中等职业教育新目录新技术新形态系列教材

Python 程序设计基础

吕宇飞　主　编

苏豫全　副主编

電子工業出版社
Publishing House of Electronics Industry
北京・BEIJING

内 容 简 介

本教材内容涉及 Python 语言的基本语法、数据类型、运算符、流程控制、函数和模块的使用、数据结构、基本算法、正则表达式等。以海龟绘图、摩尔斯码、文件读写、Excel 办公自动化、网络爬虫等应用案例和工作任务，按照项目教学法编写，理实一体，定位清晰。让初学者在应用过程中学习 Python 基础知识，并使其逐步具备简单的办公自动化的编程能力。

本教材配有相关素材和资源，请登录华信教育资源网免费获取。本教材既可以作为职业院校程序设计入门课程的教学用书，也可以作为各类编程培训班的教学用书，还可供对计算机编程感兴趣或有办公自动化和自动化运行维护管理需求的从业人员参考使用。

图书在版编目（CIP）数据

Python 程序设计基础 / 吕宇飞主编. —北京：电子工业出版社，2023.9

ISBN 978-7-121-46590-1

Ⅰ. ①P… Ⅱ. ①吕… Ⅲ. ①软件工具－程序设计－教材 Ⅳ. ①TP311.561

中国国家版本馆 CIP 数据核字（2023）第 204350 号

责任编辑：郑小燕
印　　刷：
装　　订：
出版发行：电子工业出版社
　　北京市海淀区万寿路 173 信箱　邮编　100036
开　　本：880×1 230　1/16　印张：18.75　字数：432 千字　插页：1
版　　次：2023 年 9 月第 1 版
印　　次：2023 年 9 月第 1 次印刷
定　　价：61.80 元

凡所购买电子工业出版社图书有缺损问题，请向购买书店调换。若书店售缺，请与本社发行部联系，联系及邮购电话：（010）88254888，88258888。

质量投诉请发邮件至 zlts@phei.com.cn，盗版侵权举报请发邮件至 dbqq@phei.com.cn。

本书咨询联系方式：（010）88254550，zhengxy@phei.com.cn。

前言

《中华人民共和国国民经济和社会发展第十四个五年规划和2035年远景目标纲要》提出“迎接数字时代，激活数据要素潜能，推进网络强国建设，加快建设数字经济、数字社会、数字政府，以数字化转型整体驱动生产方式、生活方式和治理方式变革”。随着数字经济蓬勃发展，Python 广泛应用于云计算、物联网、大数据、智慧城市、人工智能、区块链等诸多领域，程序设计基础将为紧跟技术发展趋势的人才培养构筑基石。

本教材专为 Python 程序设计入门而设计，具有以下特点：

1．注重培养学习者的编程兴趣，以多个应用实例贯穿介绍 Python 语言的基础知识，例如以海龟绘图介绍流程控制，以摩斯密码介绍列表、元组、字典等数据结构，以一首诗介绍正则表达式。

2．注重体现学习者的学习价值，以文件读写、文件和目录运行维护、Excel 办公自动化、合法爬取数据等工作任务学习 Python 内置模块及第三方库的使用，并提高编程综合能力，使学习者在入门阶段已经能够将 Python 应用于办公自动化。

3．注重锻炼学习者的核心能力，各种信息处理最终都表现为数值或字符的处理，入门阶段重点掌握流程控制和处理数值、字符串、列表的能力，为今后学习第三方库等专门领域的编程应用打下坚实基础。

除此以外，教材着力做好以下方面：

1．做一本学习者可以阅读的教材，而非技术手册。文中结合阅读方式穿插配图，在阅读过程中，以设问和引导探究的方式修改与优化代码。

2．做一本明理透彻、概念准确的教材，而非止于基础。知识点无难易之分，明晰则易，含糊则难，例如分析变量的类型和在内存中的存储情况有助于深刻理解代码，应对人为错误。

3．做一本有教学设计思想的教材，而非内容呈现。内容的编排和组织符合学习者的认知规律，多处做了独到的设计，从日常经验入手讲解知识点，例如分析如何构建循环体培养设

计代码能力，以遮罩效果理解正则表达式，寻找公共结构解决递推问题等，以此启智增慧，注重编程素养的培养。

本教材提供配套资源，包含各章节源代码以及相关的素材和效果图等，读者可以登录华信教育资源网免费获取。本教材采用文件和目录维护、Excel 办公自动化等工作任务，教师通过解决问题的程度即可评价学习者的编程能力，解决了 Python 程序设计因为实现方法众多而难以评价的问题，有利于学校在教学过程中开展评价。

本教材由中等职业教育、高等职业教育计算机专业教研团队和企业软件工程师共同编写，吕宇飞任主编，苏豫全任副主编，参编人员有陈云志、余运祥、葛巧燕、邵泽城、刘晓梅、罗炎香、王永森、刘陈亮、汪忠校、朱思俊、林聪太、方昱霞、王宗政。感谢杭州职业技术学院、广州中望龙腾软件股份有限公司、杭州古德微机器人有限公司、杭州有渔智学科技有限公司提出宝贵意见并给予大力支持！书中难免出现疏漏之处，敬请广大读者予以批评指正。在此表示衷心感谢！

编者

目 录

1 词汇

1.1 Python 的起源与应用

python['paɪθən] 蟒蛇

1.2 Python 的编程环境

path[pɑ:θ] 路径

download[ˌdaʊn'ləʊd, 'daʊnləʊd] 下载

PyCharm['paɪ tʃɑ:m] 一款由 JetBrains 公司开发的 Python 集成开发环境（IDE）

1.3 第一个 Python 程序——我爱我的祖国

print[prɪnt] 打印

input['ɪnpʊt] 输入

编程环境：

project['prɒdʒekt, prə'dʒekt] 项目，方案

file[faɪl] 文件

run[rʌn] 运行

提示信息：

IndentationError: unexpected indent

缩进错误：意外的缩进

indentation Error[ˌɪnden'teɪʃn] ['erə(r)] 缩进错误

unexpected[ˌʌnɪk'spektɪd] 想不到的

indent[ɪn'dent, 'ɪndent] 缩进

invalid character[ɪn'vælɪd]['kærəktə(r)] 无效字符

syntax Error['sɪntæks]['erə(r)] 语法错误

Name Error[neɪm]['erə(r)] 命名错误

process['prəʊses, prə'ses] 过程

finish['fɪnɪʃ] 完成

exit['eksɪt] 出口，退出

code[kəʊd] 代码，密码

traceback[treis] [bæk] 回溯，反向追踪

defined[dɪ'faɪnd] 定义（define 的过去分词和过去式）

invalid[ɪn'vælɪd, 'ɪnvəlɪd] 不承认的，无效的

character['kærəktə(r)] 文字，字母，符号

scan[skæn] 扫描，浏览

string[strɪŋ] 字符串

literal['lɪtərəl] 文字的，逐字的

1.4 第二个 Python 程序——代码编辑与调试

output['aʊtpʊt] 输出

prompt[prɒmpt] 提示符

format['fɔ:mæt] 格式

debug[ˌdi:'bʌg] 调试

show[ʃəʊ] 显示

console[kən'səʊl, 'kɒnsəʊl] 控制台

变量名：

student ['stju:dnt] 学生

第 1 章

认识 Python

本章节涉及的内容

- Python 的起源和应用介绍
- 搭建 Python 的编程环境
- 使用 PyCharm 编辑和运行程序
- 运用 print( )输出字符串数据
- 识别运行程序时常见的出错信息

