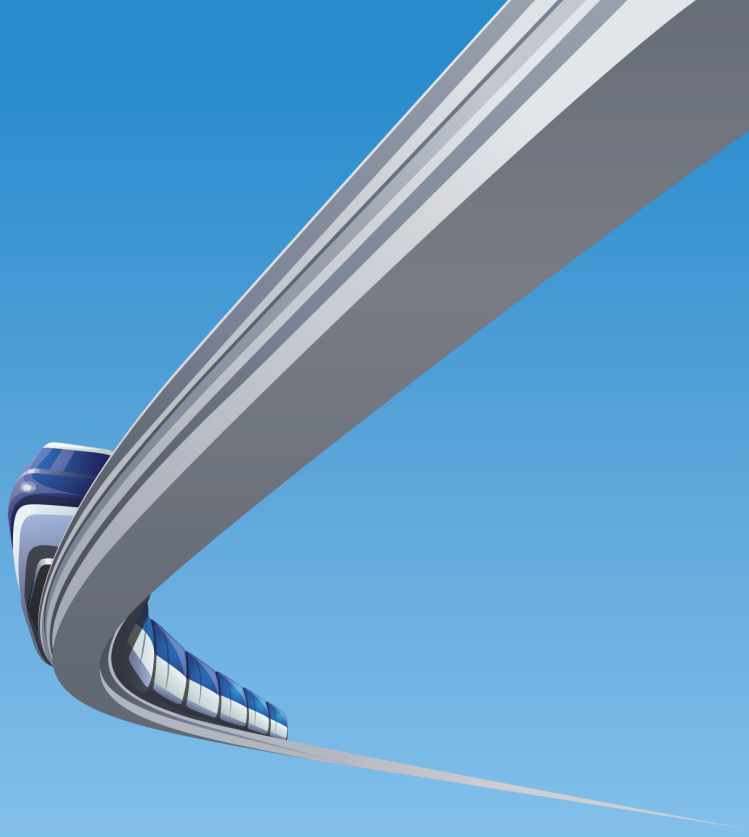


通过实例学Ruby编程

詹智敏　詹沁萌

通过实例学 Ruby 编程

Zhimin Zhan and Courtney Zhan

This book is for sale at http://leanpub.com/learn-ruby-programming-by-examples-zh-cn

This version was published on 2017-04-04

This is a Leanpub book. Leanpub empowers authors and publishers with the Lean Publishing process. Lean Publishing is the act of publishing an in-progress ebook using lightweight tools and many iterations to get reader feedback, pivot until you have the right book and build traction once you do.

Contents

前言

2013 年 12 月 8 日，美国总统奥巴马在电视讲话里“要求每一个美国人都要尝试学习写代码”（观看视频[1]），以此拉开计算机科学教育周刊 2013 年发起的” 每天学习代码一小时” 的活动序幕. 奥巴马说：“学习这些技能不仅对你的未来十分重要，它对国家的未来也意义重大.”

在 2013 年 2 月，“马克·扎克伯格，比尔·盖茨和其他几个 IT 业巨头号召孩子们学习编程”(视频[2]). 我特别喜欢在视频开头引用的一句话：

> “在这个国家, 每个人都应该学习如何使用电脑编程 ..., 因为它能教会你如何思考.” — 史蒂夫·乔布斯

他们要传达的信息：编程是这个信息时代的一项重要技能，当然，它不只限于美国人. 除了强调编程的重要性，这些 IT 巨头也试图在传达，“你也可以做到”.

作为一个软件工程师和自动化软件测试顾问，我曾和许多手动测试人员 (software tester) 共事. 在很多人眼里，测试并不算是一个有趣的职业. 有的人甚至把手动测试人员叫做稍带歧视性的“测试猴子”(Testing monkey), 暗指他们总是干重复的工作. 然而，自动化测试工程师，使用程序化的测试脚本来驱动应用软件，是一个有趣味，有创造性和令人满意的工作（事实上，它被评为2012 年美国最幸福的职业[3]，请注意，不是” 之一”）. 拥有这些技能的测试人员在如脸书 (Facebook) 和谷歌 (Google) 这样的世界顶级高科技公司都得到高度的重视，而编程是就其中最重要的技能.

当我把使用测试脚本来进行自动化测试的方法介绍给手工测试人员时，我马上可以感觉到他们的恐惧：“编写程序太难”. 这个反应是很典型的和常见的. 不要让人云亦云的“太难”挫败你的学习兴致. 我告诉你，编程不但不难，而且很有趣.

学习编程，从某种意义上说，就是掌握与计算机进行沟通的方式，通过指令让它们为你执行任务. 程序语言是我们编写代码时用的一组格式和规则，计算机能理解并运行它. 当前流行的编程语言有 Java，C#，Ruby 和 PHP 等. 对于初学者来说，不要抱着非要学某一个所谓最流行的编程语言不可的想法. 计算机内部的工作方式是相同的，如果你掌握了编程的思维方式，使用什么编程语言并不重要. 用一个可能不太准确的比方，众多编程语言像是中国各地的方言, 而编程思维方式是承载中华文化的汉字. 我学习并使用过十几种编程语言，我的经验是一旦你掌握了一种语言，再学习另一种是很容易的.

