

Python模块与包

Python模块与包关系



Python实战怎么学

学习建议

- ◆ 认清现实：师傅领进门，修行靠个人
- ◆ 掌握方法：掌握方法，培养自学能力
- ◆ 勤做笔记：专业名词太多，逐个了解
- ◆ 眼高手低：实践中思考、思考中进步



学习建议

◆ 思考：工作中遇到问题如何解决？

◆ 勤动手：动手还出错该如何解决？



Python模块

课程介绍

◆ Python模块介绍

◆ 模块导入及定位

◆ 模块属性

◆ python实战

◆ 包的介绍及使用

◆ 标准模块 (os,datetime,sys...)

◆ 第三方模块

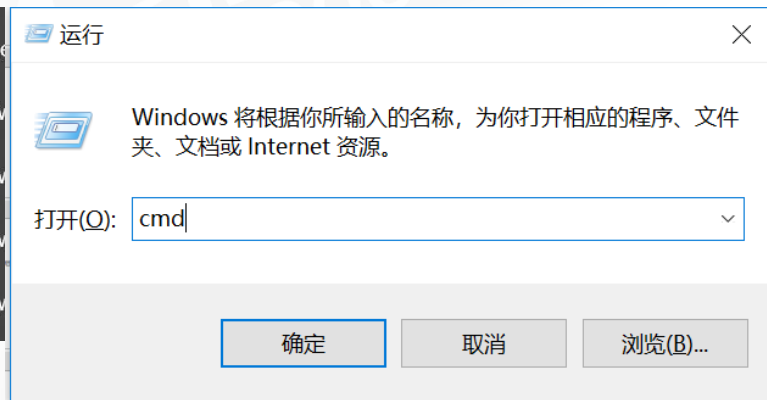
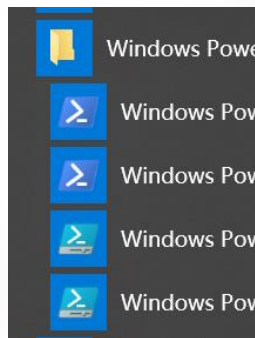
命令行/终端知识

