计算机类专业“互联网+”创新型精品教材

Python程序设计

主　编　刘小园　关晓颖　陈荣征

北京出版集团
北京出版社

Python程序设计
主编　刘小园　关晓颖　陈荣征
北京出版集团
北京出版社

图书在版编目（CIP）数据

Python 程序设计 / 刘小园 , 关晓颖 , 陈荣征主编
. —北京：北京出版社，2023.5
ISBN 978-7-200-17880-7

Ⅰ . ① P… Ⅱ . ①刘… ②关… ③陈… Ⅲ . ①软件工具—程序设计—高等学校—教材 Ⅳ . ① TP311.561

中国国家版本馆 CIP 数据核字（2023）第 061487 号

Python 程序设计

Python CHENGXU SHEJI

主　　编：刘小园　关晓颖　陈荣征
出　　版：北京出版集团
　　北京出版社
地　　址：北京北三环中路 6 号
邮　　编：100120
网　　址：www.bph.com.cn
总 发 行：北京出版集团
经　　销：新华书店
印　　刷：定州启航印刷有限公司
版 印 次：2023 年 5 月第 1 版　2023 年 5 月第 1 次印刷
成品尺寸：185 毫米 ×260 毫米
印　　张：16
字　　数：360 千字
书　　号：ISBN 978-7-200-17880-7
定　　价：49.50 元

教材意见建议接收方式：010-58572162　邮箱：jiaocai@bphg.com.cn
如有印装质量问题，由本社负责调换
质量监督电话：010-82685218　010-58572162　010-58572393

目 录

项目一

初识 Python 语言

计算机是 20 世纪最伟大的发明之一，自诞生至今，一直推动人类社会进步，改变人们原有的生产、生活方式。可是，你知道吗？其实计算机都是按照程序员预先设计的程序在工作的，而程序则是一系列计算机指令，是由程序设计语言编写而成。目前，伴随着大数据和人工智能的发展，Python 程序设计语言以其简单易学、开发效率高、功能强大和应用广泛，特别是其在数据处理领域的突出表现，迅速成为程序开发最热门的语言之一。接下来，让我们一起从 Python 的由来入手，逐步了解其编码规范、特点、开发环境，在学习中我们重点通过几个简单案例来体验 Python 程序设计的乐趣。

任务一　揭开 Python 的神秘面纱

任务描述

小明大学选择的是计算机专业，并且很喜欢编程，入学后常去一些编程类博客看帖，发现最近网上在流传着一句话：“人生苦短，我用 Python。”他通过了解发现 Python 语言目前在 Web 开发、大数据开发、人工智能开发、后端服务开发和嵌入式开发等领域都有广泛的应用，且人才市场的缺口仍在与日俱增。于是小明下决心认真学习 Python 语言，本任务我们就跟着小明一起揭开 Python 的神秘面纱，最后完成我的第一个 Python 程序。

Python 的输入和输出函数

任务目标

1. 能熟悉 Python 的程序结构和编码规范。
2. 能了解 Python 的由来和特点。

3. 能了解 Python 的输入、输出函数。

4. 能编写一个简单的 Python 程序。

5. 培养学生对 Python 的学习兴趣，理解创新驱动发展战略。

知识链接

一、Python 的由来

Python 语言是由荷兰人吉多·范罗苏姆（Guido van Rossum，以下简称“吉多”）于 1989 年圣诞节期间为了补充和继承自己参与开发的 ABC 语言，决心开发一个新的脚本解释程序。

最初的 Python 完全由吉多个人开发，后来得到其同事认可并纷纷利用业余时间参与改进，2000 年 10 月 Python 2.0 发布，之后 Python 也从最初的基于邮件列表的开发方式转变为完全开源的开发方式。随着 Python 社区日渐成熟，2010 年，Python 2.x 系列发布了最后一个版本，2.x 系列的主版本号为 2.7。

之后，几乎每年都有新的版本发布，截至 2022 年 2 月，Python 的最新版本为 Python 3.10.2，本书所有程序都在该版本测试验证过。

Python 语言的最大特点是将许多机器层面上的细节隐藏，交给编译器处理，并凸显出逻辑层面的编程思考，成为最符合人类期待的编程语言之一。Python 程序员可以花更多的时间用于思考程序的逻辑，而不是具体的实现细节，最大程度释放了程序员的思维，这一特征吸引了广大的程序员，Python 开始广泛流行。

二、Python 的特点

Python 作为高级程序设计语言的新成员，能在诸多的程序设计语言中取得一席之地，必有其独到之处。

（一）简单易学

Python 在语法上沿用了大量 C 语言的语法，但抛弃了较为复杂的指针，降低了初学者的学习难度。Python 程序有点类似于伪代码，当我们学习了 Python 基本的语法知识后，再去看 Python 程序源代码就像阅读小学英语短文一样简单；当我们编写 Python 程序时，可以更专注于解决问题，而不需要过多地关注语言本身。

（二）免费开源

由于吉多认为 ABC 语言失败的主要原因是非开放性，所以在设计 Python 之初就采用了基于邮件列表的开放式开发方式，2.0 版本之后则建设 Python 社区采取了完全开源方式，通过基于团队的知识分享，让一群精益求精的热爱 Python 的人不断迭代、改进、优化 Python。

（三）可移植性好

由于 Python 是一种解释型语言，理论上可以在任何预装了 Python 解释器的平台上执

行，能有效避免程序必须依赖于特定系统的特性，所以 Python 语言编写的程序可以不做任何修改就能在 Linux、Windows、Windows CE、OS/2、Android 等几乎所有平台上运行。

（四）可扩展性好

Python 在高层可引入 .py 文件，在底层则可通过接口和库函数调用其他高级程序设计语言编写的程序。比如希望关键代码运行更快或者希望某些算法不公开，则这部分程序可用 C 或 C++ 语言编写，然后在 Python 程序中调用它们，或者将 Python 程序作为脚本嵌入 C 或 C++ 程序中。

（五）类库丰富

Python 解释器拥有丰富的内置类和函数库，再加上开源社区 Python 爱好者们贡献的覆盖各个应用领域的第三方函数库，程序员可以直接调用来帮助我们处理各种工作，比如处理正则表达式、文档生成器、处理电子邮件、密码系统，等等。记住，只要安装了 Python，所有这些功能都是直接可用，除此之外，还有 NumPy(数值计算)、Twisted（网络工具）和 Pillow（图像处理）等许多其他高质量的扩展库。

（六）面向对象，也面向过程

Python 既支持面向过程编程，也支持面向对象编程。在“面向过程”的语言中，程序开发是以实现执行过程为设计思想，使用函数为程序主体构建起来的。在“面向对象”的语言中，程序开发是以描述执行“人”（即对象）的特征及其相互作用为主要设计思想，使用由数据（属性）和功能（方法）组成的对象为程序主体构建起来的。与其他主要的语言（如 C++ 和 Java) 相比，Python 以一种功能强大而使用简单的方式实现面向对象编程。

（七）通过缩进来规范代码

Python 采用强制缩进的方式使得代码具有极佳的可读性，不符合缩进规则的语句将无法执行。另外，行尾不需要使用分号，类的定义体和方法的定义体通过缩进的方式呈现，条件判断和循环体内的语句块也不再需要使用花括号进行“包裹”，这些代码规范在一定程度上提高了开发者的开发效率。

