第 5 章

科学 Python 生态系统简介

第 4 章中，学习了 Python 3 的历史和基础知识，还在 Windows 计算机和树莓派 Raspbian OS 上运行了第一个 Python 程序。

本章将学习科学 Python 生态系统的基础知识，并详细了解 PyPI 和 pip，以及学习如何使用 Jupyter Notebook 和 IPython 进行交互编程。

5.1 Python 包索引(PyPI)和 pip

Python 附带了一个庞大的库，可以满足许多编程需求。但是，Python 的内置函数库中许多高级功能和数据结构不可用，因此存在第三方库项目可用。其中很多项目都可以在 PyPI 中找到。可以通过 https://pypi.org/访问 PyPI。其主页上介绍了 PyPI 是 Python 编程语言的软件库。

可以使用名为 pip 的命令行实用程序安装 PyPI 上托管的 Python 软件包和软件库。pip 代表 pip install packages 或 pip installs python。它是一个递归的首字母缩略词，在扩展时会包含自身。对于 Python 3，有 pip3 命令，可以在 Windows 或树莓派上运行它。它包含了 Python 3 的所有发行版，因此不必单独安装。

使用以下命令查看当前 Python 3 环境中安装的软件包列表。

```
pip3 list
```

使用以下命令在 PyPI 中搜索特定的软件包。

```
pip3 search numpy
```

使用以下命令从 PyPI 中安装特定的软件包。

```
pip3 install numpy
```

使用以下命令从计算机中卸载特定的软件包。

```
pip3 uninstall numpy
```

如果在树莓派上运行命令时遇到权限问题，则必须在命令前面加 sudo。

5.2　科学 Python 生态系统

科学 Python 生态系统由 SciPy 和相关项目主导，首先详细了解 SciPy。

SciPy 代表科学 Python，是一个用于数学、科学和工程的基于 Python 的开源软件生态系统。可以在 https://www.scipy.org/中了解更多细节。SciPy 有许多不同但相关的、协作开发的库，以下是其核心库。

- NumPy。NumPy 是使用 Python 进行科学计算的基本软件包。科学 Python 生态系统中的几乎所有其他库都使用 NumPy 的 Ndarray 数据结构。我们将在后面的章节中详细介绍 NumPy。访问 http://www.numpy.org/可以获取详细信息。
- SciPy 库。SciPy 库是构成 SciPy 堆栈的核心软件包之一。它提

供了许多用户友好和高效的数值例程，如数值积分和优化例程。可以在 https://www.scipy.org/scipylib/index.html 上找到有关它的详细信息。

- Matplotlib。它是 Python 中的 2D 绘图库，用于创建出版品质的可视化图像。它可以创建科学数据的 2D 和 3D 可视化图像，在 https://matplotlib.org/上可以找到有关它的详细信息。本书中有专门章节介绍 Matplotlib。
- SymPy。它是一个用于符号计算的库。访问 https://www.sympy.org/en/index.html 可以获取详细信息。
- Pandas。Pandas 是一个开源、BSD 授权的软件库，提供高性能、易于使用的数据结构和数据分析工具。可以在 http://pandas.pydata.org/上找到有关 Pandas 的更多信息。Pandas 主要用于数据科学计算和可视化图像。
- IPython。IPython 为 Python 提供了交互式计算环境。将在本章详细讨论 IPython。访问 http://ipython.org/可以获取详细信息。

5.3 IPython 和 Jupyter

IPython 是一个用于 Python 的交互式计算工具，可以在命令提示符窗口以及浏览器中运行。IPython 仅支持 Python 语言。但是，由于它的许多功能（如 Notebook 和 NoteBook 服务器）非常吸引人，因此 IPython 演变成另一个相关的项目，称为 Jupyter。Jupyter 支持许多其他编程语言。IPython 仍在 Jupyter 项目中提供 Python 内核。可以在 https://jupyter.org/上找到有关 Jupyter 项目的详细信息。本书将使用 Jupyter 编写所有编程示例。在 Windows 命令提示符窗口中运行以下命令。

```
pip3 install jupyter
```

