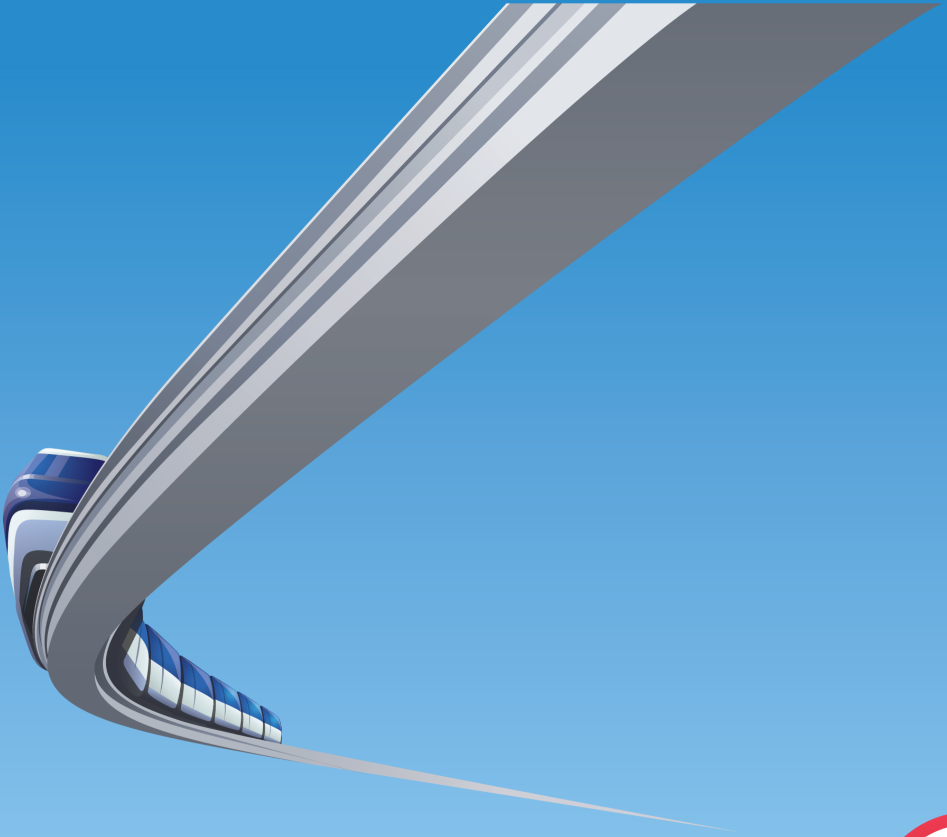


Learn Ruby Programming by Examples

Zhimin Zhan . Courtney Zhan

Learn Ruby Programming by Examples

Zhimin Zhan and Courtney Zhan

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

This version was published on 2018-09-23

Contents

Preface

On December 8, 2013, US President Barack Obama "asked every American to give it a shot to learn to code" (watch it here[1]), kicking off the Hour of Code campaign for Computer Science Education Week 2013. "Learning these skills isn't just important for your future, it's important for our country's future," President Obama said.

In February 2013, Mark Zuckerberg and Bill Gates and several other big names in IT want kids to learn code (video[2]). I particularly like the quote at the beginning of the video:

> **"Everybody in this country should learn how to program a computer… because it teaches you how to think." - Steve Jobs**

You don't have to be an American to get the message: coding (aka. programming) is an important skill for this information age. Besides the importance of programming, the other message those VIPs also tried to convey is that "you can do it".

As a programmer and software test automation coach, I have worked a lot with manual testers. Manual testing can be repetitive and boring. However, automated testing (using test scripts to drive software applications) is a fun, creative and satisfying job. Writing automated test scripts requires programming skills.

When I introduced the idea of automated test scripts to manual testers, I could immediately sense their fear: "*programming is too hard for me*". This reaction is very typical and common. Don't let the "too hard" phrase discourage you from learning. Let me tell you, programming is not that hard, and it is fun.

Learning programming is a way to master communication with computers, by giving them instructions to perform tasks for you. A programming language is a language that is used to write instructions for computers to understand and execute. There are several popular programming languages such as Java, C#, Ruby, and PHP. For beginners, don't fixate on one. Computers internally work the same way, mastering thinking in programming is more

[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

important than one language syntax. In my opinion, different programming languages are like dialects. I learned and taught myself over a dozen of programming languages. Once you have mastered one, it is easy to learn another.

In this book, I will use Ruby, a popular and elegant programming language. Ruby is widely used in enterprise business applications and software testing (*Twitter was initially developed in Ruby*). The main reason I chose Ruby is that it is concise. Therefore, learners can focus more on thinking rather than the syntax.

