称之为包,一个目录要当做包,必须要有一个文件 `__init__.py` 文件

一个模块就是一个 `py` 文件,其中包含函数和类,包是用来更好的管理模块

在一个包下面必须要有一个 `__init__.py` 文件

1 用来初始化模块或者子包(类似于 `class` 中的 `init` 初始化函数)

2 通过 `__init__.py` 明确的声明这是一个包结构

3 可以使用 `import` 或者 `from ... import` 的方式导入需要使用的模块

调用有包管理的模块:

方式一: 使用 `import` 包名.模块名 导入需要的模块

```
import util.mathutil
util.mathutil.test2()
```

方式二: 使用 `from import`

```
from util import mathutil
mathutil.test1()
```

## 5. 模块的发布

当我们的模块测试通过,需要把模块发布给别人使用,别人安装我们提供的模块即可使用模块中的功能

### 5.1. 制作一个 `setup.py` 文件

```
from distutils.core import setup
setup(
    name = 'util',
    version = '1.0',
    author = 'wolfcode',
    author_email = 'wolfcode@wolfcode.cn',
    url = 'http://www.wolfcode.cn',
    download_url = 'http://www.wolfcode.cn',
    description = 'util module',
    py_modules=['util.mathutil', 'util.stringutil']
)
```

### 5.2. 构建模块

```
python setup.py build
```

### 5.3. 生成发布压缩包

```
python setup.py sdist
```

## 6. 模块的安装

### 6.1. 模块查找的路径

```
import sys
from urllib import request

if __name__ == '__main__':
    for item in sys.path:
        print(item)
    # 查找模块所在的路径
    print(request.__file__)
```

### 6.2. pip 在线安装

pip install requests 安装一个 http 模块 request

pip uninstall requests 卸载模块 requests

```
C:\Users\heshengjun>pip install requests
Collecting requests
  Downloading https://files.pythonhosted.org/packages/49/df/50aa1999ab9bde74656c2919d9c0c085fd2b3775fd3eca826012bef76d8c
/requests-2.18.4-py2.py3-none-any.whl (88kB)
    100% |#####| 92kB 420kB/s
Requirement already satisfied: certifi>=2017.4.17 in c:\users\heshengjun\appdata\local\programs\python\python36\lib\site
-packages (from requests)
Requirement already satisfied: urllib3<1.23,>=1.21.1 in c:\users\heshengjun\appdata\local\programs\python\python36\lib\site
-packages (from requests)
Requirement already satisfied: idna<2.7,>=2.5 in c:\users\heshengjun\appdata\local\programs\python\python36\lib\site-pac
kages (from requests)
Requirement already satisfied: chardet<3.1.0,>=3.0.2 in c:\users\heshengjun\appdata\local\programs\python\python36\lib\site
-packages (from requests)
Installing collected packages: requests
Successfully installed requests-2.18.4
You are using pip version 9.0.3, however version 10.0.1 is available.
You should consider upgrading via the 'python -m pip install --upgrade pip' command.
```

### 6.3. pip 离线安装

去对应的网站(<https://pypi.org>)下载需要使用的模块的包(whl 文件)

# requests 2.18.4

pip install requests

Latest version

Aug 15, 2017

Python HTTP for Humans.

## Navigation

[Project description](#)[Release history](#)[Download files](#)

## Project links

## Download files

Download the file for your platform. If you're not sure which to choose, learn more about installing packages.