---

[1] https://www.adafruit.com/blog/2013/12/09/president-obama-calls-on-every-american-to-learn-code/

[2] http://www.psfk.com/2013/02/mark-zuckerberg-bill-gates-coding-school.html

[3] http://www.forbes.com/sites/jacquelynsmith/2012/03/23/the-happiest-jobs-in-america/

在这本书里，我将使用 Ruby，当下很流行的一种编程语言. 很多 Ruby 程序员谈及 Ruby 时，用“喜爱 (love)”而不是“喜欢 (like)”来描述它. Ruby 被广泛应用于企业业务应用程序和软件测试中，AirBNB 就是用 Ruby 开发的. 我选择 Ruby 的主要原因是，它很简洁且优雅. 正因如此，学习者可以把精力集中在思考上，而不是在语法上.

> **简历上最有价值的编程技术 - Ruby**
>
> 根据 Buring Glass 与布鲁金斯学会的经济学家乔纳森·罗斯韦尔根据从数以千计的美国招聘广告（2014 年 7 月）的研究数据, Ruby on Rails 是最有价值的编程技技术，平均工资为 109,460 美元[a].
>
> [a] http://qz.com/298635/these-programming-languages-will-earn-you-the-most-money

我激励我 13 岁的女儿 Courtney（在美国总统奥巴马的视频的帮助下）使用这本书学习编程. 她是这本书的第一个读者. 实际上，我在这本书中记录了她的想法和问题，以及她的一些完成的代码练习. 我想 Courtney 所犯的错误和遇到的困难可能会对他人有帮助. Courtney 还设计了书的封面和给所有的问题画了可爱的插图. 这使她当之无愧地成为这本书的第二作者.

本书的独特之处

一个典型的教授如何编程的书将通过编程概念，语法和用简单的例子做示范等方式来教学. 我以前看过几十本这样的编程语言或工具书，而且在大学任教时也是用这种方式教学. 但我认为这并不是一种有效的方法，因为它更像是一个老师对学生的知识倾倒. 我当时不知道有更好的办法，直到我发现了米歇尔·托马斯方法[4].

米歇尔托马斯方法是由米歇尔·托马斯开发的教授外语口语的方法. 托马斯声称，他的学生可以“在三天内达到的会话能力在其他任何大学用两三年的时间都无法达到”. 我对这个方法的理解是，老师开始一个简单的对话场景，然后逐步扩大每个场景，每次增加几个新词汇. 这样，学生熟悉谈论的话题和大部分单词/句子，同时通过实用有趣的对话，学习一些新的词汇.

我相信这种语言教学方法同样可以适用于编程. 不仅编程语言被称之为“语言”，而且编程是实践性极强的科学. 学习语言所做的“对话”就像是编程的练习. 当人们从学习中得到满足和反馈：看到他们的程序在电脑中运行，成就感会让他们学得更好.

没有什么比在实际的练习中应用编程更好的学习编程办法. 在这本书中，我选择了非常简单易懂的练习. 除了教学价值外，这些练习也是既有用处又有趣味性的.

市面上有不少如编程题集之类的书籍，我经常发现有些这样的练习又长又难以理解. 很常见的是，作者似乎喜欢炫耀自己的编程技能和巧妙的解决方案. 而这本书的不同之处在于: 通

[4] http://www.michelthomas.com

过练习循序渐进地教授实用的编程技能和思考方式. 在完成所有的练习后，你将能够编写可运行的程序（可能还不是完美的）并有信心继续学习和提高.

在本书第 11 章（自动化测试）中，我会告诉你从这本书所学到的知识可能会引导你找到一个有前途的职业：Web 应用程序的自动化测试工程师. 时下 Web App 已成为主流，它的快速更新和多浏览器支持表明自动化测试是唯一可行的解决方案. 然而，很少有人具备这样的技能. 编程 + 自动化测试是现代软件公司，如 Facebook，特别需求的技能.“所有的 Facebook 工程师负责给他们的代码编写自动测试”[5].

谁应该读这本书？

所有人，无论是工作需要，寻求转行，写应用程序或者游戏，甚至是为了更好地了解计算机程序的工作原理，都可以从此书中受益.

我尤其希望年轻人来一展身手.

英文术语

我们都知道，和计算机相关的很多术语来自英文, 如 CPU, USB, iPhone 等. 虽然在翻译成中文时，我尽量准确，但我建议即使对初学者来说，也最好知道其英文术语. 这样会更容易理解其涵义. 其实换个角度来讲，在学习另一个文化的过程也很正常，比如我的孩子在澳洲参加跆拳道班，他们的教练（当地澳洲人）喊的动作口令用的都是韩语.

在书中，大多数编程术语我都会在后面的括号里用英文注明，有的术语翻译成中文后可能更难理解，如哈希表，那我就索性直接用英文 Hash.

如何读这本书

我强烈建议按章节顺序来阅读这本书. 该书的练习被分为 11 章，每章一般遵循由易到难的模式.

所有练习的解决方案，也可在本书网站[6]上找到.

反馈

我们欢迎读者在本书网站上提出宝贵意见，以便今后不断改进，使之更趋于完善.

