第 3 章

编写第 1 个 Python 程序

本章以 Hello World 作为切入点,介绍如何编写和运行 Python 程序代码。Python 程序主要有两种运行方式:①交互式方式运行;②文件方式运行。

3.1 使用 Python Shell

进入 Python Shell 可以通过交互式方式编写和运行 Python 程序。启动 Python Shell 有以下 3 种方式。

(1) 单击 Python 开始菜单中的 Python 3.8(32-bit).lnk 快捷方式,启动 Python Shell 界面,如图 3-1 所示。

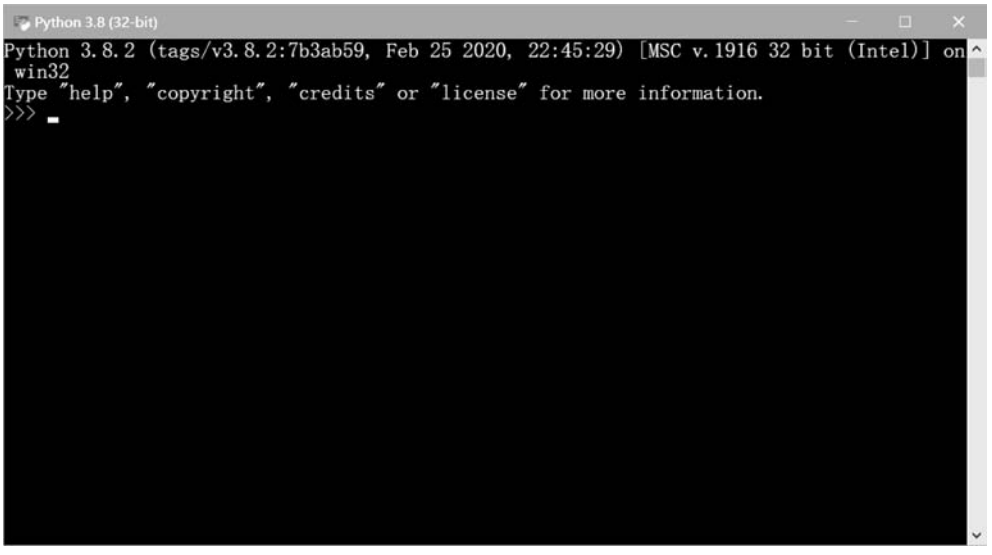

图 3-1 快捷方式启动 Python Shell

(2) 在 Windows 命令提示符(即 DOS)中使用 Python 命令启动。启动命令不区分大小写,也没有任何参数,启动后的界面如图 3-2 所示。

(3) 通过 Python IDLE 启动 Python Shell,如图 3-3 所示。Python IDLE 提供了简单的文本编辑功能,如剪切、复制、粘贴、撤销和重做等,且支持语法高亮显示。