命令行/终端知识

- ◆ 别名：命令行、终端、terminal、控制台、小黑窗、交互式界面
- ◆ 使用场景：DOS系统、Linux

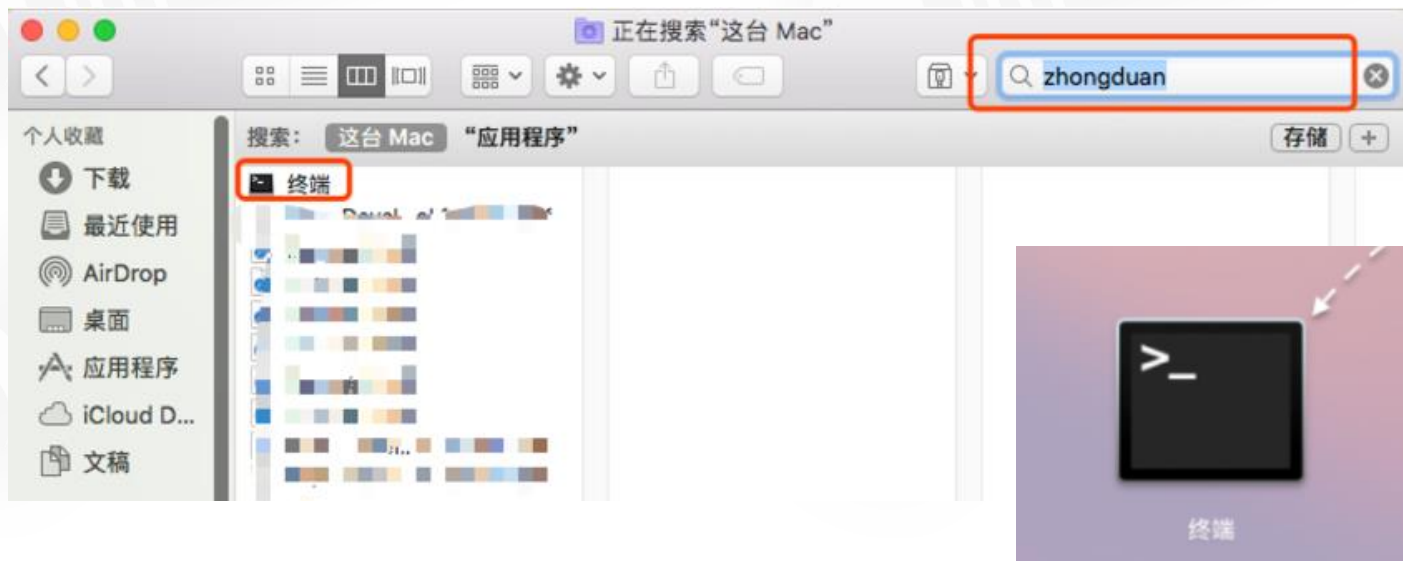
打开终端方式

◆ Windows



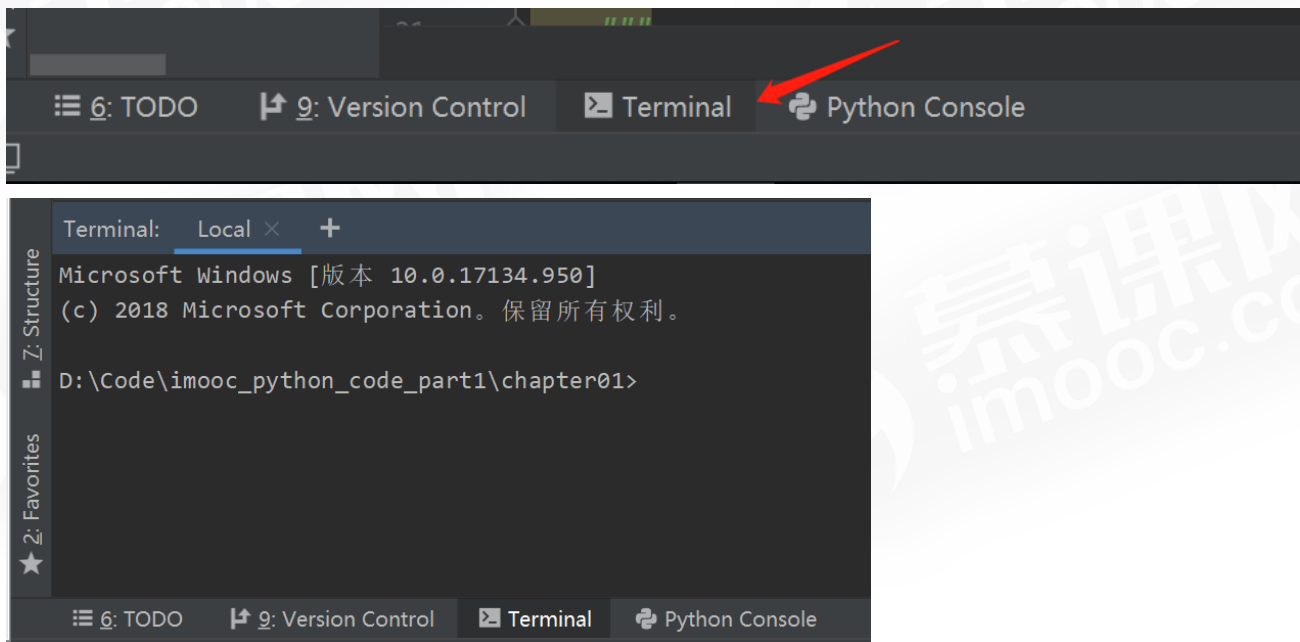
打开终端方式

◆ Mac



打开终端方式

◆ PyCharm



DOS常用命令

- ◆ cd 切换目录
- ◆ dir 查看目录下的文件和文件夹
- ◆ cls 清除屏幕显示的内容
