# 第1章 Python介绍及学习前的准备

在正式介绍如何使用Python进行数据分析之前，先来简单地了解一下Python的定义，以及学习Python时使用的相关软件的安装和配置方法。

## 1.1 Python是什么

在IEEE发布的2017年编程语言排行榜中，**Python高居首位**。对于这样一门流行的编程语言，很多Python入门图书都给它进行了定义，但本书作者认为，较为清晰明了的定义来自维基百科：

Python是一种广泛使用的高级编程语言，属于通用型编程语言，由吉多·范罗苏姆创造，第一版发布于1991年。Python可以被视为一种改良（加入一些其他编程语言的优点，如面向对象） 的LISP。作为一种解释型语言，Python的设计哲学强调代码的可读性和简洁的语法（尤其是使用空格缩进划分代码块，而非使用大括号或者关键词）。相比于C++或Java，Python让开发者能够用更少的代码表达想法。无论是小型程序还是大型程序，Python都试图让程序的结构清晰明了。

这段文字囊括了读者需要了解的关于Python的基本信息。

1）Python目前被广泛使用。

2）Python属于高级编程语言，这区别于C语言这样的中级语言或底层的硬件编程、汇编等语言。

3）Python由吉多·范罗苏姆创造，于1991年发布 。

4）Python支持面向对象编程（OOP）。

5）Python属于解释型语言。解释型语言以文本的方式存储程序代码，不需要在运行前进行编译（为大众所熟知的C语言就不是解释型语言，在运行前必须编译为机器识别的语言）。

6）强调代码的可读性和简洁的语法是Python的设计哲学，这一点尤其需要注意和理解，因为这是Python在形式上有别于其他编程语言之处。Python使用空格的缩进来划分不同的代码块，其他一些常见语言一般使用大括号或者关键字，正是这个特点，让Python代码无论大小长短都看起来非常简单清晰，易于使用（读者将会在本书学习的过程中深入理解这一特点）。

## 1.2 为什么使用Python进行数据分析

近年来，Python的科学计算库（如结构化数据操作库pandas、机器学习库scikit-learn）不断进行改良，使得利用Python进行数据分析成为了优选方案。 Python还有一个“胶水语言”的外号，因为它能够非常轻松地集成C、C++等底层代码，进行计算优化。与SAS和R这样的分析建模领域特定编程语言相比，Python可以同时用于项目原型的构建和生产，从而避免了使用多个语言的麻烦。加上Python本身多年来不断提升的强大编程能力，用户只使用Python就可以完成以数据为中心的建模、分析与应用。

## 1.3 科学计算核心库简介

Python拥有着众多的软件包/库，本书难以全部涉及，这里仅介绍几个构成Python科学生态系统的核心“成员”。

* NumPy：NumPy是Numerical Python的简称。NumPy是Python科学计算基础的库，基本上涉及数据分析的软件包都基于它构建。
* pandas：pandas的名字来源于Python数据分析（Python data analysis）和面板数据（panel data）的结合。该库提供了多个数据存储对象，其中的DataFrame对象可以表征数据分析常见的二维表格。除此之外，它还提供了非常多便捷处理结构化数据的函数。
* Matplotlib：它是起源于矩阵实验室MATLAB中的绘图函数，是Python中比较流行的绘图库，可以轻松进行二维数据甚至多维数据可视化。
* SciPy：该库提供了一组专门用于科学计算中各种标准问题包，如数值积分、微分、信号处理，它与NumPy结合可以处理诸多科学计算问题。
* Jupyter：它是一个交互和探索式计算的高效环境。其中两个组件较为常用，一是IPython，用于编写、测试和调试Python代码；二是Jupyter Notebook，它是一个多语言交互式的Web笔记本，现在支持运行Python、R等多种语言，与Markdown结合可以创建良好、可重复的动态文档。这也是我们进行Python数据分析的学习环境。

## 1.4 搭建环境

目前，Python存在Python 2（现在一般指Python 2.7）和Python 3（现在一般指Python 3.5及以上）两个不同的版本，而且Python官方宣布将于2020年停止Python 2的更新和维护，全面进入Python 3时代。因此，本书的介绍以Python 3版本为基础。

目前流行的Python集成开发环境（IDE）有很多，如PyCharm、Sublime Text、Eclipse+PyDev和Anaconda中的Spyder。不同软件、系统的安装和配置方式各不相同，本书使用Anaconda平台的Jupyter Notebook进行介绍。Anaconda是非常强大的跨系统开源计算平台，支持个人PC使用的Windows、Linux和macOS，提供的近1000个软件包可以完全满足个人工作任务的需求。

为了满足不同用户的需求，本书介绍两种Python线上平台，以及本地机器环境下相关软件的安装，读者可任意选择使用。

### 1.4.1 线上平台

目前，网络上有很多在线的Python解释器，可以在有网络服务的情况下通过浏览器运行代码。因为包的导入和计算都在服务器端，所以用户不需要较高配置的计算机就能进行学习和分析。本书推荐两个免费的Jupyter Notebook网站，读者可结合自己计算机的配置和网络情况进行选择。

1）Jupyter官方提供的使用Ipython学习教程网站，其中包含学习Ipython使用的例子和练习，读者可以在该教程的主页运行、调试代码，并下载Jupyter笔记本到本地存储。



图1-1 XXXXXX

2）微软公司提供的Jupyter数据探索学习平台Azure，如图1-2所示，支持在线运行多种语言进行数学科学探索，比较常用的是Python和R。用户可以通过微软账户创建仓库，新建Jupyter Notebook并书写代码和探索数据，完成后可以保存、与他人分享（使用过GitHub等开源仓库的读者会发现这个平台的操作和它们极为相似）。