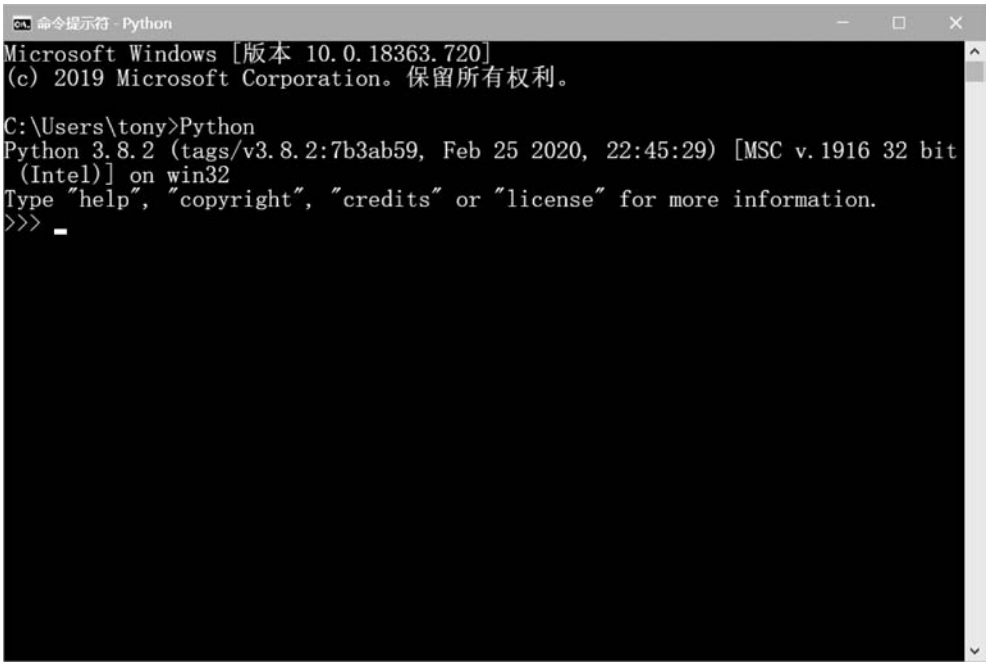

图 3-2　在命令提示行中启动 Python 解释器

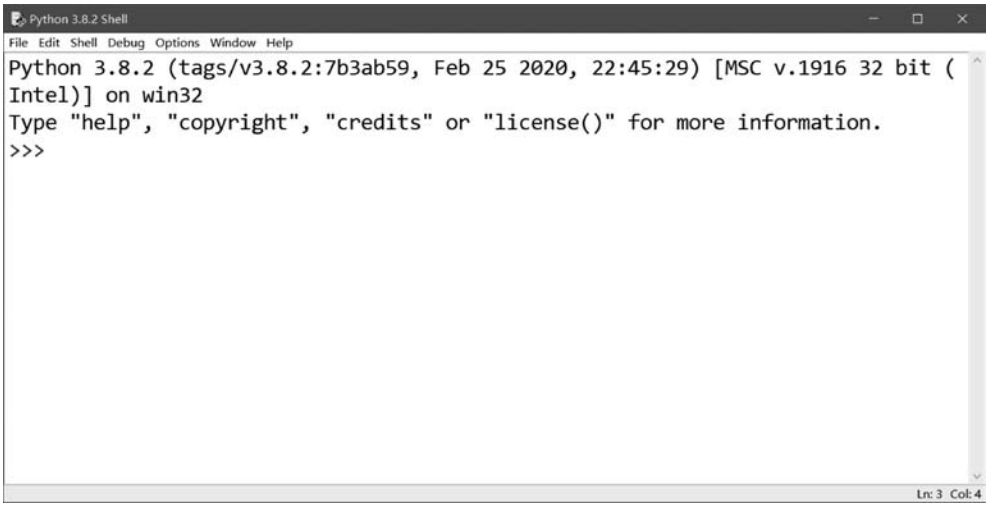

图 3-3　IDLE 工具启动的 Python Shell

无论采用哪种方式启动 Python Shell,其命令提示符都是“>>>”。在该命令提示符后可以输入 Python 语句,然后按 Enter 键即可运行 Python 语句,Python Shell 马上输出结果。图 3-4 为执行几条 Python 语句示例。

如图 3-4 所示,PythonShell 中执行的 Python 语句解释如下:

```
>>> print("Hello World.")      ①
Hello World.                   ②
```

```
>>> 1+1                          ③
2                                ④
>>> str = "Hello, World."       ⑤
>>> print(str)                   ⑥
Hello, World.                    ⑦
>>>
```