（八）支持多种文字

Python 3.x 系列的解释器采用了 UTF－8 编码，该编码不仅支持英文，还支持中文、韩文、法文等多种文字，这让我们在 Python 程序中对中文字符的处理变得更加灵活。

Python 作为一门解释型高级程序设计语言，有着诸多优点，但其执行效率不是很高，只有 C 语言的十分之一，并且 Python 3.x 系列与 Python 2.x 系列是不兼容的。

三、Python 的编码格式规范

Python 代码格式规定通过代码缩进体现语句间的逻辑关系，该规定是 Python 语法的组成之一，不符合格式规范的 Python 代码无法正常运行，因此，所有 Python 代码都具有相对统一规范的风格，提升了 Python 代码的可读性。

（一）Python 注释

在实际软件开发过程中，为了让其他人更容易理解代码的功能和方便后期维护，注

释是非常有效的方法。所谓注释就是在代码中穿插的辅助性文字，用于标识代码的含义与功能，提高程序的可读性，但注释并不会被编译运行。Python 程序中的注释分为单行注释（行注释）和多行注释（块注释）。

1. 单行注释

单行注释以“#”开头（Python 官方建议“#”后面先添加一个空格），既可以单独占一行，放在被注释代码行之上，也可以位于标识语句之后，与标识语句共占一行。示例如下：

```
print ('中国，加油！')                    # 这是与标识码同行的单行注释
# 这样则是单独占一行的单行注释
# 与第一条语句功能一样，都是使用输出函数输出：中国，加油！
print ('中国，加油！')
```

2. 多行注释

Python 中多行注释使用三个单引号（''' '''）或者三个双引号（""" """）来标记，将注释写在引号中间，但实际上这是多行字符串的书写方式，并不是 Python 本身提倡的多行注释方法。示例如下：

```
print ('中国，加油！')
''''
这样则是多行注释
上述语句功能是使用输出函数输出：中国，加油！
'''
```

（二）代码缩进规范

在代码语句块的表示方式上，Python 与其他程序设计语言使用花括号表示不同，缩进是其表示代码语句块的唯一方式。标准 Python 风格规定每级缩进 4 个空格，如果使用 Tab 键，建议在编辑器中设置一次 Tab 键显示 4 个空格，以免不同编辑器处理 Tab 键不同，而引发 Python 代码缩进错误。

在 Python 中，缩进规定非常严格，一个语句块的所有语句缩进必须相同，源文件的第一行不允许以空格开始，例如下面的代码将报错。

```
if(a>0):
    Print('a 大于 0')
else:
    Print('a 不大于 0')
Print('if 语句结束了。')          # 缩进不一致，导致运行错误
```

小贴士

一条语句代码过长怎么换行？

如果我们在编写代码时，碰到一条较长的文本或代码，可以使用括号将它们括起来处理。示例如下：

```
demo_str=("Python 作为高级程序设计语言的 " 新 " 成员，能在诸多的程序设计语言中取得一席之地，必有其独到之处。")
```

还可使用反斜杠连接多行代码，示例如下：

```
a=15+ \          # 反斜杠连接上下两行代码
    15
print a          # 输出 30
```

四、Python 的输入和输出函数

输入和输出是任何程序最基本的操作，后续学习中我们将反复使用，这里我们先分别介绍一个输入、一个输出函数。

（一）输入函数 input()

Python 中提供了一个输入函数 input()，该函数用于收集用户输入的字符串。如下示例中，当用户在命令行中输入 name=input()，并按回车键，命令行将等待用户输入，待用户输入：谷爱凌，并按回车键，则会将“谷爱凌”保存在变量 name 中，我们可以直接在命令行中输入 name 来查看变量 name 保存的内容“谷爱凌”。

```
>>> name=input()
谷爱凌
>>> name
'谷爱凌'
```

（二）输出函数 print()

Python 中提供了输出函数 print()，该函数有 4 个参数，如下示例所示。

```
print(*objects,sep='',end='/n',file=sys.stdout,flush=False)
```

其中四个参数的含义如下：

sep：在值之间插入的字符串，默认是一个空格。

end：追加在后面值的字符串，默认是一个换行符。

file：类似文件（流）的对象，默认是 sys.stdout。

flush：表示是否强制刷新文件（流），Python3.3 版本之后默认是 False。

print() 函数可以用于输出各种类型的数据、变量、表达式等，具体可见如下示例：

```
print(160)              # 直接输出数值型常量
160
print('中国特色！')      # 直接输出字符串
```

```
中国特色！
var1=168                    # 变量赋值
print(var1)                 # 输出变量
168
list=[1,3,5,7,9]            # 变量赋值
print(list)                 # 输出列表
[1, 3, 5, 7, 9]
var2='hello!'               # 变量赋值
var3=len(var2)              # 函数 len() 求得字符串的长度，赋值给变量 var3
print('字符串：%s 的长度为：%d' %(var2,var3))      # 格式化输出
字符串：hello! 的长度为：6
print(12+1,18.11,'python')      # 同时输出多个不同数据类型的数据或表达式
13 18.11 python
```

上述函数 print() 的示例显示结果如图 1－1 所示，图中一个地方报错了，还有一个地方注释没有写完，找找看是什么原因？（在任务实施中，可以找到答案）

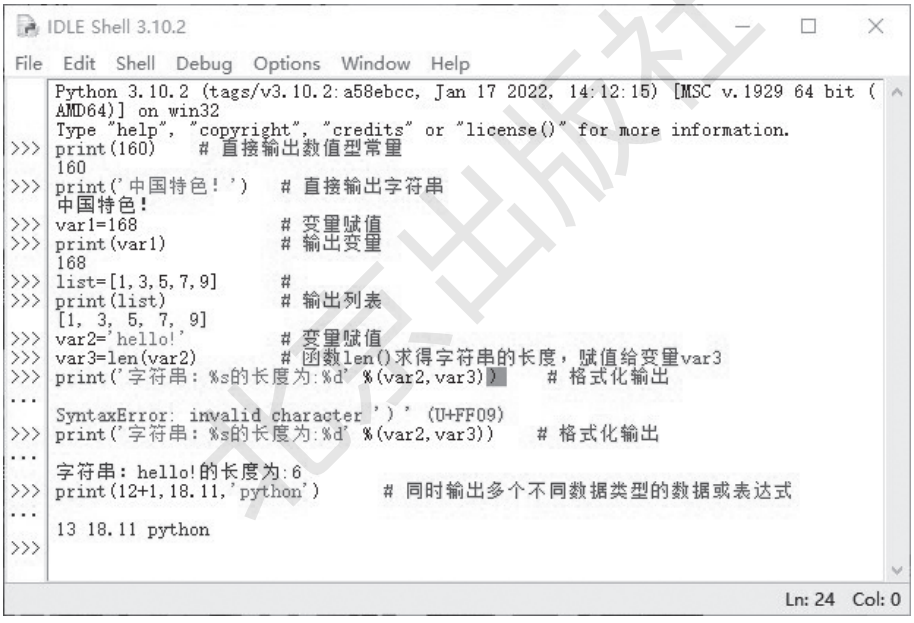

图 1-1　函数 print() 的示例显示结果

任务实施

打开开发工具 Python IDLE，如图 1－2 所示看到其首界面，在提示符（>>>）后面输入“name＝input()”，按回车键，命令行将等待用户输入姓名“小明”，按回车键，在提示符（>>>）后面输入“print(name)”，命令行会输出刚刚用户输入的姓名（小明），依次输入其他命令，最后我的第一个 Python 程序的显示如图 1－3 所示。