The most valuable programming skills to have on resume

According to the job research data, compiled from thousands of American job ads, by Buring Glass with Brookings Institution economist Jonathan Rothwell in July 2014, Ruby on Rails is the most valuable programming skills with average salary of $109,460[a]. Also, according to Mashable's findings from the CyberCoders database of hundreds of thousands of job postings, Ruby on Rails Developer is the one of top 5 lucrative tech careers to pursue in 2015[b].

[a]http://qz.com/298635/these-programming-languages-will-earn-you-the-most-money
[b]http://mashable.com/2015/02/22/highest-paid-tech-jobs-2015/

I motivated my 13-year old daughter Courtney to learn programming with this book (with the help of President Obama's video). She is the first reader of this book. In fact, I included her thoughts and questions in this book, as well as some of her finished code for the exercises. I think the mistakes Courtney made and the hurdles she faced could be helpful to others. Courtney also designed the book cover and cute illustrations for all the questions, which entitled her the co-author of this book.

What is unique about this book?

A typical how-to-program book will go through the programming concepts, syntax and followed by demonstrations with simple examples. I have read dozens of them (for different programming languages or tools) before and have taught this way at universities. It was not an effective approach. It is more like a teacher dumping knowledge upon students. But I did not know a better way, until I discovered The Michel Thomas Method[3].

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

The Michel Thomas Method is developed by Michel Thomas for teaching foreign languages. Thomas claimed that his students could “achieve in three days what is not achieved in two to three years at any college”. My understanding of this method is that the teacher starts with a simple conversation scenario, then gradually expands the scenario with a few new words each time. That way, students are familiar with the conversation topic and the majority of words or sentences, while learning some new, in real interesting conversations.

I believe this teaching method can be applied to programming. Not only a programming language may also be considered as ‘a language’, but also very practical. The ‘conversation’ in speaking languages are exercises in programming. People learn better when they get satisfaction or feedbacks and see their programs running.

As I said before, thinking in programming is much more important than being familiar with a programming language. There is no better way than writing real programs for practical exercises. In this book, I have chosen the exercises that are very simple to understand, besides teaching values, they are useful and fun to do.

Besides programming tutorial books, there are also programming quiz books. I often find some of those exercises are long and hard to understand. Quite commonly, the authors seem to be fond of showing off their programming skills or smart solutions. It won’t be the case in this book. This book is a guide to programming and its purpose is to teach. After you finish all the exercises, you will be able to write working programs, and with confidence to continue to learn and grow.

In Chapter 11 (Automation), I will show what you have learnt may lead you to a promising career: test automation engineer for web applications. Web applications are the main stream nowadays. Due to its nature of rapid changes and multi-browser support, automated testing is on demand. However, very few possess the skill. Programming + Automated Testing skills are highly valued in software companies like Facebook: “All Facebook engineers are responsible for writing automated tests for their code”[4].

Who should read this book

Everyone, for whatever reasons: job needs, career change, writing apps or games, or simply to better understand how a computer program works.

In particular, I would strongly encourage young people to give it a go.

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

How to read this book

It is highly recommended to read this book from page to page. The exercises are organized into chapters, exercises within each chapter generally follows an easy-to-hard pattern.

The solutions for all exercises are also available on the book's website[5], for access code see the Resources section of this book.

Send me feedback

We'd appreciate your comments, suggestions, reports on errors in the book and code. You may submit your feedback on the book's site.

Zhimin Zhan and *Courtney Zhan*

November 2014

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

1. Introduction

I still remember my first programming lesson. The first sentence the coach said was "computers are not mysterious". Nobody uses the term 'mysterious' to describe computers nowadays. It was the case in 1980's, computers were rare back then.

We are in the "Information Age" now, computers are a large part of our lives. It seems to me that programming remains mysterious and difficult to the majority of people despite the fact that they spend most of their working hours in front of computers.

Once you have mastered programming, there are many things you can do, such as:

- Instantly rename hundreds of file with a script instead of doing it one by one
- Generate a nice Excel report from raw data instead of typing it in
- Write a document once and use scripts to generate several different formats: HTML (online) and PDF
- Turn on or off certain electronic devices when a certain condition is met
- Write a cool iOS or Android App
- Develop a web application

The bottom line is that when you know how software works you will definitely use computers better.

