Data Science and Analytics with Python

用Python进行数据科学与分析

1. Trials and Tribulations of a Data Scientist

数据科学家的试验与磨难

The ever increasing availability of data requires the use of tools that enable businesses and researchers to draw conclusions and make decisions based on the evidence provided by the data itself. From performing a regression analysis to determining the relationship between data features, or improving on recommendation systems used in e-commerce, data science and analytics are used every day by all of us. This book is intended to provide those interested in data science and analytics a perspective into the subject matter using Python, a popular programming language available for various platforms and widely used both in business and academia.

工具的使用使得数据能够不断发挥越来越大的价值，帮助企业和研究人员基于数据本身提供的证据来做出结论和决定。从进行回归分析到确定数据特征之间的关系，或用于改进电子商务所用的推荐系统，我们每个人每天都会用到数据科学和分析。本书旨在为那些对数据科学和分析感兴趣的人提供一个使用Python的视角来了解主题，Python是一种流行的编程语言，可用于各种平台，并广泛应用于商业和学术界。

In this chapter we will cover what data science is and how it related to various disciplines from mathematics to business intelligence and from programming to design. We will discuss the characteristic that make a good data scientist and the composition of a data science team. We will also provide an overview of the typeical workflow in a data science and analytics project and shall see the trials and tribulations in the work cycle of a data scientist.

在本章中，我们将介绍什么是数据科学，以及它与从数学到智能商业，从编程到设计的各个学科之间的关系。我们将讨论如何成为一个好的数据科学家和构建一个数据科学团队。我们还将概述数据科学和分析项目中的典型工作流程，并将以此感受到数据科学家在工作中的所遇到的各种挑战和磨难。

* 1. Data? Science? Data Science!

数据？科学？数据科学！

The use of Data as evidence in support for decision making is nothing new. You only have to take a look at the original meaning of the word statistics as the analysis and interpretation of information relating to states such as economic and demographic data. Nowadays, the word statistics is either understood as a branch of mathematics that deals with the collection, analysis, interpretation and presentation of data; or more colloquially as a fact or figure obtained from a study based on large quantities of data. Simply take a look at the news on any given day and you will surely get to hear about statistics, proportions and percentages, all in support (or not) of a new initiative, plan or recommendation. The power of data is all around us and we use it all the time.

使用数据作为支持决策的证据并非什么新鲜事。“统计学”的原始本意，就是分析和解释与国家有关的信息，如经济和人口数据。现在，“统计学”被理解为数学的一个分支，它涉及数据的收集、分析、解释和呈现；或者更通俗地说是基于大数据得研究从而获得某些事实或直观的数据呈现。只要看一看任意某天的新闻，你一定会听到有关统计数字、比例和百分比的消息，关于支持（或不支持）某项新举措，计划或者提案。由此可见，数据的力量无处不在，我们一直在使用它。

Now, what about the word science? Well, you may remember from your school days that science is system that enables the organization of knowledge, based on testable evidence and predictions. Notice that key word evidence mentioned there again.

那么，什么是“科学”？好吧，你可能还记得在你上学的时候老师教的定义：科学一个[建立](https://baike.so.com/doc/6247046-6460455.html)在可检验的证据和(对[客观事物](https://baike.so.com/doc/1292750-1366818.html)进行)预测的知识系统。注意这里又提到了关键词-“证据”。

No surprises here so far, right? From a very simplified point of view, the scientific method makes use of data and their analysis to acquire, correct and integrate knowledge. Nonetheless, data science is not just simply the direct use data science of statistics, or the systematization of data. How shall we understand that much loved combination of the words data and science?

到目前为止，并没有什么让人吃惊的，对吧？从简单的角度来看，科学的方法是利用数据及其分析来获取、修正和整合知识。然而，数据科学不仅仅是简单的统计数据或直接分类数据那么简单。那么我们应该如何理解“数据+科学”？

* + 1. So, what is Data Science?

因此，到底什么才是数据科学？

Data science and analytics are rapidly gaining prominence as some of the more sought after disciplines in academic and professional circles. In a nutshell, data science can be understood as the extraction of knowledge and insight form various sources of data, and the skills required to achieve this range from programming to design, and from mathematics to storytelling.

数据科学和分析在学术和专业领域中正在迅速地获得突出地位。简而言之，数据科学可以理解为从各种数据来源中获取知识和相关洞察力，实现这一目标需要各种技能：从编程到设计，从传统数学到如何生动的讲述该故事(数据)。

There is no doubt that the term data science is a true neologism of our time. The term has started being used and, to a certain extent, even abused. As we have mentioned before data science is rather more than the sum of data on the one hand and science on the other one, although it is inevitably related to both concepts.

毫无疑问，数据科学一词是我们这个时代特有的新词。然而，这个词已经开始被使用，甚至在某种程度上被滥用。正如我们前面提到的，数据科学比数据和科学的涵盖面要广，尽管它不可避免地涉及到这两个概念。

Currently, data science can be considered a budding field with applications in a wide range of areas and industries, as well as in academic research. It is fair to say that it is elusive to define this emerging field, and throughout this book we shall consider data science and analytics as a portmanteau for a number of overlapping tasks related to data-from collection, provision and preparation, analysis and visualization, curation and storage-that exploit tools from empirical sciences, mathematics, business intelligence, machine learning and artificial intelligence. The aim of these tasks is to enable effective, pragmatic and most importantly actionable decisions.

目前，数据科学可以被认为是一个萌芽领域，广泛应用在各个行业，以及学术研究领域。可以公平地说，这个新兴领域的定义是难以捉摸的，在本书中，我们将把数据科学和分析看作是与数据相关的一系列任务的叠加：如数据收集，提供和准备、分析和可视化、管理以及储存，然后利用经验科学、数学、商业智能、机器学习和人工智能来开发相关工具。最终通过这些任务的形成有效、务实以及可采取具体行动的决定。

The motivation for data science and analytics in deriving valuable insights from data is great, and widely welcomed by businesses. However, this is a very challenging task. Companies such as Google, Netflix and Amazon have demonstrated that careful storage and analysis of data delivers a very competitive edge. These days there are easier and cheaper ways to collect large amounts of data than ever before, and mobile is becoming a ubiquitous presence. This have allowed companies, particularly start-ups, to develop in-house data science capabilities.

对于企业来说，如何应用数据科学和分析在数据中获得有价值的洞察力的商机巨大，并且受到广泛欢迎。然而，这是一项非常具有挑战性的任务。谷歌(Google)、Netflix和亚马逊(Amazon)等公司已经证明，对用户数据进行存储和分析将提升品牌的竞争力。如今，收集大量数据的方法比以往任何时候都更容易、更便宜，并且移动设备正在成为一种无处不在的存在。这使得公司，特别是初创企业，具备了能够在家就开发数据科学的能力。

Typical examples of data science products are better explained by the questions they aim to answer; these questions are the drivers to the acquisition and selection of the appropriate data to be interrogated in order to provide insight into an area of interest. I am sure you can come up with a few of examples relevant to you, but there are some that come to mind:

* What product will sell better in conjunction with another popular product?
* Who will be declared Prime Minister (or President, or winner; depending on the flavor of the government system of interest) in the next general election?
* How can customers be encouraged to spend a longer time in an online portal?
* Are there any discernible patterns that allow us to characterize different groups of sales agents, customers or businesses?

|  |  |
| --- | --- |
| * What advertisement should be placed on what site? | Advertising and marketing |
| • Given the interests of a customer, what other products can be recommended to them? | Recommendation systems |
| • What are the latest developments and breakout reports in newspapers and social media that may affect the industry of interest? | Social media analysis |
| • Given someone’s interests and hobbies, who may be suitable potential partners? | Online services |
| • How can we keep potentially sensitive information protected and react proactively to information we store? | Cybersecurity |
| • How can we distinguish valid, relevant documents such as emails (ham), from invalid, irrelevant ones (spam)? | Classification analysis |
| • How to determine if a retail transaction is valid or not? | Fraud prevention |
| • What is the demand for a particular service at a particular time or place? | Demand forecasting |

典型的数据科学产品的例子将更好地解释他们(企业)所想要回答的问题; 这些问题是获取和选择要询问的适当数据的驱动因子，以便对该领域提供有价值的洞察。我相信你也可以拿出一些相关例子，下面是我所想到的：

* 哪种产品与另一种受欢迎的产品结合起来会卖得更好？
* 下一届谁将当选为总理(或称为总统以及胜利者，基于政府利益体系的价值最大化)？
* 如何吸引客户在线门户网站上花更长时间？
* 是否有任何可识别通用模型, 使我们能够描述不同的销售代理、客户或企业组？
* 如何根据广告类型精准投放到对应的网站上？
* 如何根据客户的兴趣爱好，推荐其相关产品？
* 在报纸和社交媒体上有哪些最新报道和大新闻可能会对相应的行业产生相关影响？
* 考虑到某人的兴趣和爱好，谁可能是其合适的潜在伴侣？
* 我们如何保护自己潜在的敏感信息，并对所存储的信息作出积极的反应？
* 我们如何区分有效的文件？(例如在电子邮箱中从无效的、无关紧要的垃圾邮件甄选出有价值的邮件)
* 如何确定零售交易记录是否有效？
* 在特定的时间或地点, 对某一特定服务的需求是什么？

These are not questions that decision-makers, businesses and industries, large and small, have recently started formulating. So, why the resurgence in seeking answers to them? The main answer is the availability of potentially useful data, big or small, together with the impact of technology, computer science and statistics in everyday life.

这些都是决策者、企业和行业(无论大小)长久以来所思考的问题。那么，我们为什么要重新寻找答案呢？主要的答案在与我们想知道相关数据(无论大小) 进行计算机科学和技术统计，如何对日常生活产生影响。

Out of the ingredients mentioned above, accessible datasets may be the most important one since without them the insight provided by technology alone is rather limited. After all, the plural of anecdote is not data. Having said that, it is important to note that this does not mean that every single data science case to be tackled falls into the category of so-called big data, particularly when we take into account that the adjective big can be used in a relative manner. We shall expand on this point later on in Section 1.3.1.

在上述要素之外，可访问的数据集可能是最重要的数据集，因为没有这些数据集，技术本身所提供的洞察力就相当有限。毕竟，逸闻趣事道听途说并非真实数据。尽管如此，需要重点注意的是，这并不意味着每一个(要处理的)数据科学案例都属于所谓的大数据范畴，特别是当我们考虑到形容词Big可以是个相对的概念时。我们稍后将在第1.3.1节中详述这一点。

One important thing to bear in mind about the outputs of data science and analytics is that in the vast majority of cases they do not uncover hidden patterns or relationships as if by magic, and in the case of predictive analytics they do not tell us exactly what will happen in the future.

关于数据科学和分析的结论，需要记住的一件重要的事情是，在绝大多数情况下，它们不会像魔术一样揭示数据里面隐藏的模式或关系，而在预测分析的情况下，它们不会确切地告诉我们未来会将发生什么。

Instead, they enable us to forecast what may come. In other words, once we have carried out some modelling there is still a lot of work to do to make sense out of the results obtained, taking into account the constraints and assumptions in the model, as well as considering what an acceptable level of reliability is in each scenario

相反，它们使我们能够预测将来也许会发生什么。换句话说，一旦我们进行了一些建模，考虑到模型中的约束和假设，以及考虑每个场景中可接受的可靠性水平，仍然有许多工作要做，以便理解所得到的结果。

Similarly, there is the tacit prerequisite of having accurate, timely data that can be readily utilized to make sense out of the modelling results, and reflect the state-of-the-art in an application. It is therefore imperative that decision makers as well as IT and business stakeholders take time to understand the information that will be needed, as well as being prepared to realize that certain data may not be fit for their purpose. It is indeed disheartening to come to terms with the fact that some data may not have the necessary features to be used in building a prediction, for example. Nonetheless, it is better to realize that is the case at an early stage, rather than relying on unsuitable results to make important decisions that impact the business.

同样，拥有准确、及时的数据也是一个必要的前提，这些数据可以很容易地被用来解释建模结果，并实时反映出应用程序的状态。因此，决策者以及IT和业务利益攸关方必须花时间了解所需要的信息，并做好准备:某些数据可能不适合。事实上，令人沮丧的是，有些数据可能缺乏必要的特性来建立预测模型。然而，最好在建模之前就认识到这一点，而不是依靠不适当的结果输出来做出影响企业的重要决定。

P43:

Even if data science may not yet be considered a well-defined subject, the number of academic and training programmers being offered by universities and at various workplaces has seen a healthy increase. This is a natural result of the need that exists for well-informed, capable experts that we get to call data scientists. So what do data scientists do and what do they look like? It will all shall be uncovered.

虽然数据科学可能还没有被认为是一个明确定义的学科，但大学相关的学术研究和很多企业提供的培训程序员的数量却已经出现了健康的增长。这是一个自然而然的结果，因为我们需要信息灵通的、有能力的专家，我们可以把这些专家称为数据科学家。那么，数据科学家究竟是做什么的，他们长什么样？一切谜底都将在本书中被揭开。

*1.2 The Data Scientist: A Modern Jackalope*

*数据科学家：现代鹿角兔*

The new term used to describe the person that deals with the seemingly disparate array of tasks described above may seem to be yet another, more fashionable way to describe a statistician or a business analyst. However, we can certainly agree that there is a gap between the latter two, and that the skills required by a data scientist involve aspects that include both statistics and a strong business acumen, but also foundations in computer science, mathematics, modelling and programming, not to mention good communication skills. A simplified diagram of these skills and their relationship is shown in Figure 1.1.

用来描述上述一系列看似不相关任务的人的新术语（现代鹿角兔）似乎是描述统计学家或业务分析师的另一种更时尚的方式。然而，我们当然必须承认统计学家或者说业务分析师和数据科学家存在差距，一名数据科学家所需的技能所涉及的方面不但包括统计和强大的商业头脑，而且还需要计算机科学、数学、建模和编程方面的基础，更不用说良好的沟通技巧。图1.1显示了这些技能及其关系的简化图。

In that sense a data scientist role goes beyond the collection and reporting on data; it must involve looking at a business application or process from multiple vantage points and determining what the main questions and follow-ups are, as well as recommending the most appropriate ways to employ the data at hand.

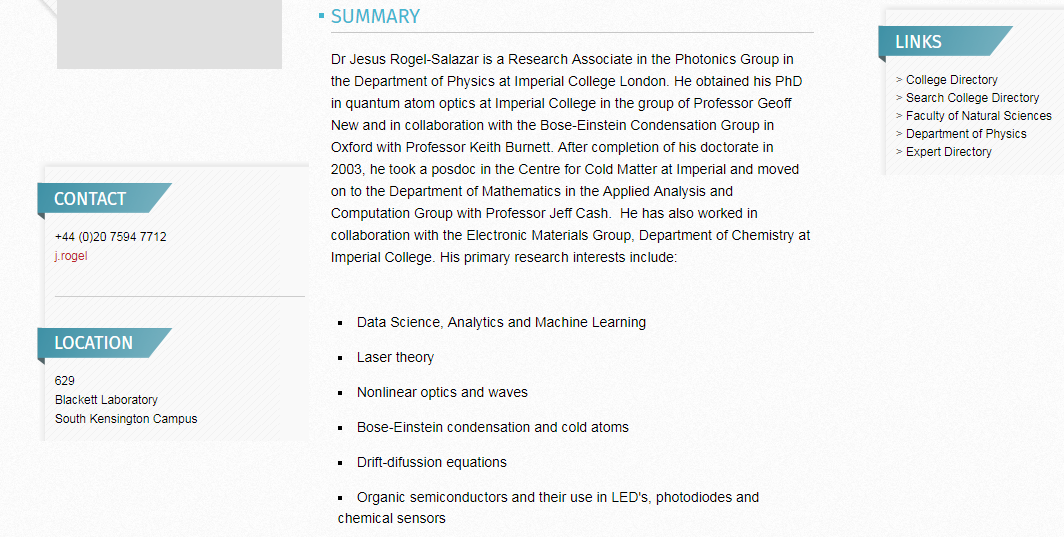
从这个意义上说，数据科学家的角色不仅仅是收集和报告数据；它必须包括从多个有利的角度看待业务应用程序或流程，确定主要问题和后续行动，以及如何使用手头数据的最佳实践。

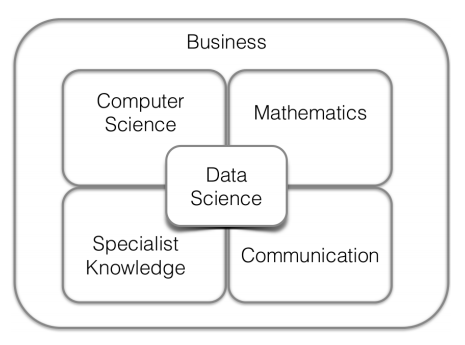
P44

8 j. rogel-salazar

<https://www.researchgate.net/profile/J_Rogel-Salazar2> ？？

<http://www.imperial.ac.uk/people/j.rogel> ？？





In terms of characteristics, a data scientist has an inquisitive mind and is prepared to explore and ask questions, examine assumptions and analyze processes, test hypotheses and try out solutions and, based on evidence, communicate informed conclusions, recommendations and caveats to stakeholders and decision makers.