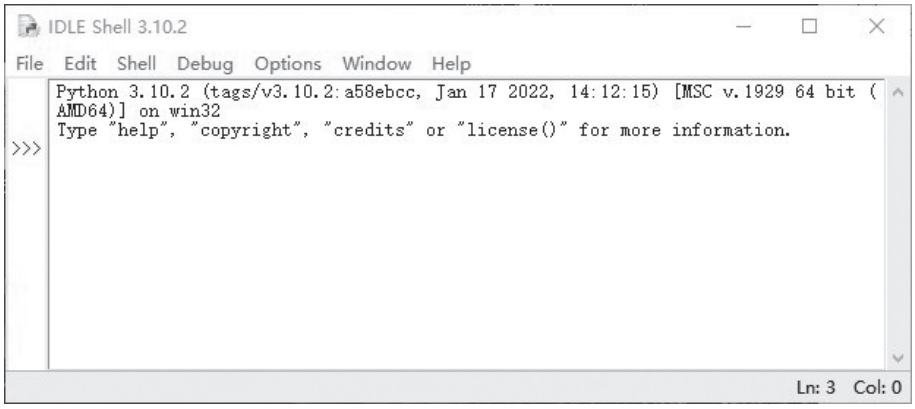

图 1-2 Python IDLE 的首界面

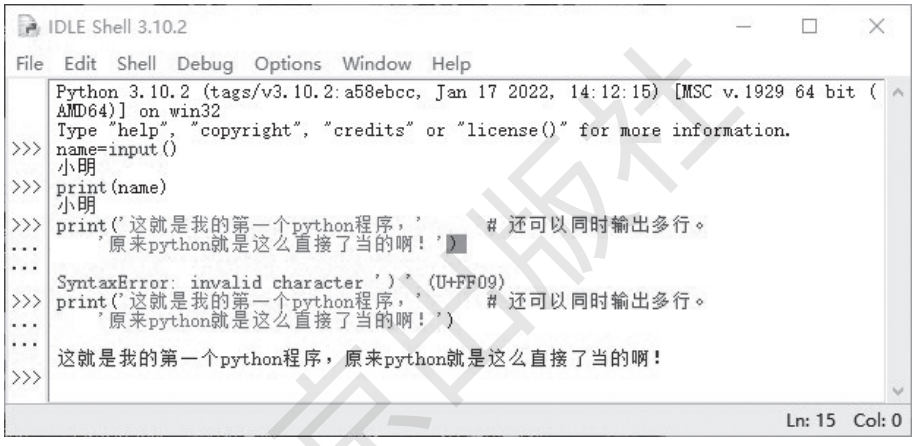

图 1-3 我的第一个 Python 程序

小贴士

第一次编写 Python 程序需要注意什么呢？

（1）print() 函数里面的字符串，可以双引号（" "）或者单引号（' '）包围，且字符串中可以包含英文、数字、中文以及各种符号，当然也可以将多段文本放在一个 print 函数中，如图 1-3 所示的第 11~12 行。

（2）引号和小括号都必须在英文半角状态下输入，而且 print 的所有字符都是小写。Python 是严格区分大小写的，print 和 Print 代表不同的含义。否则将如图 1-3 所示的第 10 行报错，错误的原因是第 9 行最后的小括号是中文状态输入的。

思考与练习

一、选择题

1. 以下说法正确的是（　　）。

A. Python 属于低级语言　　B. Python 是面向过程的

C. Python 属于解释性语言　　D. Python 是非开源的

2. 下列选项中，不是 Python 语言特点的是（　　）。

A. 简单易学　　B. 免费开源

C. 可移植性好　　D. 依赖平台

3. 下列关于 Python 语言的说法，错误的是（　　）

A. Python 只能编写面向对象的程序

B. Python 编写的程序执行效率比 C 语言低

C. Python 主要由吉多开发

D. Python 是一门解释型高级程序设计语言

二、填空题

1. Python 是既支持面向_____，也支持面向_____的高级程序设计语言。

2. Python 语法_____大小写。

3. Python 的语句块是采取_____方式来标识。

4. Python 的注释可以分为_____和_____两种。

5. Python 的常用输入函数是_____；输出函数是_____，其中输出函数包含_____个参数。

三、思考题

简述 Python 的主要特点。

四、实践题

打开开发工具 Python IDLE，编写一段我的初识牛刀代码实现如图 1-4 所示结果。

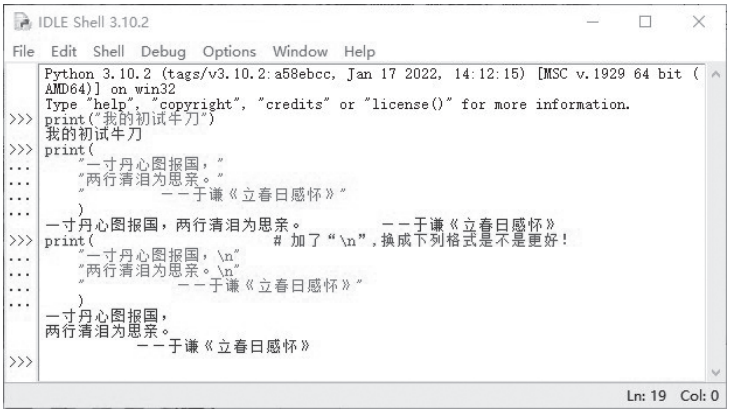

图 1-4　我的初试牛刀显示效果

任务二 安装和配置 Python 开发环境

任务描述

小明通过初试牛刀的尝试后，觉得用 Python 语言编写程序越来越有意思，于是决定在自己的电脑上安装前端开发工具软件。本任务就让我们一起来了解 Python 版本及解释器、执行原理。然后下载安装 Python 3.10.2 和 PyCharm 两个开发工具，并熟悉其基本操作，最后分别用两个开发工具写一个 Python 小程序。

Python 开发环境安装与配置

任务目标

1. 能了解 Python 的不同版本及解释器。
2. 能熟悉 Python 3.x 与 Python 2.x 的主要区别。
3. 能了解 Python 程序运行方式。
4. 会下载安装 Python，并熟悉其基本操作。
5. 会下载安装集成开发环境 PyCharm，并熟悉其基本操作。
6. 培养学生的程序设计素养，贯彻创新驱动发展战略。

知识链接

一、Python 的版本和过渡

Python 发展到今天，由于是完全开源的原因，经历的版本很多，目前市场上有两个 Python 的版本系列并存，分别是 Python 2.x 系列和 Python 3.x 系列，为了不带入过多的累赘，Python 3.0 在设计之初就放弃了向下兼容，也就是说早期 Python 2.x 系列版本设计的程序都无法直接在 Python 3.x 系列上直接执行。从其官方查看 Python 开发人员指南可以了解到更多的版本信息，表 1-1 所列举的是目前仍旧比较活跃的 5 个版本。

