第 1 章

初识 Python

学习目标

- 了解 Python 的发展历程。
- 了解 Python 语言的特点及应用领域。
- 掌握在不同操作系统下 Python 编程环境的搭建。
- 掌握 PyCharm 及 Sublime Text 的安装。

1.1 认识 Python

1.1.1 Python 的发展历程

随着大数据及人工智能的飞速发展，Python 已成为当今最流行的开源编程语言之一，越来越受到世界各国的重视。

Python 的创始人为荷兰的 Guido van Rossum(吉多·范·罗苏姆，图 1-1)。Python 一词来自 Guido 所挚爱的电视剧 *Monty Python's Flying Circus*，因此 Python 便作为该编程语言的名称。Guido 在 1989 年的圣诞节假期，开始编写 Python 语言的解释器。

图 1-1 吉多·范·罗苏姆

Guido 曾经参与 ABC 语言的设计与实现。ABC 语言是由荷兰的 CWI(Centrum Wiskunde & Informatica，数学和计算机科学中心)开发的以教学为目的的语言，具备良好的可读性和易用性，但也存在一些问题，如非开放性、可拓展性差、不能直接进行 I/O、过度革新及传播困难等。因此，Guido 决心在 Python 语言的实现中，继承 ABC 语言的优点，克服 ABC 语言的问题，并结合 C 和 shell 语言的特点，使 Python 成为功能全面、语法简洁优美、易学易用、可拓展的编程语言。

1991 年，Python 的第一个编译器诞生，它是面向对象的解释型语言，用 C 语言实现，并能够调用 C 语言的库文件。Python 具有面向对象的类、函数、文件及异常处理等机制，包括以字符串、列表、字典及元组为核心的数据类型，是以模块为基础的拓展系统。

Python 最初完全由 Guido 本人开发，随着 Python 功能的完善，Guido 的同事也加入了 Python 的改进中来，逐渐形成 Python 的核心团队。随后，Python 拓展到 CWI 之外。Python 将许多机器层面上的细节隐藏起来，交给编译器处理，这样，Python 程序员就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。正是这一特点，吸引了更多的程序员使用 Python，使得 Python 逐渐流行起来，2011 年 1 月，Python 赢得了 TIOBE 编程语言排行榜 2010 年度语言的桂冠。

1.1.2 Python 语言的特点

Python 语言主要具有以下几个特点。

1. 简单易学

Python 的语法极其简单，虽然 Python 是用 C 语言写的，但它抛弃了 C 语言中的指针，从而使 Python 的语法得到简化。Python 代码具有伪代码的本质，这就使程序编写者能够专注于解决问题，而不是去搞明白语言本身。

2. 解释性

Python 可以直接从源代码运行。在计算机内部，Python 解释器把源代码转换为字节码的中间形式，然后把它翻译成计算机使用的机器语言。不需要担心程序的编译问题，这就使得使用 Python 变得更加简单，更加易于程序的移植。

3. 可移植性

由于 Python 开源的本质，Python 已经被移植到很多平台上，包括 Linux、Windows、Mac OS X 及 Google 基于 Linux 开发的 Android 平台。

4. 可扩展性

如果需要一段关键代码运行得更快或希望某些算法不公开，可以把部分程序用其他语言编写出来，如 C 或 C++，然后在 Python 程序中使用它们。

5. 面向对象

Python 既支持面向过程的编程，也支持面向对象的编程。在面向过程的编程语言中，程序是由过程或可重用的代码构建起来的。在面向对象的编程语言中，程序可以通过组合和继承定义类构建起来，但与 C++ 或 Java 等面向对象的语言相比，其实现面向对象编程的方式更为简单。

6. 丰富的库

Python 标准库非常庞大，包括正则表达式、文档生成、单元测试、线程、数据库、网页浏览器、CGI、FTP、电子邮件、XML、XML-RPC、HTML、WAV 文件、密码系统、GUI、Tk 和其他与系统有关的操作。除了标准库外，还有许多其他高质量的库，如 wxPython、Twisted、Python 图像

库等。

1.1.3 Python 的主要应用领域

Python 的主要应用领域如下。

1. Web 应用开发

Python 现已成为 Web 开发的主流语言之一，其有丰富的 Web 开发框架，如 Django、Flask、Weppy、Zope2、Tornado、CubicWeb 及 Web2py 等。其中，Python＋Django 框架最为流行，其应用范围广、开发速度快、学习门槛低，可以让程序员轻松地开发和管理复杂的 Web 程序。

2. 自动化运维

Python 在自动化运维方面已经成为运维工程师的首选编程语言。业内使用最为广泛的自动化运维平台 Ansible 和 SaltStack 都是基于 Python 的，所以其具有良好的二次开发扩展能力。

3. 网络安全

自 2013 年“棱镜门”事件后，网络安全问题越发引起世界各国的重视。Python 作为高级编程语言的一种，编程简单、易学及易用，且有强大的第三方编程模块的支持，越来越受网络安全维护人员的青睐。一些常用的网络安全工具都是基于 Python 语言来编写的，如 Scapy、Pcapy 及 Sulley 等。

4. 网络爬虫

在编写网络爬虫程序的众多语言中，Python 起到了举足轻重的作用，其中 Scrapy 爬虫框架应用得非常广泛，可以应用在数据挖掘、信息处理或存储历史数据中。此外，Python 还提供了多个网络爬虫框架以满足不同的应用需求，如 PySpider、Crawley、Newspaper、Beautiful Soup、Grab 及 Cola 等。

5. 游戏开发

Python 在游戏开发方面也有很多应用，如专门为游戏开发而设计的模块 pygame，能够使开发者快速开发出自己的游戏。相比于 Lua 或 C＋＋，Python 比 Lua 具有更高阶的抽象能力，可以用更少的代码描述游戏的业务逻辑。例如，用 C＋＋开发游戏，有时必须写一些扩展，从而增加游戏的代码量。

6. 数据分析

Python 拥有丰富的第三方库，如 Pandas、NumPy、Matplotlib 及 SciPy 等，可以让程序编写人员完成各种数据分析需求。

7. 人工智能

Python 在人工智能、神经网络、深度学习方面，有强大而丰富的库。Python 是面向对象的动态语言，且适用于科学计算，这就使得 Python 在人工智能方面备受青睐。目前，世界优秀的人工智能学习框架，如 Google 的 TensorFlow、Facebook 的 PyTorch 及开源社区的神经网络库 Keras 等均是用 Python 来实现的。

1.2 搭建 Python 编程环境

Python 是一种跨平台的编程软件，这就意味着它能够运行在所有的主流操作系统中。但在不同的操作系统中，Python 的安装方法略有不同，本节将介绍在 Linux、Mac OS X 和 Windows 系统中安装 Python 的方法。

可以通过访问 Python 官网 https://www.python.org/，获取 Python 语言的最新版本、技术文档及相关新闻资讯等。

1.2.1 在 Linux 系统中搭建 Python 环境

Linux 系统是为编程而设计的，因此，绝大多数 Linux 系统在安装时默认安装了 Python。

1. 检查 Python 版本

在系统中打开一个终端窗口，执行命令 python（注意 p 是小写的）。若系统中已安装 python，将会输出如下内容：

```
#python
Python 2.6.6 (r266:84292, Nov 22 2013, 12:11:10)
[GCC 4.4.7 20120313 (Red Hat 4.4.7-4)] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

它指出了 Python 的版本号——Python 2.6.6；最后一行的“>>>”为 Python 命令提示符，提示用户输入 Python 命令。若要退出 Python 并返回终端窗口，可以按 Ctrl+D 快捷键或执行 Python 命令 exit()。

2. 安装 Python 3

在 Linux 系统中检查是否安装 Python 3，只需在终端窗口中执行命令 python3。下述输出表明该系统未安装 Python 3。

```
# python3
bash: python3: command not found
```

现在以 Python-3.6.3 版本为例，讲解如何在 Linux 系统中安装 Python 3，具体步骤如下：

```
❶# mkdir /usr/local/python3
❷# cd /usr/local/python3
❸# wget https://www.python.org/ftp/python/3.6.3/Python-3.6.3.tgz
❹# tar -xvf Python-3.6.3.tgz
❺# cd Python-3.6.3/
❻# ./configure
```

```
❼# make
❽# make install
```

在 Linux 系统中的路径/usr/local 下创建目录 python3(见❶),进入该目录下(见❷),从 Python 官网下载需要的 Python 版本(见❸),在 python3 目录下解压 Python-3.6.3.tgz 包(见❹),进入解压出的文件目录 Python-3.6.3 下(见❺),执行./configure 命令进行软件配置(见❻),执行 make 命令进行编译(见❼),最后执行 make install 命令完成 Python-3.6.3 的安装(见❽)。

检查 Python 3 是否安装成功,只需在终端窗口中再次执行命令 python3。若安装成功,则会输出如下内容:

```
# python3
Python 3.6.3 (default, Apr  7 2019, 16:21:54)
[GCC 4.4.7 20120313 (Red Hat 4.4.7-4)] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

3. 在终端运行 Python 代码

在终端窗口中输入并执行 python 或 python3,在 Python 命令提示符>>>后,输入如下代码并按 Enter 键:

```
>>> print('Hello Python world!')
Hello Python world!
>>>
```

消息直接显示在当前终端窗口中,若要关闭 Python 解释器,可按 Ctrl+D 快捷键或执行命令 exit()。

1.2.2 在 Mac OS X 系统中搭建 Python 环境

与 Linux 系统一样,大多数 OS X 系统也都默认安装了 Python,可以检查一下自己的 OS X 系统是否安装了 Python。

1. 检查 Python 版本

在系统终端中输入命令 python 并按 Enter 键,若系统已安装 Python,将会出现如下输出:

```
% python
Python 2.7.15 (default, Jan 21 2019, 18:19:19)
[GCC 4.2.1 Compatible Apple LLVM 10.0.0 (clang-1000.11.45.5)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

它指出了 Python 的版本号——Python 2.7.15;最后一行的>>>为 Python 命令提示符,提示用户输入 Python 命令。同样地,若要退出 Python 并返回终端窗口,可以按 Ctrl+D 快捷键

或执行 Python 命令 exit()。

2. 安装 Python 3

(1)访问 Python 官网 https://www.python.org/downloads/,即可进入 Python 的下载页面,如图 1-2 所示。单击矩形框中的链接地址,即可进入 Mac OS X 系统的 Python 下载页面。Mac OS X 系统的 Python 下载页面提供了多个 Python 版本,用户可以根据自身的需求,选择适合自己需要的版本,本书以 Python 3.6.3 版本为例,演示 Python 的安装过程。

图 1-2 Python 下载页面 1

(2)双击下载好的.pkg 文件,进入 Python 3.6.3 的安装界面,单击“继续”按钮,如图 1-3 所示。

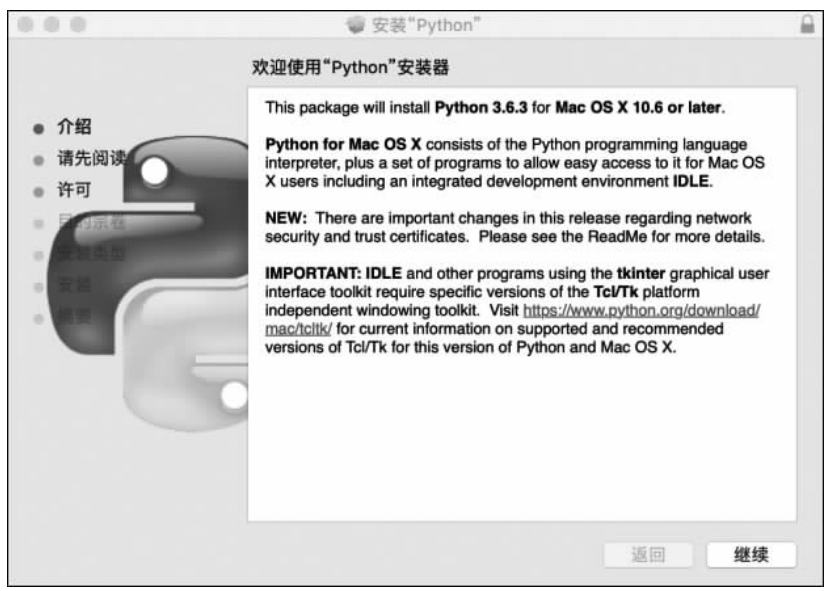

图 1-3 Python 安装界面

(3)进入“请先阅读”界面,单击“继续”按钮,如图 1-4 所示。

图 1-4 “请先阅读”界面

(4)进入“许可”界面,单击“继续”按钮,如图 1-5 所示。

图 1-5 “许可”界面

(5)弹出“若要继续安装软件,您必须同意软件许可协议中的条款”提示框,单击“同意”按钮,如图 1-6 所示。

图 1-6　询问是否同意软件许可协议

(6)进入"安装类型"界面，单击"安装"按钮，如图 1-7 所示。

图 1-7　"安装类型"界面

(7)进入安装界面，等待安装完成后，单击"继续"按钮，如图 1-8 所示。

图 1-8 “安装”界面

(8)安装成功的“提示”界面如图 1-9 所示。

图 1-9 “提示”界面

检查 Python 3 是否安装成功，只需在终端窗口中再次执行命令 python3，若安装成功，则会输出如下内容：