就特征而言，数据科学家有一种好奇的头脑，随时准备探索和提出问题，审查假设和分析过程，测试假设和尝试解决办法，并根据结论向利益攸关方和决策者传达知情的结论、建议和警告。

In other words, a data scientist is a true new Renaissance woman or man. No wonder that despite being branded the sexiest job of the 21st century, as well as the increasing demand for these individuals, it is hard to find people with the right skills to fill in these roles. This has led to branding data scientists as Unicorns.

换句话说，数据科学家是真正的文艺复兴时期的女性或男性。难怪尽管被评为21世纪最时髦的工作，以及对这些人的总体需求不断增加，但很难找到具备适当技能的人来填补这些职位。这导致数据科学家被称为独角兽(Unicorns)。

To a certain extent, the symbolism of a Unicorn as a creature that is beautiful, mysterious and difficult to tame or even capture may be applicable to describe a data scientist. However, in my opinion, it may not be totally appropriate given the fact that, as majestic as Unicorns can be, they are way too common as far as popular culture goes.

独角兽作为一种美丽、神秘、难以驯服甚至难以捕捉的生物来说，其象征意义在某种程度上可以用来描述数据科学家。然而在我看来，独角兽更具有一种威严庄重感，而数据科学家就流行文化而言，更加普遍通俗一些，因此可能并非完全合适。

(Branding data scientists as Unicorns is a result of the quixotic expectations businesses and industries have and thus is not appropriate.)

(将数据科学家冠以“独角兽”的称号其实是商业和工业对其所抱有不切实际的期望的结果，因此是并不合适。)

The shortage that businesses experience when trying to attract data scientists is more likely due to the fact that they have created internal expectations for the role and that no single individual can fulfil, thus appealing to the magical nature of a common mythical beast. They have created their idea of the Data Scientist Unicorn, and unfortunately the fascination prevails.

企业试图吸引数据科学家的原因往往是因为他们的缺陷，在于该角色他们创造了内在的期望值，目前没有任何个体能够达到该期望，因此寄托于神话怪兽独角兽般存在的数据科学家能够做到这些。不幸的是，这样想法的企业非常普遍。

To tackle the prevailing image, I am convinced that the use of a new symbol is needed. And a silly one at that! There is an allegory I usually propose to colleagues and those that talk about the data science Unicorn. It seems to me to be a more appropriate one than the existing image: It is still another mythical creature, less common perhaps than the Unicorn, but more importantly with some faint fact about its actual existence: a Jackalope. You can see an artistic rendition of a couple of Jackalopes in Figure 1.2.

为了解决这一流行现象，我坚信需要使用一个新形象来代表数据科学家。真是愚蠢的家伙！我通常向同事说那些谈论数据科学是独角兽的人。新形象在我看来，它似乎比现存的更合适：它仍然是神话中的生物，但并不像独角兽那么常见，更重要的是，基于一些模糊的事实，他又似乎曾经真实存在过：这就是鹿角兔。你可以在图1.2中看到一对鹿角兔(Jackalopes)的艺术再现。

( The Jackalope is the one we propose.)

我们建议的形象就是鹿角兔

A Jackalope is said to be a strange beast that looks like a jackrabbit with a pair of stag horns. It is described to be a shy but clever and cunning animal, and if threatened it can be dangerous. If you are ever in the Mountain West in the United States you may stumble into Jackalope heads mounted as trophies; but of course that is not the only place where Jackalopes are endemic; there are tales of the Hasenbock in Austria1 or you may hear the Huichol stories about how the tátSu (rabbit) lost its antlers to the Colorado kaukamali (deer)2

鹿角兔(Jackalope)据说是一种奇怪的野兽，看起来像一只长着一对鹿角的野兔子。它被描述为一种羞怯但聪明又可爱的动物，当然如果受到威胁，它也可能变得很危险。如果你在美国的西部山区，你可能会跌倒在Jackalope的头上，作为奖杯；当然，这并不是Jackalopes所特有的地方；在奥地利的哈森博克也有鹿角兔的故事，甚至在维乔人那里你也可以听到兔子在科罗拉多失去了鹿角的故事

（Figure 1.2: Jackalopes are mythical animals resembling a jackrabbit with antlers.）

图1.2：鹿角兔是神话中的动物

1 Toelken, B. (2013). The Dynamics of Folklore. University Press of Colorado

1托尔肯，B（2013）。动态民俗学。科罗拉多大学出版社

2 Zingg, R., J. Fikes, P. Weigand, and C. de Weigand (2004). Huichol Mythology. University of Arizona Press

2 Zingg，R .，J . Fises，P . Weiand，and C . de Weigrd（2004）。维乔人的神话故事 亚利桑那大学出版社

(A Jackalope is a mythical being similar to a jackrabbit with a pair of stag horns)

鹿角兔是一种神话般的动物，类似于一头长着一对鹿角的野兔子

No need to explain that a Jackalope is indeed an imaginary, mythical being, much like the Unicorn, but it seems to be a better metaphor for the data scientist. We can argue that it is rather difficult at best, and impossible at worst, getting hold of a single individual that is able to be an all rounded ninja programmer, with vast expertise in mathematics, statistics and probability, plus knowledge of computer science and well-versed in business. This offers no solution to businesses interested in getting the benefit of exploiting the available data.

无需解释，Jackalope确实是也一个虚构的、神话般的存在，这点很像独角兽，但对于数据科学家来说，它似乎是一个更好的比喻。因为对于我们来说，要抓住一个能全能型程序员的人，并且他同时在数学、统计和概率方面拥有丰富的专业知识，并且精通商业造诣，是相当困难的一件事，在最坏的情况下几乎是不可能的任务。这对于那些有兴趣利用攫取大数据中商业价值的企业来说等于无法提供解决方案。

（It is indeed difficult to get hold of a Unicorn.）

抓住一个独角兽确实很难。

Well, if you cannot get them in the wild, make them up from various parts - in the best style of Dr Frankenstein and his monster - and that is where the image of the Jackalope comes handy. In 1932 Douglas Herrick did indeed put together his creation when he stuck a pair of deer horns on a dead jackrabbit and mounted it as a trophy3. The rest is history, as the Converse County city of Douglas, Wyoming became the Jackalope capital of the United States.

1932年，道格拉斯·赫里克把一对鹿角贴在一只死掉的豺兔上，并把它作为艺术品悬挂起来，这的确是他的创意。时间一长，这便成了一段历史，美国怀俄明州作为道格拉斯的一个县，因此成了鹿角兔的发源地。

（If you cannot get data scientists in the wild, make them up.）

如果你不能在野外找到数据科学家，那就造一个出来吧

（3 Martin, D. (2003, Jan 19th). Douglas Herrick, 82, Dies; Father of West’s Jackalope. The New York Times）

Furthermore, you do not have to get a fake hunter’s trophy to see a Jackalope. As I mentioned before, there is faint fact to the existence of horned rabbits. That is definitely more than one can say about a one-horned horses. This is thanks to the existence of a virus, the cottontail rabbit papilloma virus (CRPV), which makes infected rabbits grow bone-like structures in their skulls4. The virus was discovered in the 1930s by Richard E. Shope and was the first example a cancer caused by a virus.

此外，你并不需要编造一个假猎人的关于鹿角兔的故事。正如前文所述，鹿角兔的存在并非空穴来风。然而对于一匹独角兽来说，并没有人见过其真实的依据。由于存在这样一种病毒，即棉铃虫兔乳头状瘤病毒（CRPV），它使受感染的兔子在其头骨中生长骨样结构。该病毒是在20世纪30年代由Richard E. Shope发现的，也是第一例由病毒所引起的癌症。

（Plus, there is a faint fact to the existence of horned rabbits.）

（因此，鹿角兔的存在并非空穴来风）

（4 Zimmer, C. (2012). Rabbits with Horns and Other Astounding Viruses. Chicago Shorts. University of Chicago Press）

（有角的兔子和其他令人吃惊的病毒。芝加哥新闻猎奇 芝加哥大学出版社）

The use of this allegory is proposed to show how silly it is to simply employ wishful thinking in the pursuit of exploiting data and hoping that a single individual will come to the rescue. What I am trying to say is that one should think optimistically about the prospect of finding capable data scientists if we are prepared to be realistic about distinguishing mythological aspirations from messy reality.

(It is possible to find capable data scientists if we are prepared to be realistic about our expectations.)

使用这个寓言是为了表明简单分析一下数据就能盈利完全是不切实际的想法，就像依靠个体力量就能单枪匹马拯救整个团队一样的愚蠢。我想表达的是：如果我们准备真正区分神话的愿望和混乱的现实，那么我们应该持有积极乐观地态度来思考如何寻找（或是打造）一个有能力的数据科学家。

只要我们满怀期待并加以准备，就有可能找到（打造）一个有能力的数据科学家

What I propose is that the best way to tackle the data science needs of a business - a startup or a large conglomerate - is to put together a rangale of jackalope data scientists, than daydreaming of a bliss of non-existant Unicorns. After all, there are indeed better chances of seeing a Jackalope-like animal than a Unicorn, right?

如何解决企业(无论是一家初创企业抑或是一家大型企业集团)数据科学家需求的最佳方式，我的提议是是聚集一大批鹿角兔般的数据科学家，而不是白日梦般地幻想着去寻找到一个独角兽般存在的数据科学家（因为这样完美的数据科学家是永远不存在的）。毕竟，比起独角兽，看到一只鹿角兔般的动物更加现实，对吧？

(I propose therefore to put together a rangale of jackalope data scientists.)

（因此，我建议把鹿角兔般的数据科学家集合起来。）

The next question is thus related to how the rangale of data scientists should be put together, what roles they should have and what resources to provide them with. These points are perhaps not easy to answer, as they depend to a large extent on the area in a business where the insight is beingsought, and for what purpose (see Section 1.4). Nonetheless,there are ome general guidelines that can be taken into account when tackling the data scientist conundrum.

因此，下一个问题涉及如何将这一类的数据科学家集合在一起，他们应该扮演各自什么角色，以及提供相应的什么资源。这些问题也许不容易回答，因为它们在很大程度上取决于企业所在的领域的洞察力，以及相关需求（见第1.4节）。然而，在处理数据科学家难题时，还是有一些通用准则。

(Not only is it important to know what qualities a data scientist should have, but also what role they are expected to play and what tools they will use to do their jobs.)

(不仅要知道一个数据科学家应该具备什么素质，而且要知道他们应该扮演什么角色，以及他们将使用对应的什么工具来完成相应的工作。)

(1.2.1 Characteristics of a Data Scientist and a Data Science Team)

(1.2.1数据科学家和数据科学团队的特点)

It seems that everyone loves, or would love to have, a data scientist, and as we have seen, the wishful list of desired characteristics makes it more difficult for businesses to choose among otherwise capable candidates.

似乎每个人都喜欢或者希望拥有一位完美的数据科学家，正如我们所看到的，这种一厢情愿的期望使得企业很难在候选人中做出选择。（因为每个候选者总会在某个领域或多或少有一定的欠缺）

(Everyone would like to have their own data scientist and knowing what is important for the business needs is a major aspect to consider.)

(每个人都希望有自己领域的数据科学家，并且将其对于该领域业务需求的了解作为主要方面的考察方向)

The analogy that comes to mind is that of the everlasting dating puzzle where everyone is waiting for Princess or Prince Charming, but is unable to find “the one”. For a data scientist to be considered “the one” the skills required include those discussed in the previous section and summarised in Figure 1.1.

脑海中浮现的比喻是永恒的相亲之谜：每个人都在等待公主或白马王子，却永远也找不到“那个人”。对于一个被认为是“最佳”的数据科学家来说，所需的技能已经在上一节中讨论并总结在图1.1中。

Let us pause for a moment before we tackle the subject at hand and consider what the purpose of the data science team is or will be. This is a crucial step in building that team as these objectives will help identify the important traits that the data scientists are expected to have.

让我们先停顿片刻，再讨论手头的主题，即数据科学团队的目标究竟是什么。因为这是建立该团队的关键步骤，这些目标的确立将有助于确定大家所期望的数据科学家的重要特征。

(Having a clear idea of how a potential data scientist will fit in the organisation and what they will work on is important.)

（形成一个清楚的概念关于一个有潜力的数据科学家将如何契合企业以及他们的工作将是如何的至关重要）

Furthermore, having a clear idea of how they will fit in the organisation and what problems they are expected to solve will aid in defining the size of the team and the type of expertise needed. It is not uncommon to hear of organisations that are interested in riding the data science wave, but do not have a clear goal regarding the purpose of their data science journey.

此外，清楚地知道他们将如何契合企业，以及期望他们解决什么问题将有助于确定团队的规模大小和所需的业务知识类型。常常听闻很多企业期望驾驭数据科学浪潮，但对其数据科学之旅的目的却没有明确的目标，这并不罕见。

With the objective of the data science team in mind, it becomes much easier to decide what is relevant in a particular case. In general, what makes a good data scientist is a linear combination of some of the following traits:

根据数据科学团队的目标，在特定条件下来决定相关要点要容易得多。一般来说，一个好的数据科学家通常具有以下特征组合：

• Curiosity

• Grasp of machine learning

• Data product building and management

• Effective communication of data insights

• Programming and data visualisation abilities

• Knowledge of statistics and probability (other mathematical areas are welcome)

• Healthy skepticism, in the scientific tradition: Carry out experiments, test hypotheses, etc

* 好奇心
* 掌握机器学习
* 数据产品的建模和管理
* 有效沟通数据洞察力
* 编程和数据可视化能力
* 了解统计和概率（熟悉其他数学领域也可）
* 科学怀疑主义：大胆猜想，进行实验、检验猜想假设等

(Some important traits in a data scientist.)

（数据科学家的一些重要特征）

The important thing to realise here is that the linear combination of the features mentioned above do not necessarily have to be equally weighted, and that is the main reason for the persistance of the Unicorn fallacy we have been discussing. Should your data scientists lack some more developed branches in their antlers, all you need to do is give them a helping hand and provide them with colleagues that will help in developing those skills, but more importantly cover the gap in those desirable features. In other words, much like Mr Herrick, put together your very own Jackalope team with people who have a broad-range of generalist interests, but a deep expertise in a certain area or two.

这里需要认识到的重要一点是，上述特征的组合并非缺一不可，而这正是我们前文一直在讨论的独角兽谬误的主要原因。如果你的数据科学家在某些方面有所欠缺(就像鹿角缺少一些发达的分支那样)，你所要做的就是向他们伸出援助之手，提供同事帮助他们发展这些欠缺的技能，弥补与理想特征之间的差距。换句话说，就像赫里克先生所带队的team那样，把你数据科学家团队打造成大部分人具有广泛的通才，少数几人拥有深厚的专业知识在单个或多个领域即可

(The features mentioned do not have to be combined in equal measures.)