1.1 Python 的起源与应用

1.1.1 Python 的起源

Python 的设计者是吉多·范·罗苏姆（Guido van Rossum），荷兰人，早在 1982 年就获得了阿姆斯特丹大学数学和计算机硕士学位。虽然他是一位小有名气的数学家，但是他更对计算机专业情有独钟，享受着计算机带来的乐趣。据说在 1989 年圣诞节期间，吉多·范·罗苏姆为了打发圣诞节的假期，决心开发一种新的编程语言，后来命名为 Python。之所以命名为 Python（中文译名为“蟒蛇”），是因为他是英国肥皂剧《Monty Python》的忠实粉丝。

1.1.2 Python 的应用

目前，Python 是最流行、应用最广泛的编程语言之一，之所以它的热度越来越高，是因为它拥有丰富和强大的内置“库”，利用这些“库”可以快捷地完成一些功能模块，因此是程序员的首选语言。它主要被用于 Web 开发、网络爬虫、桌面软件开发、游戏开发、云计算开

发、科学计算和人工智能等领域。

1. Web 开发

Python 提供了多种 Web 框架，如图 1-1-1 所示，其中 Django 为常用的 Web 框架之一，它自动生成 admin.py 文件可用于 Web 配置；models.py 文件用于配置和管理与数据库相关的操作；views.py 用于设计视图。这些文件的自动生成，大大减少了编程人员的代码编写量，提高了编程效率。国内许多著名网站，例如豆瓣等，都采用 Python 进行编写。

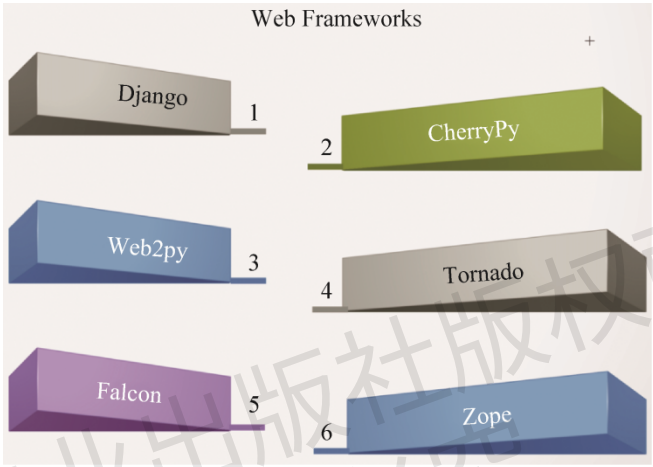


图 1-1-1　Web 框架

2. 网络爬虫

如图 1-1-2 所示，网络爬虫按照一定的规则自动抓取网络上的信息和数据。例如，先抓取电子商务平台销售商品的评论及销售数量，然后进行分析，对用户消费前景进行预测等。利用 Python 的 Urllib 库、Requests 库和 Scrappy 等框架，用几行代码便可实现从网络上获取有用的数据和信息。

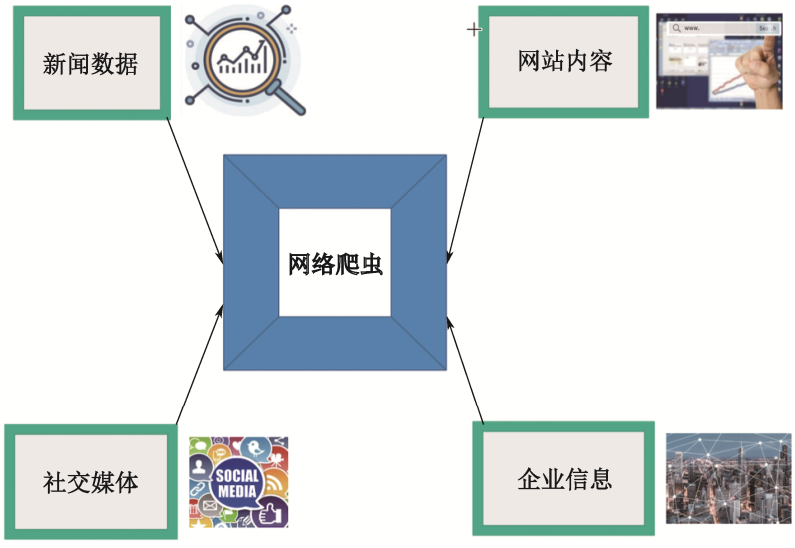


图 1-1-2　网络爬虫应用

3. 桌面软件开发

Python 在图形界面和桌面软件开发方面功能十分强大。例如使用 PyQt、Tkinter 框架开发桌面软件，使用 wxWidgets 框架开发图形界面。

4. 游戏开发

Python 有 Pygame、PyOpenGI 等丰富的游戏开发库，飞扬的小鸟、Python Shooter、接水果、坡道驾驶等游戏都是采用 Python 编写的经典游戏，如图 1-1-3 所示。

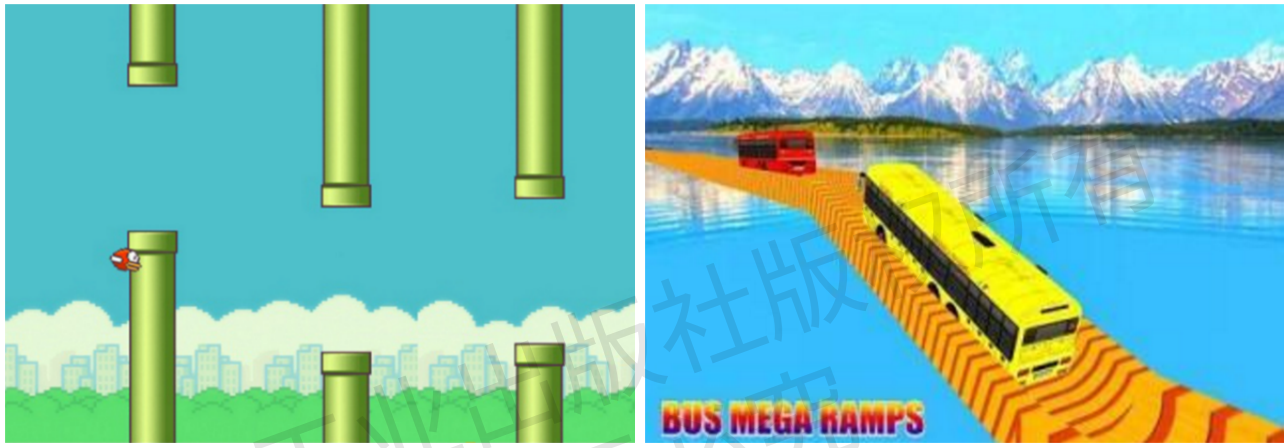


图 1-1-3　Python 开发的经典游戏

5. 云计算开发

Python 是从事云计算开发需要掌握的编程语言。目前，云计算的框架 OpenStack 就是用 Python 语言开发的。如果需要对 OpenStack 进行二次开发，那么必须精通 Python 语言。

6. 科学计算

随着 NumPy、SciPy、Matplotlib、Enthought librarys 等众多库的开发，Python 越来越适合于做科学计算。基于 NumPy 和 SciPy 等基础的数值运算软件包提供的矩阵对象（ndarray）和运算方法，使用户可以方便地进行数值分析和处理。

7. 人工智能

目前，Python 提供了大量的第三方库，例如 Pandas、PyBrain、Sklearn、Keras 等用于数据分析和可视化、数据建模、神经网络、深度学习等人工智能领域，很多学习者甚至将 Python 与人工智能画等号。