```
% python3
Python 3.6.3 (v3.6.3:2c5fed86e0, Apr 8 2019, 20:32:10)
[GCC 4.2.1 (Apple Inc. build 5666) (dot 3)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

3. 在终端运行 Python 代码

在终端窗口中输入并执行 python 或 python3，在 Python 命令提示符>>>后输入如下代码行：

```
>>> print('Hello Python world!')
Hello Python world!
>>>
```

消息直接显示在当前终端窗口中，若要关闭 Python 解释器，可按 Ctrl＋D 快捷键或执行命令 exit()。

1.2.3 在 Windows 系统中搭建 Python 环境

Windows 系统没有默认安装 Python。在 Windows 系统中安装 Python 的具体安装步骤如下：

(1)访问 Python 官网 https://www.python.org/downloads/，进入 Python 的下载页面，如图 1-10 所示。单击矩形框中的链接地址，即可进入 Windows 系统的 Python 下载页面。Python 同样为 Windows 提供了多个版本，用户可以根据自身的需求，选择适合自己需要的版本。本书以 Python 3.6.3 版本为例，演示 Python 的安装过程。

图 1-10 Python 下载页面 2

(2)双击下载好的.exe 文件，进入 Python 3.6.3 的安装界面，如图 1-11 所示。建议在安装的过程中选中“Add Python 3.6 to PATH”复选框，如图 1-12 所示。这样，Python 路径就会自动添加到系统的环境变量中，无须再手动为 Python 配置环境变量。

图 1-11　Python 安装方式选择

图 1-12　选择添加 Python 路径

(3)选择第 1 种安装方式 Install Now,安装界面如图 1-13 所示。若选择第 2 种安装方式 Customize installation,则可以选择安装路径及可选项等内容。

(4)安装成功界面如图 1-14 所示。

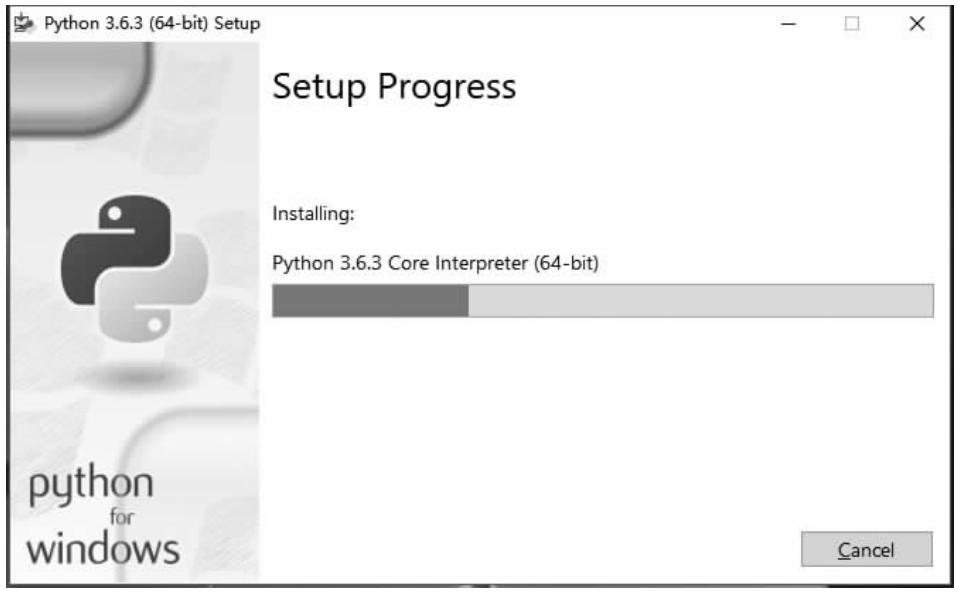

图 1-13　安装过程界面

图 1-14　安装成功界面

检查 Python 3 是否安装成功，只需在 DOS 命令行窗口中执行命令 python，若安装成功，则会输出如下内容：

```
    >python
    Python 3.6.3 (v3.6.3:2c5fed8, Oct  3 2017, 18:11:49) [MSC v.1900 64 bit (AMD64)]
on win32
    Type "help", "copyright", "credits" or "license" for more information.
    >>>
```

在 DOS 命令行窗口中输入并执行 python 命令，在 Python 命令提示符>>>后输入如下代码行：

```
>>> print('Hello Python world!')
Hello Python world!
>>>
```

消息直接显示在当前终端窗口中，若要关闭 Python 解释器，则可以执行命令 exit()。

1.3 Python 集成开发环境

目前，有多种 Python 集成开发环境，本书仅以 PyCharm 及 Sublime Text 为例，演示两种集成开发环境的安装方法。

1.3.1 PyCharm

PyCharm 是由 JetBrains 打造的一款 Python IDE，带有一整套可以帮助用户在使用 Python 语言开发时提高工作效率的工具，如调试、语法显示、Project 管理、代码跳转、智能提示、自动完成、单元测试及版本控制等功能。

1. 安装 PyCharm

(1)访问 PyCharm 官网 http://www.jetbrains.com/pycharm/download/#section=windows，即可进入 PyCharm 的下载页面，如图 1-15 所示。下面演示在 Windows 系统中安装 PyCharm 的过程。

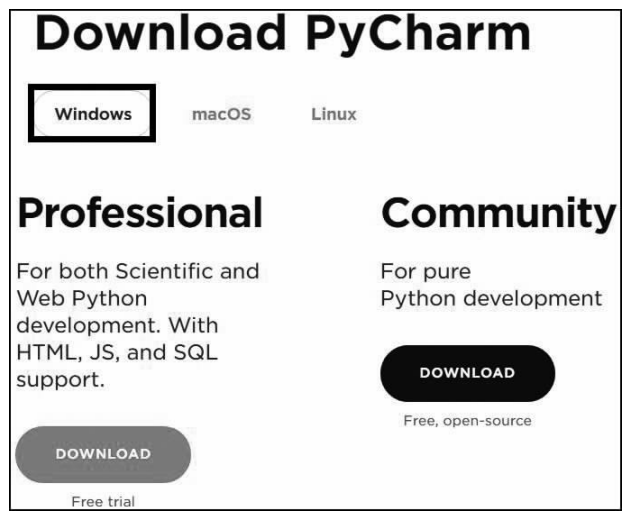

图 1-15 PyCharm 下载页面

提示 PyCharm 分为专业版(Professional)和社区版(Community)两个版本。专业版与社区版相比，功能更加丰富，专业版比社区版增加了 Web 开发、Web 框架、Python 分析器、远程开发、数据库和 SQL 等功能。