所提到的特征没有固定的组合模式。

(Start a data science team with a solid core, perhaps made out of more than one person.)

（一个稳定的数据科学团队，很可能由多人组成。）

The sensible thing to do is to start with a solid core and not let the list above let you get carried away. In other words, setting the foundations of the data science team is similar to having strong foundations in a building; without them the whole tower may collapse in an instant. Furthermore, use this core to your advantage and bank some of the easy wins to start with. The three pillars in this data science triumvirate I am referring to may include, with variations in the titles, the following main roles:

明智的做法是先从打造一个坚实的团队核心成员开始，而不要被上面的清单所迷惑。换言之，数据科学团队的核心成员如同建筑物中打下坚固的地基；没有它们的存在，整个塔可能在瞬间倒塌。此外，通过核心团队的建议有助于先打赢一些小战役(如签下小单，做演讲demo等)。我所提到的数据科学三大支柱包括下列主要角色：

• Data Science Project Manager

• Lead or Principal Data Scientist

• Data Architect

(The data science triumvirate.)

* 数据科学项目经理
* 首席数据科学家
* 数据架构师

（数据科学三部曲）

Having a person that is able and experienced in managing technical teams is an important role to have in the mix. The main idea is to cover the fact that many a data scientist is far more interested in tackling questions and problems head on, rather than dealing with managing a project from end to end. One way to help them deliver is to have a knowledgeable individual that is able, on the one hand, to keep track of how projects are going, attend meetings and manage relationships. On the other hand, they should have a general understanding of techniques, algorithms and technology to be able to liaise with the team effectively. The project manager does not have to be a ninja programmer, but should be able to understand what the rest of the team are working on and the challenges they may be facing.

在管理团队中有有能力和经验的项目经理是一个至关重要的角色，其存在的主要目的是涵盖这样一个事实：多数数据科学家更感兴趣的是如何直接处理一个接一个的技术问题，而不是从始至终的管理项目。因此帮助他们实现目标的一种方式是需要一个知识全面的人，一方面，能够跟踪项目进展情况，参加会议和管理关系。另一方面，也对算法和各类技术栈有着全面的了解，以便能够有效地与团队进行联系。项目经理不必是忍者型程序员(即能够完全解决问题的程序员)，但是应该能够理解团队的其他成员正在做什么，以及他们可能面临的挑战。

(First, a Data Science Project Manager is needed.)

（首先，需要一个数据科学项目经理。）

The second figure in the triumvirate is that of the principal data scientist. Not only is it necessary to have a good project manager, but also have someone with a strong background in a quantitative field: Physics, mathematics, computer science, etc. Ideally the academic credentials this person would speak for themselves. In terms of programming, this person may not be a developer in the full sense of the word, however, they should have a firm background in coding and solving problems with the use of technology. An important ingredient of the role is to be able to act as an advisor or guide to other data scientists and analysts in the team.

核心三角阵型的第二个关键是首席数据科学家。团队中不但要有一个好的项目经理，而且要有一个在专业领域有很强技术背景的人：物理、数学、计算机科学等。理想情况下，这个人的学术资历就能证明一切。在编程方面，这个人可能不是一个完全意义上的开发人员，但是，他们应该具有使用技术编码和解决问题的坚实能力。这个角色的重要作用在于能够扮演为团队中其他数据科学家和分析师的顾问或向导的角色。

(Followed by a Lead Data Scientist.)

（其次是首席数据科学家。）

The third pillar in the team is the data architect, who will provide expertise in terms of data structures, databases, software engineering and computational capability. It is important for the data architect to be able to disentangle the data resources that the business may (or may not) have, and be able to use their expertise to assess what data is available, when it is available, and manage the constraints that the business, regulation and security impose on the workflow. Ideally, the data architect would be interested in quantitative topics, but most importantly their programming skills must be spot on. Note that the data architect will use the same technology that the data scientists employ in their day-to day activities.

核心团队的第三边是数据架构师，该角色将在数据结构、数据库、软件工程和计算能力方面提供专业知识。对于数据架构师来说，重要的是能够分离业务可能拥有（或可能没有）的数据资源，并且能够使用他们的专业知识来评估哪些数据是可用的，何时是可用的，以及制定合理的工作流程来管理业务、法规和保证项目开发安全。理想情况下，数据架构师会对某些特定主题感兴趣，但是最重要的一点是，他们的编程技能必须是非常精确。需要注意的是，数据架构师和数据科学家在日常工作中使用的技术应该完全相同。

(And finally a Data Architect completes the trio.)

（最后，数据架构师完成核心的第三边。）

Finally, there are four aspects that are important to remember when considering putting together that data science team. First, consider who the main stakeholders of the data science team are, and clarify the lines of reporting. Remember that everyone wants their own data scientist, and confusing or conflicting messages can lead to undesired results.

最后总结一下，如何建立一支数据科学团队有四个重要的方面需要要记住：

首先，考虑数据科学团队的主要利益相关者是谁，并澄清报告的内容。记住，每个利益相关者都想要自己的数据科学家，而混淆或冲突的消息会导致不期望的结果(因此要分清主次)

(Clear reporting lines are also important.)

(明确报告范围也很重要。)

Second, for data scientists to be able to work independently and (more importantly) productively it is important for them to be able to navigate the stack entirely. This enables extracting relevant data with appropriate tools (see Section 1.3). A data science team without strong IT skills or engineering support will have a hard time doing the job they do best.

第二，对于数据科学家来说，能够独立完成工作，如高效地完成数据模型搭建是非常重要的。包括能够用适当的工具提取相关数据（见第1.3节）等。如果没有强大的专业IT工具支持，数据科学团队将很难做到尽善尽美。

（Having appropriate tools to work with is paramount.）

(合适的工具对工作来说至关重要)

Third, once data has been identified for tackling a problem, proper interpretation is not necessarily easy, and misrepresentation of the results can be very damaging. It is not uncommon to see the use of tools such as machine learning algorithms to be seen as a black box; in practice, knowing the capabilities, limitations and trade-offs requires experience.

第三，一旦数据被确定用于处理某个问题，正确的解释（拟合）不一定是容易的事情，错误的结果可能是非常有害的。一些机器学习算法类工具被视为黑箱并不少见，往往只有实践才能出真知.

（It is necessary to have appropriate expertise to interpret and rework results.）

(必须依靠对应专业知识来解释和修正结果。)

Fourth, have the product always in mind: Not only is it important to have the right IT and statistics/machine learning skills, but also the team has to have a clear idea of the final product of their efforts, as well as their target audience. You may be able to come up with the most amazing models and results, but they may not be of much use if the product is of no interest to stakeholders or if the data scientist fails to communicate the results to them.

第四，始终牢记产品准则：拥有正确的IT和统计/机器学习技能固然很重要，但团队必须清楚地了解他们所努力打造的最终产品，以及目标受众是谁。您可能能够提出最令人惊叹的模型和结果，但是如果涉众对产品不感兴趣(未能抓住痛点)，或者数据科学家未能将结果完美展示出来，那么该产品则意味着失败。

（Also have a clear idea of the final product and communicate results clearly.）

（对最终产品有着清晰的认识并能够将其清楚地传达出去）

Consider as well the tools used to present results; in other words, there may be technology out there that lets the data scientist dazzle his/her target audience, but if that audience is not able to even access the technology, then you have lost the battle before starting.

还要考虑用于呈现结果的工具；换句话说，假如数据科学家的产品所采用的技术能够令他/她的目标受众眼花缭乱的，但是如果大家无法从结果展示中看出该技术的价值，那么产品还未展示就意味着已经失败了

（Appropriate technology for presenting and delivering results is also important.）

（适当的呈现和传递产品的技术工具也是很重要的。）

A point in case in my experience is the use of great JavaScript libraries such as D3. I am an advocate for their use as they can be effective and even great fun to use. However, they only work on “modern” browsers and unfortunately a large number of institutions out there only support old browsers unsuitable to render the created assets. This becomes a relevant point when considering the deployment of solutions (dashboards, reports, etc.).

我的经验中是使用JavaScript庞大的类库如D3(Data-Driven Documents)。我倡导使用的原因是因为它们对受众是有效的，甚至使用起来非常的有趣。然而，它们通常只在较新版的浏览器上工作，不幸的是，许多机构使用的浏览器版本比较陈旧。因此如何进行结果解决方案的呈现（如仪表板、报告等）时，就成为了一个需要探讨的话题。

译者注：D3支持的主流浏览器不包括IE8及以前的版本。D3测试了Firefox、Chrome、Safari、Opera和IE9。D3的大部分组件可以在旧的浏览器运行。

Page53

1.3 Data Science Tools

数据科学工具

With our newly acquired data science team and the individual high-calibre data scientists and analysts that compose it, we are able to keep abreast of the latest developments in the field of analytics and data science, and are able to extract actionable insights from our data.

通过我们新组建的数据科学团队以及由其组成的个人高素质数据科学家和分析师，我们能够及时了解分析和数据科学领域的最新发展，并能够从我们的数据中提取可操作的洞察力

However, not only do we need to be flexible, agile and expert, we are also required to have the right tools and infrastructure to enable the team to fulfil the objectives agreed with the team sponsors. To that end, there are a number of considerations that we would need to think about in helping the team decide on the tools needed as well as some other points such as:

但是，我们不仅需要灵活，灵活和专业，还需要拥有合适的工具和基础架构，以使团队能够实现与团队赞助商期望的目标。为此，在帮助团队进行工具选型以及其他一些要点时，我们需要考虑许多因素，例如：

(The tools chosen need to enable us to be flexible agile and expert.)

（所选择的工具需要使我们变得灵活灵活和专业）

• Regulatory and security requirements of hosting and manipulating the data

• Locations of data sources - and related subjects such as whether we would need/have immediate access to them, or would get them in batches for upload

• Responsiveness requirements for queries - e.g. Real-time v Fixed Reporting

• Volume of queries/searches to be run

• Format of the data source

• Quality of the data

* 托管和操纵数据的法规和安全要求
* 数据源的位置——以及相关的项目，例如我们是否需要/立即访问它们，还是成批地将它们上传
* 查询的响应能力要求 - 例如实时或定时报告
* 要运行的查询/搜索量
* 数据源格式
* 数据质量

(Some considerations when choosing appropriate tools.)

(选择合适工具时的一些注意事项)

The security consideration above is usually a big question for any business that requires their data to be in a particular jurisdiction and does not plan to create their own cloud service. For instance, Google to date will not guarantee that data will stay in Europe, for example.

对于要求其数据位于特定管辖范围且不打算创建自己的云服务的任何企业而言，上述安全考虑通常是一个大问题。例如，到目前为止，谷歌并不能保证全部数据都留在欧洲。

(Security of the data is very important.)

(数据的安全性非常重要。)

Data science and analytics is all about data, statistical analysis and modelling. It is therefore important to have the technology that enables those functions. A data warehouse, ETL software, statistical, modelling and data-mining tools are necessary. Similarly, an appropriate hardware and network environment are required (perhaps even in the cloud).

数据科学和分析都是关于数据、统计分析和建模。因此，拥有能够实现这些功能的技术非常重要。数据仓库，ETL软件，统计，建模和数据挖掘工具都是必须的。同样，需要适当的硬件和网络环境支撑（甚至可能在云端搭建）。

(A data warehouse, ETL software, statistical, modelling and data mining tools are necessary.)

（数据仓库，ETL软件，统计，建模和数据挖掘工具是必要的。）

The technologies used in the analytics arena have evolved at a fast pace in the last few years, and a number of open source projects, with lots of support have emerged, for instance:

分析领域中使用的技术在过去几年中发展迅速，并且出现了许多开源项目，并提供了大量支持，例如：

• Data Framework: MapReduce, BigQuery, Hadoop, Spark.

Hadoop is probably the most widely deployed (if sometimes under-utilised) framework to process data.

Hadoop is an open source implementation of the MapReduce programming model from Google. Other

technologies are aimed at processing streaming data, such as S4 and Storm. BigQuery (by Google) is a web service that enables interactive analysis of massive datasets and can be used in conjunction with MapReduce.

Enterprise versions of Hadoop are available from vendors such as HortonWorks. More recently the use of Spark has captivated the imagination of the big data connoisseurs

(Data framework technologies)

数据框架：MapReduce, BigQuery, Hadoop, Spark.

Hadoop可能是处理数据最广泛部署（但有时未充分利用）的框架。

Hadoop是谷歌的MapReduce编程模型的一种开源实现。其他的技术旨在处理流数据，例如S4和Storm。BigQuery（由Google提供）是一种Web服务，可以对大量数据集进行交互式分析，并可与MapReduce结合使用。

Hadoop的企业版可以从HortonWorks等供应商处获得。Spark框架最近也受到一些大数据使用者的青睐。

(大数据框架技术)

• Streaming data collection: Kafka, Flume, Scribe. The models may be different but the aim is similar: Collect data from many sources, aggregate it and feed it to a database, or a system like Hadoop, or other clients

流数据收集框架：Kafka, Flume, Scribe。这些模型之间可能有所不同，但是目的相似：从多个源收集数据，聚合数据并将其输出到数据库，或者像Hadoop这样的系统，或者其他客户端以供分析

(Streaming data collection technologies)

(流数据收集技术)

• Job scheduling: Azkaban and Oozie manage and coordinate complex data flows

工作调度：Azkaban和Oozie管理和协调复杂的数据流

(Job scheduling technologies)

(工作调度技术)

• Big Data Query languages: Pig and Hive are languages for querying large non-relational datastores. Big data frameworks such as MapReduce and Hadoop can be made more “user friendly” with them. Hive is very similar to SQL. Pig is a data-oriented scripting language

大数据查询语言：Pig和Hive是查询大型非关系数据库的语言。MapReduce和Hadoop等大数据框架和其配合“用户易用性较好”。Hive与SQL非常相似。Pig是一种面向数据的脚本语言

(Big data query languages)

(大数据查询语言)

• Data stores: Voldemort, Cassandra, Neo4j and HBase. These are data stores designed for good performance on very large datasets

数据存储：Voldemort，Cassandra，Neo4j和HBase，这些是专为在巨型数据集上获得良好性能而设计的数据存储

(Data stores)

（数据存储）

1.3.1 Open Source Tools

1.3.1 开源工具

The model of developing tools whose source code is made available for contribution has shifted the environment for their deployment both in small and large enterprises.The collaborative nature of the various projects provides a pool of knowledge and quality assurance that is difficult to beat. A rich and wide set of tools in the open source domain has contributed to the expansion of data science.They include tools that process large datasets as well as data visualisation, together with prototyping tools:

开源工具中贡献的源代码模型已经改变了在小型和大型企业中部署的环境。各种项目的协作性质提供了难以击败的知识壁垒和质量保证。开源领域中丰富而广泛的工具为数据科学的扩展做出了贡献。它们包括处理大型数据集以及数据可视化的工具以及原型设计工具：

(There are many open source tools that can be readily used in the data science workflow.)

（有许多开源工具可以在数据科学工作流程中使用。）

• Python: Data manipulation, prototyping, scripting, and the main focus in this book

• Apache Hadoop: Framework for processing big data

• Apache Mahout: Scalable machine-learning algorithms for Hadoop

• Spark: Cluster-computing framework for data analytics

• The R Project for Statistical Computing: Data manipulation and graphing

• Julia: High-performance technical computing

(We will be using Python in this book.)

(R is a noteworthy software package widely used by the data science community.)