1.2 Python 的编程环境

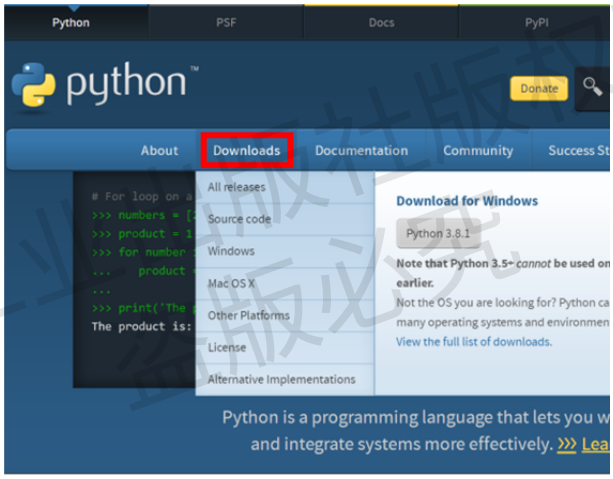
你将获取的能力：

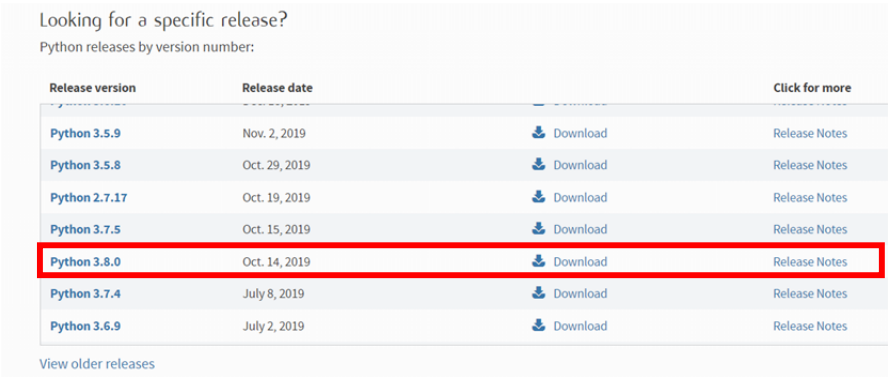
能够搭建 Python 的编程环境。

1.2.1 Python 的安装与测试

步骤 1：Python 的下载

打开 Python 官网下载地址，找到 Downloads 菜单下 All releases 栏目，选择 Python 3.8.0 版本，如图 1-2-1 所示单击 Windows x86-64 web-based installer 进行下载。





Looking for a specific release?

Python releases by version number:

Release version	Release date		Click for more
Python 3.5.9	Nov. 2, 2019	Download	Release Notes
Python 3.5.8	Oct. 29, 2019	Download	Release Notes
Python 2.7.17	Oct. 19, 2019	Download	Release Notes
Python 3.7.5	Oct. 15, 2019	Download	Release Notes
Python 3.8.0	Oct. 14, 2019	Download	Release Notes
Python 3.7.4	July 8, 2019	Download	Release Notes
Python 3.6.9	July 2, 2019	Download	Release Notes

View older releases

图 1-2-1　Python 软件下载

步骤 2：Python 的安装

（1）双击 Python 3.8.0 安装程序，打开后一定要勾选 Add Python 3.8 to PATH 复选框，如图 1-2-2 所示，再选择 Customize installation（自定义）安装。

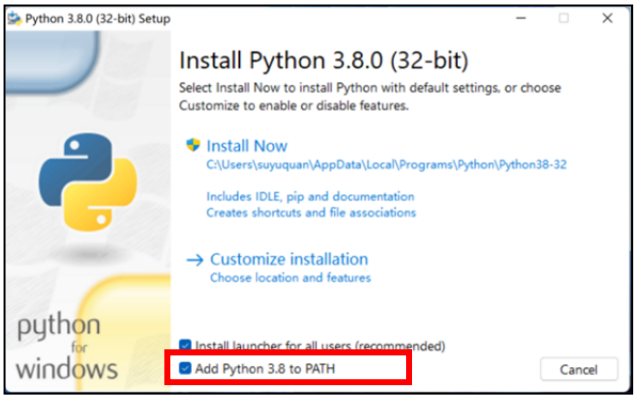


图 1-2-2　Python 安装（1）

（2）在单击自定义安装中，勾选 Documentation、pip、tcl/tk and IDLE、Python test suite 等全部复选框。单击“Next”按钮，进入高级选项，注意选择 Install for all users，下面 Customize install location 位置是 Python 的安装目录。如图 1-2-3 所示单击“Next”按钮直至安装完成。

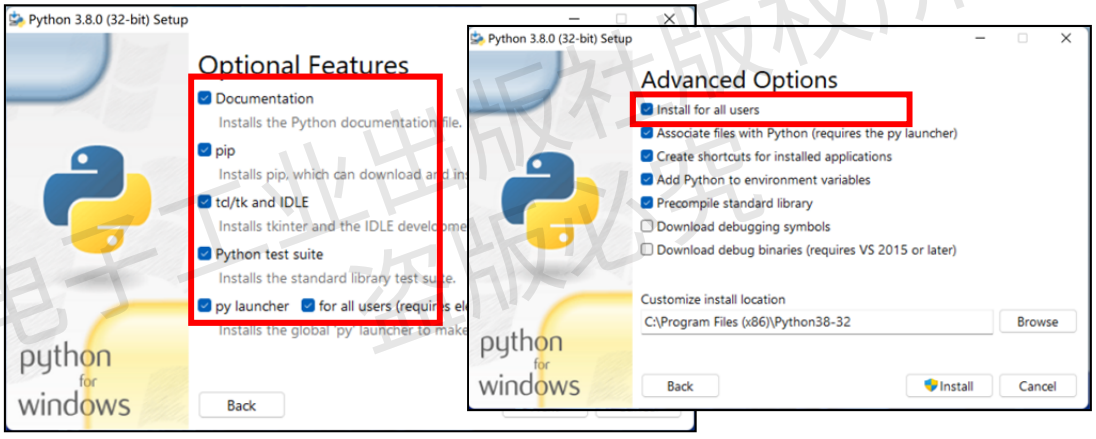


图 1-2-3　Python 安装（2）

步骤 3：安装测试

按 Win+R 键在运行中输入 cmd，在 cmd 命令窗口的提示符>后输入 Python 进入交互模式窗口，出现如图 1-2-4 所示信息表示安装成功。

图 1-2-4　Python 测试

在 Python 交互模式窗口提示符>>>后输入 quit( )或按下 Ctrl+Z 后按下回车键均可退出 Python。

在 cmd 命令窗口的提示符>后输入“where python”，可以查看 Python 解释器 python.exe 所在的位置。

1.2.2 PyCharm 的安装与启动

Python 常用的开发环境有 IDLE、PyCharm、Wing IDE、Eclipse、IPython 等。这些开发环境本质上都是 Python 解释器 python.exe 的封装，换言之，它们是解释器的外挂程序，便于编程开发。

其中 IDLE 是 Python 内置的开发环境，随着 Python 一同安装完成，包括交互式命令行、编辑器、调试器等基本组件，几乎具备 Python 开发的所有功能，非常适合初学者使用。在 Windows 窗口中，如图 1-2-5 所示即可启动 IDLE，默认进入交互模式。