- ◆ exit 退出控制台

在控制台编写Python代码

- ◆ 第一步：进入控制台
- ◆ 第二步：进入Python控制台
- ◆ 退出Python控制台 `>>> quit()/exit()`

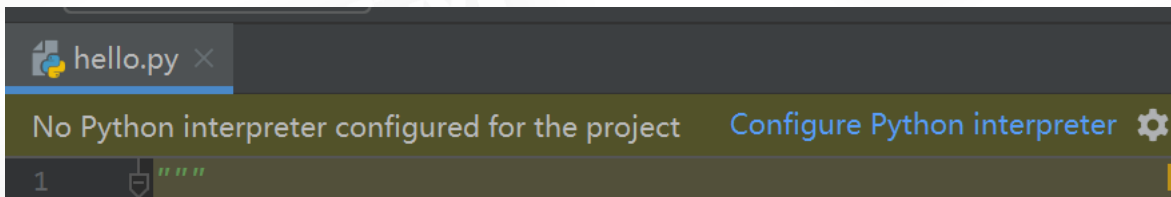
在控制台执行Python代码

- ◆ 第一步：进入控制台
- ◆ 第二步：找到对应的py文件，执行 `python C:/test.py`

PyCharm使用技巧

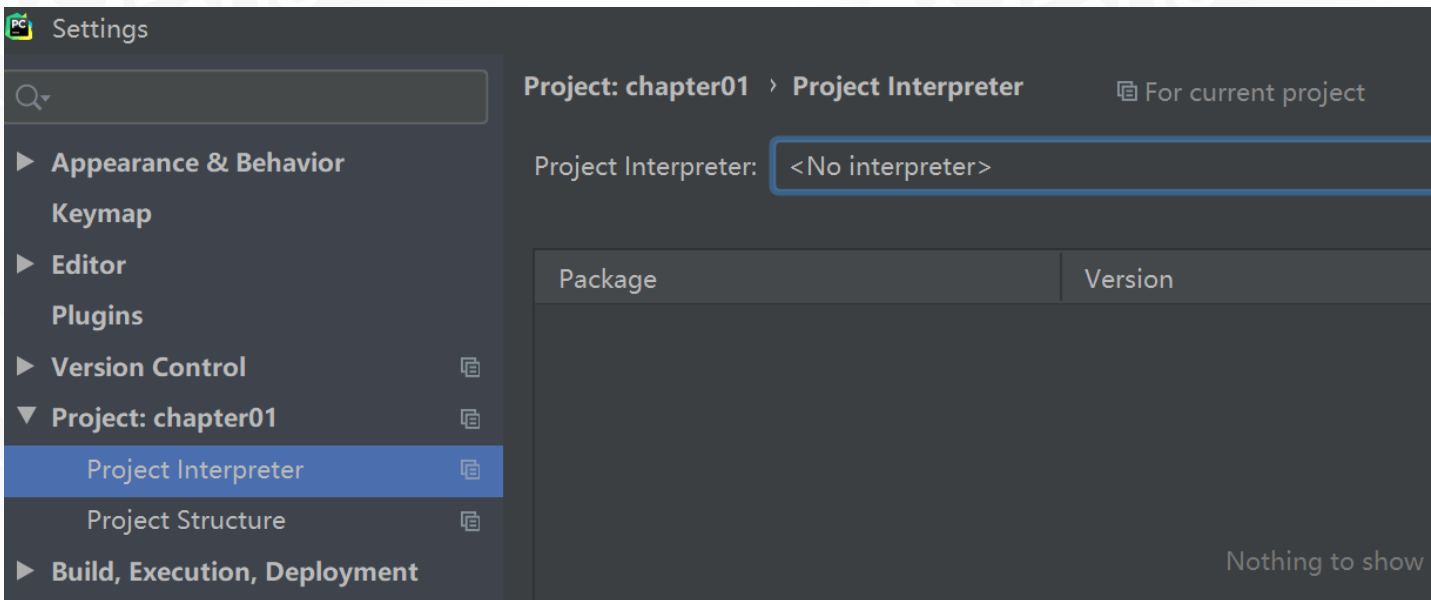