• GitHub, Subversion: Software and model management tools

• Ruby, Perl, OpenRefine: Prototyping and production scripting languages

* Python：数据操作，原型设计，脚本编写，这也是本书的主要重点
* Apache Hadoop：处理大数据的框架
* Apache Mahout：用于Hadoop的可扩展机器学习算法
* Spark：用于数据分析的集群计算框架，用于统计计算
* R项目：数据处理和图形
* Julia：高性能技术计算
* GitHub，Subversion：软件和模型管理工具
* Ruby，Perl，OpenRefine：原型和生产环境脚本语言

（本书中将使用Python）

（R是数据科学界广泛使用的值得注意的软件包。）

As mentioned above, Hadoop is rapidly becoming ubiquitous for processing massive datasets. The framework is scalable for distributed data processing, but as remarked in Section 1.1.1, in my view not all data science problems require big data processing. The Hadoop “hype” has caused many organizations to deploy MapReduce-like systems that are effectively used to dump data - without a big picture of the information management strategic plan or without understanding how all the pieces of a data analytics environment fit together.

如上所述，Hadoop在处理海量数据集方面正迅速变得无处不在；该框架可扩展用于分布式数据处理，但正如1.1.1节所述，在我看来，并非所有的数据科学问题都需要大数据处理；Hadoop“热”已经导致许多企业部署了类MapReduce系统，这些系统可以有效地用于转储数据，从而无需了解信息管理战略计划的大局，也不必了解数据分析环境相关部分如何组合在一起。

(Not all data science is about big data.)

(不是所有的数据科学都关于大数据

R is seen as the programming language for statistical computing. It is not characterized by the beauty of its code, but the results are great. The number of packages that is available in the R repository (CRAN) makes it very flexible.

R被视为统计计算的编程语言。它的特点并非代码的美丽整洁，而在于输出很棒（译者注：此处理解为输出结果绘图方便并且容易与其他编程语言和数据库进行交互）。在R存储库（CRAN）中有各类包使得它非常灵活且易于扩展。（译者注：截至2018年10月16日，CRAN已经收录了各类包13181个，涵盖经济计量、财经分析、人文科学研究以及人工智能。）

The use of scripting languages such as Python provide a professional platform for application development and deployment. It is very well suited for prototyping and testing new ideas. Furthermore it supports various data storage and communication formats, such as XML and JSON, plus there is a large number of open source libraries for scientific computing and machine learning.

Python等脚本语言的使用为应用程序开发和部署提供了专业平台。他非常适合原型设计和测试新的想法，而且它支持各种数据存储和通信格式，如XML和JSON。除此之外还有大量用于科学计算和机器学习的开源库可供选择

(In recent times, Python has seen a resurgence thanks to the data science scene.)

（最近，多亏了数据科学领域的旺盛需求，Python已经复苏并有了王者归来的架势。）

Python has a number of very useful libraries such as SciPy, NumPy and Scikit-learn. SciPy extends Python into the domain of scientific programming. It supports various functions, including parallel programming tools, integration, ordinary differential equation solvers, and even extensions for including C/C++ code within Python code. Scikit-learn is a Python-based machine learning package including many algorithms for supervised learning (support for vector machines, naïve Bayes), unsupervised learning (clustering algorithms), and other algorithms for dataset manipulation. It is for these reasons that we will use Python in the rest of this book.

Python有许多非常有用的库，如SciPy、NunPy和Scikit-learn。SciPy将Python扩展到科学编程领域。它支持各种功能，包括并行编程工具、集成、常微分方程求解器，甚至在代码中支持C/C++代码的扩展。Scikit-Learn是基于Python的机器学习包，包括许多用于监督学习的算法（支持向量机、朴素贝叶斯）、无监督学习（聚类算法）和其他用于数据集操作的算法。出于以上原因，我们将在本书的剩余部分使用Python进行演示。

(Python is a well-supported language with a wide variety of modules and libraries.)

（Python是一个被各行各业广泛支持的语言，具有各种各样的模块和库。）

Page58

*1.4 From Data to Insight: the Data Science Workflow*

*从数据到洞察力：数据科学工作流*

As we have seen above, the role of a data scientists is an interesting one, and at times a challenging one. Not only do we need the right combination of skills (either in a team or an individual), but also the right tools and business questions. In this section we will address the steps that adata science project may follow. It is important to emphasise that although we may categorise and separate the various steps, the workflow is not necessarily a linear one as we shall see.

正如我们上面所看到的，数据科学家的角色是一个有趣的角色，有时是一个具有挑战性的角色。这不仅需要正确的技能组合（无论是团队还是个人），还需要正确的工具和业务问题。在这一节中，我们将讨论数据科学项目可能遵循的步骤。需要强调的是，尽管我们对各个步骤已经进行了归类和标序号，但是实际工作流并不一定是线性的，正如下图我们将看到的那样。

（The workflow of the data science expert is worth discussing too.）

（数据科学专家的工作流程也值得讨论。）

With our newly acquired data science team and the right combination of skills, we are ready to tackle out first assignment, and it is now when key measurements of success for the project should be identified. Furthermore, I tis important to realize from the start that in the vast majority of cases there will not be a unique, final answer. It is thus better to frame the problem as an iterative process where a better solution is reached on each iteration. The various steps in the data science workflow include:

具备新组建的数据科学团队和各种正确的技能组合后，我们已准备好解决首次任务，现在应该确定项目成功的关键衡量标准。此外，从一开始我们就应该意识到，在大多数情况下，不会有唯一的、最终的答案。因此，最好将问题框架化为迭代过程，在每个迭代中达到更好的解决方案。数据科学工作流中的各个步骤包括：

（The data science process is iterative.）

（数据科学过程是迭代的。）

• Question identification

• Data acquisition

• Data munging

• Model construction

• Representation

• Interaction

（The steps in the data science workflow）

问题识别

数据采集

数据挖掘

模型构建

表征

交互作用（互作）

（数据科学工作流中的步骤）

Page 59 图

Figure 1.3: The various steps involved in the data science workflow.

图1.3：数据科学工作流中涉及的各个步骤。

The fact that they have been listed in that order does not mean that they have to be followed one after the other. In some cases you may start, for example, with an internal dataset and immediately create some plots even before cleaning the data. Also, once you have started the project, you may move between steps in every iteration. Figure 1.3shows a diagram of the steps mentioned above, note that they do not necessarily follow one another in the order listed above.

（The workflow outlined above is not necessarily followed in sequence.）

上述步骤按照该顺序列出的事实并不意味着它们必须一个接一个地被遵循。在某些情况下，您可以使用内部数据集开始，甚至在清理数据之前立即创建一些图表。此外，一旦启动项目，您可以在每次迭代中的步骤之间移动。图1.3显示了上述步骤的图表，注意它们不一定按照上面列出的顺序彼此跟随。

（上述工作流程不一定按顺序执行。）

1.4.1 Identify the Question

1.4.1识别问题

There is no such a thing as insight without a clear and concise question, as well as having a way to measure the success or failure of the answer obtained after running your favourite machine learning algorithms. It is not a matter of getting a dataset and simply massaging it and creating some plots. On the contrary, let the questions point out the potential datasets that may be useful in answering them and to what extent.

（Without a clear question, there is no insight.）

没有清晰简洁的问题就不会形成洞察力这样的东西，也没有办法衡量在运行你最喜欢的机器学习算法后得到的答案是成功还是失败。这不是获取一个数据集和信息分析后形成一个点位图那么简单。相反，通过问题引导出那些潜在有用的数据集来并通过这些深层数据来形成洞察力（洞见）

（没有明确的问题，就无法形成洞察力）

Another important thing to bear in mind is the fact that although we may encapsulate a problem in a single question, in many cases it is much easier to break it down into smaller parts that can be tackled in a more straightforward manner. Furthermore, at every iteration there may be more, smaller or larger, follow-up questions that will also require answers. Remember that it is an iterative process!

（Breaking down the problem into smaller questions is useful.）

另一件需要牢记的重要事情是，尽管我们可以将问题封装在单个问题中，但在许多情况下，将其分解为可以更直接的方式处理的较小部分要容易得多。此外，在每次迭代中，可能会有更多、更小或更大的后续问题也需要答案。记住这是一个反复的过程！

（把问题分解成更小的问题是非常有用的。）

1.4.2 Acquire Data

1.4.2获取数据

Once you have a problem to tacke, the first thing that needs doing is figuring out if you or your organisation has the appropriate data that may be used to answer the question. If the answer is no, you will need to find appropriate sources for suitable data externally - web, social media, government, repositories, vendors, etc. Even in the case where the data is available internally, the data may be in locations that are hard to access due to technology, or even for regulatory and security reasons.

（Identify appropriate sources of suitable and useful data.）

一旦遇到问题，需要做的第一件事就是确定您或您的企业是否具有可用于回答问题的适当数据。如果答案是否定的您将需要在外部找到适当数据的适当来源 - 网络，社交媒体，政府，存储库，供应商等。即使数据在内部拥有，但也可能位于由于技术原因难以访问到，甚至是出于监管和安全的考虑而无法访问。

（确定合适的有用数据来源。）

1.4.3 Data Munging

1.4.3数据挖掘

If there is no insight without a question, then there is no data without data munging. Munging, or wrangling data is actually the most time-consuming task in the data science workflow. According to the New York Times’ Steve Lohr data scientists may spend anything between 50 and 80 percent of their time doing “data janitor work”5 and I can definitely attest to that.

（There is no data without data munging.）

如果说没有问题就没有洞察力，那么没有数据挖掘，数据也就不存在了。在数据科学工作流中，数据挖掘和数据整理实际上是最耗时的任务。根据《纽约时报》的史蒂夫·洛尔的调研：数据科学家们可能会把50%到80%的时间花在“看门人”工作(数据挖掘和整理)上，关于这一点我完全赞同。

（没有数据挖掘，数据也就不存在了）

Data preparation is key to the extraction of valuable insight and although some may prefer to concentrate only on the much more fun modelling part, the fact that you get to know your dataset inside out while munging it implies that any new or follow-up questions can probably be attained with less effort.

（5 Lohr, S. (2014, Aug 17th). For Big-Data Scientists, ’Janitor Work’ Is Key Hurdle to Insights. The New York Times）

数据准备是提取有价值的洞察力的关键，虽然有些人可能更愿意只关注建模部分（因为觉得那更有趣），但事实上当你经过数据挖掘后会对数据有了从内到外的深层次理解，这也意味着任何新的或后续的问题都可能事半功倍。

（5 Lohr, S.（2014，8月17日）对于大数据科学家来说，“看门人工作”(数据挖掘和整理)是洞察力的关键障碍。纽约时报

1.4.4 Modelling and Evaluation

1.4.4建模与评价

Having a clean dataset to feed to a machine learning or statistical model is a good start. Nonetheless, the question remains regarding what the most appropriate algorithm to use is. A partial answer to that question is that the best algorithm depends on the type of data you have, as well as its completeness. It also depends on the question you decided to tackle. Once the model has been run through the so-called training dataset the next thing to do is to evaluate how effective and accurate the model is against the testing dataset and decide if the model is suitable for deployment.

(Every model needs to be evaluated.)

将一个干净的数据集输入到机器学习或统计模型是一个良好的开端。尽管如此，问题仍然在于最适合使用的算法是什么。这个问题的部分答案是，最佳算法取决于您所拥有的数据类型，以及它的完整性。也取决于你决定要解决的问题。一旦模型已经建立并训练数据集测试通过，接下来要做的是评估模型与测试数据集的有效性和准确性，并决定模型是否适合部署。

（每一个模型都需要进行评估）

（有关训练和测试数据集的更多信息，请参见第3.11节）

（For more in training and testing datasets, see Section 3.11.）

1.4.5 Representation and Interaction

1.4.5表征与互动

They say that a picture is worth a thousand words, and it does stand to reason that the vast majority of us get further information from a couple of well-presented plots than by looking at rows and rows of data. Data visualization is more of an art than a science, and much has been written (or drawn) about by brilliant designers and data journalists. Simply take a look at the great images produced by David McCandless in his book Information is Beautiful6 or the visualizations produced by Manuel Lima in Visual Complexity7.

俗话说一图胜千言，我们绝大多数人都可以从一些精心设计的图中获得更多信息，而不是查看行和行的数据。数据可视化与其说是一门科学，还不如说是一门艺术，许多优秀的设计人员和数据记者都曾撰写（或绘制）过。例如大卫·麦克坎德莱斯（David McCandless）在《信息是美丽的》（Information is Beautiful6）一书中所创作的伟大图像，或者曼纽尔·利马（Manuel Lima）在《视觉复杂性》（Visual Complexity7）一书中所创作的可视化图像。

（Data visualization is more of an art than a science, but an important one nonetheless.）

（数据可视化更像是一门艺术而不是科学，但却是一门重要的艺术。）

（6 McCandless, D. (2009). Information is Beautiful. Collins）

（6 McCandless, D. (2009). 信息是美丽的. Collins）

（7 Lima, M. (2011). Visual Complexity: Mapping Patterns of Information. Princeton Architectural Press）

（7 Lima, M. (2011).视觉复杂性：映射信息的模式。普林斯顿建筑出版社)

You do not have to produce such artistic beauties, but make sure that the data representation that you decide to use is accurate, simple, and provides clarification to the story you want to communicate. In some cases, there is the possibility of telling the story in a way that the reader/viewer is able to interact with the data representation, for instance in the form of dashboards, reports or interactive plots. These may be fun to use, but the same principles of accuracy, simplicity and clarity do apply.

你不必要求自己的作品也达到艺术美的高度，但至少要确保所使用的数据准确性，简单性，并为你要阐述的观点提供说明。在某些情况下，有可能以读者/观众更容易被数据的直观展现所吸引，例如仪表板、报表或交互式绘图等形式。这些展现方式可能非常有趣，但同样需要遵循准确性，简单性和清晰适用等原则。

1.4.6 Data Science: an Iterative Process

1.4.6数据科学：一个迭代过程

The simple fact that a machine learning model has been run on clean data does not mean that the work of the data scientist is done and dusted. On the contrary, the effectiveness of the model needs to be carefully monitored as the outcome depends on the data that is fed to them. A simple case of garbage-in-garbage-out. Similarly, any new data with a variety of new features may degrade the accuracy of the initial model, and thus it becomes necessary to adjust parameters or acquire new data.

机器学习模型已经在清洁数据上运行的简单事实并不意味着数据科学家的工作已经完成并且已经尘埃落定。相反，需要仔细监控模型的有效性，因为结果取决于提供给他们的数据。 一个简单的例子就是garbage-in-garbage-out（若输入错误数据则输出亦为错误数据）。类似地，具有各种新特征的任何新数据都可能降低初始模型的准确性，因此有必要调整参数或获取新数据。

（The effectiveness of a model needs to be monitored.）

（一个模型的有效性需要被监控）

Furthermore, even if there were such a thing as a never-changing model, the fact is that new and follow-up questions arise naturally from the data science process. This means that the workflow is better to be treated as an upward spiral where constant iterations provide improvement and new insights.

此外，即使最开始存在一个不需改变的模型，但随着新的和后续的问题在数据科学分析过程中不断涌现，工作流程逐渐向上螺旋盘升，因此需要不断的迭代改进以提供新的洞察力。（迭代过程中模型需要改变）

（Think of this process as an upward spiral.）

（把这个过程想象成一个向上的螺旋。）

I would like to close this chapter with a few questions that the data scientist team and their stakeholders should always have in mind:

• What data was used and why?

• Where was the data acquired from and who owns it?

• Was the entire dataset used? Is a sample representative of the entire population?

• Were there any outliers? Have they been considered in the analysis?

• What assumptions were made when applying the model/algorithm? Are they easily relaxed/strengthened?

• What does the result of the model mean to the process/business/product?

（Some questions to always bear in mind during any data science project.）

我想用一些数据科学家团队及相关负责人应该始终牢记的问题来结束本章：

* 使用什么数据？为什么？
* 数据从何而来，谁拥有它们？
* 是否使用了整个数据集？这个集合能否代表整个样本空间吗？
* 有异常值吗？分析中考虑过它们吗？
* 在应用模型/算法时做出了什么假设条件？它们容易放松/加强吗？
* 模型的结果对流程/业务/产品意味着什么？

（在任何数据科学项目中都要记住的这些问题。）

*1.5 Summary*

*1.5总结*

In this chapter we addressed some crucial aspects that will enable us to implement and acquire that elusive mythical being called a data scientist. We have provided a working definition for the term data science and have described how it is a rapidly evolving multi-disciplinary field encompassing areas in mathematics, computer science, statistics and business expertise.