该命令将在 Windows 计算机上安装 Jupyter。

要在树莓派上安装 Jupyter，请在终端中依次运行以下命令。

```
sudo pip3 uninstall ipykernel
sudo pip3 install ipykernel == 4.8.0
sudo pip3 install jupyter
sudo pip3 install prompt - toolkit == 2.0.5
```

这将在树莓派上安装所有依赖项的正确版本。

安装 Jupyter Notebook 后，可以通过在 Windows 命令提示符窗口或树莓派终端上运行以下命令启动它。

```
jupyter notebook
```

运行 Jupyter Notebook 时，它将在命令提示符窗口启动一个 Notebook 服务器，并在计算机默认的 Web 浏览器上启动一个 Jupyter Notebook 的实例，如图 5.1 所示。

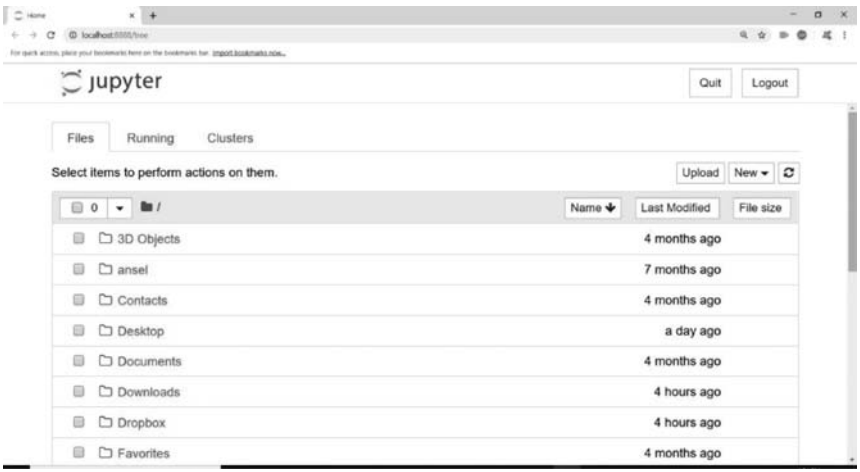

图 5.1　Jupyter 实例

图 5.1 所示的截图提供了一个 Jupyter 的实例。可以从在命令提示符窗口中启动 Jupyter 的目录中看到文件和文件夹，并将该目录视为当

前实例的根目录。

同样地，在命令提示符窗口中，它将显示 Jupyter 的执行日志。可以在该日志中找到以下字符。

```
Copy/paste this URL into your browser when you connect for the first time, to
login with a token:
http:// localhost:8888/?token = e7f8818e7673d0d4dbd46f7376e8b4bb1a07aebc91e6a64c
```

在上面的日志中，可以找到包含当前会话令牌的 URL。如果要使用其他浏览器登录 Jupyter Notebook，则可以直接使用该 URL。

也可以在浏览器的地址栏中搜索 http://localhost:8888/。在这种情况下，需要从执行日志中复制并粘贴令牌（上述日志中 token＝之后的字符串），然后单击 Log in 按钮，如图 5.2 所示。