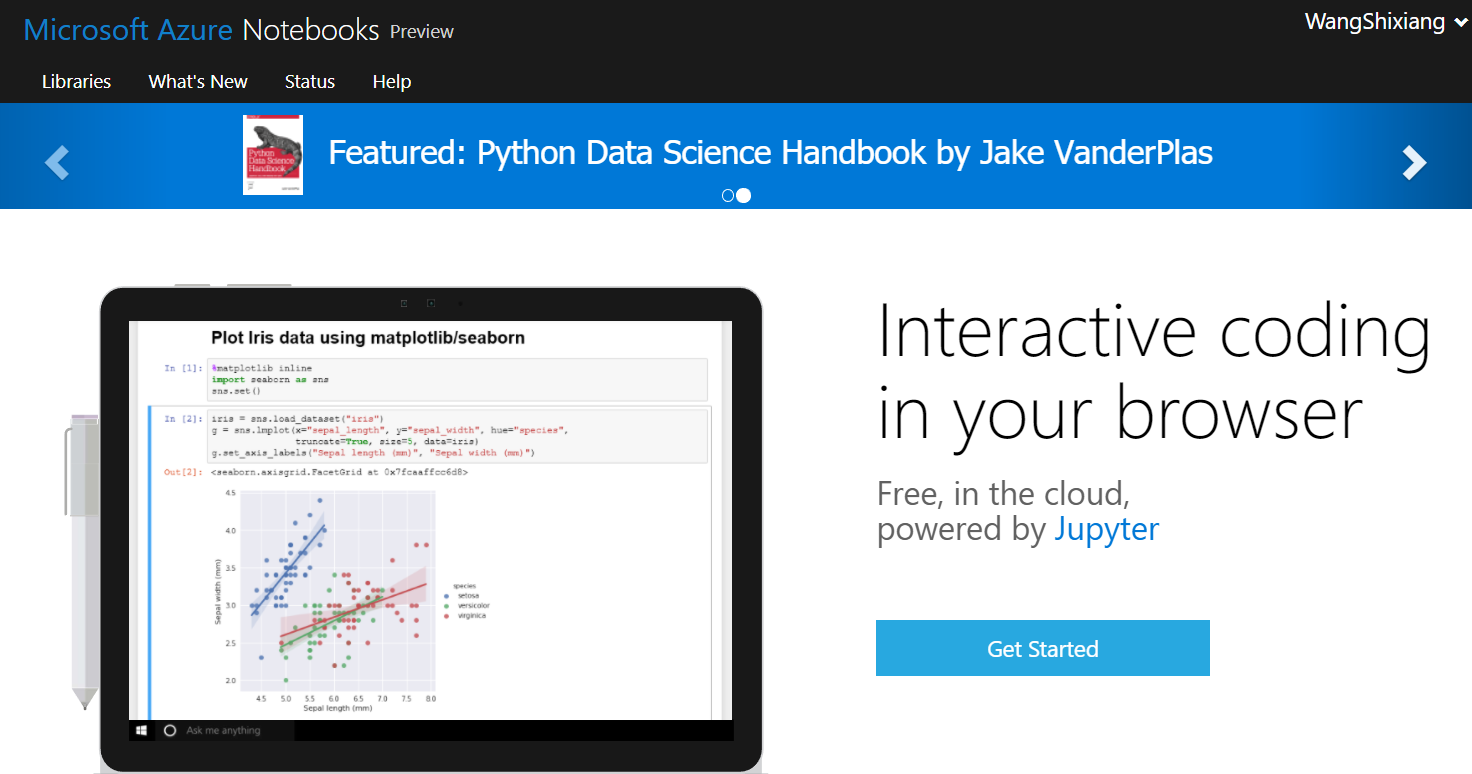


图1-2 Azure平台

本书作者推荐使用Azure平台，因为其在创建、使用、保存与分享方面占有优势，不过读者首先需要做的就是拥有一个微软账号。

### 1.4.2 本地机器环境下相关软件的安装

如果读者想要在本地部署学习环境，那么可以选择安装相关的两款软件，第一款软件是上文已经提到的Anaconda，其为必需软件；第二款软件为nteract（见图1-3），其为可选软件。与Anaconda默认提供的Jupyter Notebook不同，nteract像我们常用的文字编辑器一样，界面非常简洁，可以非常方便地编辑Jupyter Notebook文件（文件扩展名为.ipynb）。nteract目前只有alpha版本（测试版），功能还在不断完善中，当然，这不会影响我们使用它学习Python。本书推荐读者使用nteract，另外，本书后续的代码和文档展示也都会使用它。由于Jupyter Notebook与nteract运行Python都是基于IPython内核ipykernel，除了显示效果，在使用上并没有太多的不同，因此读者完全不必担心软件的选择和使用问题。