(2)双击下载好的.exe 文件,进入 PyCharm 的安装界面,单击“Next”按钮,如图 1-16 所示。

图 1-16　PyCharm 安装界面

(3)进入 PyCharm 的路径选择界面,可以自定义 PyCharm 的安装路径,然后单击“Next”按钮,如图 1-17 所示。

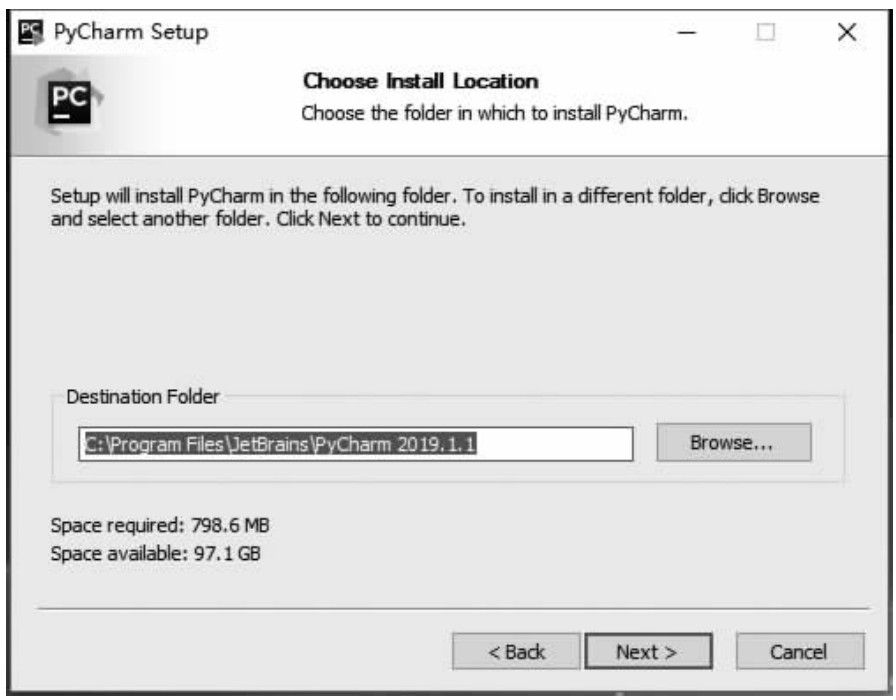

图 1-17　路径选择界面

(4)进入 PyCharm 相关配置界面,选择相关配置后,单击“Next”按钮,如图 1-18 所示。

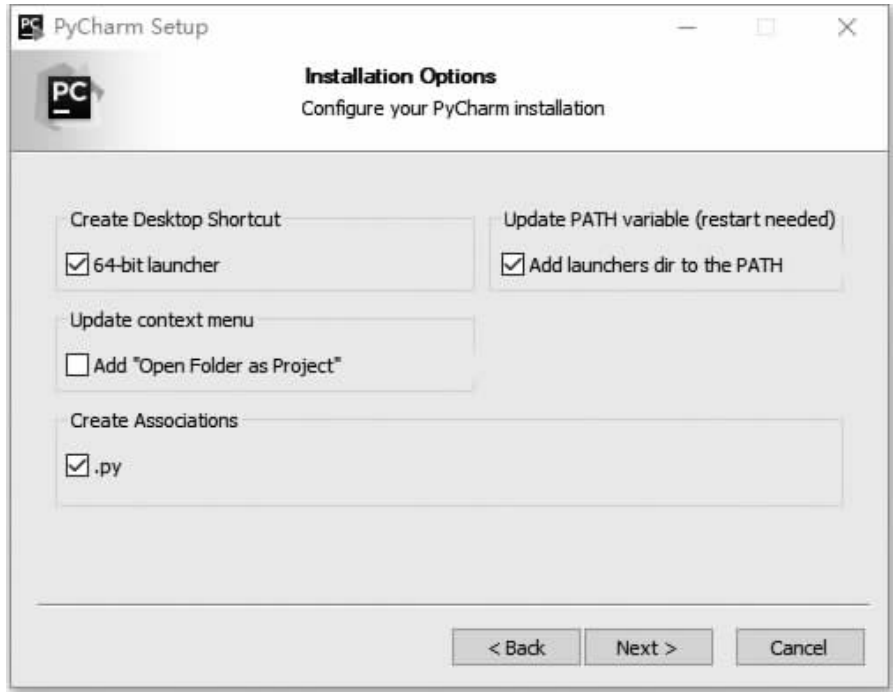

图 1-18 PyCharm 配置界面

(5)进入 PyCharm 安装进度界面,如图 1-19 所示。

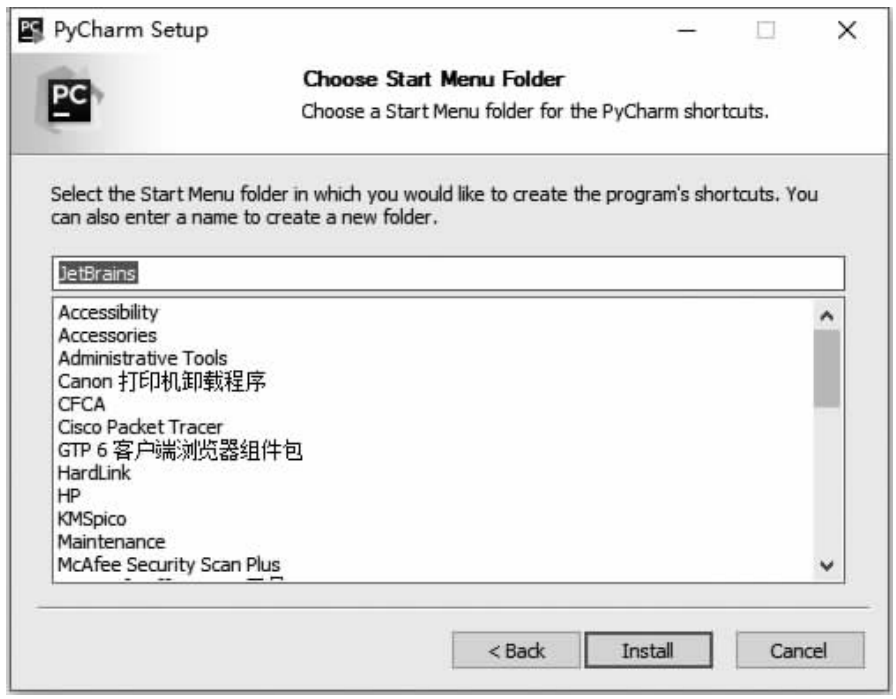

图 1-19 安装进度界面 1

(6)安装完成后单击“Finish”按钮,如图 1-20 所示。

图 1-20　安装完成

2. 使用 PyCharm

(1)完成 PyCharm 安装后,便可以使用 PyCharm 了。但在首次使用 PyCharm 时,会有一系列提示信息,完成 PyCharm 的配置后,首次打开 PyCharm,会弹出如图 1-21 所示的对话框,选中“Do not import settings”单选按钮,单击“OK”按钮。

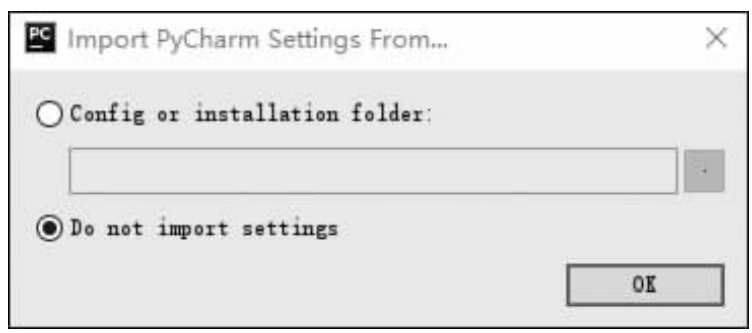

图 1-21　配置选项

(2)弹出用户协议界面,选中“I confirm that I have read and accept the terms of this User Agreement”复选框,单击“Continue”按钮,如图 1-22 所示。

图 1-22　用户协议界面

(3)进入 PyCharm 数据共享界面,用户可以根据自己的需求,单击“Send Usage Statistics”或“Don't send”按钮,如图 1-23 所示。

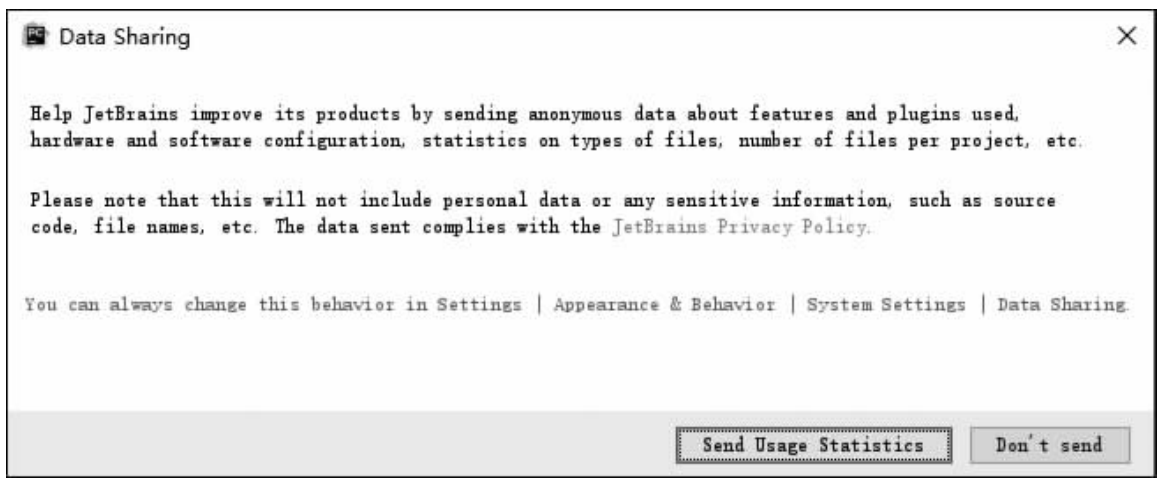

图 1-23　数据共享界面

(4)进入主题设置界面,单击“Next:Featured plugins”按钮,如图 1-24 所示。

(5)打开如图 1-25 所示的“Customize PyCharm”对话框,单击“Start using PyCharm”按钮。

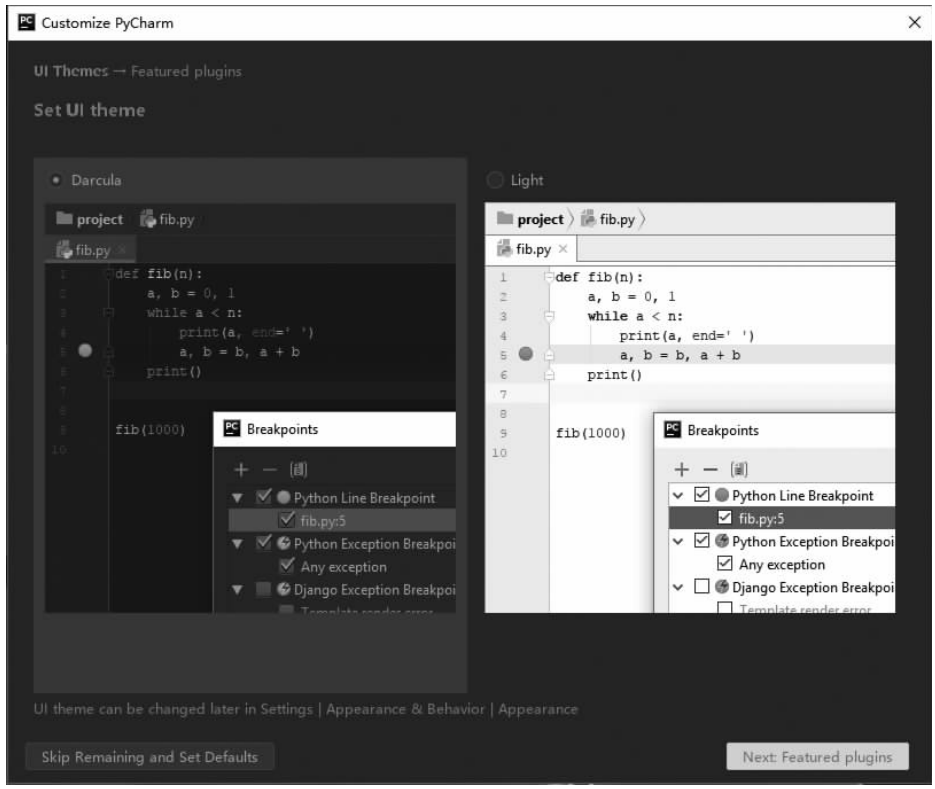

图 1-24　主题设置界面

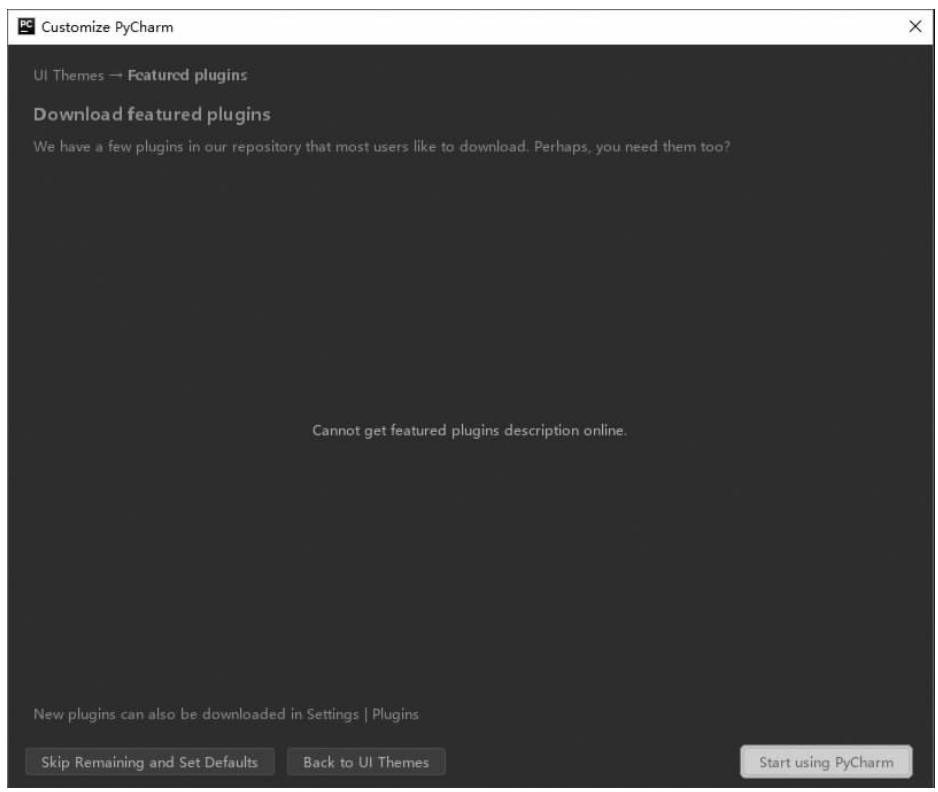

图 1-25　“Customize PyCharm”对话框

(6)进入用户激活界面,专业版 PyCharm 是收费的,会提示用户输入许可信息激活软件,这里选中“Evaluate for free”单选按钮,有 30 天的试用期。单击“Evaluate”按钮,如图 1-26 所示。

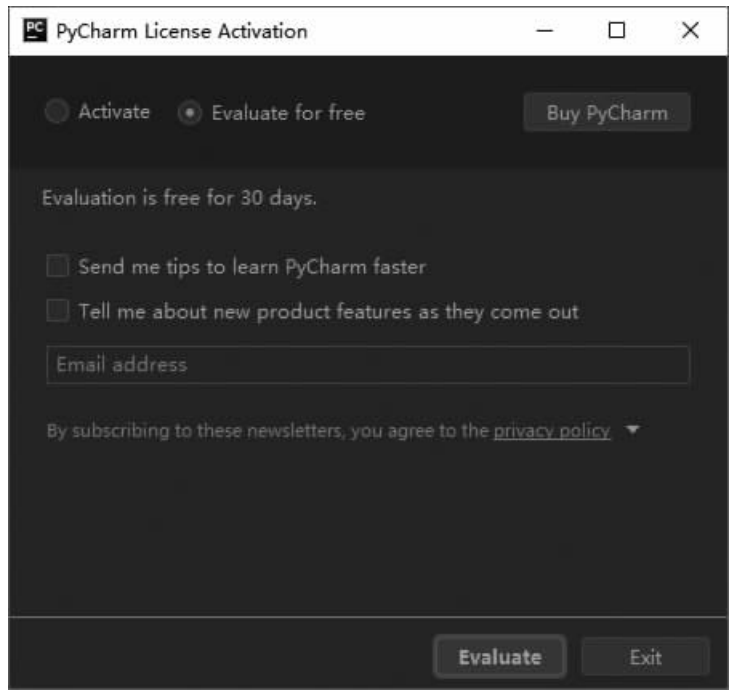

图 1-26 用户激活界面

(7)进入 PyCharm 启动界面,如图 1-27 所示。

图 1-27 PyCharm 启动界面

(8)PyCharm 启动完成后,进入 PyCharm 项目创建界面,其中有 3 个选项:“Create New

Project”用来创建新的项目，“Open”用来打开已有的项目，“Check out from Version Control”用来从版本控制中检查新的项目。单击“Create New Project”按钮，如图 1-28 所示。