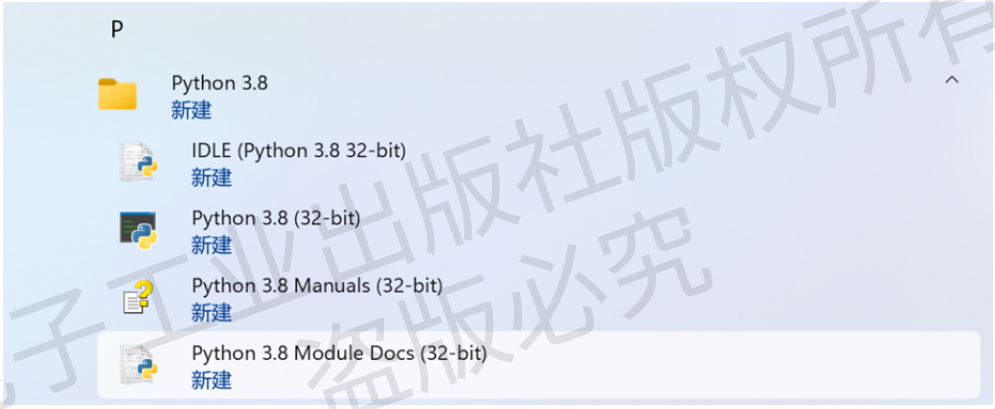


图 1-2-5 启动 IDLE

PyCharm 是一款功能强大的 Python 编辑器，具有代码调试、语法高亮、代码跳转、智能提示、自动完成等功能。主要安装步骤如下。

步骤 1：下载 PyCharm

访问 PyCharm 官方网站，可以免费下载、安装应用社区版，参考资源见信息文档。

步骤 2：安装 PyCharm

进入安装程序界面，单击“Next”按钮开始安装，并对安装路径进行修改，也可以保持默认路径安装。在安装过程中勾选安装版本、关联文件和添加安装的 Python 路径到系统路径等，便可完成安装，如图 1-2-6 所示。

步骤 3：启动 PyCharm

双击桌面上的 PyCharm 图标，选择“Do not import settings”，单击“OK”按钮，再单击“Create”按钮，便可进入 PyCharm 编程界面。

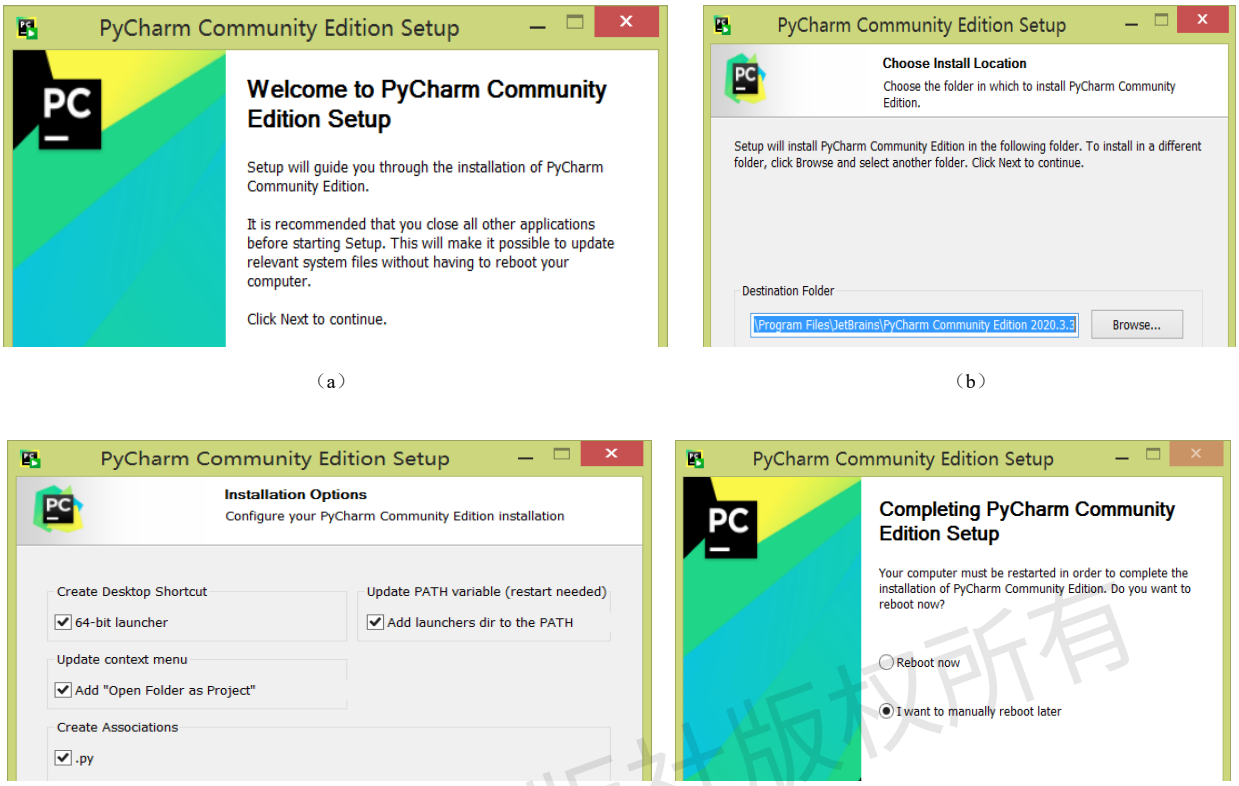


图 1-2-6　PyCharm 主要安装步骤

1.3　第一个 Python 程序——我爱我的祖国

你将获取的能力：

能够使用 PyCharm 编辑和运行程序；

能够编写运用 print( )函数输出字符串数据的 Python 程序；

能够识别运行程序时常见的出错信息。

编写程序可以从输入数据、程序处理和输出结果三个环节入手。因此本节介绍 print( )函数、input( )函数和定义变量等基础知识，讲解如何输入、存放、处理和输出数据。

1.3.1　案例：第一个 Python 程序

设计程序“第一个程序.py”，运行程序，输出结果：

```
我爱我的祖国
```

1. 示例代码

第 01 行　　`print("我爱我的祖国")`

2. 思路简析

（1）在 PyCharm 中创建“第一个程序.py”文件

步骤 1：打开 PyCharm 编辑器，选择 File 菜单下的 New Project 创建工程，工程文件夹设置为“D:\练习”，单击“Create”按钮创建，如图 1-3-1 所示。