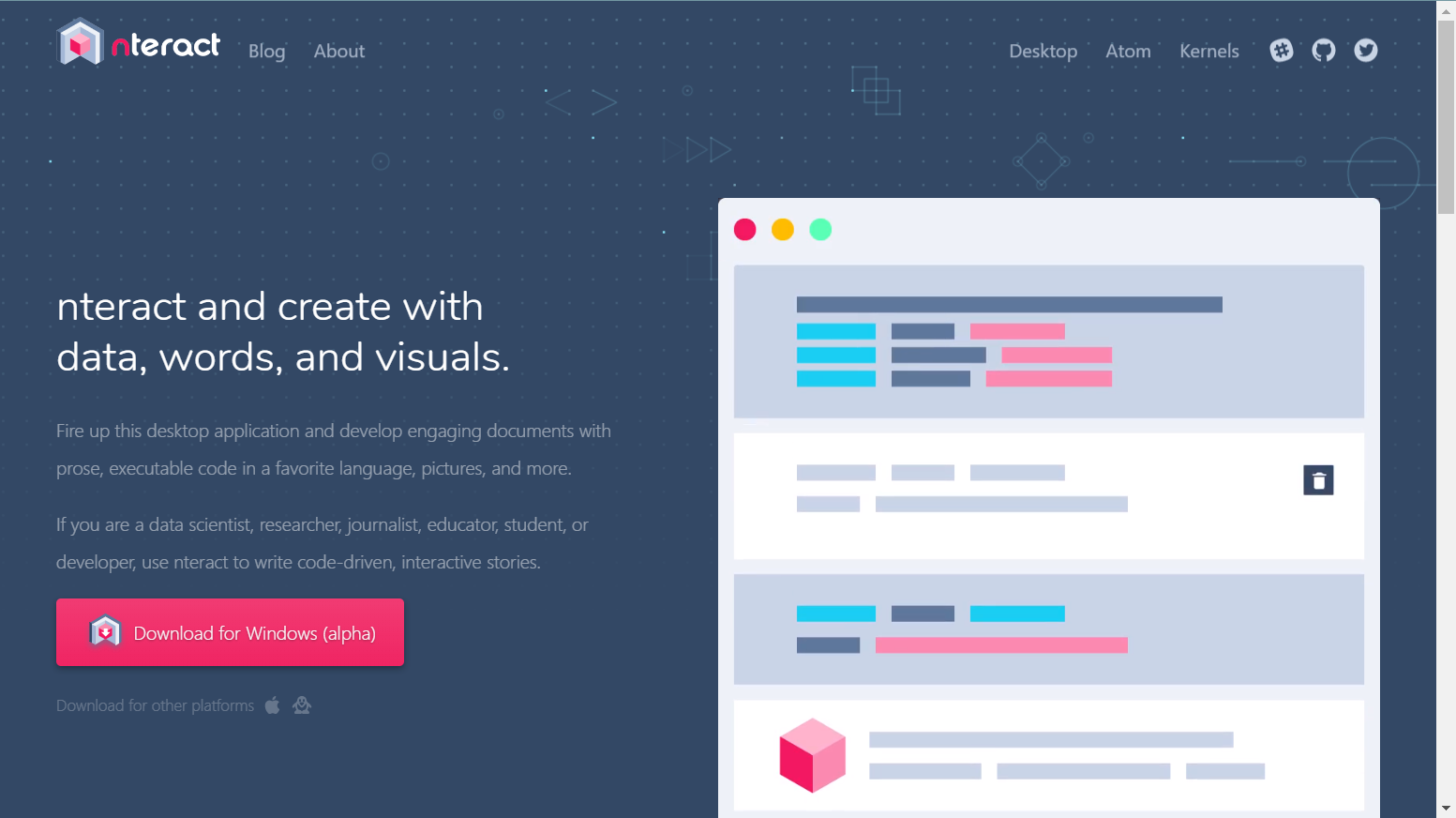


图1-3 nteract网站

#### 1.Anaconda的下载与安装

读者可以到Anaconda官网下载对应操作系统的Python 3.6版本的Anaconda。

（1）Anaconda的Windows版本的安装

安装过程在Windows和Mac系统上跟普通软件相似，一路点击下一步、同意协议等等即可。值得注意的是，需要将Anaconda路径添加到环境变量。Windows系统如下图所示：

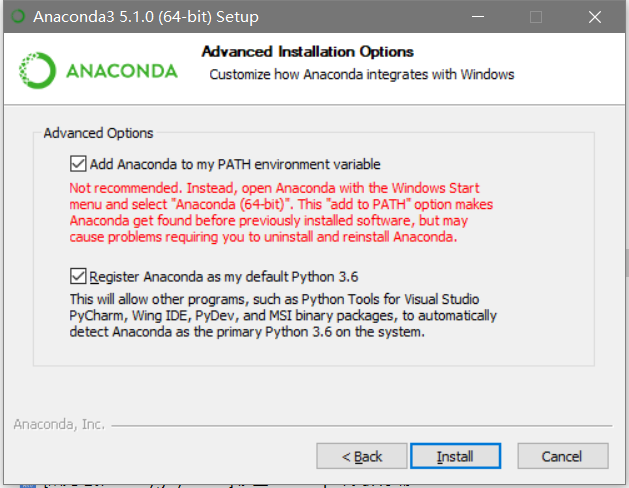


图1-x

（2）Anaconda的Linux版本的安装

Anaconda的Linux版本可以使用命令行方式进行安装（macOS也可以）。下载完Anaconda的Linux版本后，打开文件所在目录并在该目录下打开终端（也可以从其他目录使用cd命令切换）。

然后，输入命令：

# 除了使用浏览器，也可以通过终端下载Anaconda  
# https:// Anaconda官网/archive/Anaconda3-5.1.0-Linux-x86\_64.sh  
  
# 添加执行权限  
chmod u+x Anaconda3-5.1.0-Linux-x86\_64.sh  
# 执行安装  
./Anaconda3-5.1.0-Linux-x86\_64.sh  
  
# 也可以直接使用Bash进行安装  
bash Anaconda3-5.1.0-Linux-x86\_64.sh

然后按照提示按回车键即可，同样注意**要添加路径到环境变量**。

在终端中输入：

jupyter notebook

将会打开Jupyter Notebook（在Windows操作系统中，使用<Windows+R>组合键，输入“cmd”）。默认情况下，浏览器会自动打开，跳转到主页面，然后读者可以新建Python 3 Jupyter笔记本进行学习和使用。