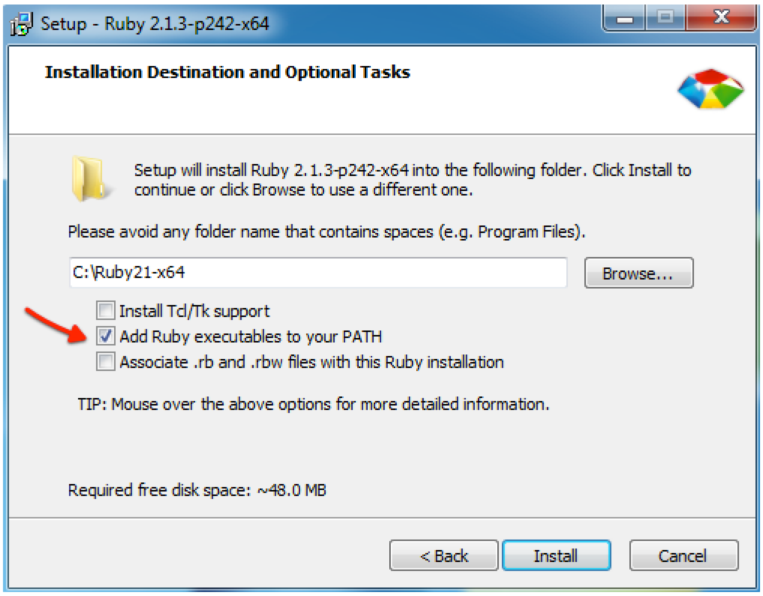
Before we start, just like my coach, I am telling you that "programming is not mysterious" and you can master it. I believe this book can guide you to the wonderful programming world.

Like many skills, you cannot master programming by reading the book only, you need to **do it**. Let's begin.

1.1 Ruby on Windows

First, we need install Ruby. Download the Ruby Installer for Windows[1] (the latest version to date is Ruby 2.1.3) and run it. Tick the 'Add Ruby executables to your PATH' checkbox on the installation dialog window (accept the defaults for the rest).

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


Setup - Ruby 2.1.3-p242-x64

Installation Destination and Optional Tasks

Setup will install Ruby 2.1.3-p242-x64 into the following folder. Click Install to continue or click Browse to use a different one.

Please avoid any folder name that contains spaces (e.g. Program Files).

C:\Ruby21-x64

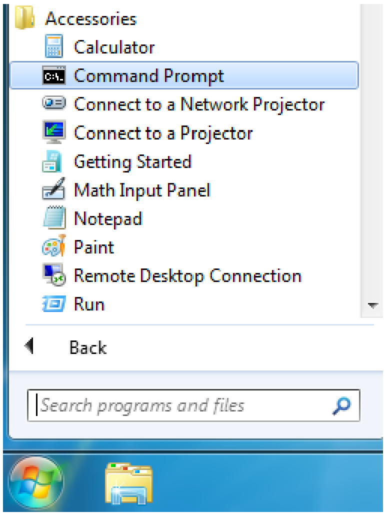
[ ] Install Tcl/Tk support
[✓] Add Ruby executables to your PATH
[ ] Associate .rb and .rbw files with this Ruby installation

TIP: Mouse over the above options for more detailed information.

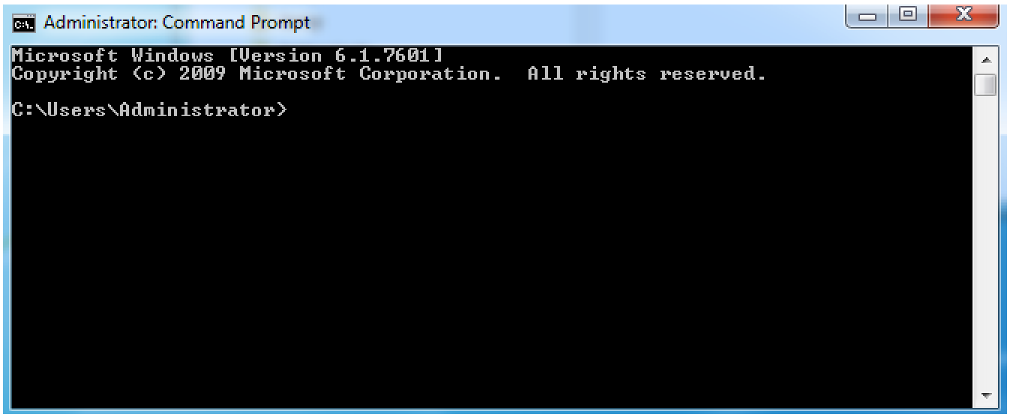
Required free disk space: ~48.0 MB

Open command line

The best way to interact with your programs is from the command line. You might have seen these scenes in some Hollywood movies: a hacker types some commands in a window, and something (usually big) happens. The windows that accept user commands (without using a mouse) are called consoles or command windows. To start a command prompt on Windows platform, press ‘Start’ → ‘All Programs’ → ‘Accessories’ → ‘Command Prompt’.