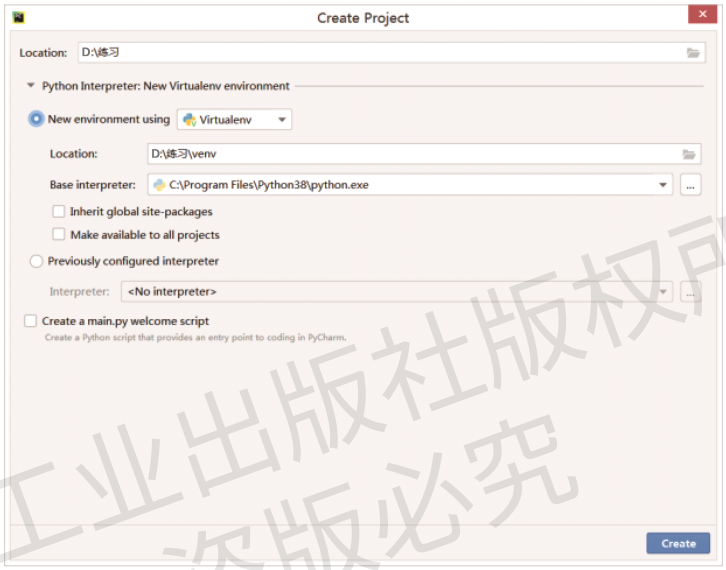


图 1-3-1　在 PyCharm 中创建工程

步骤 2：右击“练习”工程文件夹，选择“New”→“Python File”选项，如图 1-3-2 所示，新建名为“第一个程序”的 Python 文件。

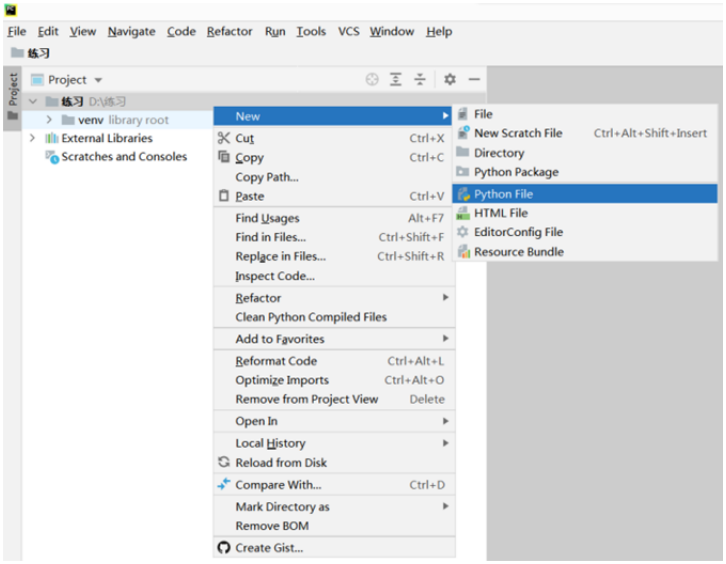


图 1-3-2　在 PyCharm 中创建文件

（2）调用 Python 内置函数 print( )，输出字符串

在代码编辑窗口输入第 1 行语句：print("我爱我的祖国")，如图 1-3-3 所示。

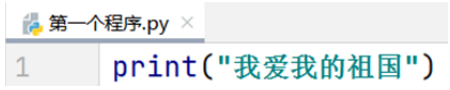


图 1-3-3　用 print( )输出字符串

输入时注意 Python 程序的基本语法。

① print("我爱我的祖国")为一行完整的代码，需要顶格输入。

② 除输入“我爱我的祖国”这几个汉字以外，其余字符包括引号和括号都需要在英文状态下输入。

③ 严格区分大小写。

④ 引号和括号需要成对出现。

⑤ 在一对单引号或一对双引号内的字符称为字符串。

（3）调试并运行代码，输出结果

如图 1-3-4 所示，在代码编辑窗口空白处单击鼠标右键，在右键菜单中选择“Run'第一个程序'”运行程序，即在 PyCharm 编辑器左下方显示如图 1-3-5 所示的界面。

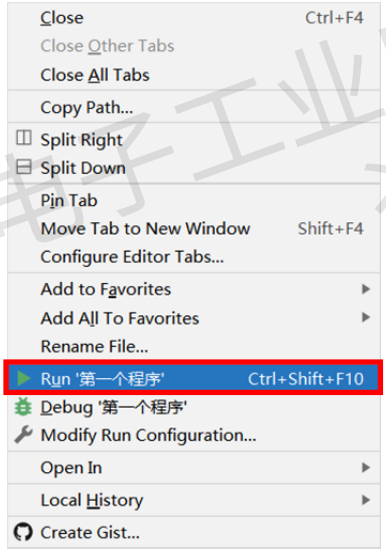


图 1-3-4　运行程序

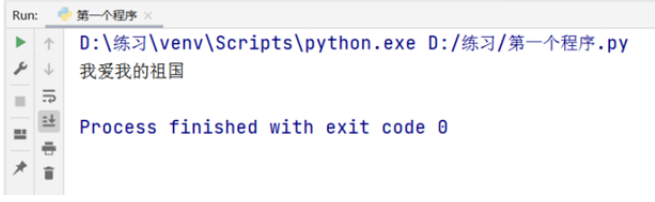


图 1-3-5　运行程序后输出结果

图 1-3-5 中的内容解释如下：

```
D:\练习\venv\Scripts\python.exe D:/练习/第一个程序.py    --->运行的程序名称
我爱我的祖国                                              --->程序运行的输出结果

Process finished with exit code 0   --->exit code 0 表示程序执行成功，正常退出
```

如此运行程序后，PyCharm 右上方工具栏如图 1-3-6 所示，可以单击下拉列表箭头图标选择要运行的程序，单击相应的按钮可继续运行和调试程序。

（4）初识出错代码

① 如图 1-3-7 所示，由于在红色箭头处添加了空格符，导致第 1 行代码没有顶格。

图 1-3-6　PyCharm 工具栏

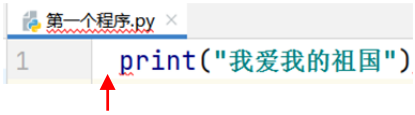


图 1-3-7　代码没有顶格错误

运行程序就其输出结果解释如下：

```
D:\练习\venv\Scripts\python.exe D:/练习/第一个程序.py
   File "D:\练习\第一个程序.py", line 1       --->指明出错位置在第 1 行代码
    print("我爱我的祖国")
IndentationError: unexpected indent
--->缩进错误：意外的缩进 Indentation 意思为缩进
Process finished with exit code 1
---> exit code 1 表示程序遇到了某些问题或者错误，非正常退出
```

② 如图 1-3-8 所示，在红色箭头处将字母 p 大写为 P。

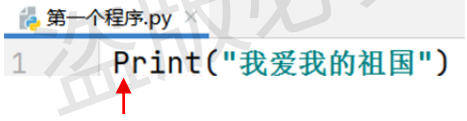


图 1-3-8　代码出现大写错误

运行程序就其输出结果解释如下：

```
D:\练习\venv\Scripts\python.exe D:/练习/第一个程序.py
Traceback(most recent call last):
  File "D:\练习\第一个程序.py", line 1, in <module>--->指明出错位置在第 1 行代码
    Print("我爱我的祖国")
NameError: name 'Print' is not defined
--->名称错误，没有定义'Print'， 因为标准库中只有print( )
Process finished with exit code 1
```

③ 如图 1-3-9 所示，红色箭头处在中文状态下输入了“（”。

图 1-3-9　代码出现中文状态“（”错误

运行程序就其输出结果解释如下：

```
D:\练习\venv\Scripts\python.exe D:/练习/第一个程序.py
  File "D:\练习\第一个程序.py", line 1          --->指明出错位置在第 1 行代码
    print（"我爱我的祖国")
          ^ --->指明出错的位置
SyntaxError: invalid character '（' (U+FF08)       --->'（'为无效字符

Process finished with exit code 1
```

④ 如图 1-3-10 所示，删除了在红色箭头处的双引号“"”。

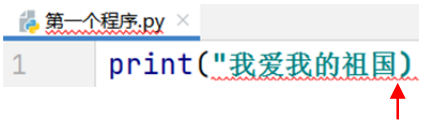


图 1-3-10　代码缺引号

运行程序就其输出结果解释如下：

```
D:\练习\venv\Scripts\python.exe D:/练习/第一个程序.py
  File "D:\练习\第一个程序.py", line 1
    print("我爱我的祖国 )                    ^ --->指明可能出错的位置
SyntaxError: EOL while scanning string literal
        --->语法错误：检测到非法结束符。因为程序没能找到字符串的另一个双引号
Process finished with exit code 1
```