图 1-28　创建项目界面

(9)进入项目路径设置界面，设置好项目存放路径后单击“Create”按钮，项目创建完成，如图 1-29 所示。

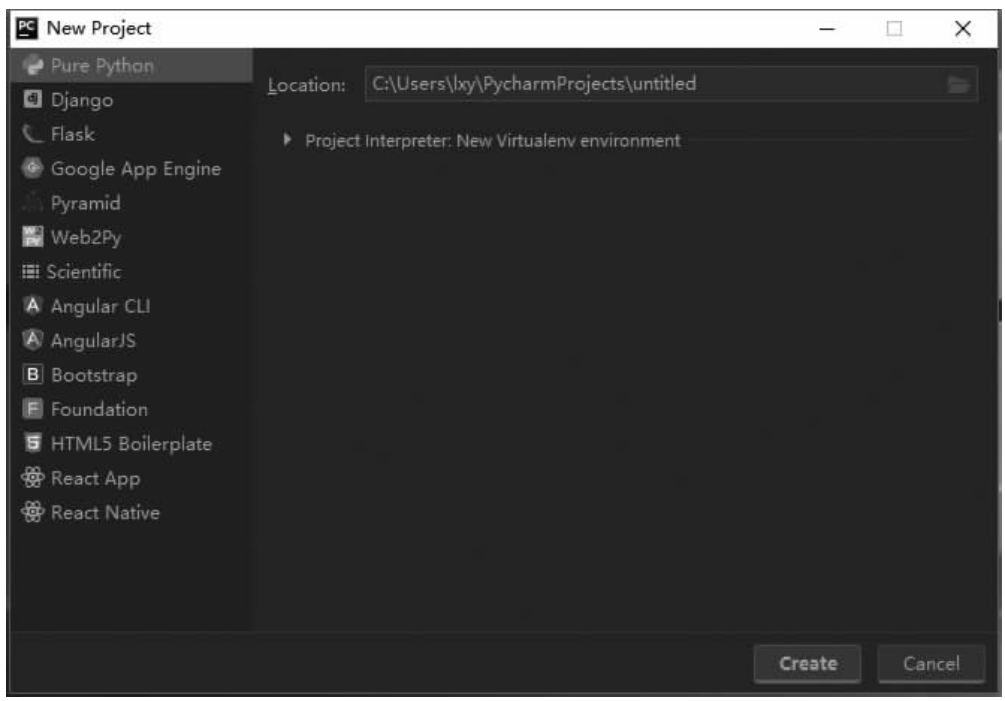

图 1-29　项目路径设置界面

(10)进入项目开发界面,如图 1-30 所示。

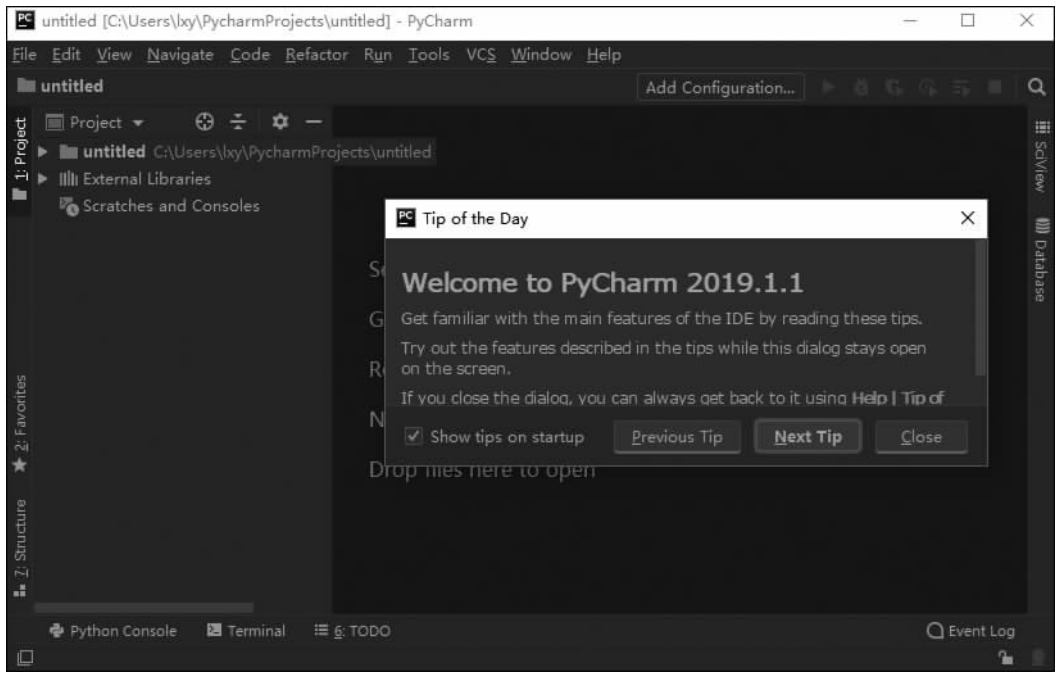

图 1-30 项目开发界面

(11)右击项目名称,在弹出的快捷菜单中选择 New→Python File 选项,新建 Python 文件,如图 1-31 所示。

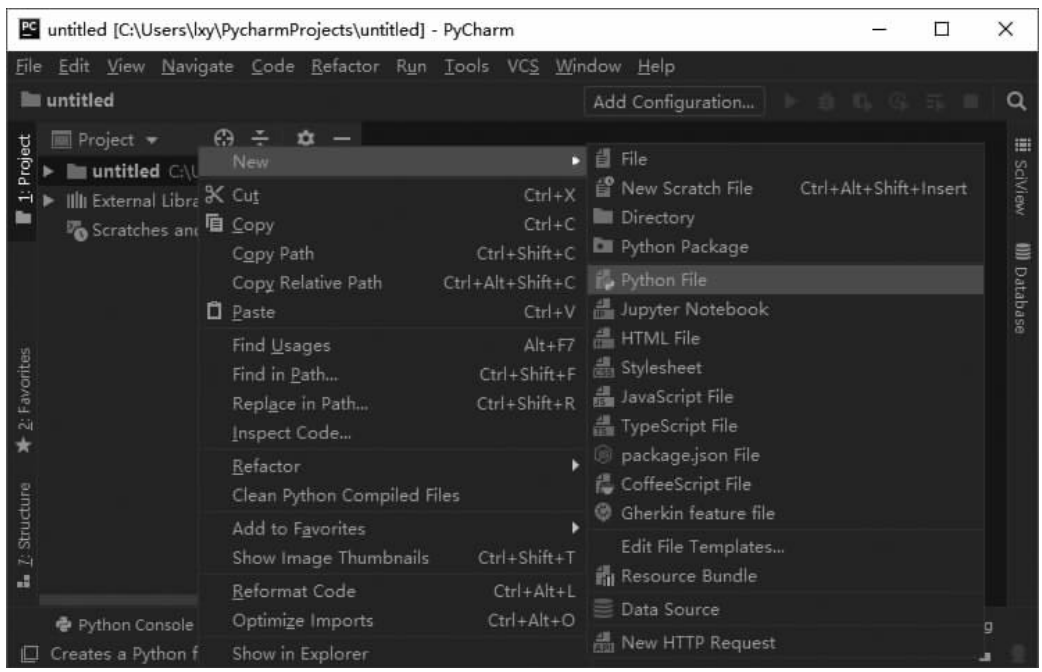

图 1-31 新建 Python 文件

(12)弹出“New Python file”对话框,在“Name”文本框中输入文件名 “HelloPythonWorld”,

单击“OK”按钮，如图 1-32 所示。

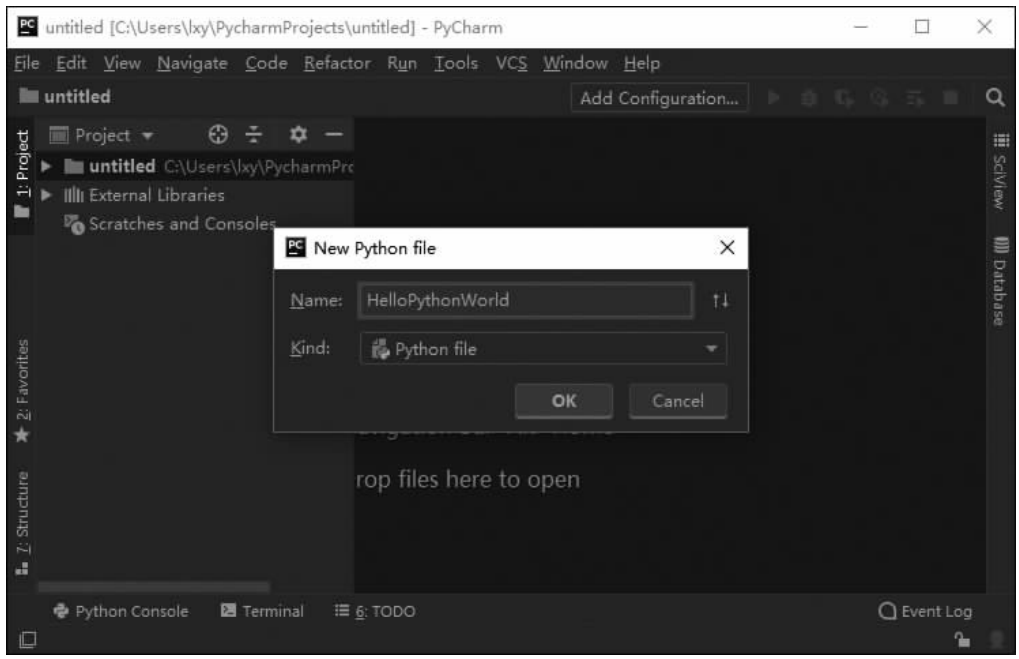

图 1-32　文件命名

(13)进入程序编辑窗口，在程序编辑区域输入“print("Hello Python World!")”，如图 1-33 所示。

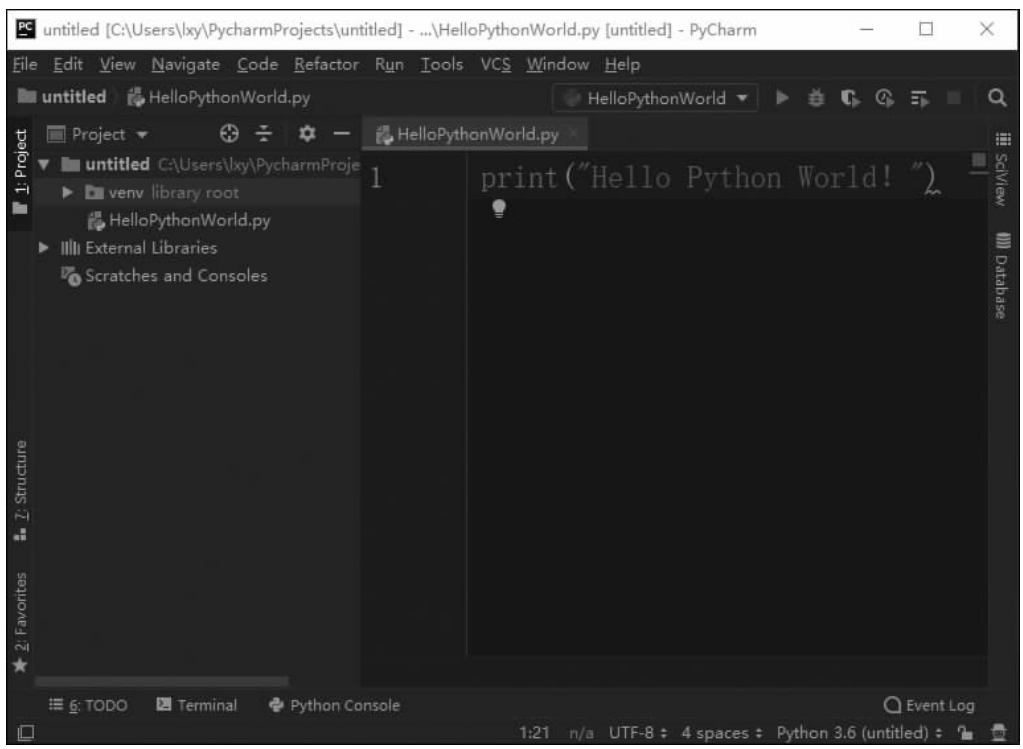

图 1-33　程序编辑窗口