詹智敏, 詹沁萌 *(Courtney Zhan)*

澳大利亚，布里斯班

[5] http://www.theinquirer.net/inquirer/news/1720797/facebook-qa-team

[6] http://zhimin.com/books/learn-ruby-programming-by-examples

1. 介绍

我还记得我的第一次编程课. 老师说的开场白是“电脑并不神秘”. 现在已没有人使用’ 神秘’ 来形容计算机了. 但是，在 20 世纪 80 年代计算机难得见到的情况下，事实确实如此.

我们现正处在“信息时代”，电脑已经是我们现代生活的一部分. 但对大多数人来说，尽管他们大部分的工作时间是坐在电脑前或使用智能手机，但是对电脑程序如何工作仍然觉得神秘，且知之甚少.

其实你一旦掌握了编程，就可以用它来做很多事，例如:

- 用脚本 (Scripts) 快速给数百种文件改名，而不是逐一的改
- 从原始数据直接产生一个漂亮的 Excel 报表，而不是费时费力地键入这些数据
- 创建只一种文件，使用脚本来生成几种不同的文件格式如 HTML(网页) 和 PDF
- 在满足特定条件下，自动打开或关闭某些电子设备
- 写一个很酷的 iOS 或安卓 (Android) 应用

至少，当你知道了软件的工作原理，你一定可以更好地使用软件.

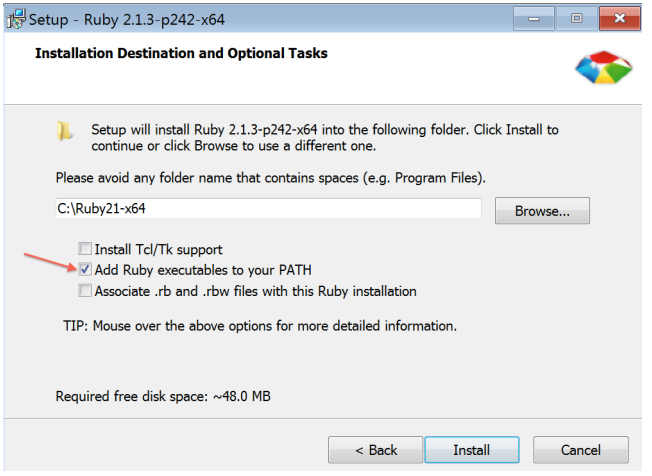
在我们开始之前，就像当初我的老师一样，我也要告诉你，“编程并不神秘”，你可以掌握它. 我相信这本书可以引导你进入精彩的编程世界.

像许多技能一样，你不能只通过阅读这本书来掌握编程，你需要实践. 现在就让我们开始吧.

1.1 在 Windows 电脑运行 Ruby 程序

首先，我们需要安装 Ruby. 在Ruby Windows 安装包[1]网站下载 (最新版本的 Ruby 是 2.3.3) 并安装. 勾选安装对话框窗口中的“添加 Ruby 可执行文件到您的 PATH”，其他设置均接受默认即可.

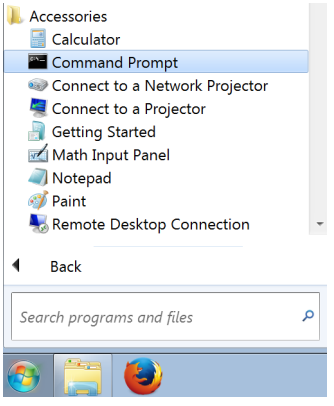
[1] http://rubyinstaller.org/downloads



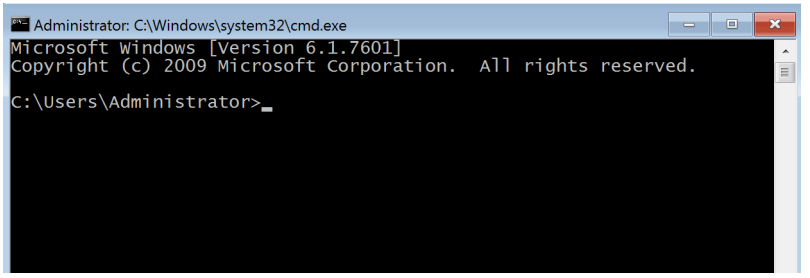
打开命令行

与程序进行交流的最佳方式是通过命令行，你可能已经在一些好莱坞电影中看到这些场景：黑客高手在某些文本窗口输入一些命令 (而不是使用鼠标)，就会有大事发生. 接受用户的文本命令的窗口被称为控制台 (Console) 或命令窗口 (Command Window).

在 Windows 中的启动命令提示符，点击“开始”-> “所有程序”->“附件”->“命令提示符”（Command Prompt）.



你会看到类似下面的一个新窗口.



在这个黑色的窗运行 `ruby -v`，再按 `Enter` 键.

```
C:\Users\Administrator>ruby -v
ruby 2.1.3p242 (2014-09-19 revision 47630) [x64-mingw32]
```

如果你得到的类似上面一样，这意味着 Ruby 安装成功，可以使用了.

选择一个 Ruby 编辑器

文本编辑器是用于编辑文本 (可识别的字符) 的工具, 例如，记事本 (NotePad) 就是一个文本编辑器. 因为程序代码是文本，在技术上讲，我们可以用记事本写代码. 然而，这将是非常低效的. 程序员一般使用代码编辑器或集成开发环境 (IDE). 简单起见，我建议初学者使用一个支持 Ruby 的程序编辑器，以下都是不错的选择.