在本章中，我们讨论了一些关键方面，这些方面将使我们能够实现并获得这种难以捉摸的神话（鹿角兔）即数据科学家。我们为数据科学这个术语提供了一个工作定义，并描述了它是如何迅速发展的多学科领域，包括数学，计算机科学，统计学和商业专业领域。

We mentioned a few examples of data science products and have seen that the main motivation for data science and analytics is deriving valuable and actionable insights from data. Our discussion pointed out that in order to fulfil that motivation, a data scientist with the appropriate skills is needed. Unfortunately, understanding about this subject has created expectations that point at the data scientist role as one akin to the mythical Unicorn. We argued that a better understanding of the role indicates that we should perhaps use a different allegory: A Jackalope. It is still a mythical being, but the fact that you can put one together out of different parts (as in a team), or the hint that there may be a scientific explanation for their potential existence provides a more hopeful panorama for many businesses interested in exploiting their data.

我们提到了一些数据科学产品的例子，并且已经看到数据科学和分析的主要动机是从数据中获得有价值和实际可操作的见解（洞察力）。我们的讨论指出，为了实现这一动力，需要具备适当技能的数据科学家。 不幸的是，对这个主题的理解产生了一些过高的期望，即数据科学家的角色应该类似于神秘的独角兽。 我们认为，更好地理解这个角色应该使用不同的寓言：即鹿角兔(Jackalope）。 它仍然是一个神话般的存在，但是事实是，你可以从不同的部分（如在团队中）拼凑出一个这样的整体(译者注：就像1.2章道格拉斯•赫里克把一对鹿角贴在一只死掉的豺兔上那样)，或者暗示它们可能存在科学的解释，为许多对利用自身大数据感兴趣的企业提供了更有希望的全景展望。

We continued our discussion with the three main pillars that would sustain a productive data science team, namely a data science project manager, a lead or principal data scientist and a data architect. We saw that not only is the team composition important, but also the tooling provided to carry out their tasks in a productive manner. We concluded this chapter by outlining the steps in a simplified data science workflow and explained their importance as part of what should be seen as an iterative process.

我们继续讨论支持高效数据科学团队的三个主要支柱，即数据科学项目经理，团队负责人即首席数据科学家和数据架构师。 我们看到，不仅团队组成很重要，而且提供高效的方式执行任务的工具是也不可或缺的。 在本章的最后，我们概述了简化的数据科学工作流程中的步骤，并解释了这些步骤应该被视为迭代过程一部分的重要性。

In the next chapter we will provide a brief refresher of some important concepts of using Python. This will enable us to have a point of reference for the rest of the book where Python will be used to implement a variety of algorithms that form part of the Jackalope’s bag of tricks.

在下一章中，我们将简要介绍一些使用Python的重要概念。 这将使我们能够为本书的其余部分提供一个参考点，其中Python将用于实现各种算法，这些算法是也是构成数据科学家(Jackalope)技巧包的一部分。

Page65

2 Python: For Something Completely Different

2 Python：完全不同的东西

There is no shortage of programming languages and paradigms. With that in mind it may seem surprising that what started up as a “hobby programming project” by Guido van Rossum in 1989 has taken on a life of its own, with a very active community, used in a wide variety of applications. Its success is perhaps due to the compactness of its code, or the fact that is open source, or even the variety of toolsets. Whatever the reason, Python is a great tool to have under your jackalope data scientist belt.

（Python started up as a hobby programming project.）

编程语言和范例史上并不缺乏，尽管如此，Guido van Rossum在1989年开始的“业余爱好编程项目”确令人惊讶的慢慢拥有了自己的生命，以及一个非常活跃的社区，并适用于各种各样的场合。 它的成功可能是归功于其代码的紧凑性，或者是开源的事实，甚至是工具集的多样性。不管是什么原因，Python都是作为组成数据科学家(jackalope鹿角兔)不可或缺的工具组的一部分。

(Python起初是一个业余爱好编程项目。)

Python is not named so after the nonvenomous snakes whose family includes some of the largest in the World. Instead it is named after the famous British comedy troupe Monty Python, and it was created to be appealing to Unix/C coders. Today, Python’s emphasis on code readability and expressive syntax has made it into a general-purpose, high level, object-oriented programming language available in multiple platforms and with a plethora of supporting packages and modules.

（We have to thank Monty Python for the name of this flexible programming language.）

Python名字的由来和世界上最大的无毒蛇其实没有任何关系。 相反，它取自的英国喜剧剧团Monty Python，它的创建是为了吸引Unix / C程序员。 今天，Python强调代码可读性和表达性的语法，使其成为一种通用的，高级的，面向对象的编程语言，可在多个平台上使用，并提供大量支持包和模块。

(我们要感谢Monty Python提供这种灵活编程语言的名称。)

译者注：在 python shell 中输入：import this

可以看到如下内容：

The Zen of Python, by Tim Peters

Python 的设计哲学，作者：Tim Peters

Beautiful is better than ugly.

优雅胜于丑陋。

Explicit is better than implicit.

明了胜于晦涩

Simple is better than complex.

简单胜于复杂。

Complex is better than complicated.

复杂胜于凌乱。

Flat is better than nested.

扁平胜于嵌套。

Sparse is better than dense.

间隔胜于紧凑。

Readability counts.

可读性很重要。

Special cases aren't special enough to break the rules.

特例并没有特殊到违背规则

Although practicality beats purity.

尽管实用性胜于纯粹性

Errors should never pass silently.

也不可对错误置之不理。

Unless explicitly silenced.

除非另有明确要求。

In the face of ambiguity, refuse the temptation to guess.

面对模棱两可，拒绝猜测。

There should be one-- and preferably only one --obvious way to do it.

尽量找一种，最好只有一种方法来做一件事。

Although that way may not be obvious at first unless you're Dutch.

虽然这并不容易，除非你是 Python 之父。

Now is better than never.

做总比不做好

Although never is often better than \*right\* now.

但不假思索就动手还不如不做。

If the implementation is hard to explain, it's a bad idea.

如果方案很难解释，那就不是好方案。

If the implementation is easy to explain, it may be a good idea.

如果方案容易解释，那么它很有可能是个好方案。

Namespaces are one honking great idea -- let's do more of those!

命名空间是一个绝妙的理念，我们应当多加利用。

（“Pythonic” coding style guidelines are followed by so-called Pythonistas.）

（所谓的Pythonistas遵循“Pythonic”编码风格指南。）

Perhaps some of the most defining features of the language include the use of indentation for grouping statements, together with was has become to be known as the “pythonic” coding style, i.e. style guidelines and idioms used by veteran Python programmers, aka Pythonistas. These guiding principles can be read in any Python installation by importing the this module. These aphorisms, compiled by the long time pythonista Tim Peters, are shown in the code listing printed above.

（The use of indentation is an important feature of Python.）

上述就是python语言的一些最具定义性的特性包括使用缩进对语句进行分组，以及被称为是“pythonic”编码样式，即由经验丰富的Python程序员使用的样式指南和习惯用法，也称为Pythonistas。通过导入this模块，可以在任何Python安装中读取这些指导原则。这些谚语，由Pyth创始人Tim Peters编写，并显示在上面列出的代码列表中。

（缩进的使用是Python的一个重要特征。）

This book assumes some knowledge of general programming principles and a level of familiarity with Python. Nonetheless, in this chapter we will briefly review some of the concepts and idiosyncrasies of the language that will be used throughout the book. If you are a seasoned Python programmer, you may want to skip this chapter and move on to the next one. If, however, you are interested in a refresher, then go ahead and read the following pages, you might find something completely different. For those of you that are beginning their Python journey, this chapter may whet your appetite to learn more. There are plenty of resources to quench your thirst1,2.

（We will assume a certain familiarity with programming principles and with Python in particular.）

1 Downey, A. (2012). Think Python. O’Reilly Media

2 Langtangen, H. (2014). A Primer on Scientific Programming with Python. Texts in Computational Science and Engineering. Springer Berlin Heidelberg

本书假定读者具有基础的编程原理的一些知识以及对Python有一定的熟悉程度。 尽管如此，在本章中，我们将简要回顾一下将在本书中使用的语言的一些概念和特性。 如果你是一位经验丰富的Python程序员，可以跳过本章并继续下一章。 但是，如果你对复习感兴趣，请继续阅读以下页面，您可能会发现完全不同的东西。 对于那些正准备开始Python之旅的人来说，本章可能会激起你学习更多东西的兴趣。有足够的资源来供你参考见1,2。

（我们假定对编程原理特别是Python有一定的了解。）

1 Downey, A. (2012). Think Python. O’Reilly Media

2 Langtangen, H. (2014). A Primer on Scientific Programming with Python. Texts in Computational Science and Engineering. Springer Berlin Heidelberg

*2.1 Why Python? Why not?!*

2.1 为何是Python？为何不是？！

We mentioned above that Python emphasises code readability, and this in turn has an impact on productivity: Not only is a data scientist able to create scripts to be executed as a batch, but also is able to start an interactive console (iPython shell for instance) or notebook (iPython/Jupyter notebook - see Section 2.1.2). Furthermore, the Python ecosystem is supported by various packages that extend and enhance the language.

（Interactive sessions are available through the iPython shell and the Jupyter notebook.）

我们在上文提到了Python强调代码的可读性，这对生产力也是一种促进：不仅数据科学家能够创建脚本作为批处理执行，而且还能够启动交互式控制台（例如iPython shell） ）或笔记本电脑（iPython / Jupyter Notebook - 参见第2.1.2节）。 此外，Python 还有各种扩展和增强语言的包以供选择。

（通过iPython shell和Jupyter notebook可以进行交互式会话操作）

For example, the NumPy module provides functions that enable the manipulation of numeric arrays and matrices. The SciPy module enables functionality usually expected in scientific computing such as optimisation, linear algebra routines, Fourier transformation, etc.

（NumPy provides numerical functions to be used with Python.）

例如，NumPy模块提供了可以操作数值数组和矩阵的函数。 SciPy模块支持科学计算中通常需要的功能，如优化算法（Optimisation），线性代数例程，傅里叶变换等。

（NumPy模块提供了很多供Python使用的科学函数。）

The support that Python has for hierarchical modularity makes it possible for programmers and developers to build further functionality. A good example is the Pandas package, which extends the NumPy arrays into dataframes for ease of data manipulation and analysis (see Section 2.5). We will be using Pandas in the rest of the book. Similarly, in this book we will extensively use packages such as Matplotlib, Statsmodels and Scikit-learn which implement plotting, statistical models and machine learning algorithms, respectively.

（The Pandas package enables data manipulation and analysis. ）

Python对分层模块化的支持使程序员和开发人员可以构建更多功能。 一个很好的例子是Pandas包，它将NumPy数组扩展为数据帧，以便于数据操作和分析（参见第2.5节）。 我们将在本书的其余部分使用Pandas。 同样，在本书中，我们将广泛使用Matplotlib，Statsmodels和Scikit-learn等软件包，分别实现绘图，统计模型和机器学习算法。

（Pandas包可以进行数据处理和分析。）

(Other popular packages are Matplotlib, Statsmodels and Scikit-learn.）

（其他受欢迎的软件包是Matplotlib，Statsmodels和Scikit-learn。）

It is true that Python is an interpreted language and as such code may be often slower than compiled code tailored to a particular machine architecture. In that respect, although the source code in Python is interpreted on the fly, the main advantage is flexibility. This is an important point in the data science workflow as we are interested in the balance between implementation time versus execution: In many cases we are more interested in getting to grips with the raw data rather than in fine-tuning the execution time for a particular machine.

（Python is an interpreted language, in other words, code is read and executed line-by-line.）

确实，Python是一种解释型语言，因此代码可能通常比为针对机器架构先编译再执行的语言执行速度要慢一些。 但是Python中的源代码是动态解释的，主要优点是灵活性。 这是数据科学工作流程中的一个重点，因为我们对实现时间而非执行时间更加关注：在许多情况下，我们更感兴趣的是掌握原始数据而不是针对机器去调优执行时间。

（Python是一种解释型语言，换句话说，代码是逐行读取和执行的。）

That brings us to another advantage: Since Python code is not compiled, we have portability. Scripts developed in one environment can be executed in any other one with the appropriate installation. Python is fast enough for the vast majority of the computational tasks in a data science workflow: It is important to get the logic right in the first place and, if needed, make the execution faster later.

（Python code is flexible and portable.）

这给我们带来了另一个优势：由于Python代码无需编译，因此具有可移植性。 在一个环境中开发的脚本只需适当的安装（Python解释器）即可在任何其他环境中执行。 Python对于数据科学工作流中的绝大多数计算任务来说运行性能足以满足需求：重要的是首先使逻辑正确，如果需要，以后可以更快地执行。

（Python代码灵活且可移植。）

（It is fast enough for the typical data science workflow）

（它对于通常的数据科学工作流程来说性能足够满足需求）

For the purposes of this book, I assume that a suitable Python installation is available to you. Furthermore, I will also assume that the installation is for version Python 3.x rather than Python 2.x. I would like to point out that although version 2 is widely used, more and more users are adopting version 3. Please note that there may be some packages that have not been updated to suit the incompatibilities between the two versions. I hope that in the near future these inconsistencies are ironed out and the community eventually moves to version 3.x.

（We will be working with version 3 of the Python distribution.）

出于本书的目的，我假设你已经安装了合适的Python版本。 此外，我还假设安装版本是Python 3.x版而不是Python 2.x. 我想指出的是，尽管版本2被广泛使用，但越来越多的用户正在使用版本3.请注意，可能有一些软件包尚未更新以适应两个版本之间的不兼容性。 我希望在不久的将来，这些不一致之处得到解决，社区最终会转向版本3.x.

（我们将使用Python发行版的第3版）

In particular, I find that the Anaconda distribution3, built by Continuum Analytics, is robust and complete enough for our purposes. Furthermore, they have suitable distributions for Windows, MacOS and Linux. You can obtain an installation package from http://continuum.io/ and follow the steps provided by the distribution. Please note that any other Python 3.x installation with the appropriate packages is equally suitable in order to follow the discussions in the rest of the book. I will explicitly mention any modules or packages that are required so that the more seasoned user is able to use pip, easy-install, homebrew or any other installation method they are comfortable with.

（3 Continuum Analytics (2014). Anaconda 2.1.0. https://store. continuum.io/cshop/anaconda/）

特别是，我发现由Continuum Analytics构建的Anaconda distribution3非常强大且完整，足以满足我们的需求。 此外，它适用于Windows，MacOS和Linux。 您可以从http://continuum.io/获取安装包，并按照对应版本提供的步骤操作。 请注意，其他软件包的Python 3.x安装都同样适用于本书其余部分的讨论， 我将明确提及所需的模块或软件包，以便经验丰富的用户能够使用pip，easy-install，homebrew或任何其他适合他们的安装方法来完成环境的相应配置。

Page 71

*2.1.1 To Shell or not To Shell*

*2.1.1 使用Shell或不使用Shell*

We have seen above that Python is an interpreted language and as such it is possible to interact with the different commands that have to be executed during the course of a session. Similarly, it is also possible to write all the commands first and execute them in a more conventional development workflow. Both approaches have their advantages and disadvantages, and fortunately you do not have to choose one over the other as Python is flexible enough to enable you to do both.

(Python shell enables interactivity. You can also develop scripts to be executed without user intervention.)