PyCharm使用技巧

◆ Python解释器的配置



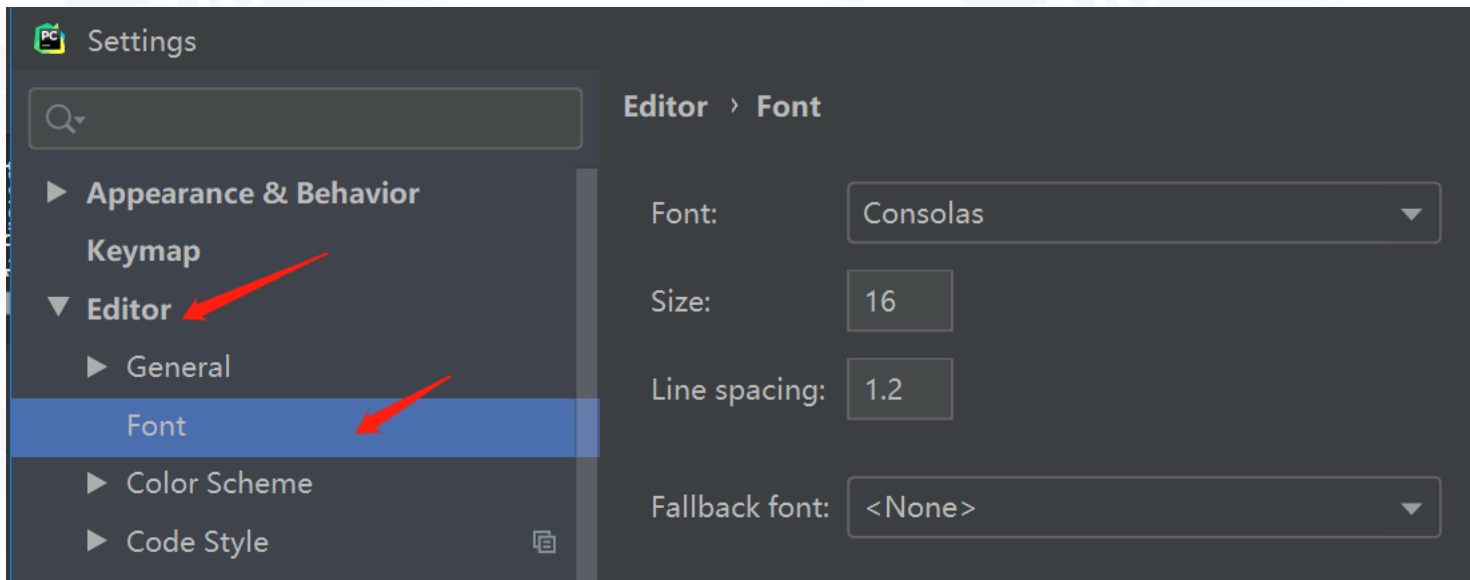
PyCharm使用技巧

◆ Python解释器的配置



PyCharm使用技巧

◆ 字体大小配置



模块介绍

Python模块介绍

◆ 什么是模块？

◆ 如果没有模块将会怎样？



Python模块介绍

◆ 似曾相识的模块

```
import random
```

```
#产生一个1~16之间的随机整数
```

```
r = random.randint(1,16)
```