商业

- Sublime Text[2]. 一个强大的程序员的编辑器. 费用：70 美元.

免费

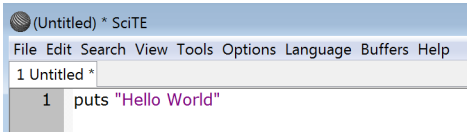
- SciTE[3]. 一个免费且轻便的程序员编辑器. 它有几种不同的安装包，最简单的可能是 Windows 安装程序 (scite-3.3.9x64.msi 约 2.7MB).
- Visual Studio Code[4]. 微软最新推出免费的程序编辑器，功能强大.

撰写您的第一个 Ruby 程序

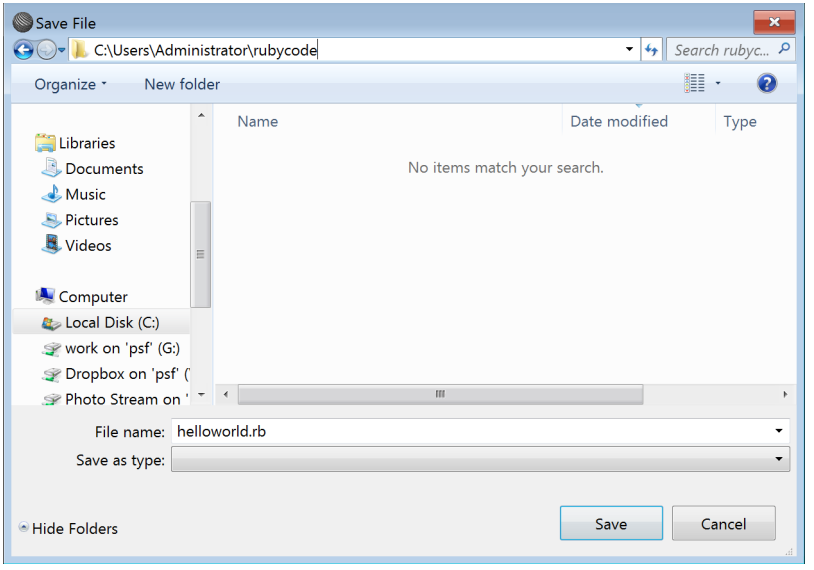
我建议建立一个专门的文件夹储存你所有的代码，例如 `C:\Users\you\rubycode`.

打开你的编辑器 (我将用免费的 SciTE 作说明)，并输入 `puts "Hello World!"`. （`puts` 把接下来的文本输出到屏幕上）

[2] http://www.sublimetext.com/
[3] http://www.scintilla.org/SciTE.html
[4] https://code.visualstudio.com/



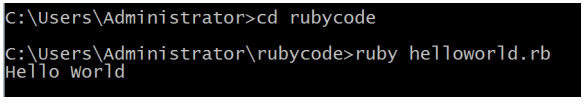
将文件保存到 `C:\Users\you\rubycode\helloworld.rb`



运行程序

现在我们来运行程序. 打开命令提示符，转到 `rubycode` 文件夹，然后键入命令 `Ruby helloworld.rb`.

```
> cd rubycode
> ruby helloworld.rb
```

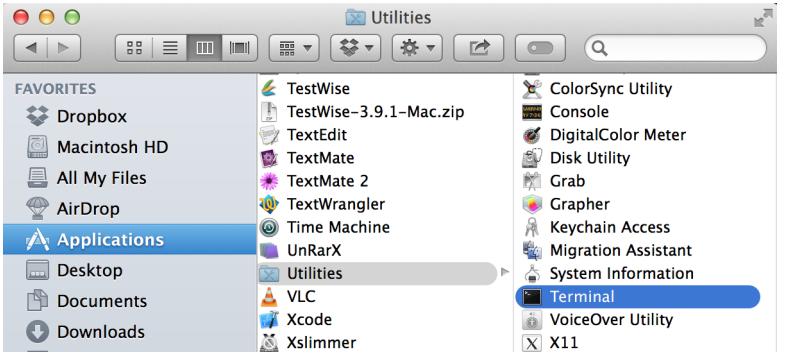


1.2 在苹果电脑 (Mac OS X) 上运行 Ruby 程序

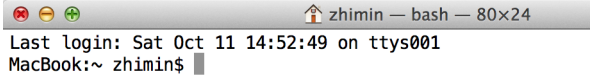
Mac OS X 已预装了 Ruby.

打开命令行

访问 Mac OS X 中的命令行应用程序被称为“终端”(Terminal). 在 Finder 中打开：“应用程序” - > “工具” - > “终端”.



它看起来应该是这样的：



键入 `ruby -v`，然后按 `Enter` 键

```
MacBook:~ zhimin$ ruby -v
ruby 2.1.3p242 (2014-09-19 revision 47630) [x86_64-darwin13.0]
```

在你的电脑上的 Ruby 版本号可能不同，但这没有关系.

选择一个 Ruby 编辑器

商业

- TextMate[5]. 它被称为“苹果的编辑器”，并在 2006 年获得了苹果最佳开发工具奖，它很受 Ruby 程序员的青睐. 费用：€39
- Sublime Text[6].Sublime Text 是由 TextMate 得来的灵感. 它们在许多方面是非常相似的. 费用：US$70.