Filename, size & hash	File type	Python version	Upload date
requests-2.18.4-py2.py3-none-any.whl (88.7 kB) <a href="#">SHA256</a>	Wheel	py2.py3	Aug 15, 2017
requests-2.18.4.tar.gz (126.2 kB) <a href="#">SHA256</a>	Source	None	Aug 15, 2017

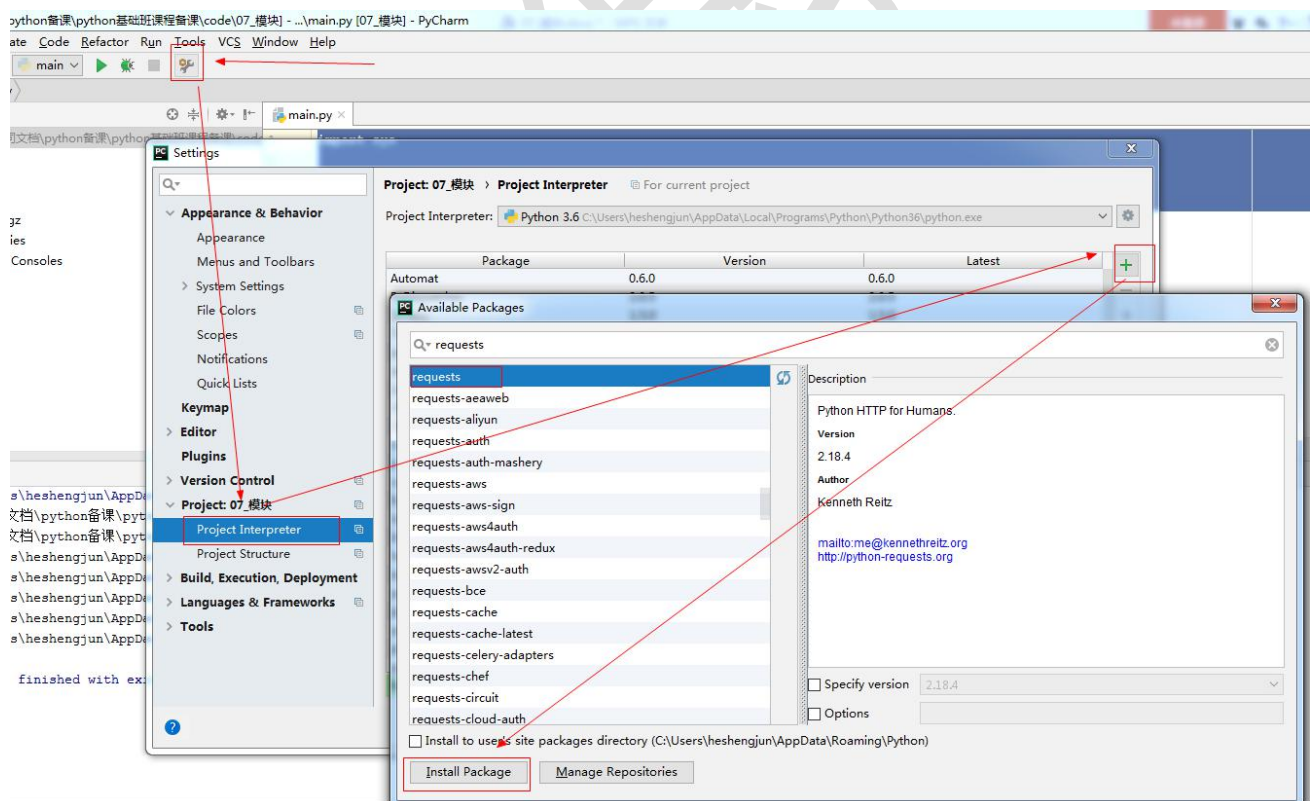
进入到 whl 文件目录进行安装

#进入到 whl 文件所在的目录进行安装

pip install requests-2.18.4-py2.py3-none-any.whl

## 6.4. 使用工具进行安装

在 pycharm 中使用插件进行安装



## 6.5. 源码的方式安装

去对应网站下载源码(<https://pypi.org>) 一般格式为\*.tar.gz 格式

- 1 找到模块的压缩包
- 2 解压压缩包 `tar -zxvf util-1.0.tar.gz`
- 3 进入到解压后的目录(含有 `setup.py` 文件)
- 4 执行命令安装 `python setup.py install`

叮丁狼教育