表 1-1 比较活跃的 Python 版本

序号	版本号	维护状态	首次发布	支持结束时间
1	3.10	错误修正	2021 年 10 月 4 日	2026 年 10 月
2	3.9	错误修正	2020 年 10 月 5 日	2025 年 10 月
3	3.8	安全	2019 年 10 月 14 日	2024 年 10 月
4	3.7	安全	2018 年 6 月 27 日	2023 年 6 月
5	2.7	停止维护	2010 年 7 月 3 日	2020 年 1 月

二、Python 3.x 与 Python 2.x 的主要区别

2008 年 12 月发布的 Python 3.0 为了不带入 2.x 系列的一些缺陷，开发者选择了不向下兼容的策略，那么 Python 3.x 系列到底在哪些方面做出了调整呢？

（一）用函数 print() 代替 print 语句

print 变成函数之后，print 就可以组件化了，作为语句的 print 是无法支持的，其真正巧妙之处在于灵活性，比如 print() 函数可以在表达式中使用，而 print 语句只能作为语句使用。另外与使用 print 语句相比，print() 函数能够指定其他的分隔符和结束符。

（二）更好地支持中文

由于 Python 2.x 中，字符串默认是 ASCII 编码的 str 类型和 unicode 类型。Python 3.x 默认使用的是 UTF－8 编码，从而可以很好地支持中文或其他非英文字符。假如同样输出一个存储中文字符串的变量，Python 2.x 和 Python 3.x 的显示结果如下示例所示：

```
# 在 python 2.x 中
>>>var='我爱北京天安门'
>>>var
'\xe6\x88\x91\xe7\x88\xb1\xe5\x8c\x97\xe4\xba\xac\xe5\xa4\xa9\xe5\
xaelx89\xe9\x97\xa8'
# 在 python 3.x 中
>>>var=’我爱北京天安门’
>>>var
'我爱北京天安门'
```

（三）更方便的除法运算

Python 语言的除法运算与其他程序设计语言有所不同，其除法运算区分为两个不同运算符，分别是普遍意义的除法“/”和向下取整除法“//”。

（1）运算符“/”在 Python 2.x 中，与其他语言一样，整数相除的结果是一个整数，浮点数相除会得到一个浮点数的结果。但在 Python 3.x 中，整数和浮点数相除的结果都是浮点数。

（2）运算符“//”在 Python 2.x 和 Python 3.x 中是一样的，即整数相除的结果是一个向下取整的整数，浮点数相除的结果是一个向下取整的小数点后保留一位数的浮点数。

具体示例如下：

```
# 在 python 2.x 中
>>>8/3
2
>>>8.0/3.0
2.6666666666666665
>>>8//3
2
>>>8.0//3.0
2.0
```

```
# 在 python 3.x 中
>>>8/3
2.6666666666666665
>>>8.0/3.0
2.6666666666666665
>>>8//3
2
>>>8.0//3.0
2.0
```

（四）优化了异常处理方式

在 Python 3.x 系列中，与 Python 2.x 系列相比，异常处理主要优化如表 1-2 所示。

表 1-2 Python 3.x 和 Python 2.x 异常处理对比

区别	Python 2. x 系列	Python 3. x 系列
对象的抛出	所有类型的对象都是直接被抛出的	只有继承自 Base Exception 的对象才可以被抛出
捕获异常的语法	except Exception，err	except Exception as err
处理异常的语法	raise Exception，args 或者 raise Exception（args)	只能用 raise Exception(args)
行为和属性	保留了异常类的序列行为 和 . message 属性	取消了异常类的序列行为 和 . message 属性

（五）改变了部分数据类型

在 Python 3.x 系列中，数据类型的改变主要有三点：

（1）取消了 long 类型，只保留了一种整型类型 int，但 int 类型行为相当于 Python 2.x 系列中的 long 类型。

（2）新增加了 bytes 类型，定义一个 bytes 字面量的方法如下所示：

```
b=b'zhongguo'       # 这里的 b 是关键字
type(b)
<class 'bytes'>
```

定义好之后，字符串对象和 bytes 对象之间是可以相互转化的，利用 .encode() 方法可以将字符串对象转化为 bytes 对象，反之则用 .decode() 方法，如下所示：

```
x=b.decode()      # 调用 decode() 方法将 bytes 对象 b 转化为字符串对象 x
x
'zhongguo'
y=x.encode()      # 调用 encode() 方法将字符串对象 x 转化为 bytes 对象 y
y
b'zhongguo'
```

（3）在 Python 3.x 中将 2.x 中的 iterkeys()、iteritems()、itervalues() 等函数都已经废弃

了，而字典中的 keys()、items() 和 values() 方法用返回动态视图代替了列表，同时废弃还有 has_key() 方法，其用 in 进行替代。

（六）废弃了不等于运算符“<>”的写法

在 Python 2.x 系列中，不等于运算符有两种写法：！ = 和 < >，在 Python 3.x 系列中只保留了！ = 这种写法了。

（七）废弃了八进制字面量“01000”的表示方式

在 Python 2.x 系列中，表示八进制字面量的方式有两种：001000 和 01000，在 Python 3.x 系列中只保留了“001000”这一种表示方式。

三、Python 解释器

由于 Python 是一门解释型语言，要想运行使用 Python 编写以 .py 为扩展名的代码文件，必须通过解释器来执行。所谓 Python 解释器，又称为 Python 虚拟机，就是用来执行 .py 文本文件的虚拟机。目前常见的 Python 解释器主要有以下五款：

（一） CPython

CPython 是 Python 社区推荐使用的 Python 解释器，也可以说是标准的 Python 解释器，目前我们大部分人安装的 Python IDE 都是 CPython，CPython 是使用 C 语言开发，将 Python 源代码编译为 CPython 的字节码，再由虚拟机解释执行。

（二） Jython（原来也称：JPython）

Jython 是在 JVM 上实现的 Python，使用 Java 语言编写。Jython 允许程序员写 Python 代码，还可以把 Java 模块加载到 Python 模块中使用。Jython 使用了 JIT 技术，也就是说运行时 Python 代码会先转化成 Java 字节码（不是 java 源代码），然后使用 JRE 执行。程序员还可以用 Jython 把 Python 代码打成 jar 包，这些 jar 和 java 程序打包成的 jar 一样可以直接使用，这样就允许 Python 程序员写 Java 程序了。

（三） IPython

IPython 是基于 CPython 之上的一个交互式解释器，相对于 CPython 而言，其交互方式有所增强，但是执行代码的功能与 CPython 是相同的。

（四） PyPy

PyPy 是使用 Python 实现的 Python 动态编译器，提供了 JIT 编译器和沙盒功能，会把代码转为机器码，因此在运行速度上要比 CPython 快很多。其起源是 Python 开发者为了更好地应用 Hack Python 而创建的项目。灵活性高、易于使用和试验，不过对于第三方模块的支持不足。