我们在上文已经提及Python是一种解释语言，因此可以与在会话过程中已经执行的不同命令进行交互。 类似地，也可以首先编写所有命令并在更传统的开发工作流程中逐一执行它们。 这两种方法都有其优点和缺点，幸运的是，您不必纠结到底选择其中哪一种方法，因为Python足够灵活，可以让您同时拥有这两种方法。

（Python shell支持交互。您还可以开发脚本，无需用户干预即可自动化执行。）

If you are interested in interacting with the code as you are writing it then starting, for instance, an iPython shell will allow you to type a command and immediately send it to the interpreter for execution. You can then take the output and continue your analysis. This way of working lets us see the results of the commands as we are working. It is an excellent way of prototyping code to be implemented in longer projects.

如果您有兴趣在编写代码时与代码进行交互，那么可以启动iPython shell将允许您键入命令并立即将其发送到解释器以供执行。 然后，您可以获取输出并继续分析。 这种工作方式让我们可以边工作边看到命令的结果。 这是在大型项目中实现代码原型的绝佳方式。

(The interactive shell lets you assess the results of each command you send to the interpreter.)

（交互式shell允许您实时看到发送给解释器的每个命令的结果。）

Unfortunately, working in this way makes the code somewhat ephemeral unless you save the commands that you are issuing to the interpreter. If you are interested in keeping track of your development and execute a series of commands repeatedly, then you can save those commands in a text file to create a Python script which by convention has a “.py” extension. The execution of these scripts does not require a Python shell and instead can be executed directly from the command line using the following syntax:

不幸的是，以这种方式工作只会使代码暂存内存中，除非你发出命令将其保存到解释器。 如果你有兴趣跟踪开发并重复执行一系列命令，则可以将这些命令保存在文本文件中以创建Python脚本，按照惯例，该脚本具有“.py”扩展名。 这些脚本的执行不需要Python shell，而是可以使用以下语法直接从命令行执行：

#python myscript.py

(Python scripts let you save the series of commands that form a program. They are saved with the .py extension.)

（Python脚本允许您保存构成程序的一系列命令。它们以.py扩展名保存。）

where we are assuming that the script called myscript.py is saved in the local path. The use of Python scripts makes it easier to launch longer jobs that do not need input from a user to be executed.

我们假设名为myscript.py的脚本保存在本地路径中。 使用Python脚本可以更轻松地启动而不需要用户重复输入较长的作业指令。

(The command above is launched directly from the terminal; no need for the Python/iPython shell.) （上面的命令直接从终端启动;不需要从Python / iPython shell执行）

In this book we will take the approach of using an interactive shell (code is compatible with Python/iPython shells) so that we are able to use the output given by the interpreter to explain the different steps we are taking. These individual commands can then be saved in a suitable script that can be run later. We will thus present code as follows:

在本书中，我们将采用交互式shell（代码与Python / iPython shell兼容）的方法，以便我们能够及时使用解释器给出的输出来解释我们正在采取的不同步骤。 然后你可以将这些单独的命令保存在可以稍后运行的脚本中。 因此，我们将提供如下代码：

#> 42 + 24

#66

（In this book we will present code as used from the Python shell for ease of presentation.）

（在本书中，我们将提供Python shell中使用的代码，以便于演示。）

Notice that the diple, >, represents the Python shell prompt where the next line of code is expected. Furthermore, if the command sent to the interpreter expects a printed result, the shell will automatically display it. For ease of explanation, in cases where we do not expect a result, or the discussion requires us to break down the code, we will show it in a script style. In other words, no shell prompt will be shown:

请注意，尖括号 >表示Python shell提示符，其中下一行为代码行。 此外，如果发送给解释器的命令需要打印结果，shell将自动显示它。 为了便于解释，在我们不期望结果或讨论要求我们分解代码的情况下，我们将以脚本样式显示它。 换句话说，不会显示shell提示符：

#a = 42 + 24

（In cases where we do not expect a result we will not show the shell prompt.）

（如果我们不期望结果，我们将不会显示shell提示符。）

Comments in Python are indicated with the use of the hash symbol, #. The interpreter will ignore any commands that follow until the end of the line. In the example below we enter a comment after carrying out a division operation

Python中的注释使用哈希符号＃来表示。 解释器将忽略直到行尾的所有命令。 在下面的示例中，我们在执行除法运算后输入注释

（A comment in Python is entered with the hash symbol, #.）

（使用符号#表示输入Python中的注释，＃。）

#> 2/3 # Python 3

#0.6666666666666666

Please note that the behaviour in Python 2 is different, as the operation above will result in an integer divison:

请注意，Python 2、3中的行为是不同的，因为上面的操作将导致整数除法：

#> 2/3 # This integer division returns 0， Python 2

#0

Since we have passed two integers to the interpreter, Python carries out an integer division, returning only the integer part. If we want a real division we can do the following:

由于我们已经将两个整数传递给解释器，因此Python执行整数除法，仅返回整数部分。 如果我们想要真正的除法，我们可以如下做到：

> 2/3.0

> 2/float(3)

（You can avoid integer division by casting integers into floats.）

(你可以通过将整数转换为浮点数来避免整数除法。)

In Python 2 we can import the functionality from Python 3.x with the \_\_future\_\_ module as follows:

在Python 2中，我们可以使用\_\_future\_\_模块从Python 3.x导入功能，如下所示：

> from \_\_future\_\_ import division Python 2

> 2/3

0.6666666666666666

（Alternatively you can use the \_\_future\_\_ module）

(或者你可以使用\_\_future\_\_模块）

*2.1.2 iPython/Jupyter Notebook*

*2.1.2 使用iPython/Jupyter Notebook*

We have seen how the interactive shell in Python allows us to assess the results of the code we are executing. That may be good enough for a number of tasks, but sometimes it may be desirable to present the code in a way that is easier to interact with, not just for the original programmer, but also with other members of a team or general audience.

我们已经看到Python中的交互式shell如何允许我们实时评估我们正在执行的代码的结果。 这对于许多任务来说可能已经足够好了，但有时可能需要以更容易与之交互的方式呈现代码，不仅对于原创程序员，而且对于团队的其他成员或一般受众。

The iPython/Jupyter notebook is a great way to do this. Not only does it let you run your code in the same way as the interactive shell and add comments to your code, but also enables you to document your code, calculations and processing all within a web-based interface. In this book, we have made a conscious decision not to use the iPython/Jupyter notebook for printing purposes, however I would encourage you to make use of it whenever you can.

使用iPython / Jupyter Notebook是一种很好的方法。 它不仅可以让你以交互式shell相同的方式运行代码并为代码添加注释，还可以让你在基于Web的界面中记录代码，计算和处理。 虽然在本书中，我们决定不使用iPython / Jupyter笔记本进行打印，但我鼓励你尽可能地使用它。

（The iPython/Jupyter notebook lets us interact with the code and also add rich media, text and maths.）

（iPython / Jupyter Notebook让我们可以与代码进行交互，还可以添加富媒体，文本和数学。）

The iPython/Jupyter notebook supports the inclusion of text, mathematical expression and inline graphics as well as other rich media such as websites, images, video, maths, etc. At its core, a notebook is a JSON document with the extension .ipynb, which makes the files very light and highly portable. The web interface is very handy, and if required, the notebook can be exported to a number of formats such as HTML, LATEX, PDF, Markdown or even raw Python. Furthermore, the Jupyter project aims to exploit the parts of the code that are not specific to Python and extend their use to other programming languages.

iPython / Jupyter notebook支持包含文本，数学表达式和内嵌图形以及其他富媒体，如网站，图像，视频，数学等。其核心是一个扩展名为.ipynb的JSON文档， 这使得文件非常轻便且便于携带。 Web界面非常方便，如果需要，可以将notebook导出为多种格式，如HTML，LATEX，PDF，Markdown甚至原始Python。 此外，Jupyter项目并非专为Python特定开发，还可以将其用途扩展到其他编程语言。

（A notebook is a JSON document, keeping with the general portability principle of Python.）

（notebook是一个JSON文档，符合Python的可移植性原则。）

Page75

*2.2 Firsts Slithers with Python*

*2.2 初次探寻Python*

We have already had the opportunity to interact with the iPython shell and have seen some simple operations such as addition (+) and division (/). We can continue our exploration of the programming language as an interactive calculator. As we would expect, the rest of the arithmetic operations are supported by Python as shown in Table 2.1. Notice that the exponentiation in Python is represented with two stars, \*\*. So far we have used integers and floating point numbers in the examples presented. It is therefore natural to ask what other types are supported by Python.

我们已经有机会用iPython shell进行交互，并看到了一些简单的操作，如加法（+）和除法（/）。 我们可以继续探索编程语言作为交互式计算器的用法。 正如我们所期望的那样，Python支持其余的算术运算，如表2.1所示。 请注意，Python中的取幂用两颗\*\*表示。 到目前为止，我们在所提供的示例中使用了整数和浮点数。 因此，很自然地会想到Python究竟支持其余哪些类型。

（Exponentiation is denoted with \*\* in Python.）

（取幂在Python中用\*\*表示。）

（Table 2.1: Arithmetic operators in Python.）

（表2.1：Python中的算术运算符。）

*2.2.1 Basic Types*

*2.2.1基本类型*

An important fact to remember about Python is that it is a dynamically typed language. In other words, we do not need to declare variables before we use them and it is not necessary to specify their type. Furthermore, each and every variable that we create is automatically a Python object.

记住Python的一个重要事实是它是一门动态语言。 换句话说，我们使用之前不需要声明变量，也没有必要指定变量的类型。 此外，我们创建的每个变量都自动成为一个Python对象。

（Python is a dynamically typed language: We do not need to specify the variable type in advance.）

（Python是一种动态类型语言：我们不需要提前声明变量类型。

*2.2.2 Numbers*

*2.2.2数字*

As we saw in the previous section, Python supports two types of numbers: Integers and floating point numbers. So we can assign the value of an integer to a variable as follows:

正如我们在上一节中看到的，Python支持两种类型的数字：整数和浮点数。 所以我们可以将一个整数的值赋给变量，如下所示：

Page77 图

（Python supports integers and floating point numbers.）

（Python支持整数和浮点数。）

Notice that assignation does not require Python to print anything as a response. We can check the type of an object with the command type:

请注意，赋值操作不需要Python打印该内容作为响应。 我们可以使用type命令检查对象的类型：

（The command type lets us see the type of an object.）

（命令type让我们看到对象的类型。）

Python will let us know what type of object we are dealing with; in this case the object Universe is of type integer. Let us see an example for a floating point number:

Python会让我们知道我们正在处理什么类型的对象; 在这种情况下，对象Universe的类型为整数。 让我们看一个浮点数的例子：

*2.2.3 Strings*

*2.2.3 字符串*

A string is effectively a sequence of characters. In Python, strings can be defined with the use of either single (’ ’) or double quotes ( ‘‘ ’’) as follows:

字符串实际上是一系列字符。 在Python中，可以使用单引号（''）或双引号（''''）定义字符串，如下所示：

Page78图

（Strings in Python can be defined with single or double quotes.）

（Python中的字符串可以用单引号或双引号定义。）

In the example above we have requested the type of the variable string1, and as expected, Python tells us that indeed it is a string. We can ask Python to print a string as follows:

在上面的例子中，我们已经请求打印变量string1的类型，并且正如预期的那样，Python告诉我们它确实是一个字符串。 我们可以让Python打印一个字符串，如下所示：

（In Python 2 the print statement does not require the brackets.）

（在Python 2中，print语句不需要括号。）

The + operator is overloaded for strings and it results in the concatenation of these objects:

+运算符可以进行字符串对象的连接操作：（python内部重载了该操作符以便于字符串连接操作）

（Concatenation of strings can be achieved with the + symbol.）

（可以使用+符号实现字符串的连接。）

In the example above we have also demonstrated the way in which Python is able to deal with simultaneous assignation in a single line. In other words, the string “Norwegian” is assigned to the variable dead and the string “blue” to the variable parrot. Note that mixing operators between strings and numbers is not allowed, and an error will be thrown if they are. Instead you will have to convert a number to a string using the str function.

在上面的示例中，我们还演示了Python能够在单行中同时分配变量的方式。 换句话说，字符串“Norwegian”被赋值给变量dead，字符串“blue”被赋值给变量parrot。 请注意，不允许在字符串和数字之间混入运算符，如果是，则会抛出错误。 你必须使用str函数将数字转换为字符串才能进行相关操作。

（Python is able to carry out multiple assignation in a single line. This is part of a Pythonic programming style!）

（Python能够在一行中执行多个赋值。 这是Pythonic编程风格的一部分！）

Strings are immutable objects in Python and this means that we cannot change individual elements of a string. We shall discuss more about immutable objects in the context of tuples in Section 2.2.6.

字符串是Python中的不可变对象，这意味着我们无法更改字符串的各个元素。 我们将在2.2.6节中讨论不可变对象元组的更多内容。

（Strings in Python are immutable.）

（Python中的字符串是不可变的。）

*2.2.4 Complex Numbers*

*2.2.4 复数*

Python also supports complex numbers, and it denotes the imaginary number *i* = *p-*1 as j, and so for a number n, nj is interpreted as a complex number.

Python也支持复数，它将虚数i = p-1表示为j，因此对于数n，nj被解释为复数。

Page79 开方

（In Python, the imaginary number i is denoted with the letter j.）

（在Python中，虚数i用字母j表示。）

Let us see an example: If we want to define the complex number z = 2 + 3i we simply tell Python the following:

让我们看一个例子：如果我们想要定义复数z = 2 + 3i，我们只需告诉Python如下：

Page79 图

Please note that although the numbers used in the example above are integers, Python recasts them as floating point numbers to suit the complex number object. In the piece of code shown above we have also demonstrated the fact that we can use the backslash (n) to break a line for code readability.

请注意，尽管上面示例中使用的数字是整数，但Python会将它们重新设置为浮点数以适合复数对象。 在上面显示的代码段中，我们还演示了这样一个事实：我们可以使用反斜杠（\n）来换行提高代码可读性。

（The backslash allows us to break a line.）

（反斜杠允许我们将一行代码分成多行。）

Remember that each and every entity in Python is an object. Each object has a number of possible actions they are able to perform, i.e. methods. In the example above we have called the real and imag methods associated with a complex number object to obtain the real and imaginary parts respectively. Another use of a method is shown in the example above for a string, in this case the format method to tell Python how to format the printing of a string.

请记住，Python中的每个实体都是一个对象。 每个对象都有许多可以执行的操作，即方法。 在上面的例子中，我们调用了与复数对象相关的real和imag方法，分别获得实部和虚部。 上面的示例中显示了一个方法的另一种用法，用于字符串，在本例中是格式化方法，告诉Python如何格式化字符串的并进行打印。

（The method of an object can be invoked by following the name of object with a dot (.) and the name of the method.）

（可以通过跟随带有点（.）的对象名称和方法名称来调用对象的方法。）

*2.2.5 Lists*

*2.2.5 列表*

A list is pretty much self-explanatory: It is a sequence of objects, and these objects can be either of the same type or not. We denote a list with square brackets, [ ]. Lists are mutable objects and therefore it is possible to change individual elements in a list:

列表几乎是不言自明的：它是一系列对象的集合，这些对象可以是相同类型也可以不是。 我们用方括号[]表示一个列表。 列表是可变对象，因此可以更改列表中的单个元素：

Page 80图

（A list is denoted by square brackets [ ].）

（列表用方括号[]表示。）

It is possible to refer to elements of a list using an index that corresponds to their position in the list: 可以使用与列表中的位置对应的索引来引用列表的元素：

（We can refer to elements in a list with an index）

（我们可以引用带索引的列表中的元素）

（Indexing in Python starts at zero.）

（Python中的索引从零开始。）

Indexing in Python starts with the number zero and thus the first element of the numbers list is referred to as numbers[0]. Also, we can refer to a sub-sequence of the list using the colon notation as start:end, where start refers to the first element we want to include in the sub-sequence and end is the last element we want to consider in the slice.