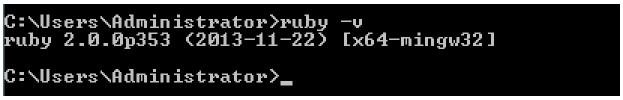


You will see a new window like the below.



```
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation.  All rights reserved.

C:\Users\Administrator>
```

Type 'ruby -v' in this black window, then press `Enter` key



```
C:\Users\Administrator>ruby -v
ruby 2.0.0p353 (2013-11-22) [x64-mingw32]

C:\Users\Administrator>_
```

If you get the output like the above, that means the Ruby is installed successfully, and is ready to use.

Choose a Ruby Editor

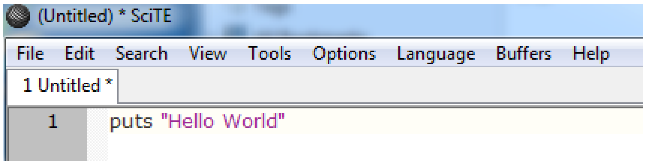
Text Editor is a tool for editing texts (e.g. NotePad). As all the code is in text format, we may use NotePad to write our code in, however, that would not be effective. I recommend the free code editors below:

- Visual Studio Code[2]. A powerful programmer's editor from Microsoft, free.
- SciTE[3]. A free and lightweight programmer's editor. There are several different packages, the easiest one is probably the windows installer (scite-4.0.0x64.msi around 3.2MB).

Write your first Ruby program

I suggest creating a dedicated folder to put all your code in, for example, `C:\Users\you\rubycode`.

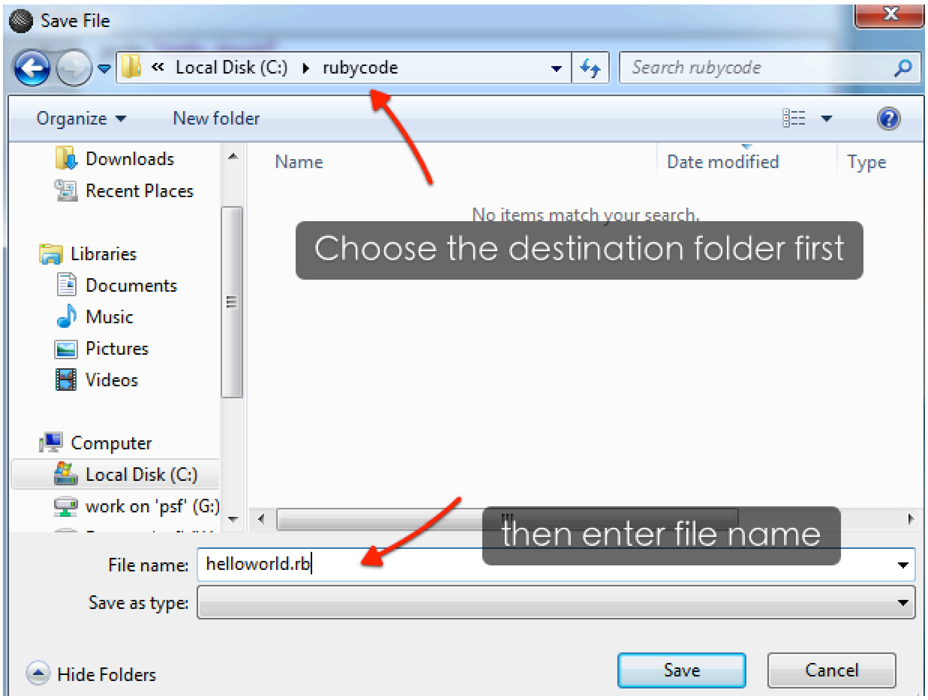
Open your editor (I use the free SciTE for illustration), and type in `puts "Hello World!"`. `puts` writes the followed text to the screen.



```
puts "Hello World"
```

Save the file to `C:\Users\you\rubycode\helloworld.rb`

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

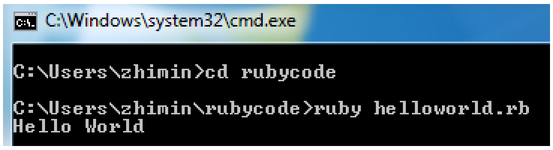


File name: helloworld.rb

Execution

To run our program, open a command prompt and change the directory to the `rubycode` folder. Type in the command `ruby helloworld.rb`.