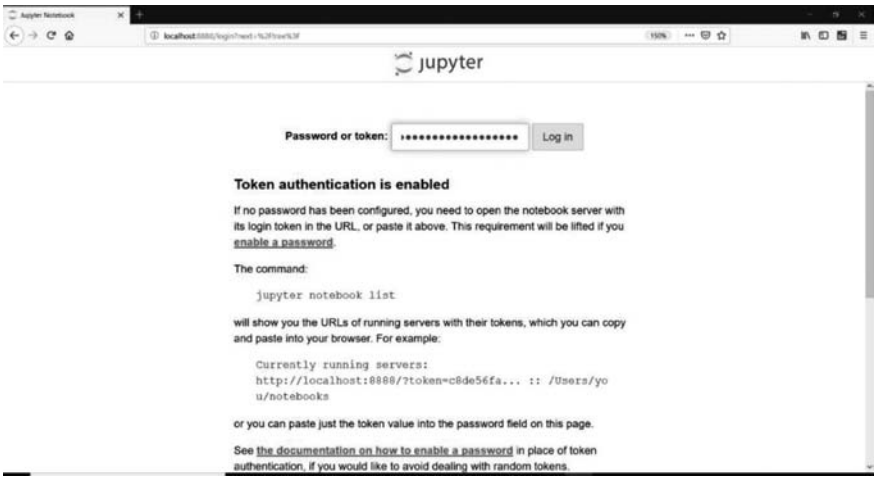

图 5.2　通过令牌登录 Jupyter

像先前的实例一样，这将启动一个 Jupyter 的实例。

下面介绍如何将其用于 Python 3 编程。单击右上角的 New 按钮，会显示一个下拉列表，其中列出了 Windows 计算机或树莓派上安装的各种编程语言。它通过引用系统中的 PATH 变量检测 Python 解释器，如图 5.3 所示。

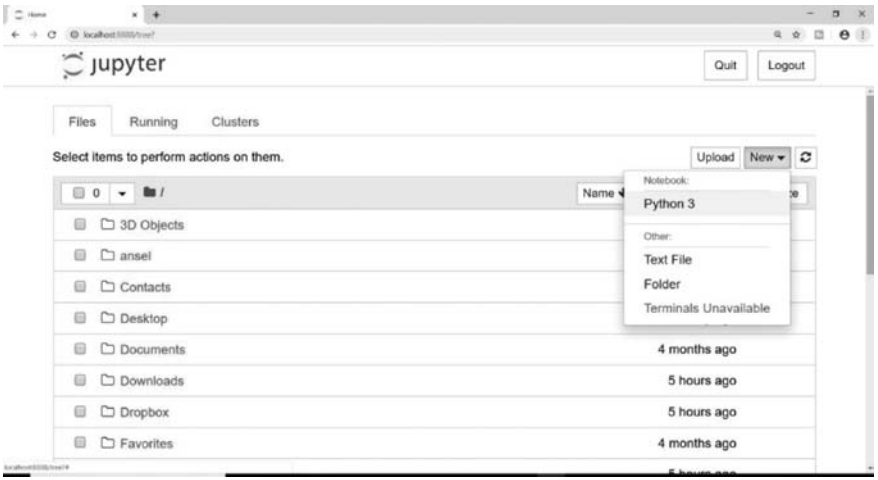

图 5.3　新的 Python 3 Notebook

它将启动适用 Python 的 Jupyter Notebook。默认情况下，它是无标题的。另外，将在命令提示符窗口中启动的 Jupyter Notebook 的当前目录中创建一个对应的文件 Untitled.ipynb。还可以选择创建新目录和文本文件。如果要将 Notebook 合并到一个目录中，则可能要创建一个新目录，并在其中创建 Notebook。以下是 Notebook 的截图，如图 5.4 所示。