Python中的索引从数字零开始，因此数字列表的第一个元素称为数字[0]。 此外，我们可以使用冒号表示引用列表的子序列，形如start：end，其中start是指我们想要包含在子序列中的第一个元素，end是我们要在其中使用的最后一个元素，从而形成一个切片。

Remember that Python interprets the slicing operation up to, but not including, the last item in the sequence. In the example above, Python reads from index 1 and up to index 3, but not including 3. That is why only the second and third elements of the numbers list are returned.

请记住，Python将切片操作解释为不包括序列中的最后一项。 在上面的示例中，Python从索引1读取到索引3，但不包括3.这就是为什么只返回数字列表的第二个和第三个元素。

（Slicing refers to the subsetting of an array-like object such as lists and tuples.）

（切片指的是类似数组的对象的子集，例如列表和元组。）

Since lists are mutable objects it is possible for us to change elements in a list:

由于列表是可变对象，因此我们可以更改列表中的元素：

Page81 图

（We are able to change the elements of a list because they are mutable objects.）

（我们能够更改列表的元素，因为它们是可变对象。）

We can also add elements to a list with the append method:

我们还可以使用append方法将元素添加到列表中：

（append lets us add elements to a list.）

（append可将元素添加到列表中。）

The new element, 1.4142, is added to the numbers list at the end, increasing the length of the list by one element.

新元素1.4142将添加到数字列表的末尾，同时列表的长度增加一。

Concatenation of lists is easily achieved with the + operator:

使用+运算符可以轻松实现列表的连接：

Page82 图

（The + symbol lets us carry out list concatenation）

（+符号让我们可以进行列表连接）

Notice that if the two lists are numerical, the result using the + operator is again the concatenation of the list elements, not the sum.

请注意，如果两个列表都是数字，则使用+运算符的结果是列表元素的串联，而不是数字总和。

Another useful method of a list is sort, which does exactly what we would expect: It allows us to sort the list’s values. This method will also enable us to see the difference between mutable and immutable objects in our discussion about tuples (Section 2.2.6).

列表的另一个有用方法是sort，它的用法完全符合我们的预期：它允许我们对列表的值进行排序。 这个方法还使我们能够在讨论元组时看到可变对象和不可变对象之间的区别（第2.2.6节）。

（The sort method allows us to sort a list in place.）

（sort方法允许我们对列表进行排序。）

Let us define a list to work with:

让我们定义一个列表来使用：

Page82 图

We can now invoke the sort method as follows:

我们现在可以调用sort方法，如下所示：

（As we can see using sort with a list results in the elements being ordered in ascending order.）

（正如我们所看到的，使用带有列表的sort会导致元素按升序排序。）

There are a couple of things to note here. First, we have called the sort method using the dot (.) notation. When executing the first line in the code above, the interpreter does not return any values, and that is a good sign: It means that the method executed correctly.

这里有几点需要注意。 首先，我们使用点（.）表示法调用sort方法。 在上面的代码中执行第一行时，解释器不返回任何值，这是一个好兆头：这意味着该方法以正确执行。

In order to see what happened we issue the second command, which lets us print the contents of List1. As shown above, the elements of the list are now ordered.

为了看看发生了什么，我们发出了第二个命令，它允许我们打印List1的内容。 如上所示，现在列表的元素已经正确排序。

This takes us to the second point to note. Since lists are mutable, we can change them and in this case the sort method has changed the elements in List1 to be in ascending order. We have sorted the list in place. There was no need to create a copy of the list and sort it.

我们第二点需要注意的地方： 由于列表是可变的，我们可以更改它们，在这种情况下，sort方法已将List1中的元素更改为升序。 我们已将列表排序到位。 无需创建列表副本并对其进行排序。

（Since lists are mutable, we are able to change their elements. In this case sorting the elements for instance.）

（由于列表是可变的，我们可以改变它们的元素。 在这种情况下，例如对元素进行排序。）

Objects in Python also have functions associated with them. Lists are no exception and in this particular case there is a sorted function too. The difference is that a function will create a new object. Let us take a look:

Python中的对象也具有与之关联的方法。 列表也不例外，在这种特殊情况下也有一个排序函数。 不同之处在于该方法函数将创建一个新对象。 我们一起来看看：

Page83 图

（Lists have a sorted function.）

（列表具有排序方法。）

So far so good, nothing has changed, we end up with a sorted list. However, let us take a look at the List1 object one more time

到目前为止一切顺利，和期望一样，我们最终同样得到一个排序列表。 但是，让我们再看一次List1对象

As you can see, the object was not changed! Instead, what the sorted function has done is create a new object with the contents of List1 in ascending order.

如您所见，对象值居然没有改变！ 排序函数所做的是创建一个新对象并返回以升序排列的List1的内容。

（The sorted function creates a new object with the elements of the original list, but in ascending order.）

（排序函数使用原始列表的元素创建一个新对象，但按升序排列其内容。）

We could have assigned the result of the function to a new variable and thus create an object that can be referred to at a later stage.

我们可以将函数的结果分配给一个新变量，从而创建一个可以在稍后阶段引用的对象。

Incidentally, if you require the elements in descending order all you have to do is pass the reverse parameter to either the method:

顺便提一下，如果您需要按降序排列元素，您只需将reverse作为参数传递给方法：

Page 84 图

or the function:

或者使用函数

A very useful pythonic way of constructing lists without the need of a full-blown loop is the so-called list comprehension. A typical usage is in the creation of lists whose elements are the result of some operations applied to each member of another sequence or iterable. For example, let us create a string with a sentence:

一个非常有用的pythonic方法是所谓的列表推导式：构建列表而不需要写出完整循环流程。 典型的用法是创建列表，其元素是应用于另一个序列或可迭代的每个成员的一些操作的结果。 例如，让我们首先创建一个包含句子的字符串：

Page85图

We can use the string above to create a list of lists with each word in the sentence in capital and lower-case letters, as well as determining the length of the word. And we can do all this in a single line of code:

我们可以使用上面的字符串创建一个列表，其中包含句子中的每个单词的大写和小写字母，以及确定单词的长度。 我们可以在一行代码中完成所有这些工作：

Page85图2

（We are using the string methods split(), upper() and lower() to separate the words in the sentence, and convert them to upper- and lower-case.）

（我们使用字符串方法split（），upper（）和lower（）来分隔句子中的单词，并将它们转换为大写和小写。）

（List comprehension is useful when we need to create a list out of operating on elements of another sequence.）

（当我们需要创建一个对另一个序列的元素进行操作的列表时，列表推导式是非常有用的。）

*2.2.6 Tuples*

*2.2.6 元组*

A tuple may be seen as a list by another name: They are also sequences of objects, and they may be of mixed type too. They are indeed closely related to lists and apart from the fact that they are defined with round brackets, ( ), the main difference is that tuples are immutable.

元组可以被视为另一种形式的列表：里面的元素不仅可以是对象的序列集合，也可以是混合类型。 它确实与列表密切相关，除了用圆括号（）定义之外，其主要区别在于元组是不可变的。

（Tuples are defined with round brackets ( ).）

（元组用圆括号（）定义。）

As we have mentioned above, immutable objects cannot be changed. In other words, we cannot add or remove elements and thus, unlike lists, they cannot be modified in place. Let us take a look at some tuples:

如上所述，无法更改不可变对象。 换句话说，我们不能添加或删除元组里面的元素，因此，与列表不同，它们不能在适当的位置进行修改。 我们来看看一些元组：

Page86图

（Tuples are immutable objects）

（元组是不可变对象）

As you can see the only change in the definitions above, compared to the lists in Section 2.2.5, is the use of the round brackets. As with lists, the elements of a tuple can be referred to by their index:

正如您所看到的，与2.2.5节中的列表相比，上述定义中的唯一更改是使用圆括号。 与列表一样，元组的元素也可以通过索引来引用：

Page86图2

（Tuples can also be sliced with the help of an index.）

（元组也可以在索引的帮助下进行切片操作。）

Let us see what happens when we try to change one of the elements of a tuple:

让我们看看当我们尝试更改元组的一个元素时会发生什么：

Page86图2

（We are not able to change elements of a tuple as they are immutable objects.）

This shows that there are manipulations that are not possible to be done with a tuple. What about sorting? Well, the sorted function still works. Let us define a tuple as follows:

（我们无法更改元组的元素，因为它们是不可变对象。）

这表明直接在元组里面进行的操作是不可能的。 排序怎么样？ 好吧，排序的功能仍然有效。 让我们定义一个元组如下：

Page87图

We can now apply the sorted function to the tuple:

我们现在可以将排序函数应用于元组：

Page87图2

（The result of the sorted function on a tuple is a list.）

Not too bad, right?, but have you noticed something odd? Well, it seems that the result is not a tuple anymore, but a list! We can see that thanks to the square brackets, and we can make sure of this by using the type command:

（对元组进行排序函数操作返回的结果是一个列表。）

还不错，对吗？但是你注意到了一些奇怪的东西吗？ 好吧，似乎结果不再是一个元组，而是列表！ 我们可以看到，是方括号而非圆括号，我们可以通过使用type命令来确保这一点：

Page87图3

This is the result of tuples being immutable: The only way to allow for the elements of the tuple to be ordered is by using the mutable nature of a list. Similarly, since the elements of a tuple cannot be changed, there is no point in having a sort method. Let us have a look:

（Since tuples are immutable we cannot change their elements in place.）

这是元组不可变的结果：允许元组元素排序的唯一方法是使用列表的可变特性将其转换为列表再排序。 类似地，由于元组的元素不能改变，因此无法直接对其使用排序方法。 我们来看看：

（因为元组是不可变的，所以我们不能改变它们的元素。）

Page88图

（As stated by the error returned by Python, tuples do not have a sort attribute.）

（正如Python返回的错误所述，元组没有sort属性。）

*2.2.7 Dictionaries*

*2.2.7 字典*

We are all familiar with the concept of a dictionary: If we are interested in finding the meaning of a new or unknown word, we simply open up a book (or access a webpage) that lists words in a specified order (alphabetically, for instance). This enables us to search for the word we are interested in. A dictionary in Python serves the same purpose and it is composed of keys and values.

（In the analogy with actual dictionaries, keys are equivalent to words and values are the definitions.）

我们都熟悉字典的概念：如果我们有兴趣查询新词或未知词的含义，只需打开一本书（或访问网页），里面会按照指定的顺序列出单词（例如，按字母顺序排列））。 这使我们能够索引到我们感兴趣的单词.Python中的字典具有相同的用途，它由键和值组成。

（与实际字典类比，键相当于单词，值是定义。）

A Python dictionary is defined with the use of curly brackets, { }. Furthermore, the key-value pairs are separated by a colon (:) as follows:

使用大括号{}来定义Python的字典类型。 此外，键值对由冒号（:)分隔，例子如下：

Page88图2

（We define a dictionary with curly brackets { }.）

（我们用大括号{}定义一个字典。）

The keywords in a dictionary can be any immutable Python object including numbers, strings and tuples. The value associated with a particular key can be changed by reassigning the new value to the element of the dictionary with the relevant entry. For instance, in our example above we can see that the value to the key spam is 4:

字典中的key（关键字）可以是任何不可变的Python对象，包括数字，字符串和元组。 可以通过索引该条目将新值重新分配给字典的关联元素从而更改相关键所对应的值。 例如，在我们上面的示例中，我们可以看到Key(关键字) “Spam”的对应的值是4：

Page89图

（The keys can be any immutable object: numbers, strings or tuples for example.）

We can change the value of this key by simply reassigning any new value. We can for example reassign the value associated to the spam key:

（键可以是任何不可变对象：例如数字，字符串或元组。）

仅需重新分配任何新值就能更改此键对应的值。 例如，我们可以重新分配与key(关键字)"Spam"相关联的值：

Page89图2

（The values in a dictionary can be modified.）

This can be done repeatedly:

（可以修改字典中的值。）

可以对同一个key重复赋值：

Page89图3

（The modification can be done as many times as required.）

It is possible to access the keys and values in the form of straight lists with the aid of the keys() and values() methods:

（可以根据需要进行多次修改。）

借助keys（）和values（）方法，可以以列表的形式遍历字典的键和值：

Page89图4

（A list of dictionary keys can be obtained with the keys() method. Similarly, values() returns a list of values in a dictionary.）

We can also obtain the key-value pairs in the form of a list of tuples with the items() method:

（可以使用keys（）方法获得字典键列表。 类似地，values（）返回字典中的值列表。）

我们还可以使用items（）方法以元组列表的形式获取键值对：

Page90图

Finally, it is possible to get rid of key-value pairs with the use of the del function:

最后，可以使用del函数删掉字典里面的键值对：

Page90图2

（We can remove entries from a dictionary with del.）

（我们可以用del删除字典中的条目。）

*2.3 Control Flow*

*2.3 控制流*

Not only is it important to understand the types and objects that are available in any programming language, but also how to control the flow of a programme to be able to follow the logic behind the way in which the programme itself is organised, in other words, the order in which the individual statements are executed.

我们不仅需要理解编程语言中可用的类型和对象，而且还需要学会如何遵循程序语言组织方式逻辑来组织控制自己的程序流程。换句话说 ，组织各个语句的先后执行顺序。

In particular it is important to mention that in Python the whitespace is a meaningful character as it enables the definition of blocks of code by having the same level of indentation. Let us see some typical structures to control the flow of a programme in Python.

特别值得一提的是，在Python中，空格是一个有意义的字符，因为它可以通过具有相同级别的缩进来定义代码块。 让我们看一些控制Python程序流程的典型结构。

（Whitespace is a meaningful character in Python.）

（Whitespace在Python中是一个有意义的角色。）

*2.3.1* if... elif... else

Conditional branching enables us to perform different actions depending on the result of boolean operations. If a condition is met, then we apply an operation, otherwise a different action is performed. In Python we can do this as follows:

2.3.1 if ... elif ... else

条件分支使我们能够根据布尔运算的结果执行不同的操作。 如果满足条件，则我们应用操作，否则执行不同的操作。 在Python中，我们可以这样做：

Page91图

（The if... elif... else... lets us test various conditions and create branches for our code.）

As you can see, each block of code is indented at the same level. Also, notice that it is possible to nest various conditions with the help of the elif reserved word. The conditions are logical expressions that can test for scalar comparison and thus we can use any of the comparison operators listed in Table 2.2. Let us see an example:

（The conditions to test are logical expression that evaluate to True or False.）

（if ... elif ... else ...让我们测试各种条件并为我们的代码创建分支。）

如你所见，每个代码块都缩进到同一级别。 另外，请注意，可以在elif保留字的帮助下嵌套各种条件。 条件是可以比较的逻辑表达式，因此我们可以使用表2.2中列出的任何比较运算符。 让我们看一个例子：

（测试的条件实质是True或False的逻辑表达式。）

Page92 图

（Table 2.2: Comparison operators in Python.）

Finally, remember that the conditions are tested one by one in the order they are provided in the code. If a condition is met, the rest of the tests are not executed.

（表2.2：Python中的比较运算符。）

最后，请记住，条件是按照代码中提供的顺序逐个执行的。 如果满足条件，则不执行其余部分。

*2.3.2* while

*2.3.2* while表达式

A while loop is used when we need to repeat a block of code until a condition is no longer met. The structure of a while loop in Python is:

当我们需要重复执行代码块直到不再满足条件时，可以使用while循环。 Python中while循环的结构是：

Page92 图2

An important thing to remember is that at the very beginning of the while loop, the logical test must evaluate to True, otherwise the block of code is never executed. Also, in order to avoid infinite loops we need to update the control variable inside the block of code.

（The while loop requires a logical test at the beginning of the block.）

We can see how this works by counting down from 10:

要记住的一件重要事情是，在while循环的最开始，逻辑表达式必须求值结果为True，否则永远不会执行该代码块。 另外，为了避免无限循环，我们需要更新代码块内的控制变量。

（while循环需要在块的开头进行逻辑测试。）

我们可以从倒计时10看到这是如何工作的：

Page93 图

（Note that countdown -= 1 is a shorthand for countdown = countdown - 1.）

（注意countdown -= 1 是countdown = countdown - 1的简写）

*2.3.3* for

*2.3.3* for表达式

Typically, a while loop is used in cases when we do not know in advance how many times the block of code will need to be executed. If we know how many iterations are needed, we can use a for loop. In Python, a for loop iterates over a sequence: a list, tuple or a string for example.