代码第①行、第③行、第⑤行和第⑥行是 Python 语句或表达式，第②行、第④行和第⑦行是运行结果。

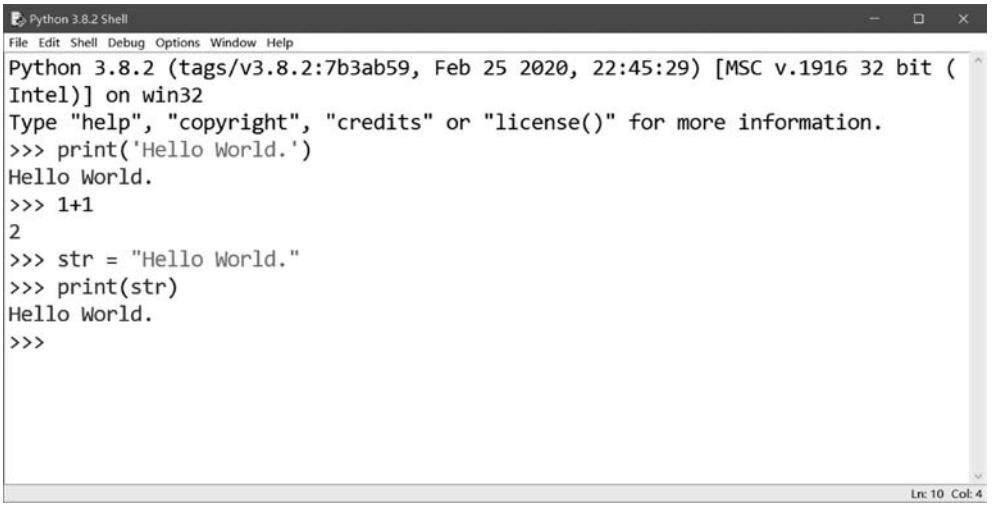

图 3-4　在 Python Shell 中执行 Python 语句

3.2　使用 PyCharm 实现

3.1 节介绍了如何使用 Python Shell 以交互方式运行 Python 代码。交互方式运行不能保存执行的 Python 文件，适合学习 Python 语言的初级阶段，但不适合开发复杂的案例或实际项目。开发复杂的案例或实际项目可以使用 IDE 工具创建项目和 Python 文件，然后再解释运行该文件。

本节将介绍如何使用 PyCharm 创建 Python 项目、编写 Python 文件及运行 Python 文件。

3.2.1　创建项目

在 PyCharm 中需通过项目(Project)管理 Python 代码文件，因此需要先创建一个 Python 项目，然后在项目中创建一个 Python 代码文件。

PyCharm 创建项目步骤如下。打开 PyCharm，在欢迎界面(如图 3-5 所示)单击 Create New Project 按钮或通过选择菜单 File→New Project 打开如图 3-6 所示的对话框，在 Location 文本框中输入项目名称 HelloProj。如果没有设置 Python 解释器或想更换解释器，则可以单击图 3-6 所示的三角按钮展开 Python 解释器设置界面，对于只安装一个版本的 Python 环境读者，笔者推荐选择 Existing interpreter(已经存在解释器)，如图 3-7 所示。