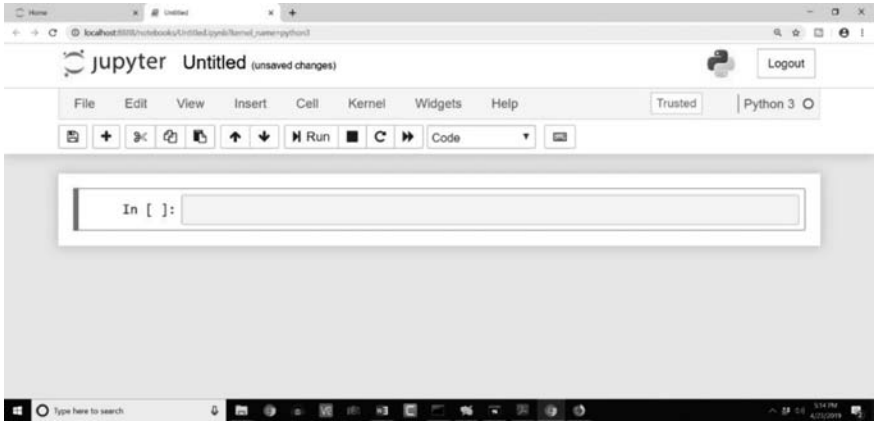

图 5.4　无标题的 Notebook

双击 Notebook 的名称(在本例中为 Untitled),将弹出一个窗口,可以在其中添加新名称,如图 5.5 所示。

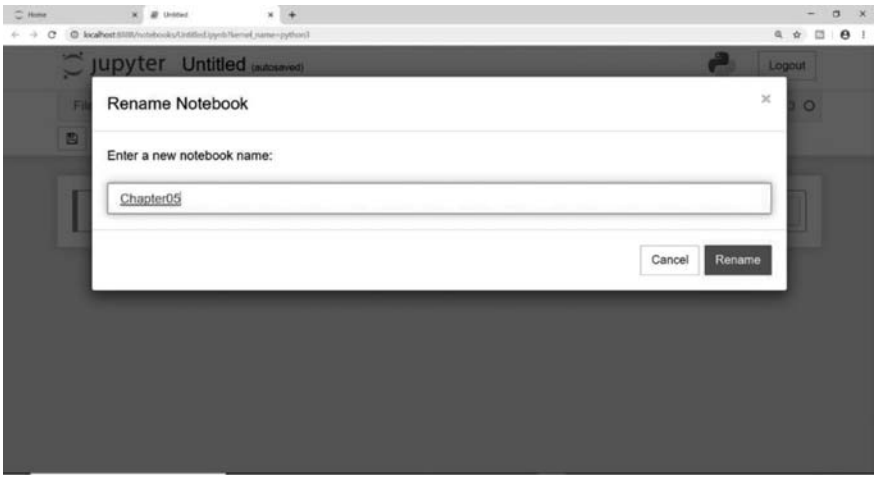

图 5.5　更改 Notebook 名称

单击 Rename 按钮后,它将更改 Notebook 以及文件系统中相应的 ipynb 文件的名称。对于本书,为每一章创建一个 Notebook,并且所有这些 Notebook 将在配套源代码包中提供。如果要使用其中任何一个 Notebook,只需要将其复制到计算机或树莓派的目录中,然后在 Windows 命令提示符窗口或树莓派终端中,从该目录启动 Jupyter Notebook。启动后,复制的所有 ipynb 文件将在 Jupyter Notebook 中显示为列表。

下面开始使用 Notebook。首先了解本书中将要使用的所有重要特性。这些是 Jupyter Notebook 最重要的特性,任何初学者都会发现它们非常有用。在工具栏中,如果单击下拉列表,将显示四个选项。第一个是 Code,第二个是 Markdown,如图 5.6 所示。

Jupyter Notebook 具有不同类型的单元格。例如,我们可能希望有一个单元格用于代码,而另一个单元格用于说明代码段用法的标题。对于标题,使用富文本格式。它由 Markdown 实现,Markdown 是一种轻量