1.3.2　定义字符串

字符串是 Python 中最常用的数据类型之一，是由 0 个或多个字符组成的有序字符序列，定义字符串有以下方法。

（1）使用一对单引号（'），例如'我爱我的祖国'。

（2）使用一对双引号（"），例如"我爱我的祖国"。

（3）使用一对三引号（'''），可以表示多行字符串，例如

'''我爱
我的祖国'''

1.3.3　print( )函数

print( )函数：是 Python 的内置函数，用于将文本或其他数据输出到控制台或终端窗口，其功能类似于打印机，如图 1-3-11 所示。

图 1-3-11　print( )函数图示

例 1：

```
print('我爱我的祖国')
```

输出结果为

```
我爱我的祖国
```

例 2：

```
print('''我爱
我的祖国''')
```

输出结果为

```
我爱
我的祖国
```

例 3：

```
print('123')
```

输出结果为

```
123
```

注意，'123'是字符串型数据，区别于数值 123，如果只输出数值 123，那么可以

```
print(123)
```

输出结果为

```
123
```

因此 print( )函数不仅可以输出字符串，还可以输出数值。

知识小结

1．在 PyCharm 中创建 Python 程序文件。

2．在 PyCharm 中编辑和运行程序。

3．Python 的基本语法。

4．常见的出错信息。

5．定义字符串。

6．print( )函数。

技能拓展

尝试并阅读以下错误信息。如图 1-3-12 所示，删除了在定义字符串时的一对单引号“'”。

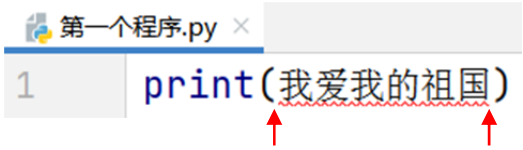


图 1-3-12　代码缺一对引号

运行程序就其输出结果解释如下：

```
D:\练习\venv\Scripts\python.exe D:/练习/第一个程序.py
Traceback (most recent call last):
  File "D:\练习\第一个程序.py", line 1, in <module>
    print( 我爱我的祖国 )                          --->指明可能出错的位置
NameError: name 我爱我的祖国 is not defined
--->命名错误：没有定义'我爱我的祖国'由于没有一对单引号，所以程序没把我爱我的祖国视为字符串

Process finished with exit code 1
```

1.4　第二个 Python 程序——代码编辑与调试

你将获取的能力：

能够理解对象、变量和赋值语句；

能够运用 input()函数输入信息并拼接字符串；

能够应用 str()、int()函数转换数据类型；

能够运用 print()函数实现格式化输出；

能够运用断点调试或交互式调试进行代码调试。

1.4.1　案例：第二个 Python 程序

设计一个能够和用户交互的程序，以“我是第×××号学员，名叫×××，我爱我的祖国!”的格式输出信息。运行程序“第二个程序.py”，输出结果：

```
请输入您的姓名：
```

此时输入：张三，程序继续运行，输出结果：

```
我是第 2000 号学员，名叫张三，我爱我的祖国!
```

1. 示例代码

```
第 01 行　stuNo=2000                                  #定义变量，变量名 stuNo
第 02 行　stuName=input('请输入您的姓名：')  #定义变量，变量名 stuName
```

第 03 行　output='我是第'+str(stuNo)+'号学员，名叫'+stuName+'，我爱我的祖国！'
　　#定义变量，变量名 output 意为将被输出的内容

第 04 行　print(output)

2. 思路简析

第 1 行代码定义变量 stuNo，赋值为 2000。

第 2 行代码调用了 Python 内置输入函数 input( )提示用户输入姓名。

第 3 行代码连接字符串。

第 4 行代码输出该字符串。

1.4.2　对象、变量与赋值语句

1. 对象

在 Python 中一切皆对象。每个对象都占有一个内存空间，至少包含类型和值。

以数值 2000 这个对象为例，如图 1-4-1 所示，程序在内存中随机申请了一个内存空间，内存地址为 2360801818192，这个对象的类型为整型，值为 2000。

如图 1-4-2 所示，一个对象就如同图书馆书架上的一本书，内存地址就是存放它的位置编号，它的类型可能属于古典文学，值即书中的内容。

图 1-4-1　局部内存空间

图 1-4-2　图书馆中的书架和书

同理，图 1-4-1 中的字符串"张三"也是一个对象，它的内存地址为 2360801888256，类型为字符串，值为张三。

id( )可以获取某个对象的内存地址，例如：

```
print(id(2000))
print(id("张三"))
```

由于内存空间为随机分配，所以输出结果通常会与图 1-4-1 所示内存地址不同。

2. 变量

变量存储的是对象的引用，它实际上只是一个标识符，用来指向内存中存储的对象，而不存储对象的实际数据，通过变量名可以访问对象。使用变量时，Python 解释器首先查找对应内存地址中存储的对象，然后执行相应的操作。

如图 1-4-3 所示，变量 stuNo 标识了内存地址 2360801818192，即指向了对象 2000；变量 stuName 标识了内存地址 2360801888256，即指向了对象"张三"。

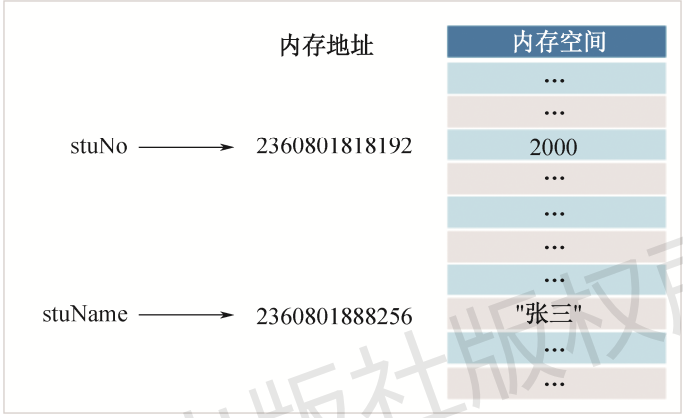


图 1-4-3　变量 stuNo 和变量 stuName

变量的命名规则：

（1）变量名只能由字母、数字和下画线组成，并且不能以数字开始。

（2）变量名中不能包含空格。

（3）变量名不能是表 1-4-1 中的 Python 保留关键字，但可以包含保留关键字。

表 1-4-1　Python 保留关键字

and	del	from	not	while	as	elif
global	or	with	assert	else	if	pass
yield	break	except	import	print	class	exec
in	raise	continue	finally	is	return	def
for	lambda	try				

例如：xingming、xingming1、xingming_1、str_xingming 等都是合法的变量名，然而 1xingming、xing#ming 和 xing ming 等则不能作为变量名。

根据变量的命名规则，变量名也可以使用汉字，例如姓名、内容等。

通俗易懂的命名能够提升代码的可读性，常见的命名方法有匈牙利命名法、驼峰命名法和下画线命名法。全书结合初学者的特点，采用小驼峰命名法。这三种命名方法，变量名命名时的约定为：

（1）变量名尽可能体现该变量的含义。

（2）变量名使用简易英文单词、汉语拼音或者缩写，避免复杂单词，例如 name、file、book、new、old、filename、fn（filename 的缩写）、xuhao（序号）、neirong（内容）和 jieguo（结果）等。

（3）变量名可以使用组合或者缩写体现该变量的含义，组合时其中第一个单词以小写字母开始，从第二个单词后的每个单词的首字母都采用大写字母。例如 stuNo 为 student 和 No 组合，stuName 为 student 和 Name 组合。