图 3-5　PyCharm 欢迎界面

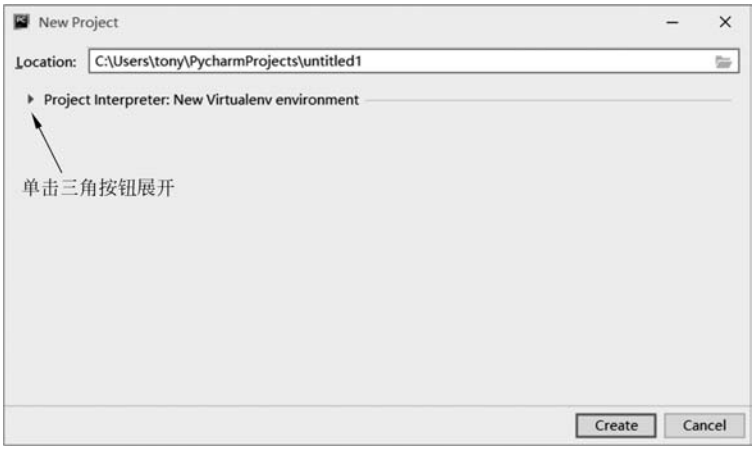

图 3-6　创建项目

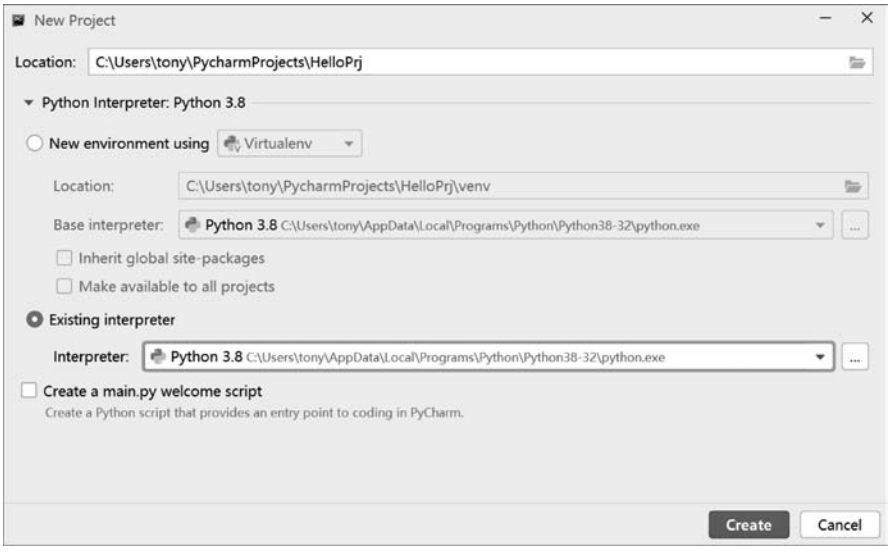

图 3-7　设置项目解释器

根据自己的情况输入项目名称,并选择项目解释器。注意不要选中 Create a main. py welcome script,因为该选项创建项目的同时将创建一个 Python 脚本文件(即代码文件,该过程将在后续介绍)。单击 Create 按钮创建项目,如图 3-8 所示。