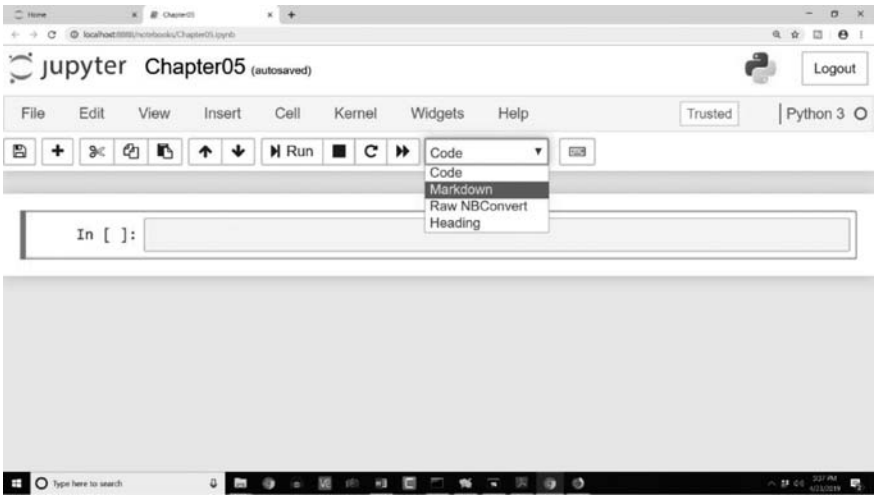

图 5.6　工具栏下拉选项

级的标记语言，具有纯文本格式语法。从工具栏中选择 Markdown 时，它将当前单元格转换为 Markdown 单元格，可以在其中添加带有 Markdown 的富文本。从工具栏中选择 Markdown 之后，将以下代码添加到当前单元格中。

```
# Hello World! Program
```

然后，单击工具栏中的 Run 按钮。Jupyter Notebook 将当前语句解释为一个 Markdown，并以 H1 样式标题阅读，如图 5.7 所示。

这样，就可以在 Jupyter Notebook 中引入具有富文本内容的单元格。Markdown 语法超出了本书的范围，因为仅用于标题和子标题。如果想使用 Markdown 的其他功能，只需要访问维基百科和 https://daringfireball.net/projects/markdown/。

一旦使用 Run 按钮执行了当前单元格的内容，除了渲染输出之外，Jupyter Notebook 会创建一个新的单元格并将光标设置在那里。默认情况下，所有新单元格的类型均为 Code。现在，在 Python 3 代码的新单元格中输入以下命令。

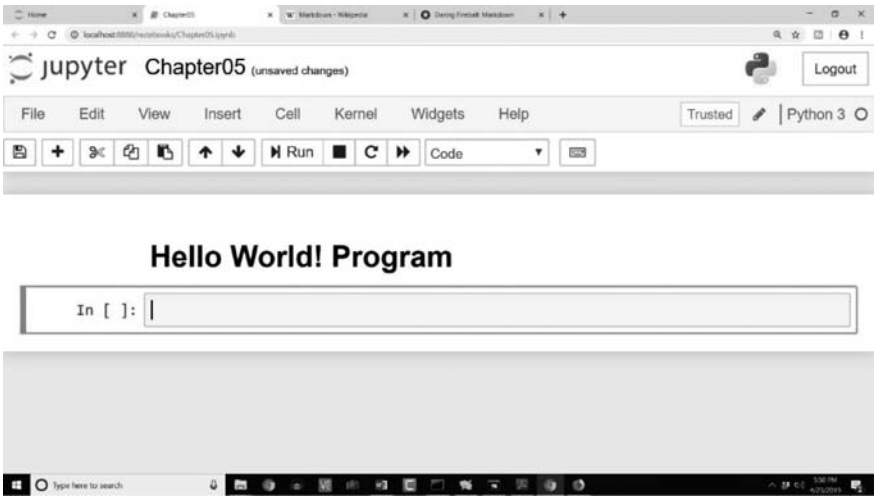

图 5.7　带 Markdown 的 H1 标题

```
print("Hello World!")
```

然后运行单元格，将其解释为 Python 语句，并显示如图 5.8 所示的输出。

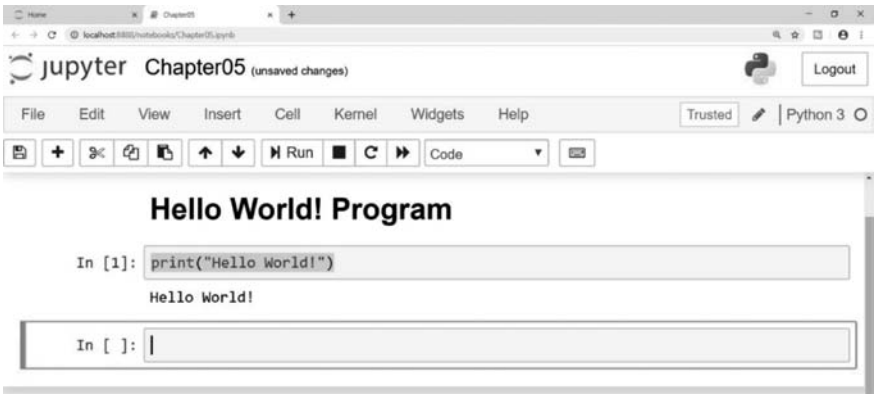

图 5.8　Python 3 代码的输出

这样，Jupyter Notebook 可以包含 Markdown 类型和 Code 类型的单元格，这使我们能够拥有多种输出类型以及作为同一 Notebook 的部分