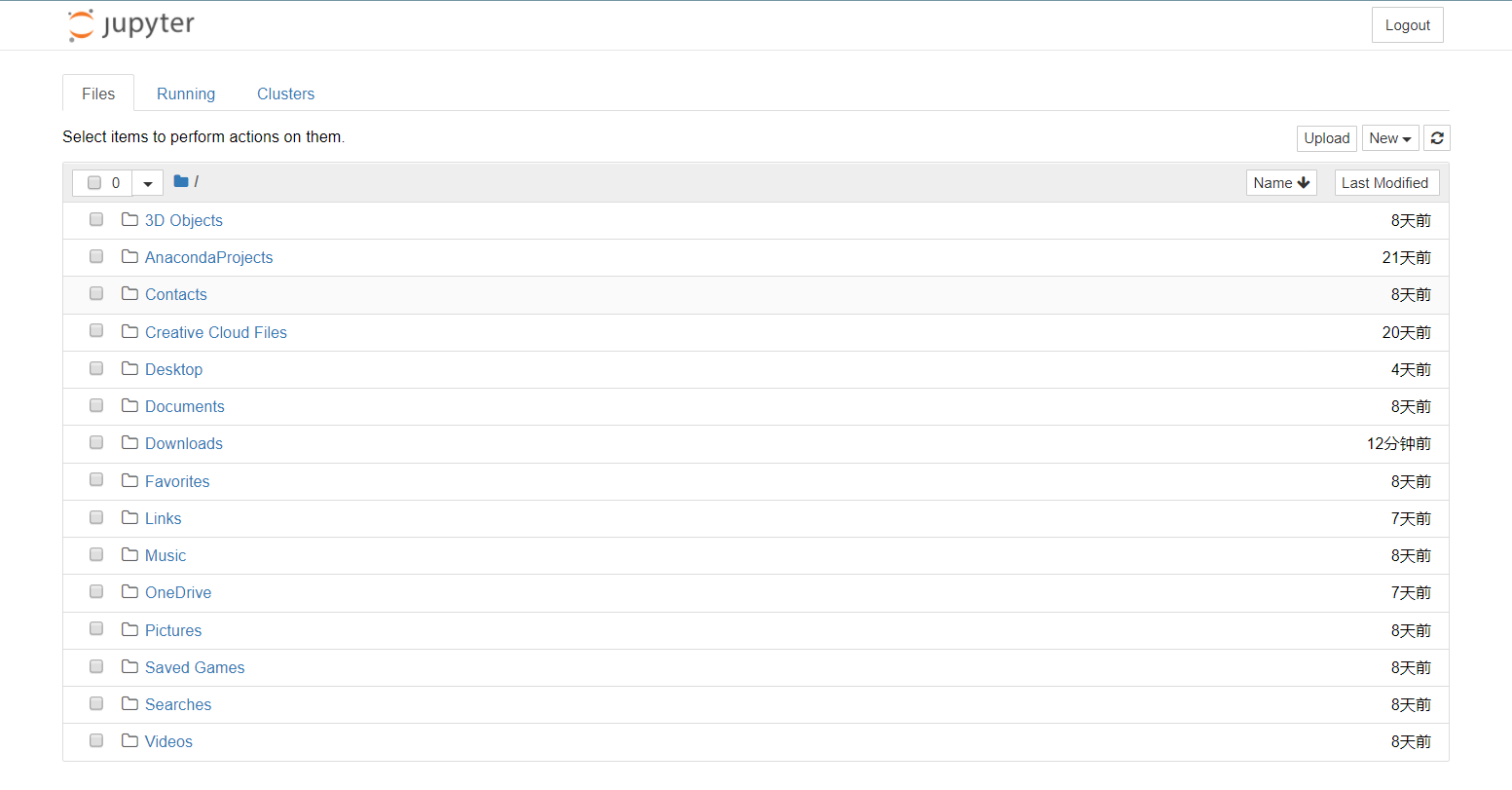


图1-x

如果读者还需要了解Anaconda及其安装，以及Jupyter Notebook的相关内容，那么可以查阅网络上的相关资料。

#### 2.nteract的下载与安装

用户可以到nteract官网下载不同操作系统对应的版本，Windows、Mac与Linux上都可以直接点击安装。安装后直接点击软件图标打开：

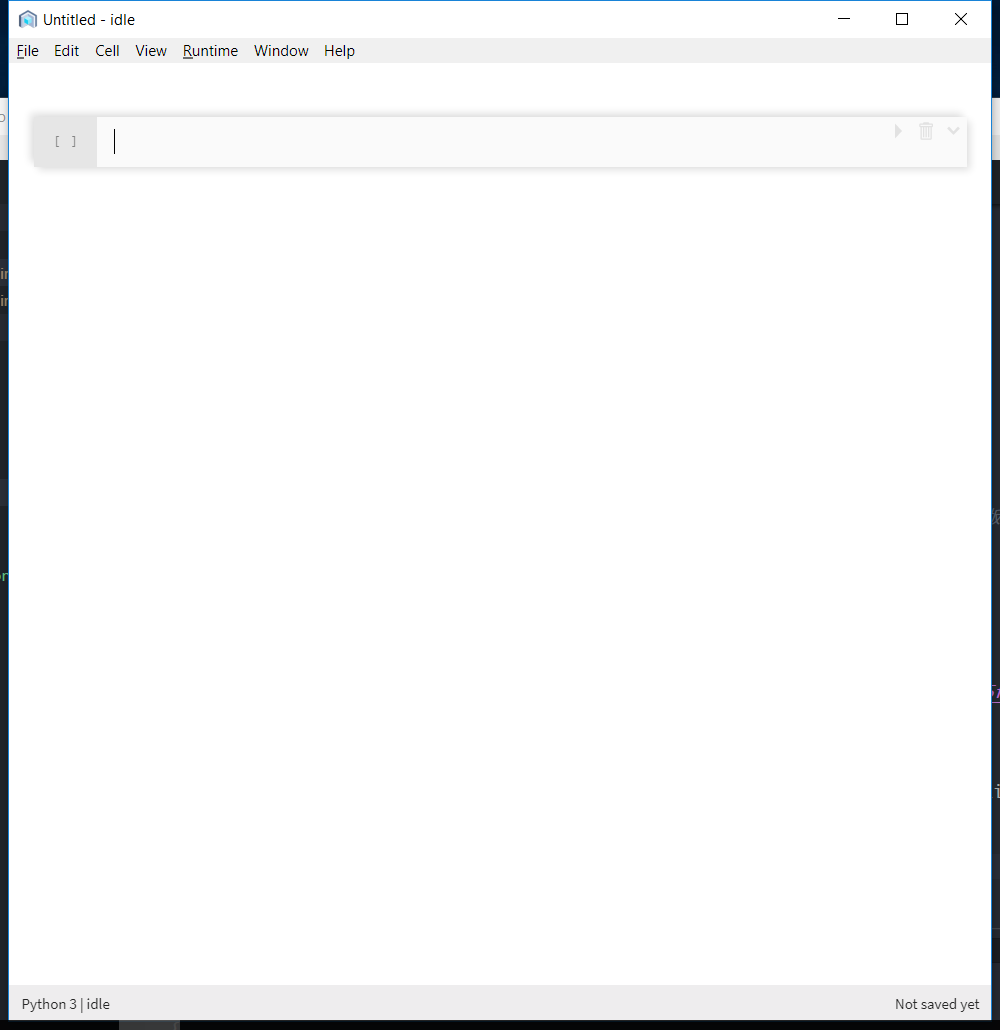


图1-x

单击图1-X菜单栏中的“Runtime”，如果出现Python，则说明可以使用了，同时左下方也会出现“Python 3”标记。如果没有出现上述内容，那么需要检查是否已经成功安装了Anaconda。

如果用户已经成功搭建了Python的使用环境，那么下面就迈出Python数据分析的第一步吧！

# 第2章 Python入门示例及基础知识

本章首先展示Python的一个入门示例，并简单介绍了nteract软件的使用，然后介绍Python的基础知识，包括Python的基本操作符、变量的命名与使用、Python的基本数据类型等。对于基础知识的介绍，本书将通过实例展示、解析其含义并进行延伸。复杂的代码总是可以拆解为基本操作的集合，Python易读易懂的语言特性让这一概念更为形象。本书作者不仅希望读者学会如何编写Python代码、分析数据，而且希望读者编写的代码易懂易维护。

2.1 Python入门示例在很多编程学习书籍或者教学中，第一个代码实现是向屏幕输出“Hello World!”，而本书作为一本中文书籍，下面就实现输出“你好啊，世界！”。

首先，打开nteract，或者使用Jupyter Notebook创建一个笔记本。

在nteract输入框中输入“print('你好啊，世界!')”，然后按下快捷键<Ctrl+Enter>（在macOS系统中，将Ctrl键换为command键即可），输出结果如图2-1所示。