(14)右击,在弹出的快捷菜单中选择“Run 'HelloPythonWorld'”命令运行程序,如图 1-34 所示。

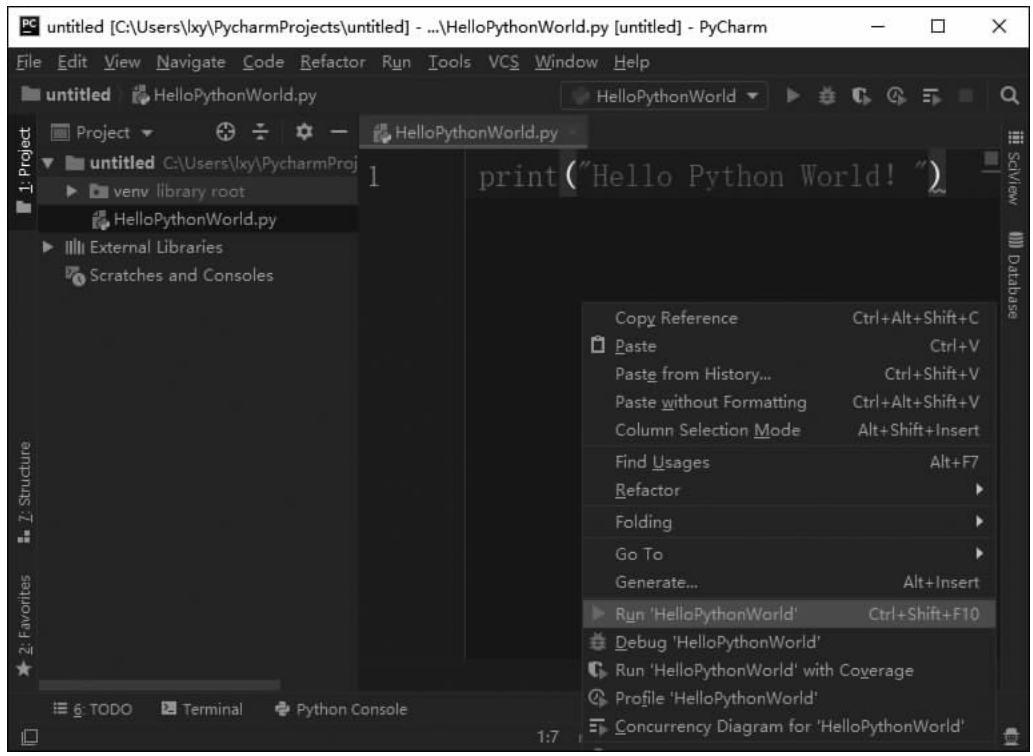

图 1-34 运行程序

(15)程序运行结果如图 1-35 所示。

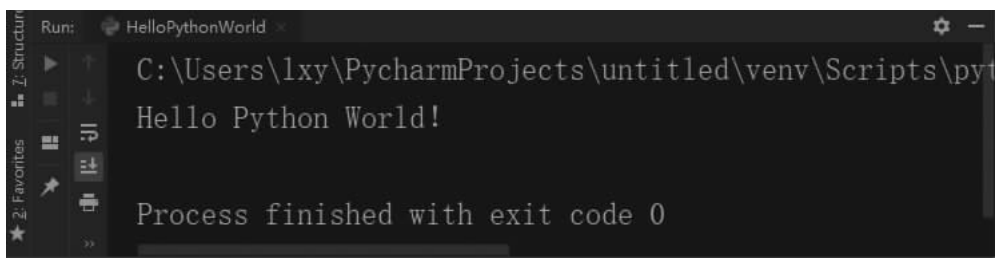

图 1-35 程序运行结果 1

1.3.2 Sublime Text

Sublime Text 是一个跨平台的编辑器,同时支持 Windows、Linux、Mac OS X 等操作系统。其主要功能包括拼写检查、书签、完整的 Python API、Goto 功能、即时项目切换、多选择及多窗口等。

1. 安装 Sublime Text

(1)访问 Sublime Text 官网 http://www.sublimetext.com/3,进入 Sublime Text 下载页面,如图 1-36 所示。下面以 Windows 系统为例,演示 Sublime Text 的安装。

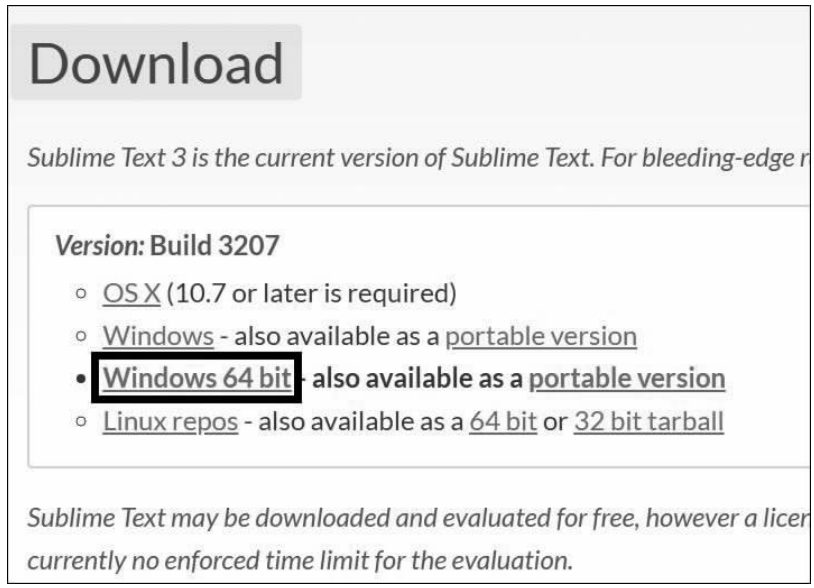

图 1-36　Sublime Text 下载页面

(2)双击下载好的.exe 文件,弹出 Sublime Text 安装界面,选择 Sublime Text 的安装路径,单击“Next”按钮,如图 1-37 所示。