TextMate 和 Sublime Text 均提供免费试用.

免费

- TextWrangler[7].

[5] http://macromates.com/
[6] http://www.sublimetext.com/
[7] http://www.barebones.com/products/textwrangler/

你的第一个 Ruby 程序

我建议建立一个专门的文件夹，储存你所有的代码，例如`/Users/YOURUSERNAME/rubycode`.

打开你的编辑器 (我建议使用 TextMate，但现在我用免费的 TextWrangler 说明)，然后键入 `puts "Hello World"`. `puts` 输出下面的文本到屏幕上.

保存到我们所选择的文件夹 `rubycode`, 名称为“helloworld.rb”.

保存后，你会发现，文字的颜色改变了. 这就是所谓的语法着色 (Syntax Highlighting)，编辑器现在“知道”它是一个 Ruby 程序 (根据扩展名*.rb*) 并着色相应的代码. 这将使得代码更易于阅读和发现错误.

运行程序

现在我们运行程序. 打开终端 (Terminal)，进入 `rubycode` 文件夹，然后键入命令 `ruby helloworld.rb`.

```
$ cd rubycode
$ ruby helloworld.rb
```

(`cd` 的意思是”更改目录”;`ruby filename` 运行一个 ruby 文件)

你会看到输出:

```
Hello World!
```

1.3 在线 Ruby 教程

虽然我相信你可以从这本书中学会基本的 Ruby 编程，你也可利用在线教程作为补充. 例如，在公交车站等车时，在你的 iPad 上阅读. 这里我将介绍两个很好的且免费的网上教程.

1. 20 分钟体验 Ruby

   20 分钟体验 Ruby (Ruby in Twenty Minutes)[8] 是官方的简单的 Ruby 教程，顾名思义，只需 20 分钟左右就可浏览完毕.

[8] https://www.ruby-lang.org/en/documentation/quickstart

2. Codecademy 的“介绍 Ruby”课程

   Codecademy[9]是一个提供免费的互动编码课程的网站. 其中之一是“介绍 Ruby”. 除了解释概念，教程也提供一些简单的可以编辑和提交代码的练习.

如果我可以从网上教程来学 Ruby，为什么还要用这本书？

在线教程教你基本的 Ruby 语法和一些编程的概念. 尽管它们很重要，这些知识只有在付诸实施时才是有用的. 举例来说，一个很好的棋手，仅仅知道国际象棋的规则是不够的.

编程技能，在我看来，是一种在电脑上体现出来的解决问题的能力. 这方面的知识只可以通过实际的编写程序才能得到，这是这本书的目的所在. 在线教程，尤其是视频教程，是把学习者放在被动模式上. 你需要像这样的一本书通过练习把被动的知识变成你自己的知识.

事实上，Courtney 在学完 Codecademy 的 Ruby 教程后，当开始真正编写程序时，还是无从下手.

1.4 做练习的步骤

每个练习有以下 6 个部分：

- 问题. 它通常是用容易理解的示例输出. 请确保你理解透彻.
- 目的. 你可以从这个练习中学到什么.
- 分析. 像程序员一样来分析问题. 这是一个重要的步骤. 很多时候，我们知道该怎么做，但无法描述得很好. 以数字排序为例，你可以立即在你的头脑中解决 5 个数字的排序. 但如何对 100 个号码排序？这需要把你解决问题的步骤一步步翻译成计算机可执行的程序.
- 编写代码. 没有人可以通过阅读学习编程，必须要通过真正动手做. 你可以参考提示 (下一节) 来协助你理解.
- 提示. 列出的提示 (Ruby 代码格式) 可以帮助你在困惑时解决问题.

如果你尽力了，但仍然没有找到一个解决方法，可以查看我们的解决方案 (见附录 II). 这完全没有问题. 事实上，这些练习都是经过精挑细选的以便推出新的编程技巧/做法以及复习以前所学的知识. 所以如果你不能在第一次做对，这很正常. 你将有机会在以后的练习中应用它. 无论你做得正确与否，只要你坚持下去，你就是在学习.