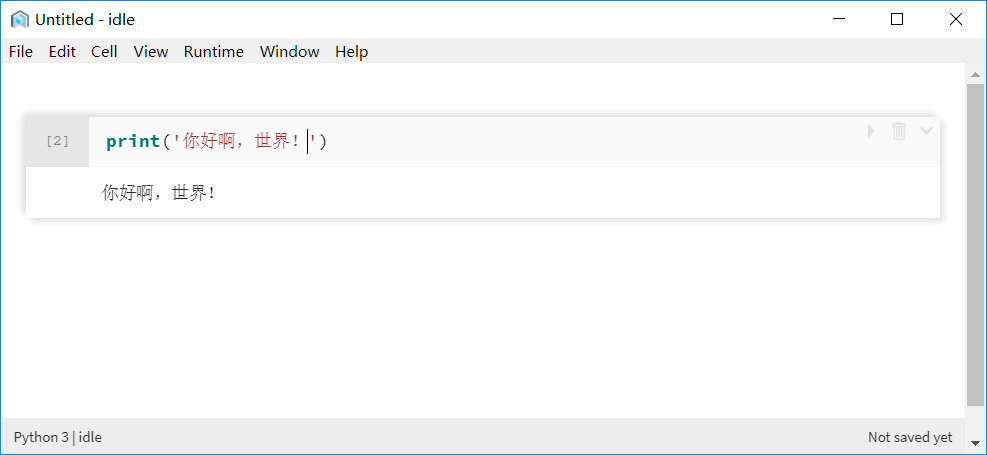


图2-1 示例及输出

在运行程序时，用户还可以单击输入框右上角的三角形图标按钮。

这行代码语句总共包含16个字符，其中7个字符是读者都认识的“你好啊，世界！”，其他9个字符是有些读者暂时还不理解的print、()和''，它们分别表示什么含义呢？

print是Python提供的一个命令，它默认可以向屏幕输出用户定义的字符（串），依赖“()”来区分用户输入的内容中什么是它要处理的对象，而这个对象就是由单引号“''”括起来的文字。

为了证明上述说明的正确性，下面向输入框中输入以下内容进行测试：

print('你好啊，世界！)  
  
print('你好啊，世界')!

结果如图2-2所示。

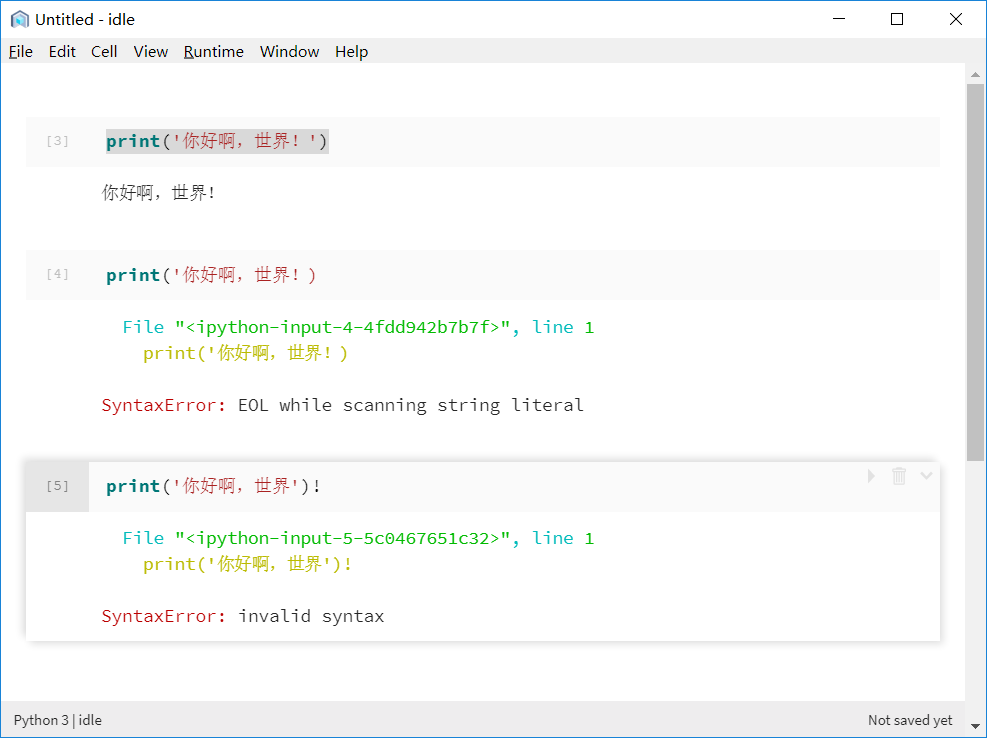


图2-2 错误的示例

显然，上述输入的这两行代码语句存在问题，因为都出现了“SyntaxError”语法错误。

可见，遵守Python的语法规则非常重要。本书后续内容也会时常提及Python的语法规则。。

2.2 nteract软件使用简介通过图2-1和图2-2，读者可以发现，每一个代码和对应的输出结果在nteract软件界面中都单独存在于一个块中，每一块都是一个相对独立的代码单元，这个代码单元可以称为单元格（Cell），如图2-3所示。

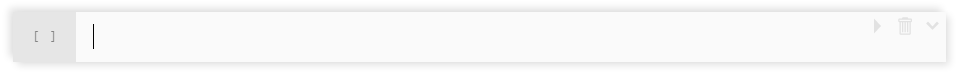


图2-3 单元格

单元格中呈现的所有信息都会被存储为Jupyter笔记本，它以特定的.ipynb作为文件扩展名，如图2-4所示。nteract软件保存的Jupyter笔记本与基于浏览器的Jupyter Notebook创建/保存的笔记本是完全一致的，只是以两种不同的形式展现出来。因此，读者完全可以将用nteract创建的笔记本上传到网络（如开源仓库GitHub）并用Jupyter的nbviewer进行查看，或者上传保存到微软的Jupyter分析平台查阅、编辑和分享。同样，读者也可以将网络上优秀的Jupyter笔记本下载到本地，使用nteract打开，并进行学习和分析。