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



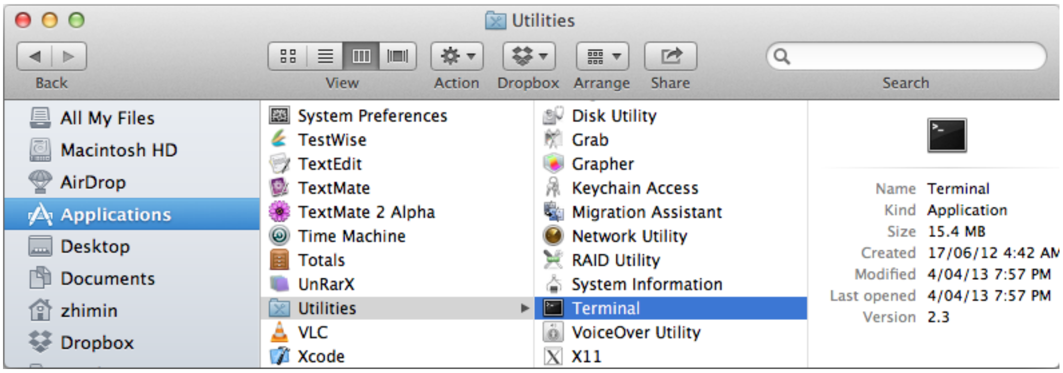
```
C:\Users\zhimin>cd rubycode

C:\Users\zhimin\rubycode>ruby helloworld.rb
Hello World
```

1.2 Ruby on Mac OS X

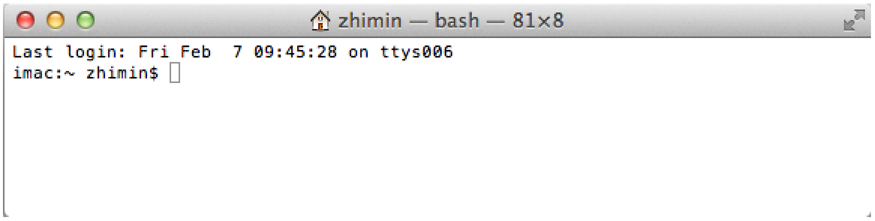
No installation is required as Ruby comes with Mac OS X.

Open command line

The application to access the command line in Mac OS X is called 'Terminal'. Open in Finder: 'Applications' → 'Utilities' → 'Terminal'.



It looks like this:



```
Last login: Fri Feb  7 09:45:28 on ttys006
imac:~ zhimin$ 
```

Type 'ruby -v', then press `Enter` key

```
imac:~ zhimin$ ruby -v
ruby 1.8.7 (2012-02-08 patchlevel 358) [universal-darwin12.0]
imac:~ zhimin$ 
```

The ruby version number might be different on your machine, this won't matter.

Choose a Ruby Editor

Commercial

- TextMate[4]. It's was called 'the editor of Mac' and won the Apple Design Award for best developer tool in 2006. It is very popular among Ruby programmers. Cost: €39.

Free

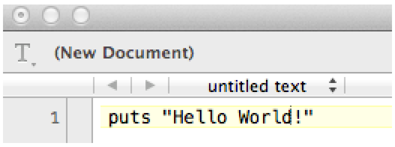
[4]http://macromates.com/

- Visual Studio Code[5].
- TextWrangler[6].

Write your first Ruby program

I suggest creating a dedicated folder to put all your code in, for example, `/Users/YOURUSERNAME/rubycode`.

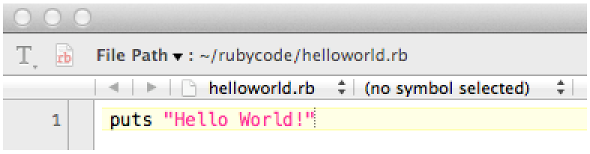
Start your editor, I would recommend TextMate, but for now (before you decide to purchase TextMate) I would use the free alternative TextWrangler. Open TextWrangler and type in `puts "Hello World!"`. `puts` writes the followed text to the screen.



```
puts "Hello World!"
```

Save to the folder `rubycode` with the name 'helloworld.rb'

After saving, you will notice that text color changed. This is called Syntax Highlighting. The editor now knows that it is a Ruby program (by the extension *.rb*) and highlights the code accordingly. This will make code much easier to read and identify problems.



```
puts "Hello World!"
```

Execution

To run our program. Open a Terminal and change the directory to the `rubycode` folder. Type in the command `ruby helloworld.rb`.

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

(`cd` means 'change directory'; `ruby filename` means running this ruby file.)

You will see the output:

[5]https://code.visualstudio.com/
[6]http://www.barebones.com/products/textwrangler/

```
Hello World!
```

1.3 Online Ruby Tutorials

While I believe you can learn basic Ruby programming with this book, there are online tutorials that you may use as supplements. For example, read them on your iPad while waiting at bus stops. Here are two good (and free) ones.

Ruby in Twenty Minutes

Ruby in Twenty Minutes[7] is the official Ruby tutorial. As its name suggests, it takes only about 20 minutes to go through.