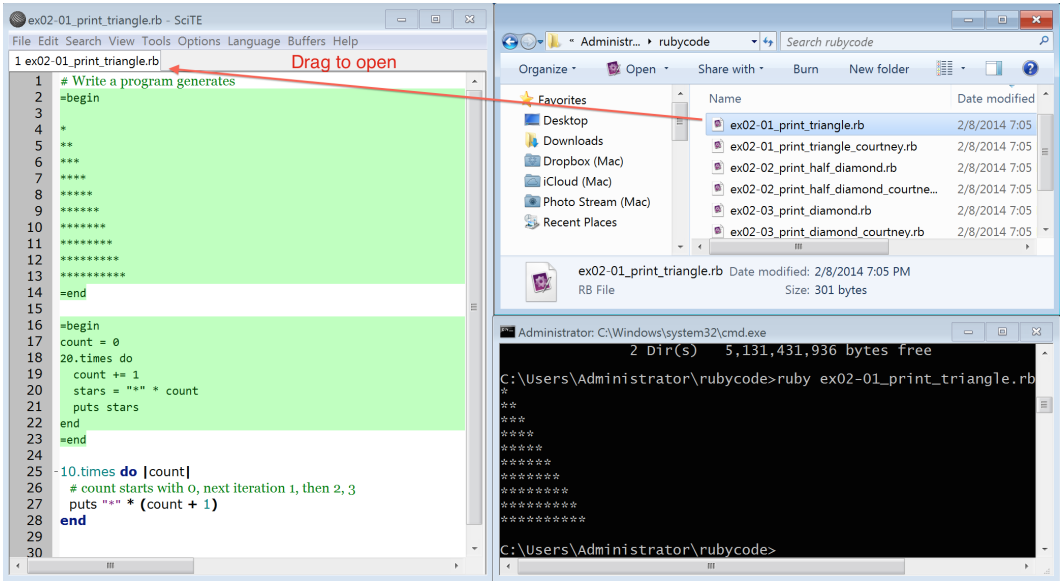
- 解决方案. 解决方案 (可在附录 II 中找到)，对多数练习来说，代码都是在 10 — 20 行之间. 我会首先显示 Courtney 的解决方案，以及她的体会. 可运行的解决方案脚本可以在本书的网站上下载.
- 复习. 想想你已经学到了什么.

[9] http://www.codecademy.com

1.5 窗口布局建议

为了让您更容易编写和运行你的代码，我建议你打开 3 个窗口，如图所示：

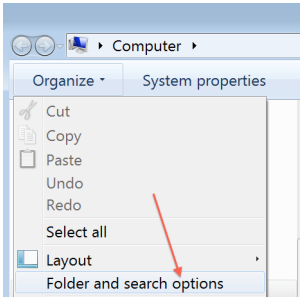
- 左侧是代码编辑器，在那里你编辑代码.（我在下面截图中使用免费的 SciTE）.
- 一个 `rubycode` 文件夹打开的窗口浏览器.
- 命令提示符（在 Mac 或 Linux 上的终端）窗口与当前目录设置为 `rubycode` 文件夹（执行命令 `cd C:\Users\you\rubycode`）



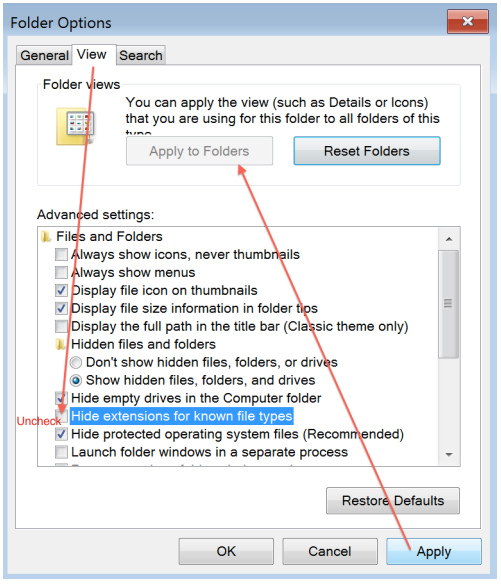
下面是编写和运行一个新的程序的步骤（*new_code.rb*）.

1. 在窗口浏览器的窗口中，单击鼠标右键，新建一个文本文件，并将其重命名为 `new_code.rb`. 更改文件扩展名'.rb' 是很重要的. 在 Windows 中，文件扩展名是默认隐藏在 Windows 的资源管理器中. 要更改窗口资源管理器中的文件扩展名，我们需要改变这个设置（显示文件扩展名）. 以下是 Windows 7 中的具体步骤.

    - 在窗口管理器的窗口中，选择“组织”->“文件夹和搜索选项”



- “查看”选项卡，取消勾选“隐藏扩展名已知文件类型”的复选框



- 点击“应用”按钮
- 为了把它作为默认设置（推荐），点击“应用到文件夹”按钮.

2. 将 *new_code.rb* 文件拽到编辑器中.
3. 在编辑器中输入和编辑代码，在它已准备好运行时保存文件.
4. 在命令提示符窗口中，键入

```
ruby new_code.rb
```

   然后按’Enter‘键运行该程序.

   快速重新运行程序的方法：按“上箭头”键，得到上次的命令.
5. 如有必要，重复步骤 3 和 4，直到得到您满意的结果.

1.6 常见错误

程序员（无论是新手还是有经验的）每一天都会经历代码错误. 不要期望你第一次的练习就做对，我们都是从错误中学习.

语法错误

我们在编写代码时打错字是正常的. Ruby 在运行之前先检查代码的语法. 如果在代码中有语法错误，程序运行中断并显示错误. 这些信息对帮助识别错误非常有用. 例如，在下面代码的第 23 行，我输入'elwe'，而不是“else”.

```
21  if row < 8
22    star_count = row + 1
23  elwe
24    star_count = (15 - row)
25  end
```

当我运行程序时，得到以下错误信息：

```
ex02-02_print_half_diamond.rb:23:in `block in <main>': undefined local varia\
ble or method `elwe' for main:Object (NameError)
from ex02-02_print_half_diamond.rb:20:in `times'
from ex02-02_print_half_diamond.rb:20:in `<main>'
```

该错误消息意味着 `elwe` 没有定义（不要担心，如果现在对你来说它没有意义，你很快就会明白的）. 更有用的部分是在错误跟踪里显示的行号'23'. 这会有助于识别错误的所在位置.