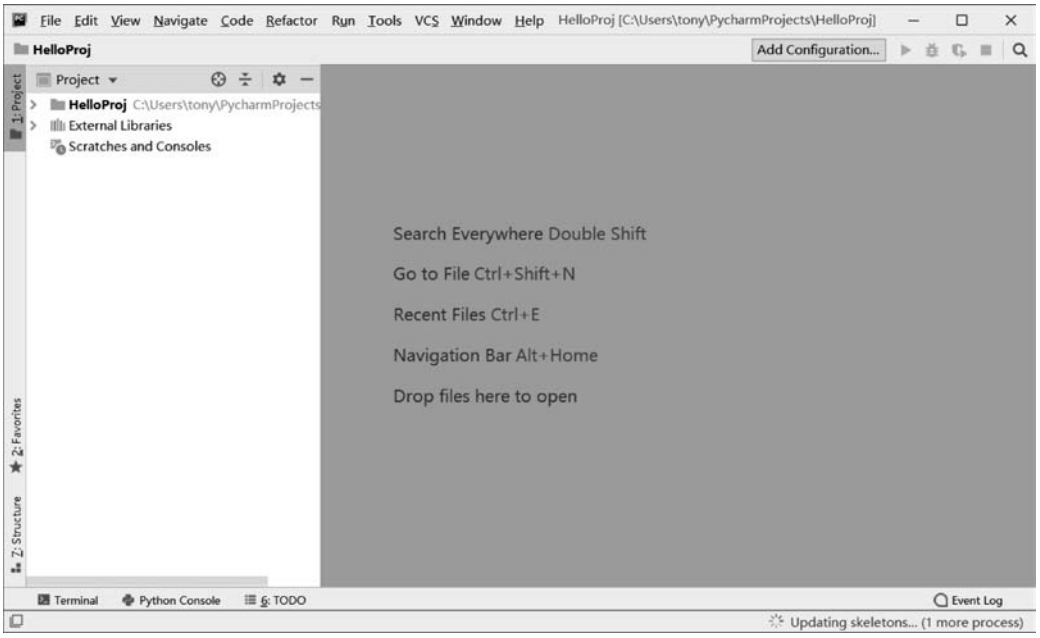

图 3-8　项目创建完成

3.2.2　创建 Python 代码文件

项目创建完成后,需要创建一个 Python 代码文件执行控制台输出操作。打开 3.2.1 节中创建的项目中的 HelloProj 文件夹,然后右击该文件夹,选择菜单 New→Python File,打开 New Python file(新建 Python 文件)对话框,如图 3-9 所示。在对话框的 Name 文本框中输入 hello,然后按 Enter 键创建文件,如图 3-10 所示,在左侧的项目文件管理窗口中可以看到刚刚创建的 hello. py 代码文件。

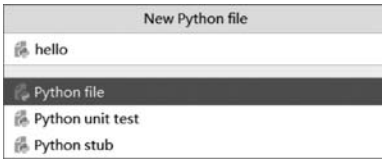

图 3-9　“新建 Python 文件”对话框

3.2.3　编写代码

Python 代码文件不需要 Java 或 C 的 main 主函数,Python 解释器从上到下解释运行代码文件。

编写代码如下:

```
string = "Hello, World."
print(string)
```

3.2.4 运行程序

程序编写完成后,第一次运行前需在图 3-10 所示的窗口左侧的项目文件管理窗口中选择 hello.py 文件,右击菜单并单击 Run 'hello'运行,下方的控制台窗口将输出 Hello, World.字符串,如图 3-11 所示。