（4）一般不直接使用汉字作为变量名。

当我们创建变量并赋值时，Python 解释器会根据右侧表达式的类型和值来创建一个对应的对象，并将该对象的引用给予左侧的变量。

3. 赋值语句

将值或者函数的返回值赋给变量的语句叫作赋值语句，格式如下，左边为变量名，“=”为赋值运算符。

变量名=值或表达式或函数

这个赋值过程的实质是：Python 解释器先运行“=”的右边，将右边最终得到的对象引用给予左边的变量，使变量标识该对象的内存地址，从而可以引用该对象。

例 1：

```
stuNo=2000
print('2000 的内存地址：',id(2000))
print('stuNo 第 1 次的内存地址：',id(stuNo))
stuNo=2000+3000
print('3000 的内存地址：'id(3000))
print('5000 的内存地址：'id(5000))
print('stuNo 第 2 次的内存地址：',id(stuNo))
```

输出结果为

```
2000 的内存地址：2360801818192
stuNo 第 1 次的内存地址：2360801818192
3000 的内存地址：2360801818160
5000 的内存地址：2360801818800
stuNo 第 2 次的内存地址：2360801818800
```

如图 1-4-4 所示，stuNo=2000 将对象 2000 的内存地址给予变量 stuNo，使变量 stuNo 指向 2000，从而 id(stuNo)为对象 2000 的内存地址，与 id(2000)的内存地址相同。

stuNo=2000+3000，首先运行右边 2000+3000 得到 5000，程序为 5000 这个对象重新申请内存空间。从图 1-4-4 中可知，其内存地址为 2360801818800，将该地址给予变量 stuNo，使其指向 5000 这个对象，因此此时 id(stuNo)与 id(5000)相同。

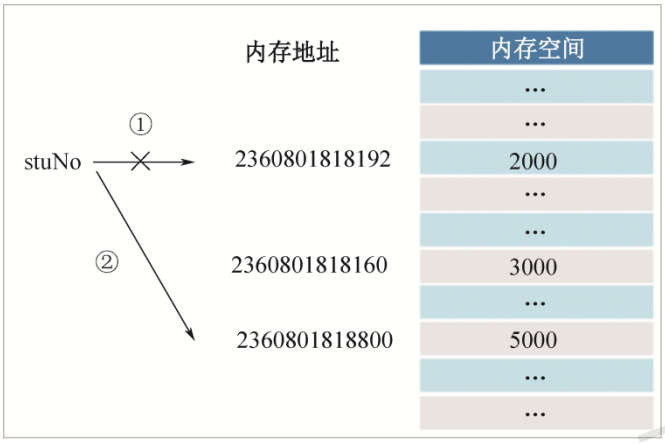


图 1-4-4　赋值运算中变量 stuNo 的指向

例 2：示例代码第二行 stuName=input('请输入您的姓名：')，将 input( )函数的返回值这个对象的内存地址给予变量 stuName，使它指向这个对象。换言之，变量 stuName 指向的内存空间存储着 input( )函数的返回值，即用户输入的内容。

特别需要注意，结合例 1，常见说法“将数值 2000 赋值给变量 stuNo”“变量 stuNo 的值为 2000”“变量 stuNo 用于存储数值 2000”均指变量 stuNo 所标识的内存地址的空间存储数值 2000。

1.4.3　input( )、str( )和字符串连接“+”

（1）input([prompt])函数：接受一个标准输入数据，返回值为字符串型。

其中在[ ]内的参数 prompt 为可选参数，表示提示信息。

如示例代码第二行：

```
stuName=input('请输入您的姓名：')
```

将返回的字符串赋值给变量 stuName，变量 stuName 的类型即为字符串型。

（2）str( )函数：将对象的值转换为字符串型并作为返回值。

如在示例代码中：stuNo=2000，变量 stuNo 的值为 2000，属于数值型；str(stuNo)则将其从数值型转变为字符串型。

```
str(stuNo)
```

将变量 stuNo 的值从数值型转变为字符串型，因此返回'2000'。
注意：变量 stuNo 本身并未改变，仍然为数值型。

（3）字符串连接“+”

在 Python 中，“+”可以用于连接两个字符串。

例 1：运行以下代码段。

```
str1='我爱'
str2='我的祖国'
str3=str1+str2
print(str)
```

输出结果：

```
'我爱我的祖国'
```

例 2：示例代码第 3 行

```
output='我是第'+str(stuNo)+ '号学员，名叫'+stuName+'，我爱我的祖国！'
```

代码中一共连接 5 个字符串，其中红色框内的均为字符串。

第一个蓝色框中 stuNo 为数值型，不能直接和字符串连接，因此使用 str(stuNo)将 stuNo 的值转换为字符串型，再和其他字符串连接。

第一个蓝色框中变量 stuName 为字符串型。

1.4.4 print( )函数的格式化输出

print( )函数有三种常见的输出格式化字符串的方式。

1. 利用 format 格式化输出

使用时{0}是指输出的第 0 个元素，同理，{1}为输出的第 1 个元素，{2}为输出的第 2 个元素，等等，使用时可以不按顺序排列。

例 1：

```
print('我名叫{0},今年{2}岁,我来自{1},我爱我的{3}!'.format('西子','杭州',18,'祖国'))
```

例 2：将示例代码简化为

```
stuNo=2000
stuName=input('请输入您的姓名：')
print('我是第{0}号学员，名叫{1}，我爱我的祖国!'.format(str(stuNo),stuName))
```

2. 利用 f-String 格式化输出

f 表示字符串，{}中为变量，是 format 格式化输出的简化形式。

例：将示例代码简化为

```
stuNo=2000
stuName=input('请输入您的姓名：')
print(f'我是第{stuNo}号学员，名叫{stuName}，我爱我的祖国！')
```

3. 利用%(称为占位符)格式化输出

指定输出类型。

例：将示例代码简化为

```
stuNo=2000
stuName=input('请输入您的姓名：')
print('我是第%d 号学员，名叫%s，我爱我的祖国！'(stuNo,stuName))
```

代码中%d 表示十进制整数，%s 表示字符串，因此数值型变量 stuNo 不需要使用 str( )函数转变值的类型。

print()函数常用的输出格式如表 1-4-2 所示。

表 1-4-2　print()函数常用的输出格式

输 出 格 式	说　　明	输 出 格 式	说　　明
%s	字符串（采用 str( )显示）	%x	十六进制整数
%e	浮点数格式	%f	浮点数
%c	单个字符	%d	十进制整数
%%	字符%	%o	八进制整数

print()函数还有其他参数，例如 sep 和 end，分别用于自定义分隔符和行尾字符。例如：

print("浙江", "杭州", "我爱我的家乡", sep="---", end="！ ")

输出结果：

```
浙江---杭州---我爱我的家乡！
```

print()函数除了输出字符串，还可以输出多种类型的数据，例如整数、浮点数、布尔值、列表、字典、集合、自定义对象等各种数据类型。当使用 print()输出不同类型的数据时，Python 会自动将它们转换为字符串形式，并将其显示在终端或控制台上。例如：

```
stuNo=2000
print(stuNo)
```

输出结果：

```
2000
```

1.4.5　代码调试之断点调试

代码调试是程序设计的重要环节，断点调试是代码调试的常用方法之一。现在运用断点

调试方法调试代码。

```
第 01 行　stuNo=2000
第 02 行　stuName=input('请输入您的姓名：')
第 03 行　output='我是第'+str(stuNo)+ '号学员，名叫'+stuName+'，我爱我的祖国！'
第 04 行　print(output)
```

