3. Python 中的模块

在实际开发过程中，不会将所有的代码都写在同一个文件中，而是将功能类似的封装在一起，这样代码结构清晰、便于管理代码。这就是模块的意义。在 Python 中使用模块来管理代码，一个模块可以被理解为一个.py 文件。在模块中可以定义函数，实现一定的功能，这样如果要使用它定义好的某个功能时，只要导入这个模块即可。

（1）导入模块

模块分为两类：

- 内置模块

只要安装了 Python 就可以使用。

- 第三方模块

如果需要使用第三方模块，首先需要安装第三方模块。第三方模块包括用户自定义模块。

导入模块使用关键字 import，语法如下。

【语法】

import 模块名

注意，需要把导入模块命令放在代码脚本的顶端。

其实我们之前已经接触过导入模块的操作了。在“幸运大抽奖”案例中我们使用了生成随机数的 random 模块：

```
#导入 random 库，实现取随机数
import random
#randint 取随机整数赋给变量 num_rand
num_rand = random.randint(1,10)
```

导入 random 模块后，使用“模块名.函数名()”的方式调用模块里的功能函数，这个示例中的“random.randint(1,10)”就是使用了 randint()函数，生成 1 到 10 之间的一个随机整数。

Python 中还允许有针对性的导入模块中的某一部分，这样在调用方法时会显得更加简洁，导入语法如下。

【语法】

from 模块名 import 函数名或类名

注意：关于类的概念，后续内容会介绍。

在导入模块时 Python 还允许给模块起一个别名，因为某些模块名字过长，使代码过于臃肿并且不利于阅读，此时就可以给它起一个别名，语法如下。

【语法】

import 模块名 as 别名

from 模块名 import 方法名或类名 as 别名

示例：使用随机数模块生成某一期的双色球中奖号码。

双色球规则：

- 6 个不重复的红球，红球的选号范围 1~33
- 1 个篮球，篮球的选号范围 1~16
- 红球以从小到大的顺序排列

分析：

- 要生成随机数，所以需要导入随机数模块 random
- 每次生成红球数，都要判断是否已经生成过，即判断是否重复，可以使用列表或集合来保存红球，这两种数据结构都有判断元素是否存在的方法。
- 红球需要排序，列表有排序方法，能够直接排序输出。故使用列表保存红球数是优选

代码实现过程如下：

```
#使用随机模块生成某一期的双色球中奖号码
from random import randint as rd
red_balls = []
#生成红球
while len(red_balls) != 6:
    red_ball = rd(1,33)
    if red_ball not in red_balls:
        red_balls.append(red_ball)
blue_ball = rd(1,16)#生成蓝球
red_balls.sort()#排序
print('红球：',red_balls)
print('蓝球：',blue_ball)
```

上机练习试一试。

小贴士

若想把一个模块的所有内容全都导入到当前的工程项目中，使用如下声明：
from 模块名 import *

（2）创建自定义模块

创建模块非常的简单，之前说过，一个.py 文件就是一个模块，所以创建模块可以认为是创建一个 py 文件。

例如如下两个文件 hello.py 和 say.py 文件：

hello.py 文件：

```
def say(name):
    print(name,'向大家问好')
```

say.py 文件

```
import hello
hello.say('张凌云')
```

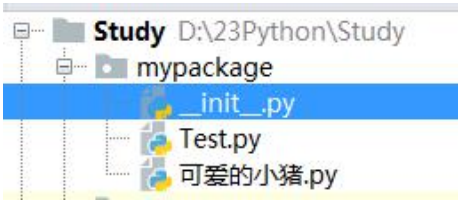
输出结果为：

张凌云 向大家问好

（3）Python 的包

包是一种管理 Python 模块命名空间的形式，简单地说，为了更好的组织模块，将多个功能详尽或者有关联的模块放在一个文件夹里，就成为了一个包。包就是文件夹，文件夹名就是包名，文件夹下必须存在一个__init__.py 文件，用来标识当前文件夹是一个包，如果缺少了这个文件，文件夹外的文件就无法导入该文件模块。

如下图所示，包名为 mypackage，该包存在__init__.py，其内的模块 Test.py 才可以被外面的文件所引用。



导入包中的模块语法如下。

【语法】

import 包名.模块名

例如：

```
import mypackage.Test
```

（4）标准内置模块

Python 本身带着一些标准的模块库，也称为内置模块库，只要安装了 Python 就可以直接使用的模块库，内置模块库通常是一些使用场景非常广泛的模块，例如随机数模块、文件模块、时间模块等等。

Python 常用内置模块库	
模块名	**功能**
time	用于获取时间戳或时间格式的转换
datetime	为日期和时间处理提供了简单和复杂的方法
math	为浮点运算提供了对底层 C 函数库的访问
os	提供了与操作系统相关联的函数
shutil	提供了针对日常的文件和目录管理任务且易于使用的函数

示例：根据指定日期，获取是星期几。

例如，2020 年 3 月 23 日，给出“周一”的提示

分析：

获取周几，有一个方法如下：

datetime.date.weekday()：返回日期的星期

```
import datetime
def getweek(y,m,d):
    week_srt = ['星期一','星期二','星期三','星期四','星期五','星期六','星期日']
    day = datetime.date(y,m,d)
    wd = day.weekday()
    print('%d 年%d 月%d 日：'%(y,m,d),format(week_srt[wd]))
y = eval(input('请输入年份(1-9999):'))
m = eval(input('请输入月份(1-12):'))
d = eval(input('请输入日期(1-31):'))
getweek(y,m,d)
```

输出结果如图：

（5）第三方模块

内置标准模块一般是通用的，使用场景广泛，而第三方模块则更具有针对性，用于处理更加专业的问题。如果要使用第三方模块，必须先安装再使用。

小贴士

由于 Python 版本较多，而且有的版本之间还存在有较大差异，所以在安装第三方模块时一定要考虑是否与当前版本匹配。