（五） IronPython

IronPython 是一种使用 C# 语言实现的解释器，它可以在 .NET 和 Mono 平台使用。IronPython 是兼容 Silverlight 的，配合 Gestalt 就可以直接在浏览器中执行。

四、Python 程序的运行方式

由于 Python 是一种解释型的程序设计语言，所以其运行方式有两种：交互式和文件式。

（一）交互式

交互式是指 Python 解释器逐行接收 Python 代码并即时响应。即在命令行窗口中直接输入代码，按下回车键就可以运行代码，并立即看到输出结果；执行完一行代码，你还可以继续输入下一行代码，再次回车并查看结果……整个过程就好像我们在和计算机对话，所以称为交互式编程，前面我们都是采用这种方式。

（二）文件式

文件式也称批量式，是指先将 Python 代码保存在文件中，再启动 Python 解释器批量解释执行代码。即先创建一个源文件，将所有代码编写在源文件中，让解释器逐行读取并执行源文件中的代码，直到文件末尾，也就是批量执行代码。这是最常见的编程方式，也是我们接下来要主要采用的方式。

打开 Python 3.10 IDLE 选择 File –> New File 菜单，新建一个 Python 文件，输入代码，如图 1–5 所示，并将其保存为：乘法运算 .py。

图 1–5　乘法运算源代码

通过 File –> Open 菜单打开“乘法运算 .py”源文件，然后在源文件中的菜单栏中选择 Run–>Run Module，或者按下 F5 快捷键，就可以执行源文件中的代码了，执行完之后在 Python 3.10 IDLE 界面可以看到运行结果，如图 1–6 所示。

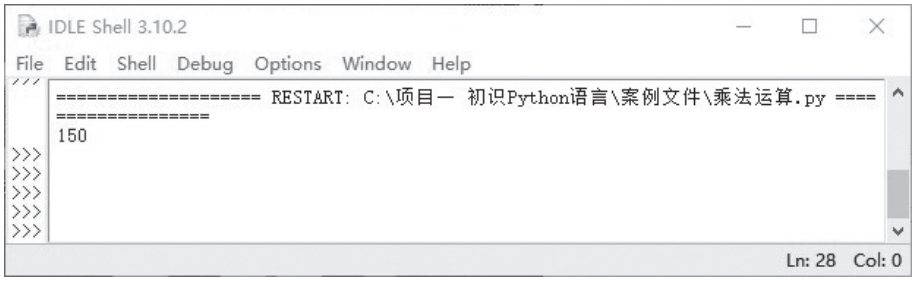

图 1–6　乘法运算运行结果

任务实施

一、下载并安装 Python

在 Windows 上安装 Python 和安装普通软件一样简单，下载安装包后单击“下一步”按钮即可。

小贴士

安装过程需要注意什么呢？

（1）尽量勾选 Add Python 3.10 to PATH，这样可以将 Python 命令工具所在目录添加到系统 Path 环境变量中，以后开发程序或者运行 Python 命令会更方便。

（2）Python 支持两种安装方式，默认安装会勾选所有组件，并安装在系统盘；自定义安装可以手动选择要安装的组件，并安装到其他盘符。

（3）IDLE 是 Python 自带的简易开发环境，安装完成以后，在 Windows 开始菜单中找到 Python 3.10 文件夹，在这里可以看到 IDLE 工具。

二、下载并安装集成开发环境 PyCharm

PyCharm 是由 JetBrains 公司研发，用于开发 Python 的集成开发环境（IDE，表示辅助程序员开发的应用软件）。

（1）进入 PyCharm 官方下载页面（如图 1-7 所示），可以看到 PyCharm 有 2 个版本，分别是 Professional（专业版）和 Community（社区版）。其中，专业版是收费的，只能免费试用 30 天；而社区版是完全免费的。强烈建议初学者使用社区版，因为该版本不会对学习 Python 产生任何影响。

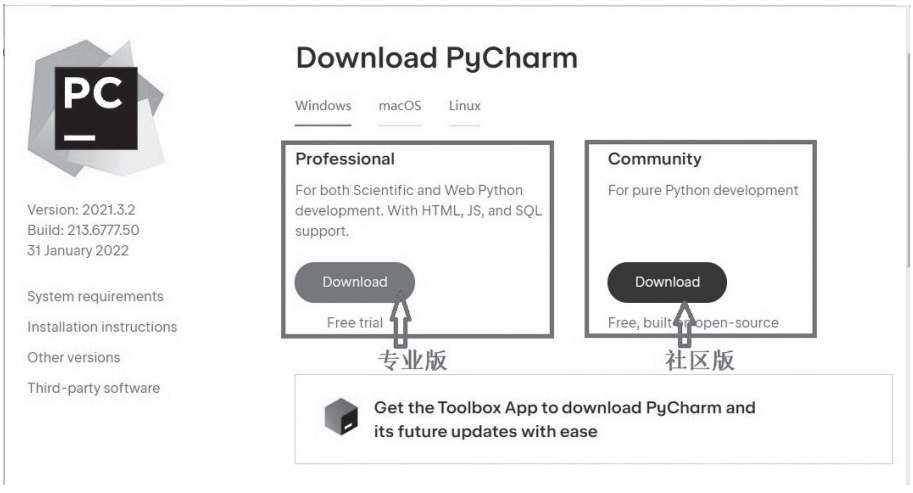

图 1-7　PyCharm 官方下载页面

（2）双击下载到的 pycharm－community－2021.3.1.exe 文件图标，打开安装向导，可以看到开始安装 pycharm 的首界面，如图 1－8 所示。