（A for loop is useful when we know how many times the code needs to be repeated.）

通常，在我们事先不知道需要执行代码块多少次的情况下推荐使用while循环。 如果我们知道需要多少次迭代，我们可以使用for循环。 在Python中，for循环可以遍历序列：例如list，tuple或string。

（当我们知道代码需要重复多少次时，for循环很有用。）

Page93 图2

（This is the same basic structure used in list comprehension.）

The example we used for the while loop in Section 2.3.2 can be written with a for loop as follows:

（这与列表推导式中使用的基本结构相同。）

我们在2.3.2节中使用的while循环的示例也可以用for循环编写，如下所示：

Page94 图

We could simplify the example above by avoiding the explicit definition of the list and instead define a range:

我们可以用列表推导式来通过range生成列表从而简化避免显示的定义一个列表：

Page94 图2

（range enables us to define a sequence of numbers as an object. This means that the values are generated as they are needed.）

（range使我们能够将数字序列定义为对象。 这意味着可以只在需要时再生成值。）

In the example above we used the range(start, end, step) function to generate a sequence of numbers from start to end-1 in steps given by step. In Python 2, similar behaviour is obtained with the xrange function.

在上面的例子中，我们使用range（开始，结束，步长）函数从 start 到 end-1生成指定步长的数字序列。 在Python 2中，xrange函数有类似的行为。

*2.3.4* try... except

*2.3.4* try... except异常捕获

It is not unusual to have syntactically correct blocks of code with statements that in certain cases may cause an error during execution. These errors are not necessarily fatal in the execution of a programme and instead they are anomalous or exceptional cases that require special processing.

（Syntactically correct code may cause errors during execution.）

即使一个代码块语法完全正确，但这些语句在某些情况下可能会在执行期间导致错误，这种现象并不奇怪。 这些错误在执行程序时不一定是致命的，而是需要特殊处理的异常或异常情况。

（语法正确的代码可能会在执行期间导致错误。）

Instances such as those described above are called exceptions and when they happen, we are interested in catching them and taking appropriate action, for example by generating an error message. This is what is known as exception handling. In Python this can be done with the try...except structure:

上述实例称为异常，当它们发生时，我们需要将其捕获并采取适当的操作，例如生成错误消息等。 这就是所谓的异常处理。 在Python中，可以通过try ... except结构来完成：

Page95 图

（Exception handling in Python can be done with the try... except structure.）

In the structure above exception1, exception2,... are standard exceptions that Python knows about and that are detailed in the appropriate documentation; we list some common ones in Table 2.3.

（For further information about standard exceptions see https: //docs.python.org/2/library/ exceptions.html）

（Python中的异常处理可以使用try ... except结构完成。）

在上面的结构中，exception1，exception2，...是Python知道的标准异常，在相应的文档中有详细说明; 我们在表2.3中列出了一些常见的。

（有关标准异常的更多信息，请参阅https：//docs.python.org/2/library/ exceptions.html）

For example, we can try to calculate the reciprocal of the elements of a list and print each of the values. However, if the sequence contains the number zero, we can try to catch the exception with ZeroDivisionError:

例如，我们可以尝试计算列表元素的倒数并打印每个值。 但是，如果序列中包含数字0，我们可以尝试使用ZeroDivisionError来捕获异常：

Page96图

Table 2.3: Standard exceptions in Python.

表2.3：Python中的标准异常。

We are using the ZeroDivisionError exception to handle this particular exception.

As you can see, instead of getting an error message and the interpreter halting programme execution, the exception is handled nicely by the code after the appropriate except entry.

我们使用异常ZeroDivisionError来处理这个特殊的异常。

正如你所看到的，通过适当的except方法之后，由代码很好地处理异常，从而避免抛出错误消息导致程序停止执行的情况出现

*2.3.5 Functions*

*2.3.5 函数*

Now that we have covered some of the elementary control flow structures in Python, we can start combining them into logical blocks to carry out specific tasks. In particular we can construct pieces of code that can be repeated when necessary and whose outcome depends on the input parameters provided. In other words, we are talking about functions.

我们已经介绍了Python中的一些基本控制流结构，并且可以开始将它们组合成逻辑块来执行特定任务。 特别地，我们可以构造出在必要时重复执行的代码片段，其结果取决于所提供的输入参数。 接下来，我们谈论的话题是函数。

（A function is a good way to write code that can be repeated, and whose outcome typically depends on inputs provided.）

A function in Python has the following syntax:

（函数是编写可重复执行代码的好方法，其输出通常取决于输入。）

Python中的函数具有以下语法：

Page97图

（The function definition starts with the word def. Remember that code needs to be indented.）

（函数定义以单词def开头。 请记住，代码需要缩进。）

Notice that the function definition starts with the reserved word def and the code inside the function is indicated with appropriate indentation.

请注意，函数定义以保留字def开头，函数内部的代码用适当的缩进表示。

The input parameters for the function are the dummy variables arg1, arg2,... , argn and as you can see it is possible to define default values for some of these parameters. Parameters with default values must be defined last in the argument list.

该函数的输入参数是变量arg1，arg2，...，argn，你可以为其中一些参数定义默认值。 定义具有默认值的参数必须放在参数列表的最后。

The second line in the function definition is called the documentation string and its purpose is to describe the actions that are performed by the function. Finally, notice that it is not necessary for a function to return a result.

函数定义中的第二行称为文档字符串，其目的是说明该函数的执行步骤和目的。 最后，请注意函数不要求必须有返回值。

（A documentation string enables us to provide information about what a function does. Make sure you use it!）

（Let us define a function to calculate the area of a rectangle sides a and b:）

（文档字符串使我们能够提供有关函数功能的信息。 一定要用它！）

（让我们定义一个函数来计算由边a和b所组成的矩形面积:)

Page98图

（We are defining a default value for the parameter b.）

（我们正在为参数b定义一个默认值。）

Notice that the parameter b has been given the default value of 1. If we were to call this function with only one parameter, the function will know how to handle the calculations and use the default values when needed.

请注意，参数b的默认值为1.如果我们调用此函数只传入一个参数，该函数将知道如何处理计算并在需要时使用默认值。

Page98图2

（We can use the function by calling it in the same any other in-built Python function is.）

（函数定义后我们可以就可以通过其名字直接调用了）

In the first line of code above, we are calling the rect\_area function with two parameters, such that we assign the value 20 to a and override the default value of b with 2. As expected the area calculated is 40. Let us try providing only one single value to the function:

在上面的第一行代码中，我们使用两个参数调用rect\_area函数，这样我们就将值20赋值给a并用2覆盖b的默认值。正如预期的那样，计算的面积是40.让我们尝试只提供一个单一值来调用函数：

Page99图

Here we have only passed the value 42.4 to the function. In this case the value is assigned to a and the default value of b = 1 is used in the calculation.

这里我们只将值42.4传递给函数。 在这种情况下，值将分配给a，并在计算中使用默认值b = 1。

We can include control flow structures in our programmes to make them more useful and flexible. Let us for instance implement a simple function to calculate the factorial of a number:

我们可以在程序中包含控制流结构，以使它们更有用和灵活。 举个例子，我们实现一个简单的函数来计算数字的阶乘：

Page99图2

（A function can use any of the other control flow structures of the language.）

（\*= and -= indicate repeated operations with the left-hand-side value.）

（函数可以使用python中的任何控制流结构。）

（\* =和 - =表示使用左侧值重复操作。）

When we pass a number smaller or equal than the one the function expects, it returns the value 1, and when the number is greater than 1 the factorial is calculated with a while loop. Let us use the function:

当我们传入一个小于或等于函数期望值的数字时，它返回值1，当数字大于1时，使用while循环计算阶乘。 让我们使用这个函数：

Page100图

There may be times when it is more convenient to define a simple function on-the-fly, without having to resort of a full def structure. In these cases we can exploit the use of the so-called lambda functions:

有时可能更方便的是即时定义一个简单的函数，而不必使用完整的def结构。 在这些情况下，我们可以利用所谓的lambda函数：

（A lambda function in Python is an anonymous function created at runtime.）

（Python中的lambda函数是在运行时创建的匿名函数。）

Page100图2

where, as before, arg1, arg2,... are the input parameters and statement is the code to be executed with the input provided.

与之前一样，arg1，arg2，...是输入参数，而statement是要根据输入参数执行的代码。

For example, if we needed to calculate the cube of a list of numbers we could try the following code:

例如，如果我们需要计算数字列表的每个元素的立方，我们可以尝试以下代码：

Page100图3

（In this case the object g is a lambda function that can be called as any other function in Python.）

（在这种情况下，对象g是一个lambda函数，可以像Python中的任何其他函数一样调用。）

So far nothing too strange: We have initialised a list with the numbers 1, 3 and 6, and then defined a lambda function that calculates the cube of the argument n. We can now apply this function, for example:

到目前为止，没有什么值得奇怪的地方：我们已经用数字1,3和6初始化了一个列表，然后定义了一个lambda函数来计算参数n的立方体。 我们现在可以应用此功能，例如：

Page101图

Lambda functions may seem very simple, but it is that simplicity that provides their strength, as it shown above. This can be seen employed for instance in the implementations of PySpark, the Python API for Spark, an open-source cluster computing framework.

Lambda函数可能看起来非常简单，但正如上面所示，简单确非常实用。 可以在PySpark的源代码中看到它的身影。PySpark是用于Spark的Python API，一种开源集群计算框架。

（Lambda functions are very useful in frameworks such as Spark.）

（Lambda函数在Spark等框架中非常有用。）

*2.3.6 Scripts and Modules*

*2.3.6 脚本和模块*

With the flexibility provided by the possibility of controlling the flow of a set of instructions, and the repeatability offered by constructing our own functions, it becomes imperative to be able to store programmes in a way that enable us to use and reuse code.

由于我们可以灵活的控制一组指令流，并且通过调用我们自己的函数来重复操作，因此能够将代码保存下来并重复使用函数变得势在必行。

In Python we are able to do this by saving the instructions that make up a programme in a plain text file saved with the extension .py. Furthermore, if we use the interactivity provided by the iPython/Jupyter notebook, it is also possible to save our notebooks in a JSON formatted notebook with the extension .ipynb.

在Python中，可以通过在扩展名为.py的纯文本文件中保存我们的代码。 此外，如果我们使用iPython / Jupyter notebook所提供的交互性，也可以将代码保存在扩展名为.ipynb的JSON格式文件中。

（Python scripts have the extension .py whereas notebooks have the extension .ipynb.）

（Python脚本的扩展名为.py，而notebooks 中保存时扩展名为.ipynb。）

It is then possible to execute a Python script from the command line by calling Python followed by the name of the script to be executed. For instance, we can create a script defining a main function and a call to it. We can save the function in a script called firstscript.py with the following contents:

然后可以通过调用Python+执行脚本的名称在从命令行执行Python脚本。 例如，我们可以创建一个定义为Main主函数和调用它的脚本， 然后将函数保存在名为firstscript.py的脚本中，其中包含以下内容：

Page102图

（We are defining a main function in this programme and calling simply with the command main().）

In this case we are asking the user for a number n with the command input. We then use this number to calculate a sequence given by the square of the numbers from 0 to n and assign it to the variable y. Finally we simply print the list stored in y.

（Notice that we have used n+1 for xrange.）

（我们在这个程序中定义一个main函数，并使用命令main（）调用。）

在这种情况下，我们通过命令行询问用户来输入数字n。 然后，我们使用此数字来计算由0到n的数字的平方给出的序列，并将其分配给变量y。 最后，我们只需打印存储列表y。

（注意我们已经使用n + 1作为xrange。）

Remember that we have saved the script above, but we have not executed it. We can do this by typing the following command in a terminal in the appropriate path:

请记住，我们已经保存了上面的脚本，但我们还没有执行它。 我们可以通过在包含该脚本所在路径的终端中键入以下命令来完成此操作：

Page102图2

（In this case we have given the value n = 4 as an input.）

This is perhaps not the most advanced algorithm to implement, but we can surely see the possibilities. In particular, we can see how we can create scripts to add further functionality to our code and as such the concept of a module becomes natural.

（在这个用例中，我们给出了值n = 4作为输入。）

这可能不是最优算法，但我们可以肯定脚本起码可用。 特别是，我们可以看到如何创建脚本并向其中添加更多功能，因此自然而然的引出了下面模块的概念。

A module is a file or collection of files containing related Python functions and objects to achieve a defined task. These modules enable us to extend the capabilities of the language, and create programmes that enable us to carry out specific tasks. Any user is able to create their own modules and packages and make them available to others. Some of these modules are readily available for us to be used and once appropriate installation is done all we need to do is import them whenever we need to use them.

模块是包含相关Python函数和对象的单一文件或文件集合，用以实现既定的任务。 模块使我们能够扩展python语言的功能，并创建使我们能够执行特定任务的程序。 任何用户都可以创建自己的模块和包，并将其提供给其他人使用。 一旦python安装完成，其中一些模块已经可以随时可供我们使用了，我们所需要做的就是在需要使用它们时导入即可。

(A module is a file containing related Python functions to achieve a specific task.)

For example, we can use the math module to access some common mathematical functions. Let us create for instance a script that implements a function to calculate the area of a circle. In this case we will need the mathematical constant p to carry out the calculations:

（模块是包含相关Python函数的文件，用于实现特定任务。）

例如，我们可以使用数学模块来调用一些常见的数学函数。 让我们创建一个脚本来实现一个计算圆的面积的函数。 在这种情况下，我们需要数学常数π来执行计算：

Page103图

（The math module contains some common mathematical functions.）

（We can use the value of π with math.pi.）

Running the program will result in the following output:

（数学模块中包含一些常见的数学函数。）

（我们可以使用在模块math.pi 中调用常量π）

运行该程序将输出如下：

Page104图

Notice that we need to tell the Python interpreter that the constant p is part of the math module by using the syntax math.pi. In the example above we are importing all the functions of the math module. This can be somewhat inefficient in cases where only specific functionality is needed. Instead we could have imported only the value of p as follows:

请注意，我们需要通过使用语法math.pi告诉Python解释器常量p是数学模块的一部分。 在上面的示例中，我们将导入math模块的所有函数。 在仅需要特定功能的情况下，这可能有些低效。 相反，我们可以只导入π的值，如下所示：

Page104图2

(In some cases it may be more efficient to load only the needed functionality from a module.)

（在某些情况下，仅从模块加载所需的功能可能更有效率。）

A large number of modules are available from the Python Standard Library and more information can be found in https://docs.python.org/2/library/. In the rest of the book we will deal with a few of these modules and packages.

Python标准库提供了大量模块，更多信息可以在https://docs.python.org/2/library/中找到。 在本书的其余部分，我们将讨论其中的一些模块和软件包。

*2.4 Computation and Data Manipulation*

With the programming structures discussed so far we are ready to take up a large number of tasks, not only in data science, but in more general settings. In our particular case, as we shall see in the rest of the book, computation with data and its manipulation can be managed more effectively and easily with the aid of linear algebra. In this section we will address some basic concepts in both data manipulation and linear algebra with Python.

(Data manipulation and computation is a very important step in the data science and analytics workflow.)

*2.4.1 Matrix Manipulations and Linear Algebra*

As we have mentioned above, linear algebra enables us to carry out computational tasks with data in a very effective way. It also provides a compact notation to express the type of manipulations we need to do to our data, from pre-processing to presenting results. The use of vectors and matrices is therefore a very important area to cover. Vectors and matrices are arrays of numerical objects with a defined set of operations such as addition, subtraction, multiplication, etc.

(Linear algebra provides us with an efficient and compact way to carry out complex calculations.)

Page105

===》 要翻译到122页 还差50页，一天翻译10页。。