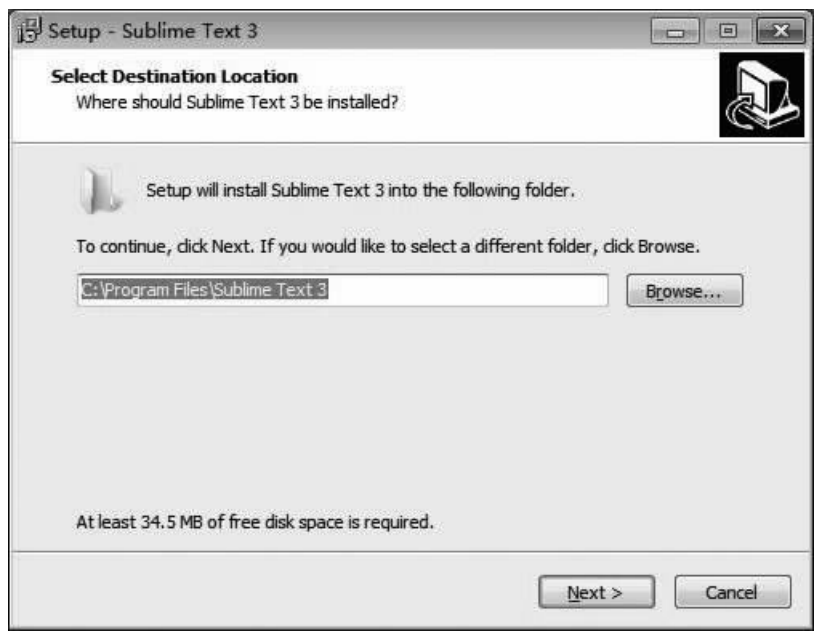

图 1-37　选择安装路径

(3)进入添加选项界面,单击“Next”按钮,如图 1-38 所示。

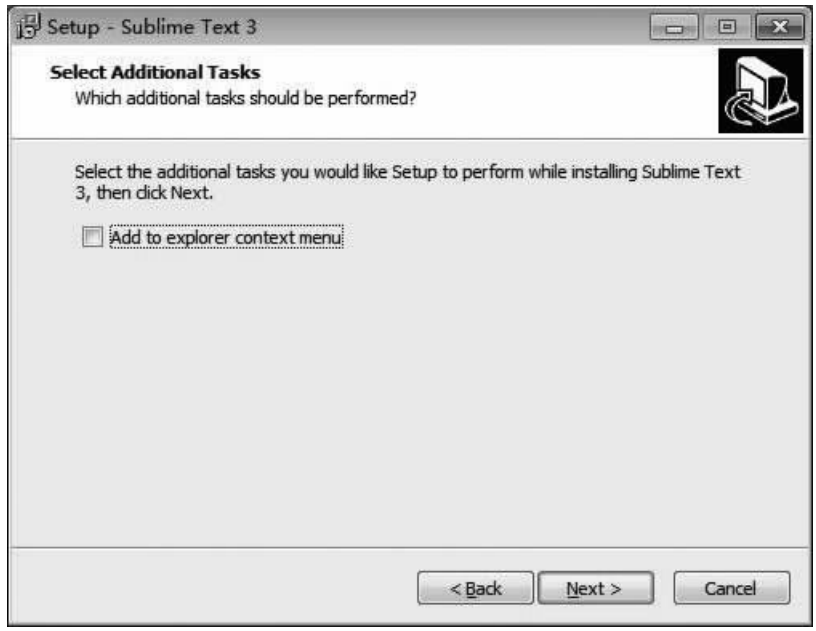

图 1-38　添加选项界面

(4)进入安装准备界面,单击“Install”按钮,如图 1-39 所示。

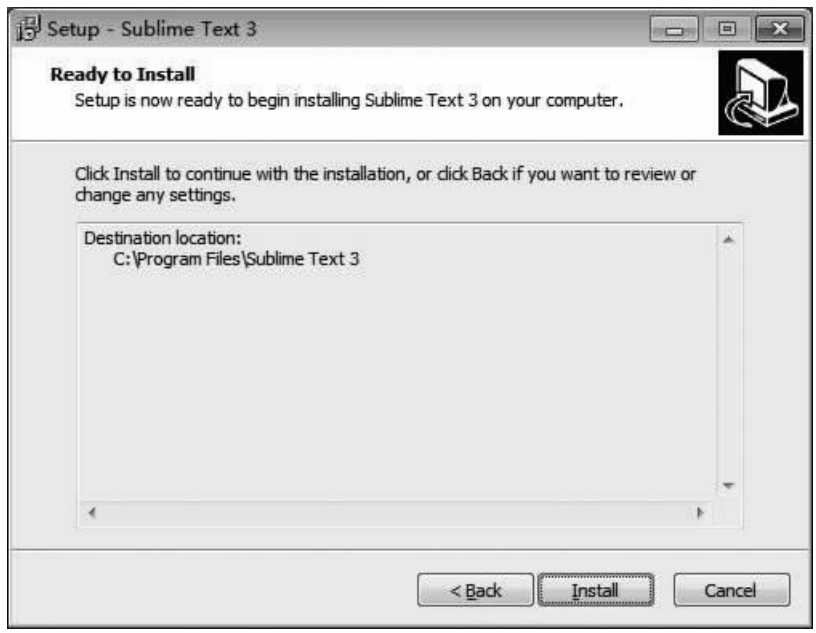

图 1-39　安装准备界面

(5)进入安装进度界面,如图 1-40 所示。

(6)安装完成后单击“Finish”按钮,如图 1-41 所示。

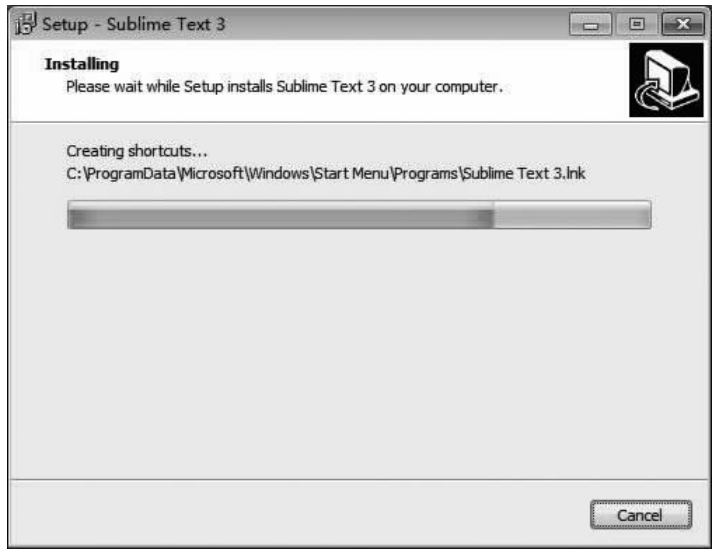

图 1-40　安装进度界面 2

图 1-41　安装完成界面

2. 使用 Sublime Text

(1)在安装好的路径中双击如图 1-42 所示的 Sublime Text 快捷方式图标。

图 1-42　Sublime Text 快捷方式图标

(2)进入 Sublime Text 程序编辑界面,如图 1-43 所示。

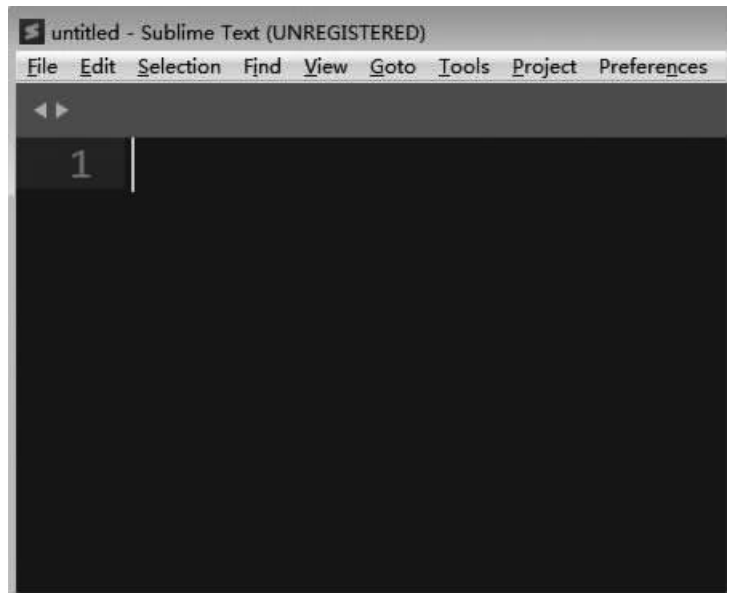

图 1-43 程序编辑界面

(3)执行 File→New File 命令,如图 1-44 所示。

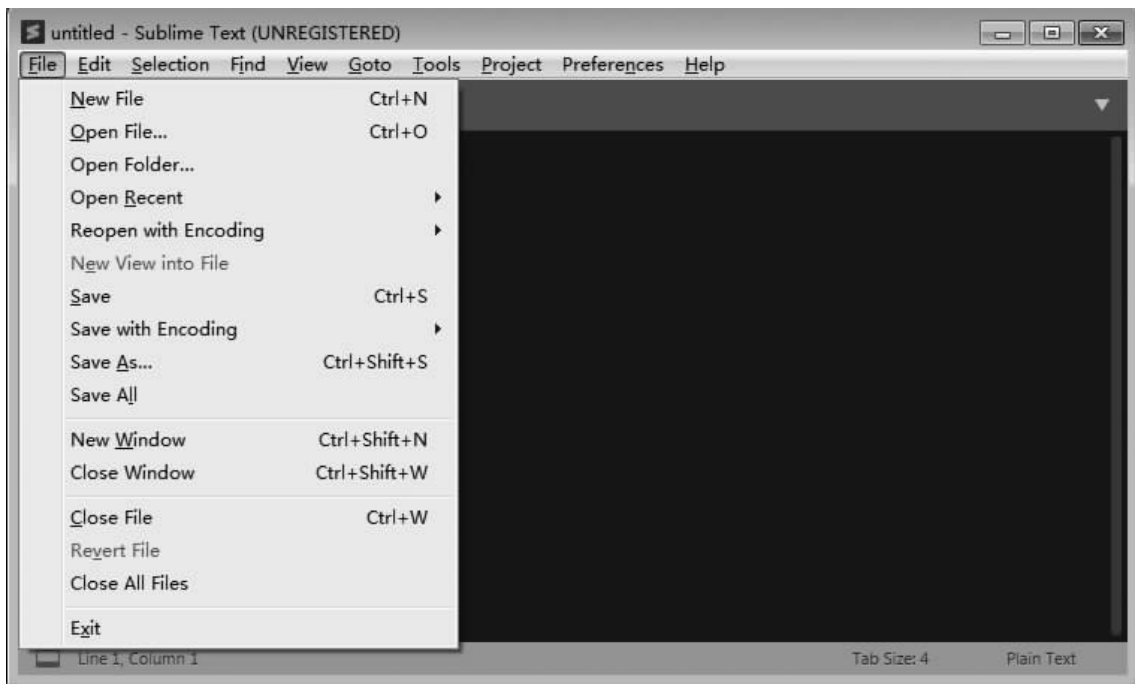

图 1-44 新建文件

(4)在程序编辑区域内可以进行程序的编写。例如,输入“print('Hello Python World!')”,如图 1-45 所示。

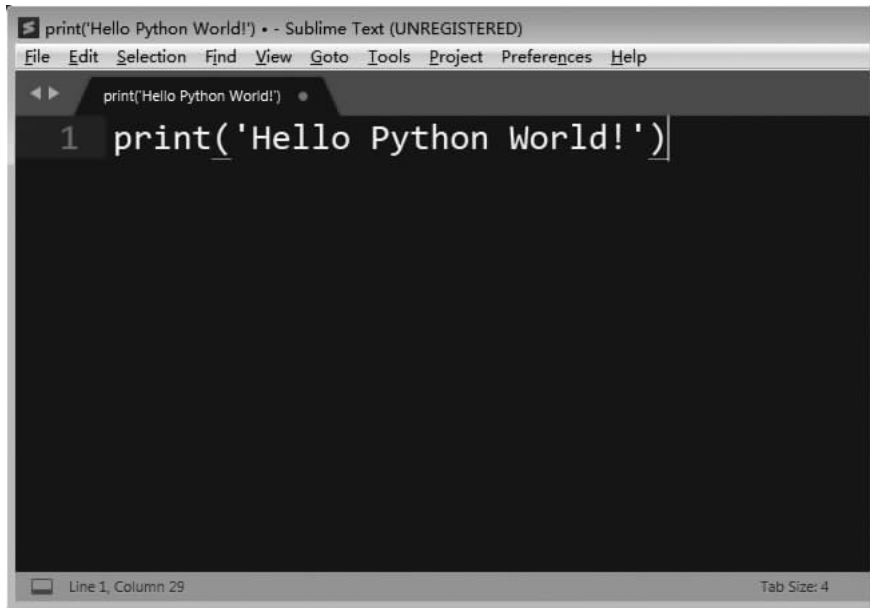

图 1-45　编辑程序

(5)程序编辑完成后需要保存。执行 File→Save 命令,弹出保存程序对话框,选择程序文件的保存路径,输入程序名“HelloPythonWorld. py”,最后进行确认。

(6)执行 Tools→Build 命令,如图 1-46 所示。

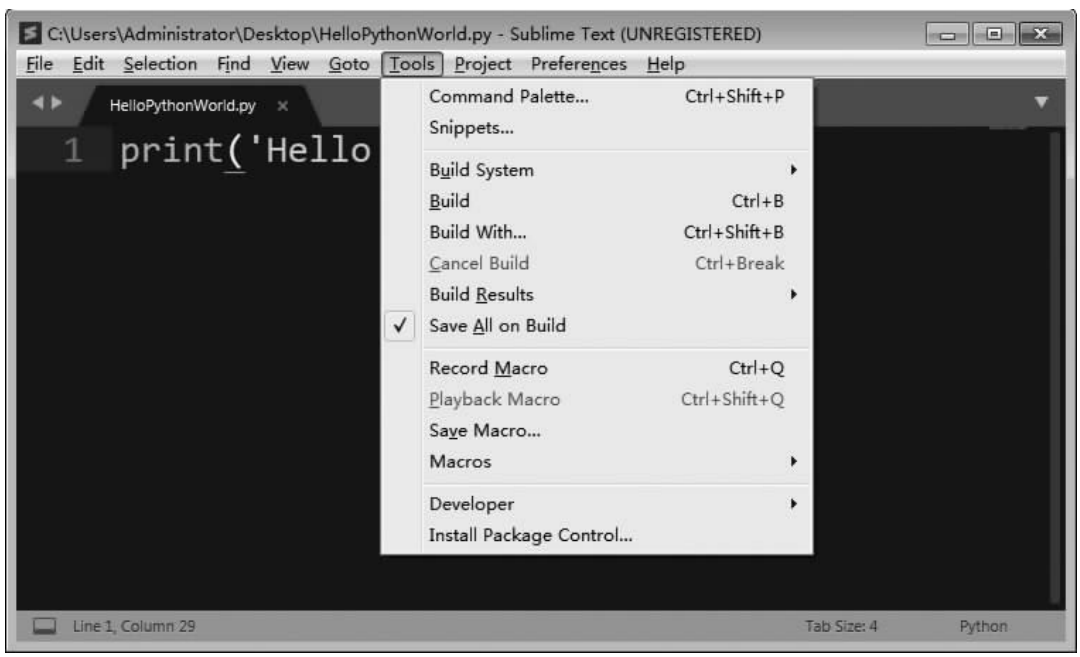

图 1-46　执行 Tools→Build 命令

(7)程序运行结果如图 1-47 所示。

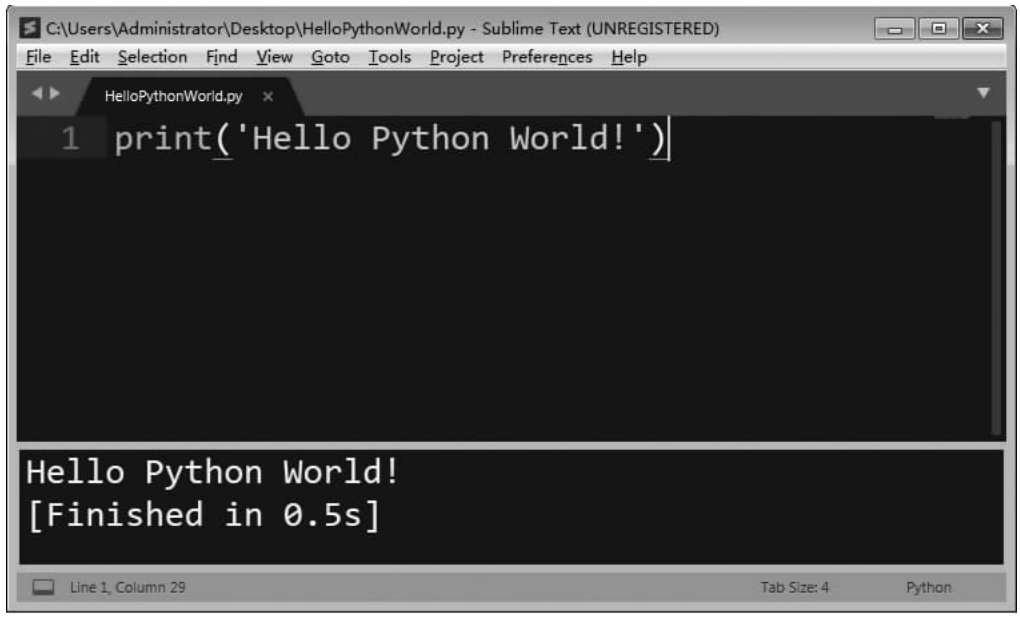

图 1-47　程序运行结果 2

1.4　本章小结

本章主要介绍了 Python 的发展历程、Python 语言的特点及应用领域；讲解了如何在不同操作系统下搭建 Python 编程环境；以 Windows 系统为例，讲解了 Python 集成开发环境 PyCharm 及 Sublime Text 的安装。