图 1-8　开始安装界面

（3）直接单击“Next”，可以看到如图 1－9 所示的对话框，设置 PyCharm 的安装路径，建议不要安装在系统盘（通常 C 盘是系统盘）。

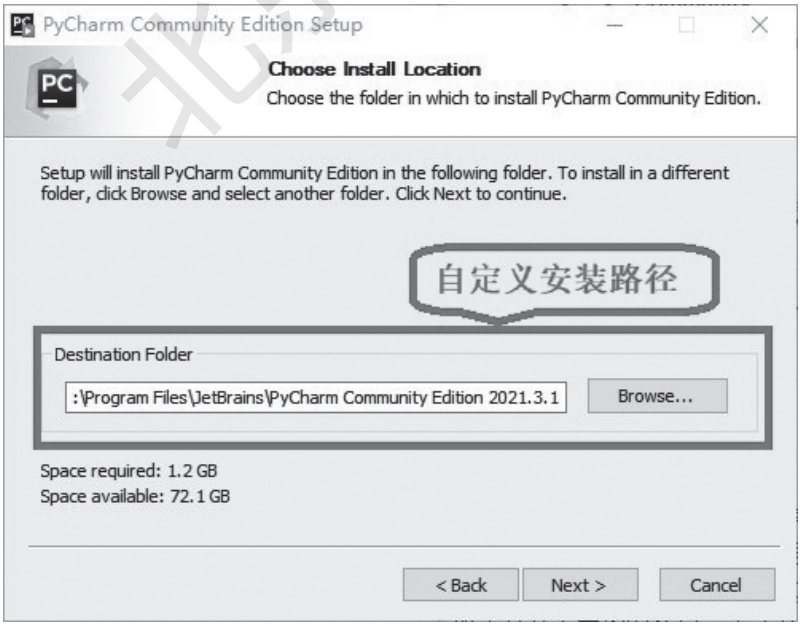

图 1-9　设置 PyCharm 安装路径

（4）继续单击“Next”，可以看到如图 1-10 所示的对话框，这里需要进行一些设置，自行选择需要的功能，若无特殊需求，按图中勾选即可。

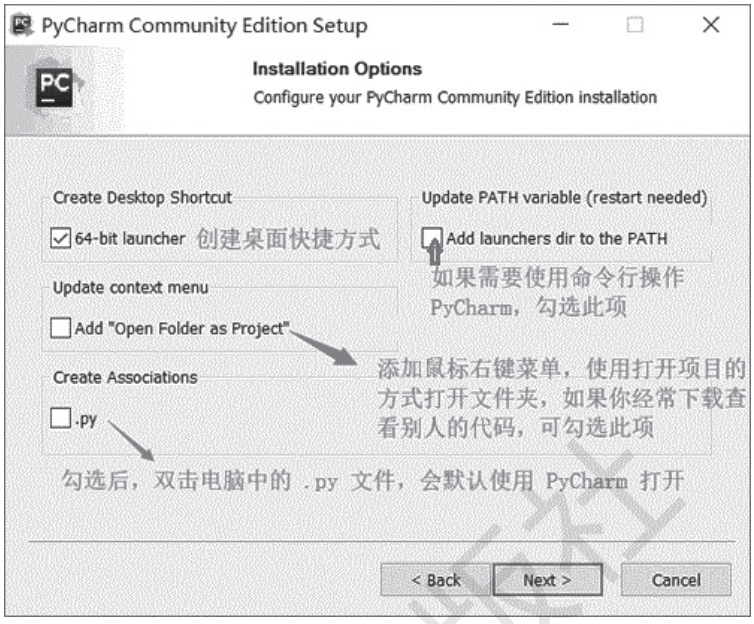

图 1-10　安装设置对话框

（5）继续单击“Next”，可以看到如图 1-11 所示的对话框，用来设置选择开始菜单文件，直接选择默认即可，单击“Install”，等待安装进度条达到 100%，PyCharm 就安装完成了。

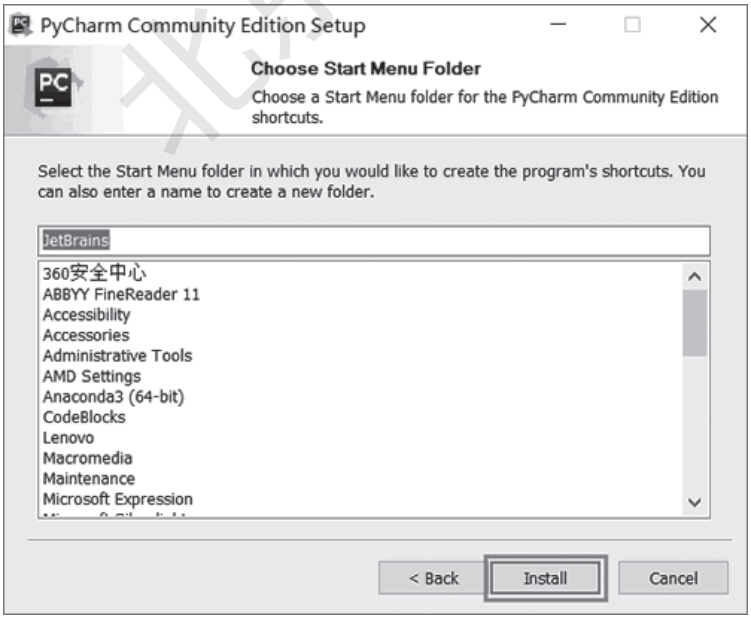

图 1-11　选择开始菜单文件

三、PyCharm 配置 Python 解释器

当我们首次启动 PyCharm，会自动进行配置 PyCharm 的过程（比如：选择 PyCharm 界面显式风格等），读者可根据自己的喜好进行配置，也可以选择退出，使用默认配置，配置过程简单明了，这里就不赘述了。下面我们一起来配置一下 PyCharm 的 Python 解释器。

（1）安装 PyCharm 完成之后，打开它会显示如下图 1－12 所示的 PyCharm 初始化界面。

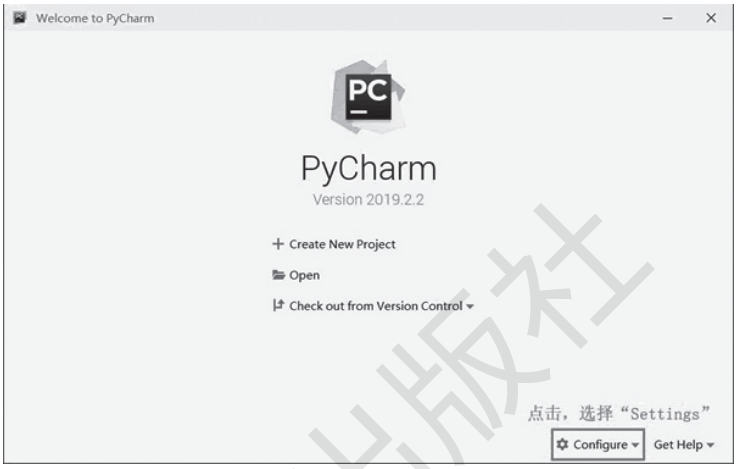

图 1-12　PyCharm 初始化界面

（2）单击图 1－15 所示的 Configure 选项，选择“Settings”，进入图 1－13 所示的界面，看到“No interpreter”，表示未设置 Python 解释器。

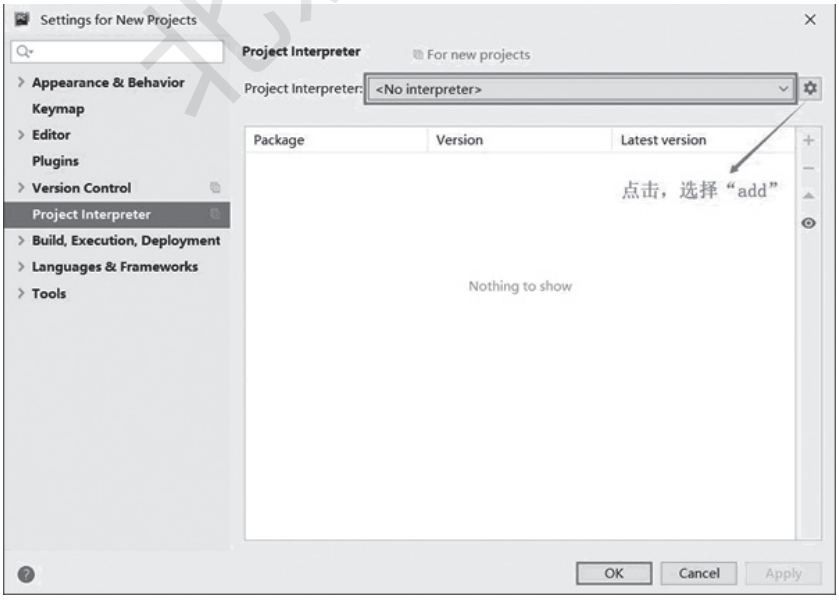

图 1-13　设置 Python 解释器界面

（3）单击图 1–13 所示的“No interpreter”后面的添加，选择“add”，会弹出如图 1–14 所示的窗口，可以手动给 PyCharm 设置 Python 解释器。