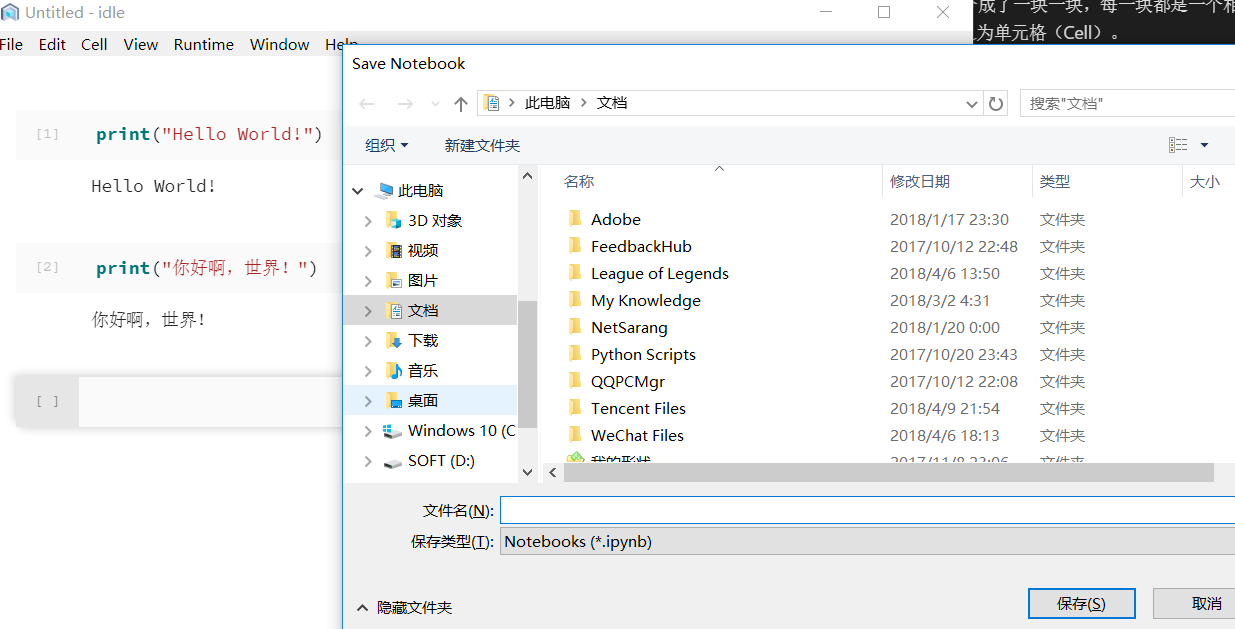


图2-4 保存为Jupyter笔记本

nteract软件除了现在界面语言只有英文以外，其他功能都设计得非常好。用户可以像平常使用Windows系统中的软件一样，用鼠标对单元格进行操作，常见的运行、删除都以图标的方式显示在单元格的右上角，使用起来很方便。习惯于使用快捷键的用户，还可以通过菜单栏中的各个选项了解快捷键的使用。

在运行代码时，常见的两个快捷键是<Ctrl+Enter>及<Shift+Enter>，前者运行当前单元格；后者先运行单元格，然后新建一个单元格。相对而言，后者更为便利。

注意，nteract中每个单元格左侧都有“[]”标记，在代码运行后，中间会出现数字，标记了单元格运行的次序。

## 2.3 算术运算操作符简介

本节介绍Python中的算术运算操作符，并给出计算身体质量指数的示例。

#### 1.算术运算操作符

在完成第一个Python代码后，对数据分析感兴趣的读者是不是想知道Python是如何实现数据计算的呢？下面从简单的四则计算开始介绍Python中的算术运算操作符，如图2-5所示。



图2-5 Python中的四则运算

图2-5中的代码语句只有数字1和算术运算操作符，可见+、-、\*和/分别对应日常的四则运算符号。1 / 1的结果怎么是1.0？不应该是1吗？

在Python的除法运算中，除了使用符号“/”，还会使用“//”，结果如图2-6所示。。另外，Python中还提供求余操作符%。

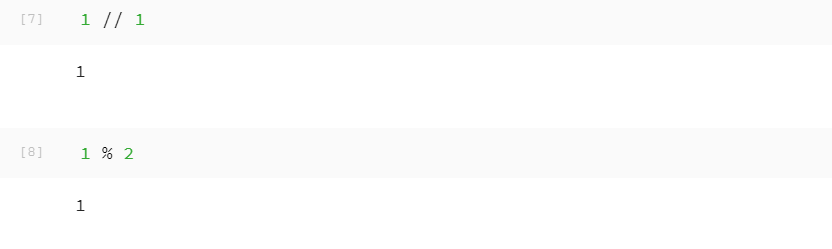


图2-6 XXXXX

Python中比较完整的算术运算汇总如下：

3 + 2 # 加法  
3 / 2 # 浮点数除法  
3 // 2 # 整除  
3 \* 2 # 乘法  
3 \*\* 2 # 指数  
3 % 2 # 求余  
abs(a) # 绝对值

#### 代码约定

通过上面的例子向读者简单介绍了如何利用nteract创建和使用Jupyter Notebook并用它输入Python代码。为了提高代码书写和利用的效率，接下来使用文本的形式输入代码和展示结果。

例如，关于图2-5中的几个运算操作，本书使用下列代码行代替。

In [3]: 1 + 1  
Out[3]: 2  
  
In [4]: 1 - 1  
Out[4]: 0  
  
In [5]: 1 \* 1  
Out[5]: 1  
  
In [6]: 1 / 1  
Out[6]: 1.0

注意： 代码左侧的[3]、[4]等标定了Python运行代码的次序，这并不是指必须在[3]运行后才能运行[4]。本书将会在后面的例子中稍加说明。

#### 2．计算身体质量指数

上文介绍了如何使用Python进行简单的计算，那么现在把所学的知识用来做一件更有意义的事情——计算自己的身体质量指数。

身体质量指数（BMI）是目前国际上常用的衡量人体胖瘦程度以及是否健康的一个标准，相信不少读者在进行体检时都会测量这个指标，它是用体重公斤数除以身高米数的平方得出的数字。也就是说，只要知道自己的身高和体重，就可以自己计算BMI。

例如作者本人，身高1.82m，体重70kg，计算后的BMI指数为：

In [11]: 70 / 1.82 \*\* 2  
Out[11]: 21.132713440405748

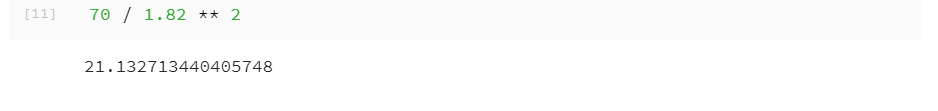


图2-x

有时，当用户将自己编写的代码发给其他人时，其他人可能完全不懂代码是什么意思，此时该怎么办呢？

解决的办法是给代码加上注释，这样其他人就能够通过注释来理解这些数学符号所代表的含义了。在Python中，使用符号“#”可以引导一行注释，因此上述代码添加注释后的效果为：

In [12]: 70 / 1.82 \*\* 2 # 计算我的BMI  
Out[12]: 21.132713440405748

注释也可以出现在对应代码的前一行，但不能出现在对应代码的前面，不然代码自身也会被当作注释而无法运行。

In [13]: # 计算我的BMI  
  
In [14]: 70 / 1.82 \*\* 2  
Out[14]: 21.132713440405748  
  
In [15]: # 计算我的BMI70 / 1.82 \*\* 2

标记为[13]和[15]的两行没有对应的out标记，看起来这两行代码并没有被Python执行，因而没有相应的输出结果。实际上，Python会在运行前检查输入的语句块，若发现它是以#开头的语句，那么跳过去（不执行）；如果代码存在语法错误，那么Python会停止运行并给出相应的错误提示， 例如：

In [16]: 计算我的BMI 70 / 1.82 \*\* 2  
 File "<ipython-input-16-b7d966d13e1f>", line 1  
 计算我的BMI 70 / 1.82 \*\* 2  
 ^  
SyntaxError: invalid syntax

为代码添加注释是编写程序时的良好习惯。注释可以使得代码便于追溯并且可以提高程序可读性。良好的代码注释能够帮助自己或他人快速理解和重用程序或代码块，提升编程的效率和能力。