没有匹配的括号

就像数学，如果有一个左括号“(”在代码中，应当有一个匹配的右括号“)”. 代码中也有匹配关键字的结构，如 `if...end`. 在下面的代码中有两个错误.

```
20 15.times do |row|
21   if row < 8
22     star_count = row + 1          missing ')'
23   else
24     star_count = (15 - row
25   puts '*' * star_count          Missing matching 'end'
26 end
27
```

1. 在第 24 行：缺少 ')', 应该是 `(15 - row)`.
2. 在第 25 行：缺少对应第 21 行的 `if` 的 `end`

运行该程序，显示错误消息：

```
ex02-02_print_half_diamond.rb:27: syntax error, unexpected keyword_end, expe\
cting ')'
```

确实，代码缺少右括号 ` expecting ‘)’ `. 但是，行号 27 不是实际错误所在. 这是因为，Ruby 是不可能检测到所有错误. 如果右括号在下一行，程序依然是有效的. 只有第 27 行，基于所述代码之后，Ruby 判定右边括号确实丢失.

改正第一个错误后，运行该程序，会看到第二个错误.

```
ex02-02_print_half_diamond.rb:27: syntax error, unexpected $end, expecting k\
eyword_end
```

再次，显示的错误行数并不是真实错误的确切位置.

代码逻辑错误

以上语法错误是比较容易被发现的. 对于程序员来说，发现代码的逻辑错误则相对困难得多. 也就是说，代码因为没有语法错误可以运行，但得不到预定结果. 查找和解决代码错误的能力把优秀的程序员和一般的程序员分开. 但随着实践，你会做得越来越好.

对于初学者来说，我有两个简单实用的建议.

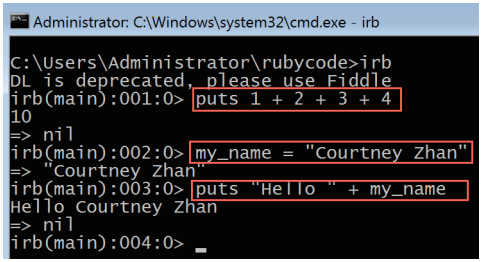
1. 一步一步地来（循序渐进）.

   写一行代码，立即运行代码. 这听起来很无趣，但在实践中，许多初学者发现它是学习编码的最有用的技巧. 如果新增加或变更的代码片段引发错误，点击（在编辑器）“撤消”按钮，会将你带回到你的代码的以前的工作状态.
2. 如果感到迷惑，重新开始.

   如果您卡在现有的代码中，很可能是代码的复杂性超出你的控制.
   靠猜测使电脑按照你编写的代码的指示工作是几乎不可能的.
   在这种情况下，最好是从头开始. 这本书中大多数的练习，程序只有15行代码左右.

1.7 交互式 Ruby (IRB)

IRB 是一个 Ruby 自带的工具，它允许运行 Ruby 代码片段并提供即时的反馈，这对于学习 Ruby 语法和调试代码中的错误非常有帮助. 要运行 IRB，只需从命令行窗口运行 `irb`，然后在那里尝试运行 Ruby 代码.



在上面的截图中的绿框中的命令是我输入的. 其它的均是从命令返回的响应.

在附录 1 中（“Ruby 概论’）我总结了核心 Ruby 的语法和在实例中的使用情况. 这些例子可以在 IRB 中快捷地运行.

2. 打印形状

打印出星号字符（*）经常被用来作为初学者的编程练习，因为它们简单，易于理解且直观有趣.

2.1 打印出三角形

编写一个程序，打印出以下面的星号形成的三角形：

```
*
**
***
****
*****
******
*******
********
*********
**********
```

目的

- 分析模式
- 使用循环

分析

行	'*' 数目
1	1
2	2
3	3
...	...
n	n

提示

打印字符

```
puts '*'
puts "**" # 将在单独一行
```

你可以在 IRB 尝试代码

可以试用 Ruby 的乘号 * 生成多个同一字符. 请注意在编程语言中数学乘号和星号是同一字符，不过在使用上是很容易区别的. 作星号时用引号括起来.

```
# 打印出多个“$ 号
'$' * 3 => '$$$'
```

用变量来存储一个整数

```
star_count = 2
puts '*' * star_count          # => '**'

star_count = star_count + 1    # 现在 star_count => 3
puts '*' * star_count          # => '***'
```

在一个固定次数循环内打印出多个同样字符.

```
5.times do
  puts '*'
end
```

do...end 标示循环的开始和结束，

在计算机上做出你的的解决方案

在动手前，请确保您了解 分析和 提示部分.

Courtney 的版本

```
count = 0
10.times do
  count = count + 1
  stars = "*" * count
  puts stars
end
```

2.2 打印半个菱形

编写一个程序，打印出用星号拼成的一个菱形的右半边（仍然是一个三角形）.

```
*
**
***
****
*****
******
*******
********
*******
******
*****
****
***
**
*
```

目的

- 在循环内递减计数

分析

问题的关键是要确定每行的相应的星星数.

```
1 至 8 行：与行相同
9 到 16 行：16 - 行数
```

提示

使用 **if ... else** 控制代码执行

代码，在它最简单的形式中，是按从上到下顺序执行. 但也有 `if` 条件和循环（以及后来的 Method 和 Class）语句改变执行流程. 条件表达式（IF-THEN-ELSE 语句）根据布尔 (Boolean) 条件（真或假）运行不同的代码语句.

```
score = 75
if score < 60
  puts("Failed!")
else
  puts("Pass!")
end
```

输出结果:

```
Pass!
```

如果你把第一行改为 `score = 59`，然后再次运行, 就会得到 “`Failed!`”.