代码。Jupyter Notebook 的另一个重要特性是可以重新编辑已经执行过的单元格，然后再次执行重新编辑的代码。这使我们可以根据需要编辑代码单元。

下面介绍工具栏上的其他选项。最左侧的为保存按钮，与其相邻的按钮（带有＋号）将在当前高亮显示的单元格之后立即创建一个新单元格，并将光标设置在此处。如果要在现有单元格之间添加新单元格，此功能非常有用。接着有一组用于剪切、复制和粘贴操作的按钮。之后，有两个按钮（向上和向下箭头），用来向上和向下移动当前单元格的位置。最后，有一组与单元格执行相关的按钮。第一个按钮是 Run；第二个按钮（实心方形）用于中断内核，如果单元格正在执行，则会导致执行的中断；第三个按钮用于重启内核；第四个按钮用于重启内核并重新运行整个 Notebook。

下面介绍工具栏上方菜单栏中最常用的选项。大多数菜单项都与任何一个其他文本或代码编辑器中提供的选项相似。此处将讨论用于立即清除整个 Notebook 执行的输出的菜单选项。在 Cell 菜单项中的 All Output 选项下，单击 Clear 选项，将清除整个 Notebook 的执行输出，如图 5.9 所示。

这样，就可以将 Jupyter Notebook 用于 Python 编程，还可以在 Notebook 中显示可视化图像，这将在第 6 章中介绍。

5.4　小结

本章简要介绍科学 Python 生态系统，还学习了 Python 的 PyPI 和 pip。然后，了解如何使用 Jupyter Notebook 程序创建交互式 Notebook。Jupyter 是进行协作的重要工具，可以通过代码示例向他人实时解释自己的想法，然后与参与者共享包含代码、富文本格式和可视化图像的 Jupyter Notebook。因此，Jupyter Notebook 在使用 Python 3 编程进行

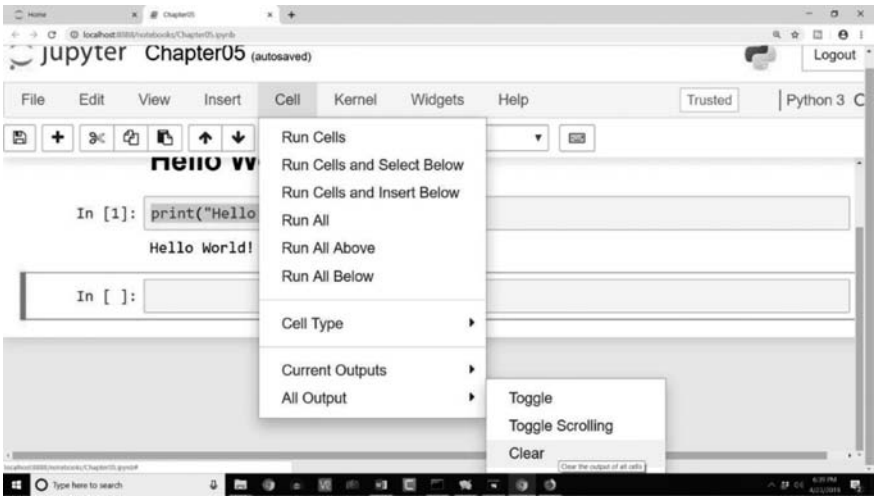

图 5.9　清除整个 Notebook 的执行输出

科学计算的学术机构、研究组织和技术型组织中是非常有用的工具。

在第 6 章中，将详细介绍 NumPy，还将对 Matplotlib 库进行非常简短的介绍。

练习

请浏览本章介绍的所有 URL，并浏览 Jupyter Notebook 菜单栏中的其余选项。