Python模块定义

- ◆ 模块就是程序，模块的名称就是不含.py后缀的文件名

模块的分类

- ◆ Python标准模块（Python内置模块、Python标准库）
- ◆ 第三方模块/库（pypi.org）
- ◆ 自定义模块

Python模块的好处

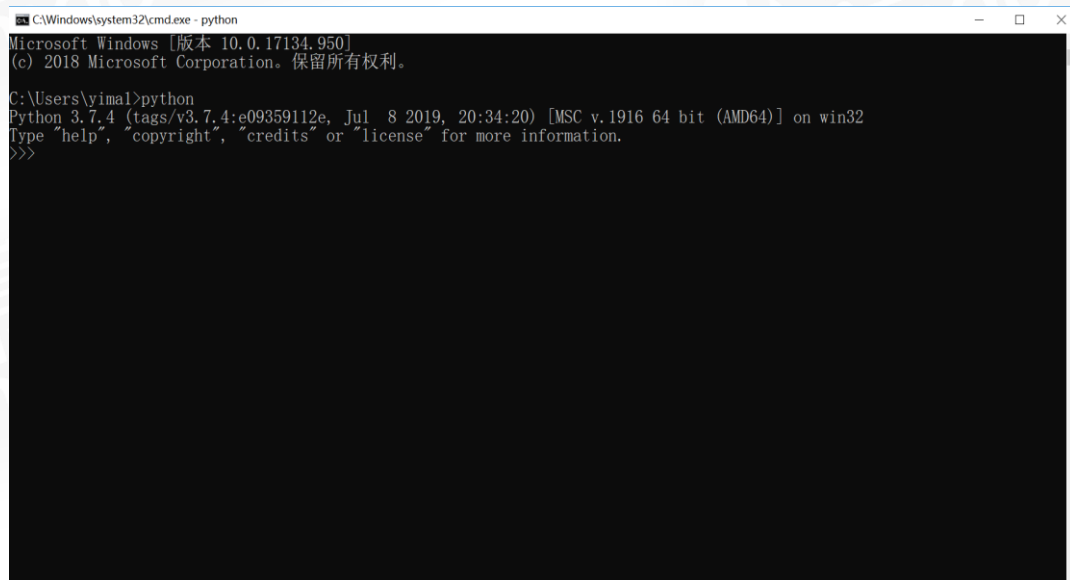
◆ 可维护性更强

◆ 方便代码重用

模块导入及定位

思考

◆ 从控制台为什么能打开python



```
C:\Windows\system32\cmd.exe - python
Microsoft Windows [版本 10.0.17134.950]
(c) 2018 Microsoft Corporation. 保留所有权利。

C:\Users\yimal>python
Python 3.7.4 (tags/v3.7.4:e09359112e, Jul 8 2019, 20:34:20) [MSC v.1916 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
```

模块导入及定位

◆ 导入: import os

◆ 定位: 当前包 ---> 内置函数---> sys.path(环境变量)

```
>>> import sys
>>> sys.path
['', 'C:\\Python36\\python36.zip', 'C:\\Python36\\DLLs',
, 'C:\\Python36\\lib', 'C:\\Python36', 'C:\\Python36\\lib\\site-packages']
>>> import hello
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
ModuleNotFoundError: No module named 'hello'
>>> sys.path.append('c:\\users\\yimal\\desktop')
>>> import hello
hello
>>>
```

模块的属性

模块属性

- ◆ `dir`——列出对象的所有属性及方法
- ◆ `help`——查看类，方法的帮助信息
- ◆ `__name__`——模块的名称
- ◆ `__file__`——文件全路径

模块属性

```
>>> sys.path.append('c:\\users\\yimal\\desktop')
>>> import hello
hello
>>> dir(hello)
['__builtins__', '__cached__', '__doc__', '__file__',
 '__loader__', '__name__', '__package__', '__spec__']
>>> hello.__name__
'hello'
>>> hello.__package__
''
>>> hello.__file__
'c:\\users\\yimal\\desktop\\hello.py'
>>>
```


包的简介

包的简介

- ◆ 可以用来组织模块（可以包含其它模块的模块）
- ◆ 目录必须包含文件 `__init__.py`
- ◆ 模块重名问题解决

包的简介

```
1 package_a
2 |— __init__.py
3 |— module_a.py
4 |— module_b.py
5 |— sub
6 |   |— bar.py
7 |   |— __init__.py
8 package_b
9 |— __init__.py
10 |— module_a.py
11 |— module_c.py
12 |— bar
13 |   |— bar.py
14 |   |— __init__.py
15 |— foo.py
16 |— setup.py
```

`__init__.py`

- ◆ 注意是英文半角的双下划线
- ◆ 将一个文件夹变为一个模块
- ◆ 导入包实际上是导入它的 `__init__.py`
- ◆ 一般为空，可以批量导入所需的模块

包的引用

包的引用

- ◆ **引入整个包:** `import module`
- ◆ **只引入所需要的属性和方法:** `from module.xx.xx import xx`
- ◆ **指定别名:** `from module.xx.xx import xx as rename`
- ◆ **引入所有:** `from module.xx.xx import *`

注意事项

- ◆ 其它叫法：包的导入、模块导入、模块引入
- ◆ 模块导入后的重命名 (as) 遵循Python变量的命名规范