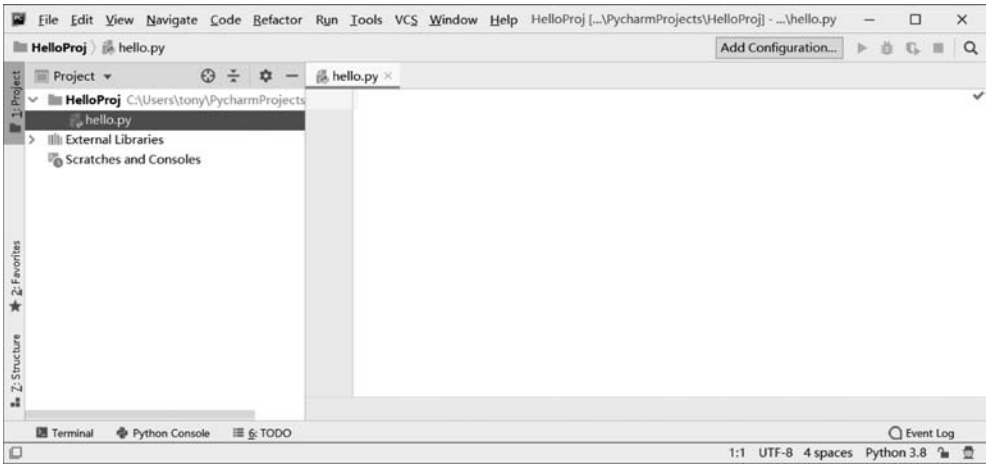

图 3-10 选择 hello.py 文件

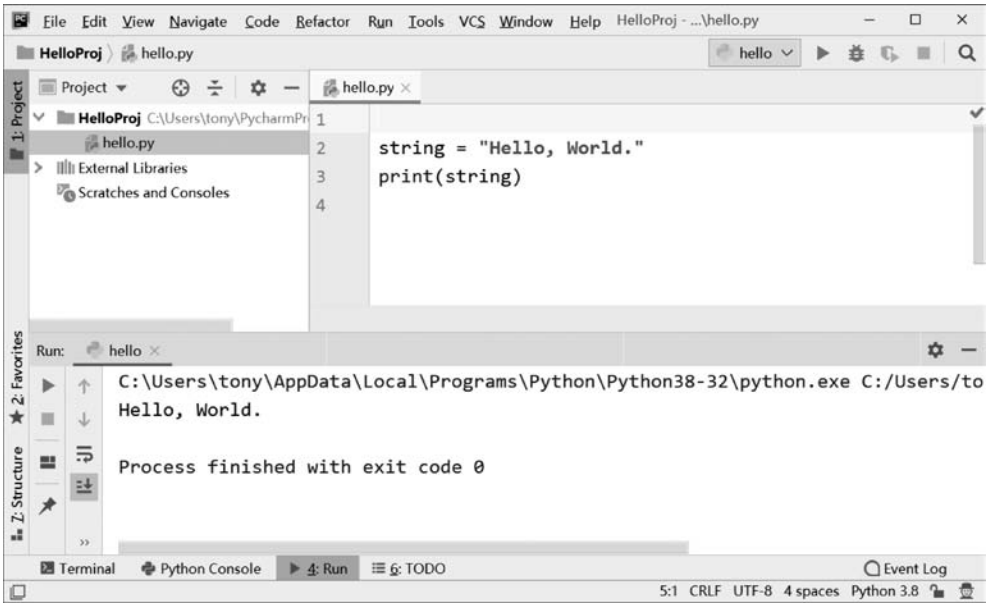

图 3-11 运行结果

注意 如果程序已经运行过,直接单击窗口下方工具栏中的三角按钮,或单击菜单 Run→Run 'hello',或使用快捷键 Shift+F10,均可运行,无须再次选择文件。

3.3　文本编辑工具+Python 解释器实现

如果不想使用 IDE 工具,文本编辑工具+Python 解释器对于初学者而言是一个不错的选择。通过在编辑器中手动输入所有代码,初学者可以熟悉关键字、函数和类,了解 Python 的运行过程,并快速掌握 Python 语法。

3.3.1　编写代码

首先使用任意文本编辑工具创建一个文件并保存为 hello.py,然后在 hello.py 文件中编写如下代码:

```
"""
Created on 2020 年 9 月 18 日
作者: 关东升
"""

string = "Hello, World."
print(string)
```

3.3.2　运行程序

如需运行之前编写的 hello.py 文件,可以在 Windows 命令提示符(Linux 和 UNIX 终端)中通过 Python 解释器指令实现。具体指令如下:

```
python hello.py
```

运行过程如图 3-12 所示。

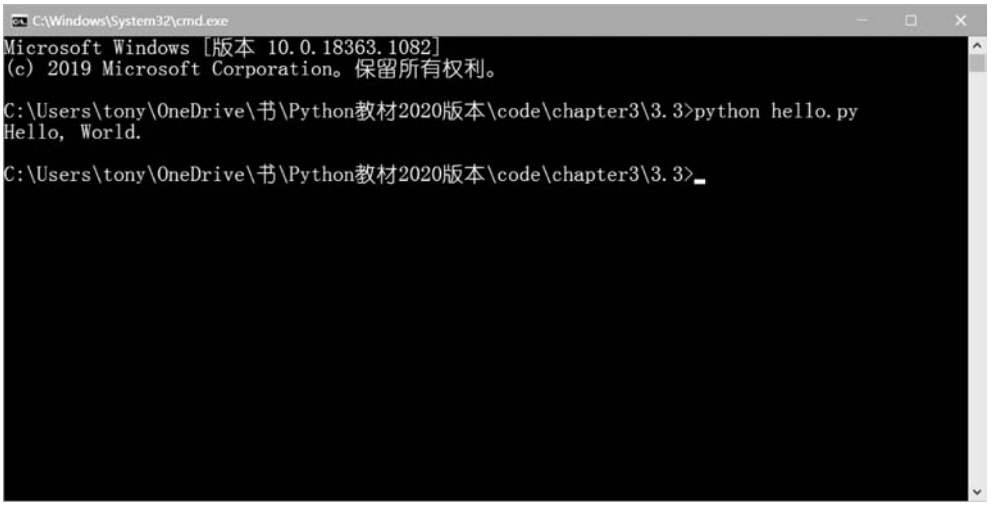

图 3-12　Python 解释器运行过程

有的文本编辑器可以直接运行 Python 文件,如 Sublime Text 工具。使用 Sublime Text 工具打开 Python 文件,按快捷键 Ctrl+B 即可运行文件,如图 3-13 所示。