Codecademy's "Introduction to Ruby" course

Codecademy[8] is a website offers free interactive coding courses. One of them is "Introduction to Ruby". Besides explaining concepts, the course also has simple exercises that you can edit and submit code.

Why bother this book if I can learn from online Ruby tutorials?

Online tutorials teach you the basic Ruby syntax and some programming concepts. While they are important, these knowledge is only useful if put into practice. For example, to be a good chess player, knowing chess rules is not enough. Though Courtney completed the Codecademy's ruby course first, she has experienced difficulties in doing even the basic programming exercises.

Programming, in my opinion, is a problem solving skill to solve problems in a computer friendly way. This knowledge can only gained by practically coding, which is what this book for. Online tutorials, especially video tutorials, put learners in a passive mode. You need a book such as this one to turn passive knowledge to your own.

[7]https://www.ruby-lang.org/en/documentation/quickstart
[8]http://www.codecademy.com

1.4 Rhythm for Working on the exercises

Every exercise has 5 sections:

- **The problem to solve**. It usually comes with sample output for easier understanding. Make sure you understand it well.
- **Purpose**. What you can learn from this exercise.
- **Analyse**. Analyze a problem like a programmer. This is an important step. Quite often we know how to do it but cannot describe it well. Take number sorting as an example; you can sort 5 numbers instantly on top of your head. But how about 100 numbers? This requires you to translate your understanding of sorting step by step into procedures that a computer can execute.

Now **write the code for the exercise**. No one can learn programming by reading, you have to actually do it. You may refer to the hints section to to help you get through.

- **Hints**. I list the hints (in Ruby code format) that may help you to solve the problem when you get stuck.

If you are struggling to solve an exercise, feel free to check out our solutions (at Appendix II). The exercises are selected to introduce new programming skills/practices as well as previous knowledge. So don't worry if you cannot get it right the first time, you will have chances to apply in later exercises. As long as you are trying, you are learning.

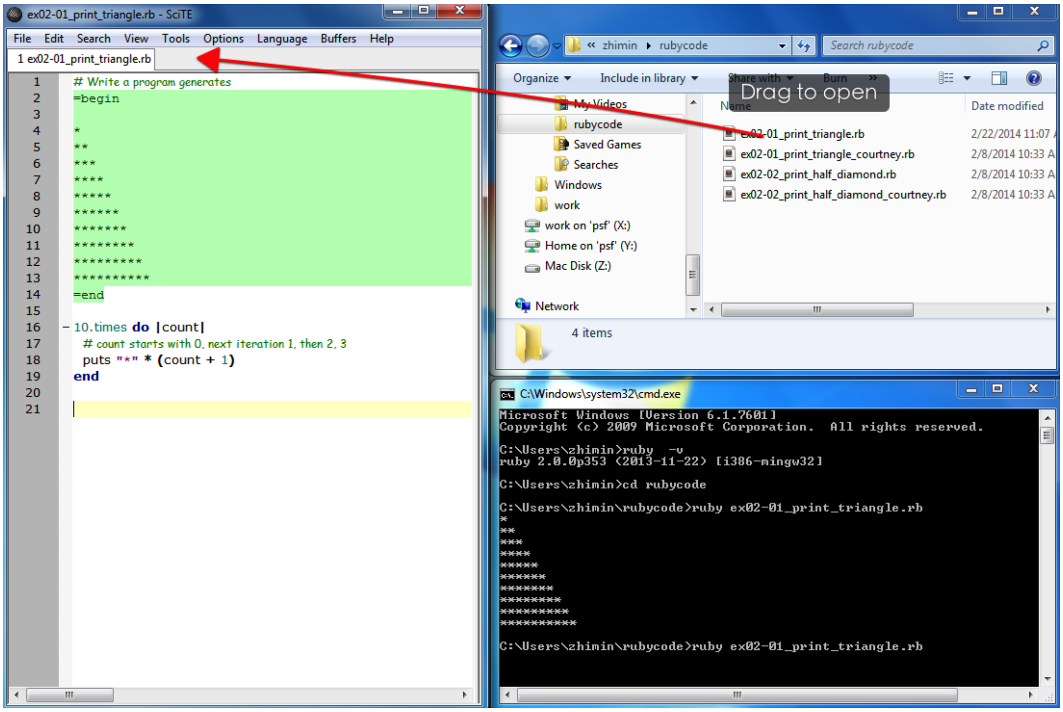
- **Solution(s)**. Solutions (can be found at Appendix II) to the most of exercises are between 10 to 20 lines of code. I may show Courtney's solution first with her comments. The runnable solution scripts can be downloaded at the book site.
- **Review**. Think about what you have learnt.