1. 设置断点

在行号的左边，双击鼠标，即可获得断点标志“圆点”，如图 1-4-5 所示。

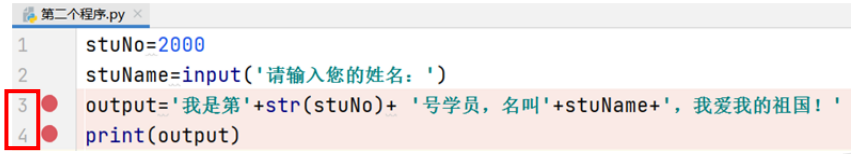


图 1-4-5　设置断点

2. 运行断点

在程序设计界面，单击“Debug”图标或在菜单中选择“Debug 第二个程序”，便可以进行断点调试，如图 1-4-6 所示，并可看见相关变量的类型、赋值情况，如图 1-4-7 所示。

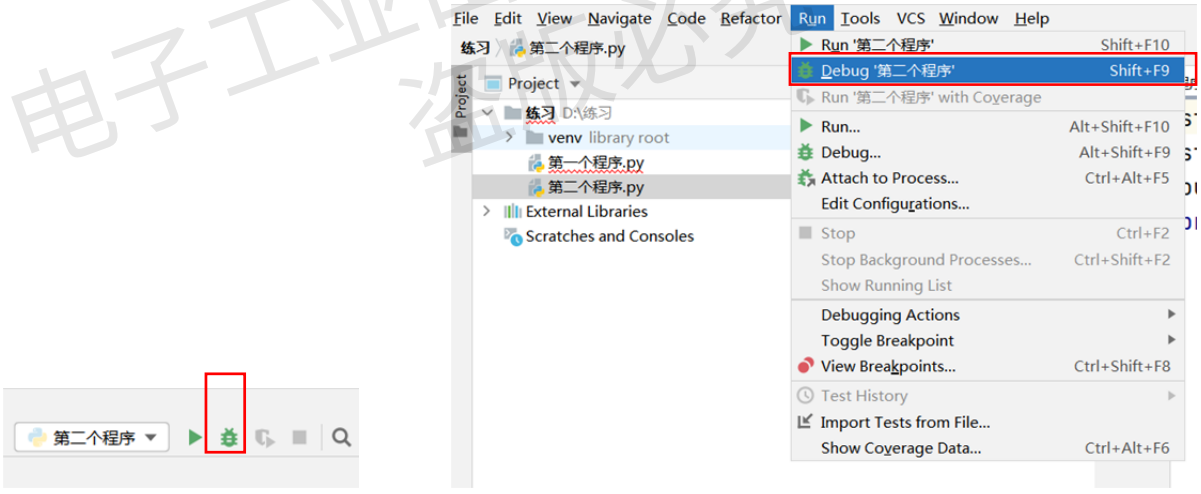


图 1-4-6　运行断点菜单

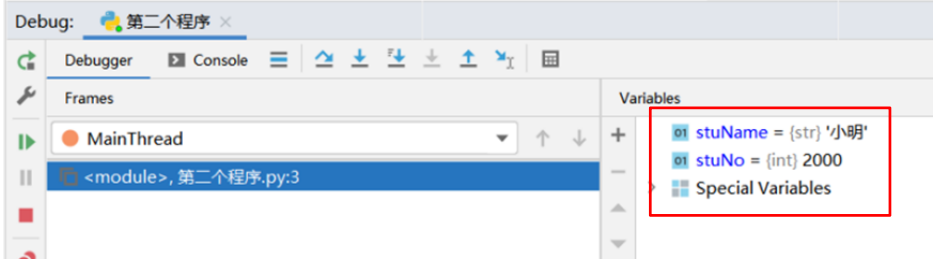


图 1-4-7　查看变量的赋值及类型

3. 跳至断点下一句

单击“ ”图标，可以进入跳至下一句，可以看见下一句变量的值及类型，如图 1-4-8 所示。

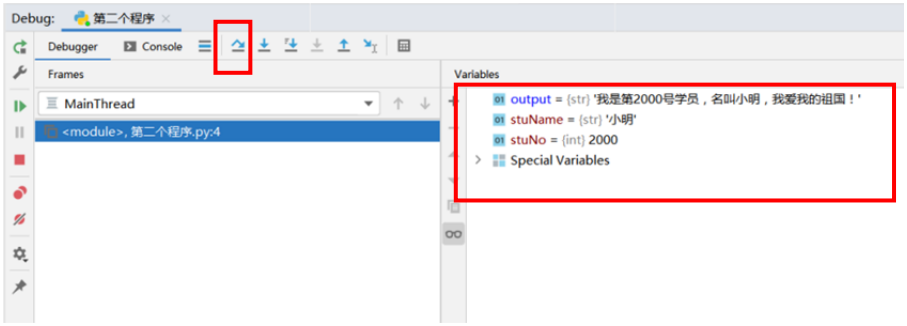


图 1-4-8　下一句变量的赋值及类型

1.4.6　代码调试之交互式调试

代码调试是程序设计的重要环节，断点调试是代码调试的常用方法之一，现在运用断点调试交互式工具进行调试。

1. 设置断点

见 1.4.5 节的设置断点。

2. 运行交互式工具

单击“Debug”图标进行断点调试，在“Console”菜单中选择“Show Debug Console”，输入“output”便可看见变量 output 的值，如图 1-4-9 所示。

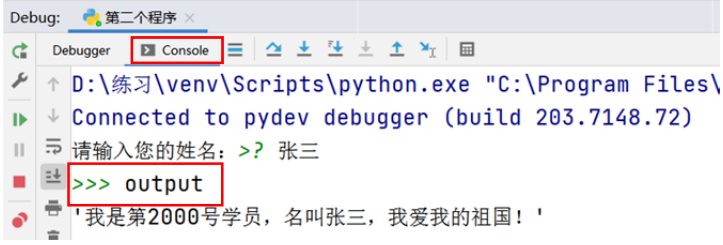


图 1-4-9　交互调试工具的操作

知识小结

1．input( )函数、str( )函数和字符串连接“+”。

2．对象、变量的定义与赋值语句。

3．print( )函数的格式化输出。

4．代码调试的两种方式。

技能拓展

1．阅读并分析以下代码：

```
我爱我的祖国='我爱我的祖国'
print('我爱我的祖国')
print(我爱我的祖国)
```

运行结果：

```
我爱我的祖国
我爱我的祖国
```

2．int( )函数可以将由整数数字组成的字符串型转变为数值型，例如 int('2000')的值为2000。阅读以下代码，并按照输出结果示例补充输出语句。

输出结果示例：假设两次输入分别为 100 和 200，输出结果：因为 a=100，b=200，所以c=a+b=100+200=300。

代码段：

```
a=int(input('请输入第一个加数：'))
b=int(input('请输入第二个加数：'))
c=a+b
print(c)
```

3．了解变量命名的常见方法

（1）匈牙利命名法

匈牙利命名法是以数据类型首字母加上标识符单词，数据类型与单词之间用下画线"_"分割，形式为数据类型_单词组合。

例如：str_bookname，从变量名就可以知道它是字符串类型，表示书名。

（2）驼峰命名法

驼峰命名法分为大驼峰法和小驼峰法，区别在于是否把第一个字母大写。

如果把变量的每个单词首字母都大写，就是大驼峰命名法，也称为帕斯卡命名法。如果除变量的第一个单词首字母小写以外，其余单词首字母都大写，就称为小驼峰命名法。

例如：

大驼峰命名法：BookName

小驼峰命名法：bookName

（3）下画线法

下画线法就是在每个单词之间使用下画线"_"进行分割，使代码阅读性更强。

例如：book_Name