布尔条件

在 `if` 之后的 `score < 60` 被称为布尔条件，它的值只能为’true’ 或’false‘（被称为布尔值）.
比较常见的 Ruby 比较运算符:

==	等于
!=	不等于
<	小于
<=	小于或等于
>	大于
>=	大于或等于

例如:

```
2 > 1   # => true
2 == 1  # => false  (等于)
2 != 1  # => true   (不等于)
2 <= 2  # true
```

Courtney 的版本

```
count = 0
8.times do
  count += 1                # 这相当于 count = count + 1
  stars = "*" * count
  puts stars
end
count = 10
8.times do
  count -= 1
  stars = "*" * count
  puts stars
end
```

Courtney 的版本循环 16 次（8 + 8），但打印出来结果依然正确（15 条线）. 这是因为当 count 被减为 0 时，没有星星被打印出来，用一个空行来代替.

Courtney 使用两个循环，这对于初学者来说，是很合乎逻辑的，并没有错.

2.3 打印菱形

打印以下用 7 行星号组成的菱形：

```
   *
  ***
 *****
*******
 *****
  ***
   *
```

目的

- 分析更加复杂的模式

分析

这和以前的练习有三个不同：

1. 中间行之前一行的星号数是 `(行 - 1)* 2 + 1`.
2. 中间行之后一行的星号数是 `(总行数 - 当前行数)* 2 + 1`
3. 在 `*` 星号之前有空格

提示

写下有多少空格（在前面）及每一行的星号数.

```
第 1 行: 打印 3 个空格 + 1 星
第 2 行: 打印 2 个空格 + 3 星
第 3 行: 打印 1 个空格 + 5 星
第 4 行: 打印 0 个空格 + 7 星
第 5 行: 打印 1 个空格 + 5 星
第 6 行: 打印 2 个空格 + 3 星
第 7 行: 打印 3 个空格 + 1 星
```

如果有困难，就一步一步来. 你可以尝试先打印出上面的三角形.

Courtney 的版本

```
space = " "
space_count = 4
7.times do |row|
  if row < 4
    space_count -= 1
    star_count  = row * 2 + 1
    print space * space_count
  else
    space_count += 1
    star_count  = (7 - 1 - row) * 2 + 1
    print space * space_count
  end
  puts '*' * star_count
end
```

我对星号数和空格数很困惑，我不得不坐下来与爸爸一起列出数学公式. 另外，多个变量让人迷惑，所以一定要给变量尽可能准确地命名. 如果你感到困惑，你可以打印 (用 `puts`) 认为可能是问题的一个变量. 这可以帮助你明白是怎么回事以及如何解决它.

Courtney 使用变量 `space` 表示一个空格字符串 (String)，这是一个很好的做法.

2.4 打印指定大小菱形

让用户输入菱形的尺寸（总行数），然后打印出一个由星号组成的菱形.

```
输入行数（奇数）：9
            *
           ***
          *****
         *******
        *********
         *******
          *****
           ***
            *
```

目的

- 读取用户的输入
- 将字符串转换为整数
- 使用控制循环次数的变量

分析

菱形的大小是不固定的，取决于用户输入的数字 (总行数). 也就是说：程序里的循环次数要根据用户输入的值的变量. 用两个步骤打印：顶部和底部，计算出每个部分的行数和星号的数目.

提示

读取用户的输入.

```
user_input = gets.chomp # 读用户输入存储在 user_input 中的字符串
"6".to_i # => 整数 6
```

在 Ruby 中，数学除法运算符用“/”表示.

```
8 / 2  # => 4
9 / 2  # => 4, 忽略余数
(1 + 2) * 3 + 3 / 2  # => 10
```

Courtney 的版本

```
puts "输入行数（奇数）:"
size = gets.chomp.to_i
space = " "
space_count = size / 2 + 1

size.times do |row|
  if row < (size / 2 + 1)
    space_count -= 1
    star_count  = row * 2 + 1
    print space * space_count
  else
    space_count += 1
    star_count  = (size - 1 - row) * 2 + 1
    print space * space_count
  end
  puts '*' * star_count
end
```

当我第一次试图以用户输入变量替换菱形的大小，我忘了改变变量，这让结果完全不同，然而纠正它并不难.

你必须先了解它，然后给电脑发指令

试着给上一个程序输入一个大数目，如 99. 这将打印一个非常大的菱形，哇！

编程的一个重要方面是，一旦你想通了一个问题的方式和逻辑，并把它正确地翻译成计算机可以理解的语言（程序），就可以解决任何规模的类似的问题. 例如，计算机计算 2×2 所需的努力和计算 12343 x 35345 时的没有太大的不同. 换句话说，我们（人类）首先必须明白如何解决问题. 编程，在某种意义上，是把‘如何’翻译成许多计算机能够理解的指令步骤.