1.5 Suggestions on Windows Layouts

To make it easier for you to write and run your code, I suggest you opening 3 windows as below:

- Code Editor on the left, where you edit the code. (I used free SciTE in the screenshot below).

- A Window explorer with the `rubycode` folder opened.
- A Command Prompt (or Terminal on Mac or Linux) window with current directory is set to `rubycode` folder (execute the command `cd c:\Users\you\rubycode`)

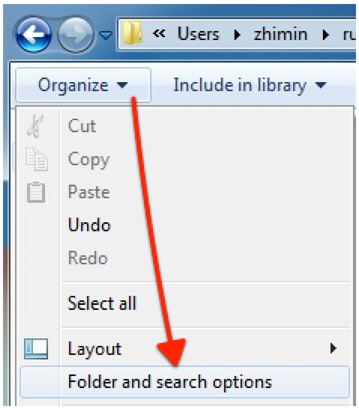

ex02-01_print_triangle.rb - SciTE
File Edit Search View Tools Options Language Buffers Help
1 ex02-01_print_triangle.rb
zhimin ▸ rubycode
Search rubycode
Organize ▾ Include in library ▾ Share with ▾ Burn
Drag to open
Name
Date modified
My Videos
rubycode
Saved Games
Searches
Windows
work
work on 'psf' (X:)
Home on 'psf' (Y:)
Mac Disk (Z:)
Network
ex02-01_print_triangle.rb 2/22/2014 11:07
ex02-01_print_triangle_courtney.rb 2/8/2014 10:33
ex02-02_print_half_diamond.rb 2/8/2014 10:33
ex02-02_print_half_diamond_courtney.rb 2/8/2014 10:33
4 items
C:\Windows\system32\cmd.exe

```ruby
# Write a program generates
=begin

*
**
***
****
*****
******
*******
********
*********
**********
=end

10.times do |count|
  # count starts with 0, next iteration 1, then 2, 3
  puts "*" * (count + 1)
end

```

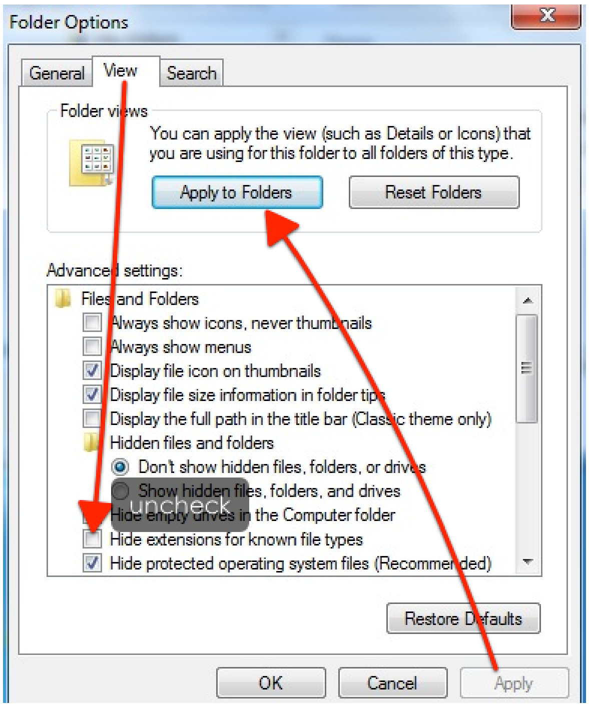
```
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation.  All rights reserved.

C:\Users\zhimin>ruby -v
ruby 2.0.0p353 (2013-11-22) [i386-mingw32]

C:\Users\zhimin>cd rubycode

C:\Users\zhimin\rubycode>ruby ex02-01_print_triangle.rb
*
**
***
****
*****
******
*******
********
*********
**********

C:\Users\zhimin\rubycode>ruby ex02-01_print_triangle.rb
```

Here are the steps to write and run a new program (*new_code.rb*).

1. In Window Explorer window, right click and create a new text file and rename it to `new_code.rb`. It is important to change the file extension to '.rb'. On Windows, the file extension is hidden in Windows Explorer by default. To change a file extension in Window Explorer, we need to change this setting (to show file extension). Here are the steps for Windows 7.
    - In a Window Explorer window, select 'Organize' → 'Folder and search options'



- Under 'View' tab, uncheck the "Hide extensions for known file types" checkbox



You can apply the view (such as Details or Icons) that you are using for this folder to all folders of this type.