通过本章的学习，读者应该对 Python 有一个初步的认识，能够独立完成 Python 开发工具的安装及使用，为 Python 后续的学习打下基础。

1.5　习题

一、选择题

1. Python 的创始人是(　　)。

A. Bruce Eckel　　B. Guido van Rossum

C. Dennis Ritchie　　D. James Gosling

2. (　　)不是 Python 的特点。

A. 解释性　　B. 简单易学

C. 面向对象　　D. 面向过程

3. Python 可以在多种操作系统中运行，这是 Python(　　)特点的体现。

A. 解释性　　B. 丰富的库

C. 可移植性　　D. 可扩展性

二、简答题

1. 简述 Python 的特点。
2. 简述 Python 的主要应用领域。

三、编程题

1. 编写一个 Python 程序，输出“Hello，World!”。
2. 编写一个 Python 程序，输出以下内容：

```
**********************************
Hello Python World!
**********************************
```

第2章

数据类型、变量与运算符

学习目标

- 掌握 Python 常用的数据类型。
- 掌握变量的命名规则及赋值。
- 掌握运算符的作用。
- 了解数据类型转换。

2.1 Python 的数据类型

Python 中的一切皆为对象，每个对象都属于某个数据类型。Python 的数据类型有内置的数据类型、模块中定义的数据类型和用户自定义数据类型。数据类型有多种，本章主要介绍系统默认的 6 个标准数据类型，分别为 Number 类型（数值类型）、String 类型（字符串类型）、List 类型（列表类型）、Tuple 类型（元组类型）、Dict 类型（字典类型）和 Set 类型（集合类型）。

2.1.1 数值类型

数值类型为不可变数据类型，分为整型（int）、浮点型（float）、复数类型（complex）和布尔类型（bool）4 种。

1. 整型

整型即整数类型。在 Python 2 中，整数的大小是有限制的，即当数字超过一定的范围时就不再是 int 类型，而是 long 型。而在 Python 3 中，无论整数的大小为多少，统称为整型，即 Python 3 的整数位数可为任意长度（只受计算机内存的影响）。在 32 位计算机上，整数的位数为 32 位；在 64 位计算机上，整数的位数为 64 位。

声明整型有以下 4 种方式。

1）十进制：0—9

```
变量＝十进制数字
```

例如，a＝0，b＝189，c＝－4。

2）二进制：0—1

```
变量＝0b(B)二进制数字
```

0b(B)是二进制的标志符号。

例如，a＝0b0，b＝0b11，c＝0B0001。

3）八进制：0—7

```
变量＝0o(O)八进制数字
```

0o(O)是八进制的标志符号。

例如，a＝0O2，b＝0o72，c＝0O265。

4）十六进制：0—9，A—F

```
变量＝ 0x(X)十六进制数字
```

0x(X)是十六进制的标志符号。

例如，a＝0x09，b＝0Xb5f，c＝0x123。

2. 浮点型

浮点型即实数的数据类型，声明浮点型有以下两种方式。

1）小数方式

```
变量＝3.14159
```

例如，a＝3.14159。

2）科学计数法

```
变量＝3.141e5      # 相当于 3.141×10^5
```

例如，b＝3.141e5

3. 复数类型

复数包括实数和虚数两部分。实数即现实存在的所有数值，虚数不是真实存在的数字。例如，数学中 x 的平方为－1，x 就是虚数的基本单位，如 1x，2x，5x，…。计算机中常用 j 来表示 x。

声明复数有以下两种方式。

1）表达式方式

```
变量＝实数＋虚数
```

例如，c＝5＋7j。

2）特定功能

```
变量＝complex(实数，虚数值)
```

例如，c＝complex(5，7)。

4. 布尔类型

布尔类型是计算机专用的数据类型，只有 True 和 False 两个值，分别对应二进制中的 0 和 1，常用于逻辑运算。

例如：

```
b1=True
b2=Flase
print(b1,b2)
```

2.1.2 字符串类型

字符串是一个有序的字符集合，是 Python 中最常用的重要数据类型。定义一个字符串类型有以下 3 种方式：

(1)单引号('')。变量='内容'。包含在单引号中的字符串可以为双引号。

(2)双引号("")。变量="内容"。包含在双引号中的字符串可以为单引号。

(3)三引号(""" """)。变量='''内容''' 和变量="""内容"""。包含在三引号中的字符串可同时包含单引号和双引号，且可以跨行。

具体示例如下：

```
str1='123'
print ('str1:',str1)
str2='He said:"I Love China".'
print ('str2:',str2)
str3="I'm 24 years old."
print ('str3:',str3)
str4="""He said:"I'm 24 years old"."""
print ('str4:',str4)
str5='''He said,
          I'm
          24 years old.'''
print ('str5:',str5)
```

程序执行结果如下所示：

```
str1:123
str2:He said:" I Love China ".
str3:I'm 24 years old.
str4:He said:"I'm 24 years old".
str5:He said,
        I'm
        24 years old.
```

2.1.3 列表类型

列表是由一系列顺序排列的特定元素组成的，它的元素可以是任意数据类型，即数字、字符串、列表、元组、字典、布尔值等，同时其元素也是可修改的。列表的标志符号为[ ]。

列表格式如下：

```
变量=[val1,[val2,val3,…]]
```

其中，val1，val2，val3，…为任意数据类型。

具体示例如下：

```
>>> list1=[3,[5,7],8]
>>> list1
[3, [5, 7], 8]
>>> list2=[12.56,-13,'London']
>>> list2
[12.56, -13, 'London']
```

2.1.4 元组类型

元组就是一系列数据的顺序组合，但是组合之后不可以修改。其特性与 list 相似，使用圆括号而不是方括号来标识。元组的标志符号为()。

元组格式如下：

```
变量=(val1,[val2,val3,…])
```

其中，val1，val2，val3，…为任意数据类型。

具体示例如下：

```
>>> t1=(3,5,7,8)
>>> t1
(3, 5, 7, 8)
>>> t2=(3,('London','China'),[5,7],9.76)
>>> t2
(3, ('London', 'China'), [5, 7], 9.76)
```

2.1.5 字典类型

字典是一系列键-值对的无序数据组合。字典的每个键-值(key-value)之间用冒号(:)分隔，每个键值对之间用逗号(,)分隔，整个字典包括在大括号({})中，键必须是唯一的，但值可以不唯一。值可以取任何数据类型，但键必须是不可变的，如字符串、数字或元组。字典的标志符号为{}。

字典格式如下：

```
变量={key1:val1, key2:val2, key3:val3...}
```

具体示例如下：

```
>>> birth_place={'Lucy':'Mexico','Tom':'Toky','Kaile':'Ottawa','Niki':'Hollywood'}
>>> birth_place
{'Lucy':'Mexico', 'Tom':'Toky', 'Kaile':'Ottawa', 'Niki':'Hollywood'}
>>> birth_place['Lucy']
  'Mexico'
>>> birth_place['Niki']
  'Hollywood'
```

2.1.6 集合类型

集合(set)是一个无序的不重复元素序列。

可以使用大括号{ }或 set()函数创建集合，注意创建一个空集合必须用 set() 而不是{ }，因为{ }是用来创建一个空字典的。

集合具有去重、无序、每个元素必须为不可变类型 3 个特性。

创建格式如下：

```
setname={value1,value2,value3,…}
```

或

```
set(value)
```

具体示例如下：

```
>>> city1={'Tokyo','Mexico', 'Tokyo','Ottawa','Hollywood','Mexico'}
>>> city1
{'Hollywood', 'Tokyo', 'Ottawa', 'Mexico'}#去重功能
>>>set1={12, 22, 32, 42, 52}
>>>set2=set((12, 22, 32, 42, 52))
#传入参数必须是可迭代的对象,即序列化数据类型,包括字符串、列表和元组
```

2.2 变量

2.2.1 变量的命名规则

变量名的命名需要遵守一定的规则，具体如下：

(1)变量名必须由字母、数字或下划线组成。

(2)不能使用空格、连字符、标点符号、引号或其他特殊字符。

(3)以字母或下划线开头,但不能以数字开头。

(4)严格区分大小写。

(5)要避免与 Python 关键字和函数名冲突。

例如,_inf_、a_list、var1 和 Name_1 是正确的变量名;12name、first city、print 和 int 是错误的变量名。

2.2.2 变量赋值

1. 直接赋值

例如:

```
变量名=值
```

具体示例如下:

```
a_int=12
a_float=2.2
a_str='string'
a_list=['a','a','a']
a_array=(1,2,3)
a_map={1:'a',2:'b',3:'c'}
```

2. 链式赋值

例如:

```
变量 1=变量 2=变量 3=值(给多个变量同时赋同一个值)
```

具体示例如下:

```
>>> x=y=z=100
>>> x
100
>>> y
100
>>> z
100
```

3. 增量赋值

增量赋值(自增或自减等)操作符:+=、-=、*=、/=、%=、**=、>>=、<<=、&=和|=。

具体示例如下：

```
>>> x=10
>>> x +=5
>>> x
15
>>> x *= 2
>>> x
30
>>> x -= 15
>>> x
15
```

4. 多元赋值

```
变量1,变量2,变量3=值1,值2,值3(给多个变量同时赋不同的值)
```

具体示例如下：

```
>>> x,y,z=1,'python',5.2
>>> x
1
>>> y
  'python'
>>> z
5.2
```

5. 变量交换

```
变量1,变量2=变量2,变量1
```

具体示例如下：

```
>>> a=1
>>> b=10
>>> a,b=b,a
>>> a
10
>>> b
1
```

变量的3个常用操作如下：

(1)获取值：可直接通过变量名获取。

用法：name。

(2)查看数据类型:使用 type()函数。

用法:type(name)。

(3)获取变量在内存中的 id 标识:使用 id()函数。

用法:id(name)。

```
>>> x,y=1,'python'
>>> z=x
>>> id(x)
1437822016
>>> id(z)
1437822016
>>> type(x)
<class 'int'>
>>> type(y)
<class 'str'>
```

2.3 运算符

2.3.1 算术运算符

Python 算术运算符包括+、-、*、**、/、//和%。下面以 x=7,y=2 为例,分别验证各个算术运算符的结果,具体见表 2-1。

表 2-1 算术运算符

运 算 符	描　　述	实　　例
+	加法运算	7+2=9
-	减法运算	7-2=5
*	乘法运算	7*2=14
**	幂运算	7**2=49
/	除法运算	7/2=3.5
//	取商运算(向下取整)	7//2=3
%	取余运算	7%2=1

具体示例如下:

```
a=7
b=2
print ('a+b:',a+b)
```

```
print ('a-b:',a-b)
print ('a * b:',a * b)
print ('a ** b:',a ** b)
print ('a/b:',a/b)
print ('a//b:',a//b)
print ('a%b:',a%b)
```

程序执行结果如下：

```
a+b:9
a-b:5
a * b: 14
a ** b:49
a/b:3.5
a//b:3
a % b:1
```

2.3.2 赋值运算符

Python 赋值运算符包括=、+=、-+、* =、/=、%=、//=、** =。前面 3 个运算符已介绍过，此处不再赘述。表 2-2 列举了常用的赋值运算符。

表 2-2 常用的赋值运算符

运 算 符	描　　述	实　　例
=	普通赋值运算	x=12
+=	加法赋值运算	x+=y 相当于 x=x+y
-=	减法赋值运算	x-=y 相当于 x=x-y
* =	乘法赋值运算	x * =y 相当于 x=x * y
/=	除法赋值运算	x/=y 相当于 x=x/y
%=	取余赋值运算	x%=y 相当于 x=x%y
//=	取商赋值运算	x//=y 相当于 x=x//y
** =	幂赋值运算	x * * =y 相当于 x=x * * y