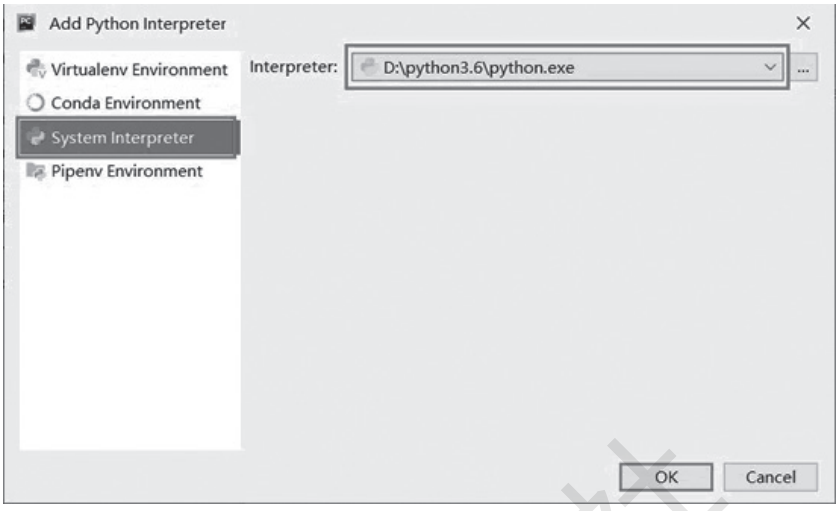

图 1–14　添加 Python 解释器界面（1）

（4）单击图 1–14 所示的“System Interpreter”（使用当前系统中的 Python 解释器），右侧找到你安装的 Python 目录，并找到 Python.exe，然后单击“OK”。此时界面会自动跳到图 1–13 所示的界面，这时会显示出可用的解释器，如图 1–15 所示，再次单击“OK”即可完成配置。

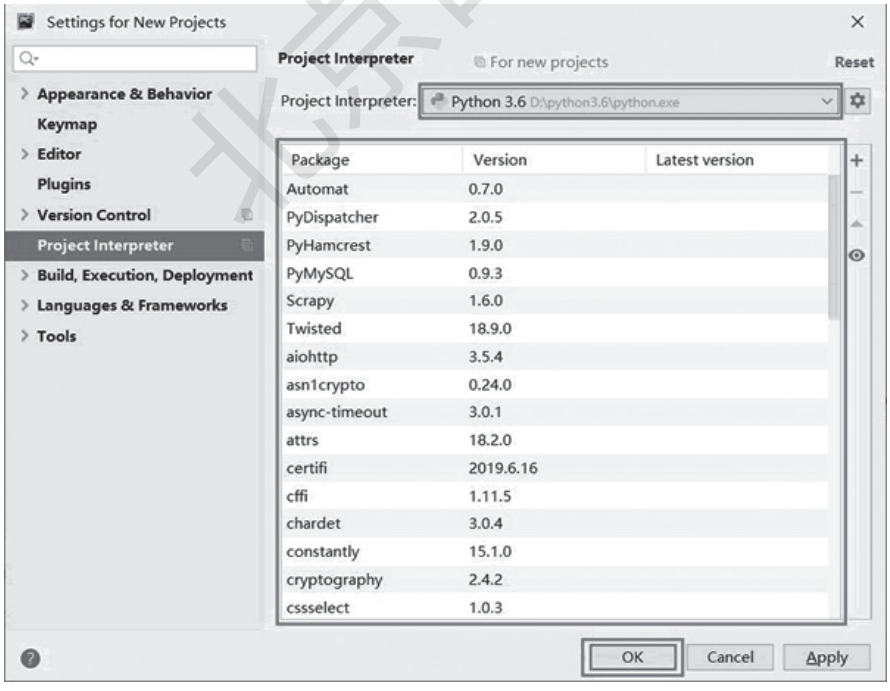

图 1–15　添加 Python 解释器界面（2）

（5）等待 PyCharm 自动完成配置，会自动回到图 1-15 所示的 PyCharm 初始化界面，接下来我们就可以使用 PyCharm 进行 Python 程序开发了。

四、使用 PyCharm 编写程序

前面我们介绍了由于 Python 是一种解释型语言，支持交互式编程和文件式编程，关于交互式编程和使用 Python 3.10 IDLE 编程我们都已经介绍了，下面我们来学习使用 PyCharm 编写第一个 Python 程序。