## 2.4 变量简介

### 2.4.1 什么是变量

仅仅使用字面意义上的70、1.82这些数字很快就会引发烦恼，比如当我们与朋友的BMI指数进行比较时，会很快对输入数字感到烦琐，甚至对编写程序失去兴趣。另外，一旦操作涉及较多的运算符，它所展现的意思已经不那么明显了（比如下面的[12]与[10]和[11]进行比较）。

In [9]: # 与朋友比较BMI  
In [10]: 70 / 1.82 \*\* 2  
Out[10]: 21.132713440405748  
  
In [11]: 48 / 1.64 \*\* 2  
Out[11]: 17.846519928613922  
  
In [12]: 70 / 1.82 \*\* 2 - 48 / 1.64 \*\* 2  
Out[12]: 3.2861935117918257

我们需要一种既可以存储信息，又可以对它们进行操作的方法。这正是变量存在的意义。变量的值可以变化，因而用户可以使用变量存储任何信息，如身高、体重、日期、天气情况、手机号码等。变量本身只是计算机中存储信息的一部分内存，为了访问它存储的信息，需要给变量命名。而将信息（数据）存为变量的操作，称为**赋值**。

例如，将自己的身高和体重存储为变量：

In [13]: height = 1.82  
  
In [14]: weight = 70

其中“=”是赋值操作符，注意，它不是表示相等！“height = 1.82”的语义为：将数字1.82赋值给height变量。

可以通过使用print()函数或者直接输入变量名来输出变量保存的信息：

In [15]: print(height)  
Out[15]: 1.82  
  
In [16]: print(weight)  
Out[16]: 70  
  
In [17]: height  
Out[17]: 1.82  
  
In [18]: weight  
Out[18]: 70

因而与朋友比较BMI可以写为：

In [19]: myBMI = 70 / 1.82 \*\* 2 # 我的BMI  
  
In [20]: friendBMI = 48 / 1.64 \*\* 2 # 朋友的BMI  
  
In [21]: myBMI - friendBMI  
Out[21]: 3.2861935117918257

### 2.4.2 变量的命名

变量的命名需要遵循下列规则。

* 第一个字符必须是字母表中的字母（大写或小写）或者一个下画线“\_”。
* 其他部分可以由字母（大写或小写）、下画线“\_”或数字（0~9）组成。
* 变量名称是大小写敏感的。例如，myname和myName不是一个标识符。

i、\_\_my\_name、name\_23和a1b2\_c3是有效的变量名称；2things、this is spaced out和my-name是无效的变量名称。

举个代码的例子：

IIn [22]: \_\_myName = "ShixiangWang"  
  
In [23]: my-name = "ShixiangWang"  
 File "<ipython-input-23-1b106abe1308>", line 1  
 my-name = "ShixiangWang"  
 ^  
SyntaxError: can't assign to operator

给变量命名是编程的基础。新手在处理简单问题时往往会选择a、b、c这样的名字，若有多组变量，可能还会使用a1、a2、a3这样的方式命名。这样简单的命名方式是不推荐的，变量的名字应该具有非常清晰的含义。对于我们现实中的事物，如小狗，你给小狗取一个好名字固然重要，但它并不会改变小狗本身。但变量不是这样的，变量和变量名本质上是同一个事物，因此，变量的好坏就在很大程度上取决于它的命名的好坏。

想要获得好的变量名，注意如下事项。

* 首先需要考虑的事项是，该变量名称要完全、准确地描述该变量所代表的事物。
* 一个好的变量名称通常表达的是“什么”（what），而不是“如何”（how）。通常而言，如果一个变量名字反映了计算的某些方面而不是问题本身，那么它反映的就是“如何”，而非“什么”。譬如，对同样一个变量命名，calcVal比sum更偏向于“how”，因此不提倡。
* 适合的变量名称长度：10~16字符。但更重要的是长度和清晰度之间的平衡，通常较短的名称适用于局部变量或者循环变量，较长的名称适用于很少用到的变量或者全局变量（全局变量与局部变量会在后面的章节介绍）。
* 变量中的计算值限定词：表示计算结果的词，如总额、平均值、最大值等，如果用户想要用类似于Total、Sum、Average、Max、Min、Record这样的限定词来修饰某个名字，那么可以把限定词加到名字的最后，如revenueTotal、expenseAverage等。这样做的目的是将为这一变量赋予主要含义的部分放到最前面，提高可读性。

上述是对《代码大全》中“如何定义一个好的变量名”部分的简单归纳，读者在学习开始时并不一定能完全理解和熟练使用它，但需要记住一个思想：**给变量命名是一件非常重要的事情**。在行动上，每次对变量命名后应当有意识地思考一下该名字是否能准确地描述出变量所代表的事物，如果不能，需要重新取一个新的名字。关于变量的命令，本书推荐一个工具——CODELF。

## 2.5 基本数据类型

“算法+数据结构=程序”，这是1984年图灵奖的获得者Niklaus E. Wirth阐述的一个经典观点。数据的结构常常由多个数据类型组成，其中数字、字符串与布尔值是Python基本的内置数据类型，它们是数据表达、存储的基础。而Python中其他常见的数据类型，如下一章会学习和使用的列表，是基于这3种数据类型的组合构建的新类型。

### 2.5.1 数字

在 Python 中，数字包括 4 种类型：整数、长整数、浮点数和复数。

* 此处所指的整数和数学中的整数是对应的，如1、2、3等。.
* 长整数是指大一些的整数。
* 3.23和5E-4是浮点数的例子。E标记表示10的幂，是一种科学计数法。在这里，5E-4表示5×10-4，即0.0005。
* (-5+4j)和(2.3-4.6j)是复数的例子。

### 2.5.2 字符串

字符串是字符的序列，基本上就是字母、数字或符号的组合。

下面列出字符串的使用方法。

* 可以使用英文单引号“''”指示字符串，如'Quote me on this'，其中所有的空白，包括空格和制表符都照原样保留。
* 也可以使用英文双引号“""”来指示字符串，它的使用方式与英文单引号完全相同，如"What's your name?"。
* 利用三引号“'''”或“"""”可以指示一个多行的字符串。用户可以在三引号中自由地使用英文单引号和英文双引号。 例如：
* '''This is a multi-line string. This is the first line.  
  This is the second line.  
  "What's your name?," I asked.  
  He said "Bond, James Bond."  
  '''

#### 1.转义符

假设用户想要在一个字符串中包含一个单引号（'），那么该怎么指示这个字符串？例如，这个字符串是“What's your name?”。'What's your name?'肯定是错误的使用方式，因为Python 会弄不明白这个字符串从何处开始，何处结束。此时，用户需要告诉Python第2个单引号不是字符串的结尾，这可以通过**转义符**来实现。用户可以用“\'”将英文单引号进行转义，也就是表示为'What\'s your name?'，这样转义符后的英文单引号就只起到单引号的作用了。

另一个方法是单双引号嵌套，如"What's your name?"，此处将英文单引号“'”嵌入英文双引号中，Python会发现英文双引号是字符串的起止符号，因而此处的英文单引号会被正确解析。类似的，要在英文双引号字符串中使用英文双引号本身的时候，也可以借助于转义符。另外，可以用转义符“\”来指示反斜杠本身。

In [26]: 'What's your name?' # 错误的表示方法  
 File "<ipython-input-26-dff8324f3597>", line 1  
 'What's your name?' # 错误的表示方法  
 ^  
SyntaxError: invalid syntax  
  
  
In [27]: 'What\'s your name?' # 使用转义符\对字符串中的英文单引号进行转义  
Out[27]: "What's your name?"  
  