所有赋值运算的格式都可以转换为[变量=变量 运算符 值]的形式。例如：var /=5 相当于 var=var/5。

具体示例如下：

```
a=5
b=3
c=12
d=2
b*=a
print('b*a=',b)
c%=a
print('c%a=',c)
d**=a
print('d**a=',d)
```

程序执行结果如下：

```
b*a=15
c%a=2
d**a=32
```

2.3.3 关系运算符

Python 关系运算符用于比较两个数，返回布尔值 True 或 False。关系运算符包括==（等于运算）、!=（不等于运算）、<（小于运算）、>（大于运算）、<=（小于等于运算）和>=（大于等于运算）。

具体示例如下：

```
a=5
b=3
print('a<b:',a<b)
print('a>b:',a>b)
print('a==b:',a==b)
print('a<=b:',a<=b)
print('a>=b:',a>=b)
print('a!=b:',a!=b)
```

程序执行结果如下：

```
a<b:False
a>b:True
a==b:False
a<=b:False
a>=b:True
a!=b:True
```

2.3.4 逻辑运算符

Python 逻辑运算符包括 and、or 和 not。具体描述见表 2-3。

表 2-3 逻辑运算符

运 算 符	逻辑表达式	描　　述
and	与运算 a and b	当所有条件都为 True 时，返回 True
or	或运算 a or b	当其中一个条件为 True 时，返回 True
not	非运算 not a	当条件为 True 时返回 False，当条件为 False 时返回 True

具体示例如下：

```
a=True
b=False
print('a and b is: ',a and b)
print('a or b is: ',a or b)
print('not a is: ',not a)
```

程序执行结果如下：

```
a and b is: False
a or b is: True
not a is: False
```

2.3.5 其他运算符

除了上述一些运算符外，Python 还支持成员运算符、同一运算符和位运算符等。本小节重点介绍前两个运算符。

1. 成员运算符

成员运算符 in 是用来判断指定序列中是否包含某个值，若包含则返回 True，否则返回 False。另一个成员运算符 not in 的作用正好与 in 相反。

具体示例如下：

```
>>> str1='Hello, Hollywood!'    #给定对象是字符串
>>> 'll' in str1
True
>>> 'c' in str1
False
>>> 'll' not in str1
```

```
False
>>> 'c' not in str1
True
❶>>> list1=[11,[22,33],44]   #给定对象是列表
>>> a=11
>>> b=22
>>> a in list1
True
>>> b in list1
False
>>> a not in list1
False
>>> b not in list1
True
```

❶中 list1 列表包含 3 个元素，其中第 2 个元素又是一个列表[22,33]，故元素 22 不在 list1 列表中，详见第 4 章列表的介绍。

2. 同一运算符

同一运算符 is 是用来判断两个标识符是否引用自同一个对象。若是则返回 True，否则返回 False。另一个运算符 not is 的作用正好与 is 相反。

具体示例如下：

```
a=10
b=a
print ('id(a):',id(a))
print ('id(b):',id(b))
print (a is b)  #内存地址相同，指向同一对象
print (a is not b)
print ('##################')
a=20
print ('id(a):',id(a))
print ('id(b):',id(b))
print (a is b)
print (a is not b)
```

程序执行结果如下：

```
id(a)：1437822304
id(b)：1437822304
True
False
##################
id(a)：1437822624
id(b)：1437822304
False
True
```

2.3.6 运算符优先级

前面介绍了不同类型的运算符，如果一个表达式中同时出现了多个运算符，这些运算符的优先级是不同的，在此对常用的运算符优先级进行归纳，表 2-4 列出了从最高到最低优先级的常用运算符。

表 2-4 常用运算符及其优先次序（由高到低排序）

运 算 符	描　　述
**	指数（最高优先级）
~	按位取反
+x，-x	一元加号和减号
*、/、%、//	乘、除、取模和取整除
+、-	加法、减法
>>、<<	右移、左移运算符
&	按位与
^	按位异或
|	按位或
<=、<、>、>=、==、!=	关系运算符
=、%=、/=、//=、-=、+=、*=、**=	赋值运算符
is、is not	同一运算符
in、not in	成员运算符
not	逻辑非
and	逻辑与
or	逻辑或

由表 2-4 可以看出，Python 关系运算符的优先级高于逻辑运算符，而算术运算符的优先级高于关系运算符，若需要改变执行顺序，则可以通过加括号来实现。

2.4 数据类型转换

将数据由当前类型转化为其他类型的操作就是数据类型转换。数据类型转换分为两类，分别是自动数据类型转换和强制数据类型转换。

2.4.1 自动数据类型转换

在变量赋值时，根据变量值的类型自动转换数据类型。

具体示例如下：

```
x=10
y=5.2
❶z=x+y
print ('type(x):',type(x))
print ('type(y):',type(y))
print ('type(z):',type(z))
```

程序执行结果如下：

```
type(x): <class 'int'>
type(y): <class 'float'>
❷type(z): <class 'float'>
```

❶中会向更精确的类型转换，所以这里转换为浮点型（见❷）。

2.4.2 强制数据类型转换

根据程序需要，人为改变数据类型的方式称为强制数据类型转换（显式转换）。

1. 将其他类型转换为整型

基本语法如下：

```
class int(x, base=10)
```

其中，x 代表字符串或数字；base 则是进制数，默认值为十进制。

(1)浮点类型转换后，会舍去小数部分。

(2)布尔值转换后，True 为 1，False 为 0。

(3)字符串转换，仅纯整型字符串可以转换（浮点型或带有其他字符的都不可以转换）。

具体示例如下：

```
>>> int()    #不传入参数时,得到结果 0
0
>>> int(1>2)   #布尔值转换之后 True 为 1,False 为 0
0
>>> int ('22')   #不带参数 base,按默认值十进制形式输出
22
>>> int('22',16)   #带参数 base,并指定以十六进制形式输出
34
```

2. 将其他类型转换为浮点型

基本语法如下：

```
class float([x])
```

其中，x 代表字符串或整数。

(1)整型转换后变为浮点型，后面加.0。

(2)布尔值转换后，True 为 1.0，False 为 0.0。

(3)字符串转换，只有纯整型字符串和纯浮点型字符串可以转换，其他都不可以转换。

具体示例如下：

```
>>> float()
0.0
>>> float(1>2)
0.0
>>> float('22')
22.0
>>> float(22.2)
22.2
```

3. 将其他类型转换为复数类型

complex()函数用于创建一个值为 real+imag * j 的复数或将一个字符串或数转化为复数。若第 1 个参数为字符串，则不需要指定第 2 个参数。

基本语法如下：

```
class complex([real[, imag]])
```

其中，real 通常为 int、long、float 或字符串；imag 为 int、long 或 float 类型。

(1)整型转换后变为(整型+0j)。

(2)浮点型转换后变为(浮点型+0j)。

(3)布尔值转化后，True 为(1+0j)，False 为(0j)。

(4)字符串、纯整型和浮点型字符串可以转换，其他字符串不可以转换。

具体示例如下：

```
>>> complex()
0j
>>> complex(2,4)   #整数
(2+4j)
>>> complex(2.2,4)   #浮点数
(2.2+4j)
>>> complex(22)
(22+0j)
>>> complex('22')   #当作字符串处理
(22+0j)
```

4. 将其他类型转换为布尔类型

转换为布尔值 False：

(1)整型：0。

(2)浮点型：0.0。

(3)复数：0+0j。

(4)布尔：False。

(5)字符串：'' 空字符串。

(6)列表：[]空列表。

(7)元组：()空元组。

(8)字典：{}空字典。

(9)集合：set()空集合。

除了以上这些，其他转换为布尔值都会得到 True。示例略。

5. 将其他类型转换为字符串类型

所有转换均改变类型为字符串类型，表示方式不变。

```
>>> a=12
>>> str(a)   #将数字转换为字符型
  '12'
>>> b=[11,[22,33],44]
>>> str(b)   #将列表转换为字符型
  '[11, [22, 33], 44]'
>>> type(str(b))
<class 'str'>
```

6. 将其他类型转换为列表类型

在 Python 中有 5 种可迭代序列可以相互转换，它们分别是字符串、列表、元组、字典和集合。

list()方法用于将元组或字符串转换为列表。具体示例如下：

```
>>> tuple1=(11,22,33,44)
>>> list(tuple1)
[11, 22, 33, 44]
>>> str1='Hollywood'
>>> print (list(str1))
['H', 'o', 'l', 'l', 'y', 'w', 'o', 'o', 'd']
```

7. 将列表转换为元组类型

tuple()将列表转换为元组，具体示例如下：

```
>>> list1=[11, 22, 33, 44]
>>> tuple(list1)
(11, 22, 33, 44)
```

8. 将其他类型转换为集合类型

set()创建一个无序不重复元素集，可进行关系测试，删除重复数据，还可以计算交集、差集、并集等。将可迭代对象返回新的集合对象。

具体示例如下：

```
list1=['H', 'o', 'l', 'l', 'y', 'w', 'o', 'o', 'd']
list2=['h', 'e', 'l', 'l', 'o']
a=set(list1) #去掉重复元素 'o' 和 'l',形成一个新的集合
b=set(list2)
print ('a:',a)
print ('b:',b)
print ('a&b:',a&b)     #取交集
print ('a-b:',a-b)     #取差集
print ('a|b:',a|b)     #取并集
```

程序执行结果如下：

```
a:{'H', 'y', 'l', 'd', 'w', 'o'}
b:{'h', 'e', 'l', 'o'}
a&b:{'l', 'o'}
a-b:{'w', 'H', 'y', 'd'}
a|b:{'H', 'y', 'l', 'd', 'e', 'w', 'o', 'h'}
```

9. 将其他类型转换为字典类型

dict()函数用于创建一个字典。

具体示例如下：

```
>>> dict()     #创建空字典
{}
>>> dict(Lucy='Mexico',Kaile='Ottawa',Niki='Hollywood')
{'Lucy': 'Mexico', 'Kaile': 'Ottawa', 'Niki': 'Hollywood'}
>>> dict([('Lucy' ,'Mexico'),('Kaile' , 'Ottawa'),('Niki' , 'Hollywood')])
#用可迭代对象方式来构造字典
{'Lucy': 'Mexico', 'Kaile': 'Ottawa', 'Niki': 'Hollywood'}
```

2.5 本章小结

本章主要讲解了 Python 中变量、数据类型及常见的运算符。通过本章的学习，读者应该能够准确判断变量命名是否符合规则，了解不同的运算符，并能够进行不同类型的数值运算。

2.6 习题

一、选择题

1. 下列选项中，不符合 Python 命名规范的标识符是(　　)。

A. _city12　　B. vcity

C. 12city　　D. city_name

2. Python 中，数字类型不包括下列哪种类型？(　　)

A. int　　B. float

C. complex　　D. bool

3. 下列语句中，(　　)在 Python 中是非法的。

A. a=b=100　　B. a-= b

C. a=(b=c+100)　　D. a, b=b, a

4. 在 Python 中，如果变量 x=10，那么，x/=10 的结果为(　　)。

A. 1.0　　B. 0

C. 10　　D. 1

5. Python 中“==”运算符比较两个对象的值，下列选项中，哪一个是 is 比较对象的因素？(　　)

A. sum()　　B. id()

C. max()　　D. min()

6. 设 t=('a',)，则 type(t)的结果为(　　)。

A. <class 'str'>　　B. <class 'set'>

C. (class 'list')　　D. (class 'tuple')

二、填空题

1. 表达式 int('11010', 2)的值为_____。

2. 查看变量类型的 Python 内置函数是_____。

3. 已知 x=22，那么，执行语句 x+=5 后，x 的值为_____。

4. 表达式 23/-4 的值为_____；表达式 23//-4 的值为_____。

5. 表达式 not 1 and 1 的结果为_____。

三、简答题

1. 简述 Python 中变量的命名规则。

2. 简述常用运算符及其优先级次序。