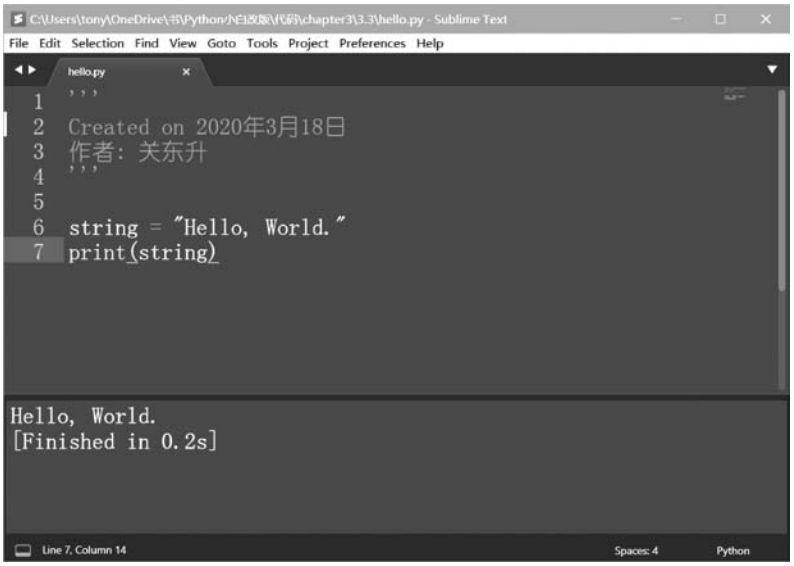

图 3-13　在 Sublime Text 中运行 Python

注意　第一次运行时将弹出如图 3-14 所示的菜单,此时单击 Python 菜单,即可运行当前的 Python 文件。

图 3-14　选择 Python 菜单

3.4　代码解释

至此只是介绍了如何编写和运行 HelloWorld 程序,下面对 HelloWorld 程序代码进行解释。

```
"""                                ①
Created on 2020 年 3 月 18 日
作者: 关东升
"""                                ②

string = "Hello, World."           ③
print(string)                      ④
```

从代码中可见,Python 实现 Hello World 的方式比 Java、C 和 C++等语言简单得多,而且没有 main 主函数。下面详细解释一下代码。

代码第①行和第②行之间使用两对三重单引号包裹,这是 Python 文档字符串,起对文档注释的作用。三重单引号可以换成三重双引号。代码第③行是声明字符串变量 string,并使用"Hello,World. "为它赋值。代码第④行是通过 print 函数将字符串输出到控制台,类似于 C 语言中的 printf 函数。print 函数语法如下:

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

print 函数有 5 个参数：*objects 是可变长度的对象参数；sep 是分隔符参数，默认值是一个空格；end 是输出字符串之后的结束符号，默认值是换号符；file 是输出文件参数，默认值 sys.stdout 是标准输出，即控制台；flush 为是否刷新文件输出流缓冲区，如果刷新字符串则立即打印输出默认值不刷新。

使用 sep 和 end 参数的 print 函数示例如下：

```
>>> print('Hello', end = ',')                          ①
Hello,
>>> print(20, 18, 39, 'Hello', 'World', sep = '|')    ②
20|18|39|Hello|World
>>> print(20, 18, 39, 'Hello', 'World', sep = '|', end = ',')
20|18|39|Hello|World,
```

上述代码中第①行用逗号作为输出字符串之后的结束符号，第②行用竖线作为分隔符。

3.5　本章小结

本章通过一个 HelloWorld 示例，帮助读者了解什么是 Python Shell，以及 Python 如何启动 Python Shell 环境；然后介绍如何使用 PyCharm 工具实现该示例具体过程；还介绍了使用文本编辑器+Python 解释器的实现过程。