（1）打开集成开发工具 PyCharm，可以看到如图 1-16 所示界面，界面中共有三个选项，这三个选项的作用如下：

- 新建项目：用来创建一个新项目。
- 打开：用来打开已经存在的项目。
- 从 VCS 获取：从版本控制中检出项目。

这里，我们选择第 1 个选项，创建一个新项目，单击“新建项目”进入项目设置界面。

图 1-16　创建项目的界面

（2）在如图 1-17 所示界面中，根据需要设置项目存储路径及项目名称，这里我们设置项目名称为：项目一 案例，然后单击“创建”按钮。

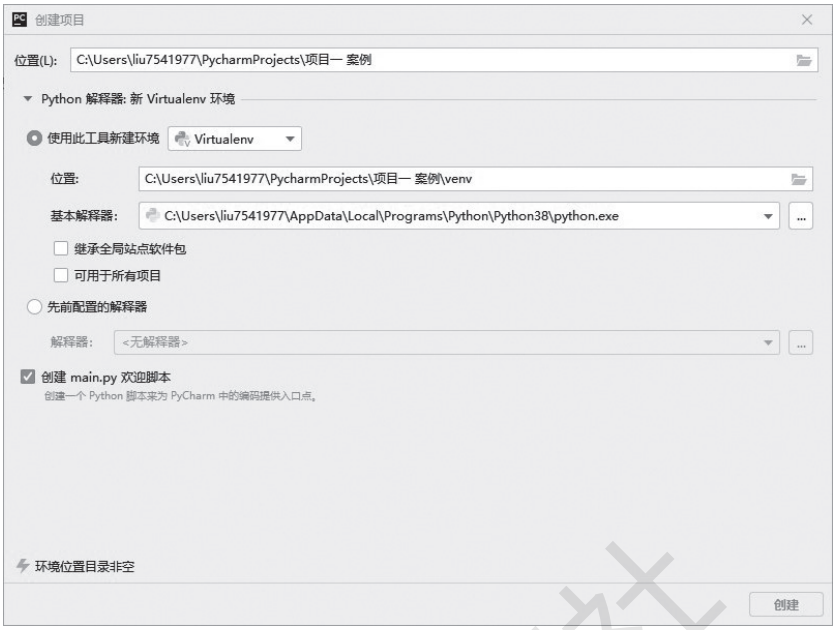

图 1-17　设置项目名及保存路径

（3）进入如图 1-18 所示界面，可以看到在“项目一案例”中系统自动创建了一个 Python 文件：main.py。

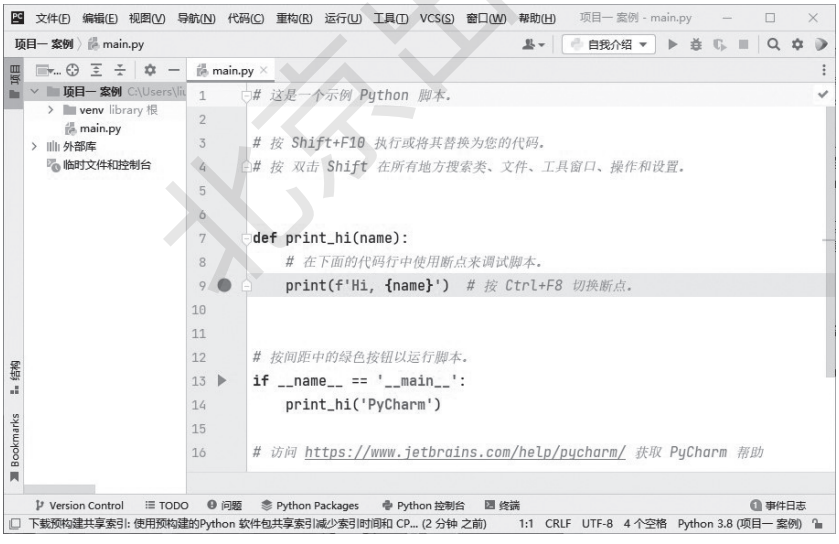

图 1-18　自动新建了一个名为 main 的 Python 文件

（4）在图 1-18 中，右键单击左侧的“main.py”文件图标，选择“另存为”，打开另存为对话框，可以看到如图 1-19 所示对话框，将 main.py 文件另存为“自我介绍 .py”，单击“确定”按钮。

图 1-19　将 main 文件另存为“自我介绍”

（5）这时在主界面可以看到如图 1-20 所示界面，在编辑区域删除其他文字，输入图中代码，右键单击上方的“自我介绍 .py”文件名，选择“运行 自我介绍 .py”。

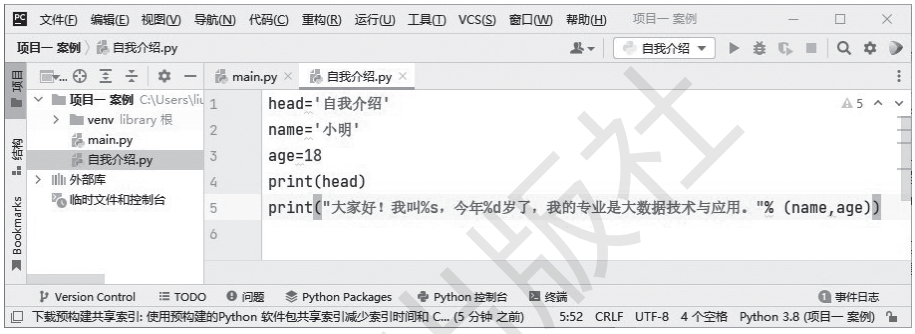

图 1-20　编写代码

（6）这时在代码编辑区域下方可以看到如图 1-21 所示的运行结果界面。

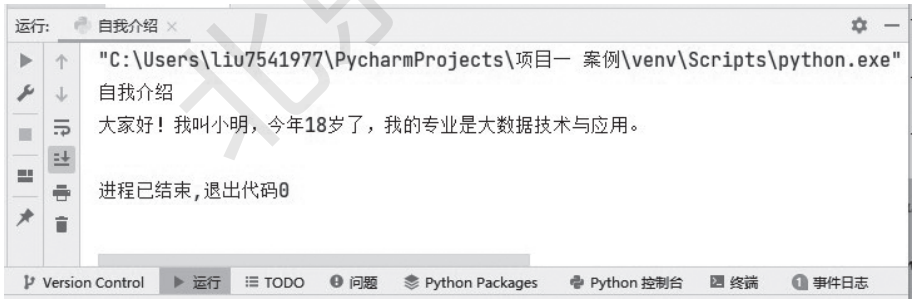

图 1-21　运行结果

小贴士

在帮助菜单中偷学一招

对于初学者，打开“帮助”菜单中的“每日小技巧”，可以看到如图 1-22 所示的 PyCharm 中自带的一些编程小技巧，还可以点击“下一条小技巧”按钮多学几招哦！

图 1-22　每日小技巧界面

思考与练习

一、选择题

1. 在 Python 3. x 系列中用函数 print() 代替 print 语句，以下说法错误的是（　　）。
   A. print() 函数能够指定其他的分隔符和结束符
   B. print 语句只能作为语句使用
   C. print() 函数可以在表达式中使用
   D. print 语句更灵活
2. 在 Python 3. x 系列中除法运算符“/”与 Python 2. x 系列不同，1/2 的结果是(　　)。
   A. 0　　B. 0.5
   C. 0.5000　　D. 1
3. Python 3. x 系列与 Python 2. x 系列相比，异常处理的表述，错误的是（　　）。
   A. Python 2.x 系列中所有类型的对象都是直接被抛出的
   B. 在 Python 3.x 系列中处理异常只能用 raise Exception(args)
   C. Python 3.x 系列与 Python 2.x 系列捕获异常的语法相同
   D. 在 Python 3.x 系列中取消了异常类的序列行为和. message 属性

二、填空题

1. Python 3. x默认使用的是_____,从而可以很好地支持中文或其他非英文字符。

2. 在 Python 3. x 系列中取消了_____类型，只保留了一种整型类型_____，新增加了_____类型。

3. 在 Python 3. x 系列中废弃了不等于运算符_____的写法。

4. 在 Python 3. x 系列中废弃了八进制字面量_____的表示方式，只保留了_____表示方式。

三、思考题

由于 Python 是一门解释型语言，Python 程序代码必须通过解释器来执行，目前常见的 Python 解释器有哪些？请简单介绍一下。

四、实践题

打开集成开发工具 PyCharm，新建一个项目，项目名称命名为：subject01，新建一个 Python 程序实现古诗格式化显示，并将文件保存为：exercise02-1. py，最终显示效果如图 1-23 所示。

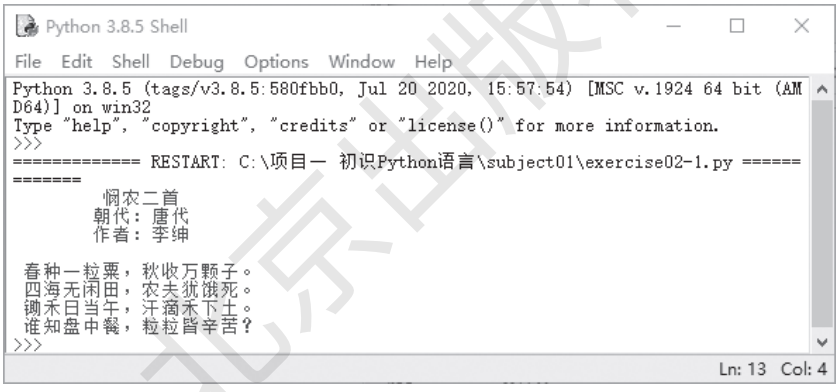

图 1-23 古诗格式化显示效果