In [28]: "What\'s your name?" # 将英文单引号嵌入英文双引号中  
Out[28]: "What's your name?"

值得注意的是，在一个字符串行末，单独一个反斜杠表示字符串在下一行继续，而不是开始一个新的行。例如：

"This is the first sentence.\  
 This is the second sentence."

等价于"This is the first sentence. This is the second sentence."。

#### 2.自然字符串

如果想要表示某些不需要如转义符那样特别处理的字符串，那么可以指定一个自然字符串。自然字符串通过给字符串加上前缀r或R来指定。

例如r"Newlines are indicated by \n"，其中“\n”是文本的换行符，读者需要理解下面结果的不同：

In [35]: "Newlines are indicated by \n"  
Out[35]: 'Newlines are indicated by \n'  
  
In [36]: r"Newlines are indicated by \n"  
Out[36]: 'Newlines are indicated by \\n'  
  
In [37]: print(r"Newlines are indicated by \n")  
Out[37]: Newlines are indicated by \n  
  
In [38]: print("Newlines are indicated by \n")  
Out[38]: Newlines are indicated by  
# 此处输出一个空行

#### 3.Unicode 字符串

Unicode是书写国际文本的标准方法（也称为万国码）。在Python中，只需要在字符串前加上前缀u或U就可以处理Unicode字符串，如u"This is a Unicode string."。需要记住，在用户知道文本中含有非英语写的文本时，最好使用Unicode字符串。

### 2.5.3 布尔值

任何一门编程语言都会存在布尔类型，用来表示“真”和“假”。这不仅仅是因为计算机的基础是二进制，而且我们人类对于事物的基本判断也是二元的，往往不是“真”就是“假”。

在Python中，True代表“真”，False代表“假”。注意，Python是大小写敏感的语言，false不等同于False。

### 2.5.4 使用type()函数查看数据类型

上文已经介绍了Python的3个基本数据类型，那么该如何知道一个变量属于哪一种类型呢？

例如，有下列进行赋值的4个变量：

In [45]: type1 = 1  
  
In [46]: type2 = 1.0  
  
In [47]: type3 = "1"  
  
In [48]: type4 = True

下面就可以使用type()函数查看变量的数据类型：

In [49]: type(type1)  
Out[49]: int  
  
In [50]: type(type2)  
Out[50]: float  
  
In [51]: type(type3)  
Out[51]: str  
  
In [52]: type(type4)  
Out[52]: bool

结果中的int、str与bool分别是integer（整数）、string（字符串）与boolean（布尔值）的缩写。

## 2.6 数据运算

上一节介绍了Python的基本数据类型，但除了数学运算，我们还没有实际演示其他数据类型是如何操作的，以及这些基本的数据类型如何相互转换和配合使用。

我们已经知道数字1+1的结果是它们的和，字符'1'+'1'的结果又是什么呢？

In [2]: '1' + '1'  
Out[2]: '11'

上述代码中的“+”可以将两个字符进行连接。如果读者想尝试一下“-”，那么可能会有点失望——它并不能从一个字符串中去除另一个字符：

In [3]: '1' - '1'  
---------------------------------------------------------------------------  
TypeError Traceback (most recent call last)  
<ipython-input-3-3f89fba82f3e> in <module>()  
----> 1 '1' - '1'  
  
TypeError: unsupported operand type(s) for -: 'str' and 'str'

可是，为什么“+”既可以求数字的和，又能连接字符呢？这里就必须提到一个称为**泛型**的概念：它就像一个黑箱子，比如这里使用的“+”，虽然只能看到外表标记了一个“+”，但它会根据不同的输入数据类型（如数字或字符）执行不同的命令。

我们现在知道使用type()函数可以判断'1'是字符1还是数字1，但该如何将字符1变成数字1呢？这就需要将str类型转换为int类型，Python恰好有相应的函数可以完成这样的操作。

In [8]: type('1')  
Out[8]: str  
  
In [9]: type(int('1'))  
Out[9]: int

除此之外，读者需要注意字符串与数字不能够进行连接（+）操作，必须将一方的类型转换为另一方的类型，不然会抛出语法错误。

In [10]: "我的语文和数学成绩之和是 " + 199  
---------------------------------------------------------------------------  
TypeError Traceback (most recent call last)  
<ipython-input-10-17fa10d93550> in <module>()  
----> 1 "我的语文和数学成绩之和是 " + 199  
  
TypeError: must be str, not int  
  
In [11]: "我的语文和数学成绩之和是 " + str(199)  
Out[11]: '我的语文和数学成绩之和是 199'

这里使用str()将数字199强制转换为字符。相似的转换函数还有int()、float()及bool()。

Python常用的内置操作符包括**算术运算操作符、字符串操作符、比较操作符与布尔逻辑操作**符，汇总如下，并给出一些简单的示例，读者可自行运行示例并仔细体会。

1）算术运算操作符：

9 + 2 # 加  
9 - 2 # 减  
9 \* 2 # 乘  
9 / 2 # 除（浮点输出）  
9 //2 # 整除  
9 % 2 # 求余  
9 \*\*2 # 幂

2）字符串操作符：

'这是一个' + '字符串' # 字符串连接  
'这是一个字符串' \* 5 # 字符串重复

3）比较操作符：

5 == 4 # 等于  
5 > 4 # 大于  
5 < 4 # 小于  
5 != 4 # 不等于  
5 >= 4 # 大于或等于  
5 <= 4 # 小于或等于

4）布尔逻辑操作符：

True and True # 逻辑“与”  
True or False # 逻辑“或”  
not False # 逻辑“非”

## 2.7 本章对应的线上笔记

为帮助读者巩固和拓展所学知识，本书提供了线上的笔记仓库，读者可通过。本章笔记对应线上笔记仓库中的Python第一步，读者可以复制或下载进行学习。