Files and Folders
- [ ] Always show icons, never thumbnails
- [ ] Always show menus
- [x] Display file icon on thumbnails
- [x] Display file size information in folder tips
- [ ] Display the full path in the title bar (Classic theme only)

Hidden files and folders
- ● Don't show hidden files, folders, or drives
- ○ Show hidden files, folders, and drives
- Hide empty drives in the Computer folder
- [ ] Hide extensions for known file types
- [x] Hide protected operating system files (Recommended)

- Click 'Apply' button
- To make it as the default settings (recommended), click 'Apply to Folders' button.

2. Drag the *new_code.rb* file to the editor.
3. Type and edit the code in the editor. Save when it is ready to run.
4. In Command Prompt window, type

```
ruby new_code.rb
```

then press `Enter` key to run the program.

A quicker way to rerun the program is pressing the 'Up Arrow' key to get the last command.

5. If necessary, repeat Step 3 and 4 until you are satisfied with the result.

1.6 Type of errors

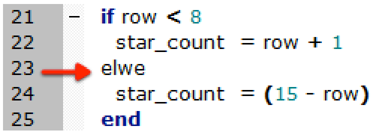
Programmers (new or experienced) encounter code errors every day. I don't expect you to get the exercises right on the first go. We learn from mistakes.

Syntax Error

Ruby checks the syntax of code before running it. If there are syntax errors in code, the error messages are usually quite helpful for identifying the error.

Typo

It is normal that we make typing mistakes when writing code. For example, in the code below, instead 'else' at line 23, I typed `elwe`.



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

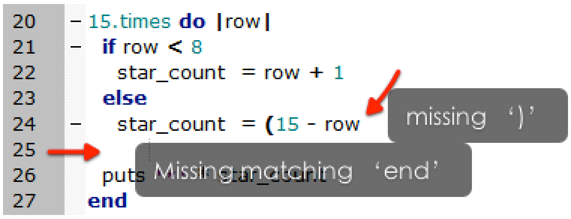
When I ran the program, I got the error message:

```
ex02-02_print_half_diamond.rb:23:in `block in <main>': undefined local variable 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>'
```

The error message means `elwe` is undefined (don't worry if this does not make sense, you will soon understand). The more helpful part in the error trace is the line number `23` (the first line number next to your code file) . It helps identify where the error is.

No matching parenthesis or brackets

Just like Math, if there is a left bracket “(“ in code, there shall be a matching right bracket “)”. There are also matching keywords for certain code structures, such as `if` … `end`. For example, there are two errors in the code below.



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

1. at line 24: missing ‘)’, shall be `(15 - row)`.
2. at line 25: missing `end` for `if` at line 21.

When you run the program, the error message *“expecting ‘)’”* is correct.

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

However, the line number 27 is not where the actual error is. That is because it is not possible for Ruby to detect all error scenarios. If the right bracket is on the next line, the program is valid. Only after line 27, Ruby detects the right bracket is actually missing.

After fixing the first error and rerunning the program, a second error occurs. Again, the error line number is not exactly where the real error is.

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

Adding `end` to line 25 will fix the code.

Runtime error

A runtime error is a software problem that prevents a program from working correctly. When a runtime error occurs, the execution of the program terminates with error messages displayed.

```
a = 5
puts("OK so far")
b = 200 / (a  - 5)
```

The above code runs with an error thrown at line 3, here is the output.

```
OK so far
runtime_error.rb:3:in `/': divided by 0 (ZeroDivisionError)
divided by 0 (ZeroDivisionError)
```

Code logic error

The above two kinds of errors are relatively easy to spot. The difficult errors for programmers are code logic errors. That is, the code can run without syntax errors, but does not behave as it is supposed to. Here is an example.

```
# score below 60 fails the subject
score = read_user_score()
if score > 60
  puts "Pass"
else
  puts "Fail"
end
```

The above code reads a user's exam score and gives the grade: "Pass" or "Fail". It runs fine, most of time, except when the score is 60. There is a code logic error on line 3, it shall be `if score >= 60`.

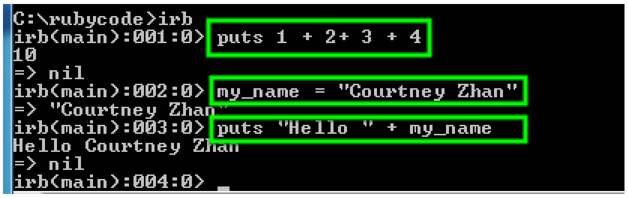
The ability to debug code errors (find and fix them) separates good programmers from the average. With practice, you will get better on this.

For beginners, I have two practical tips.

1. **One step at a time**. Write one line of code, run the code immediately. This may sound uninteresting, but in practice, many find it is the most useful tip to learn programming. If newly added or changed code fragment caused the error, a click of 'Undo' button (in your editor) will get back your code to previous working state.
2. **If feeling confused**, **restart**. If you stuck with existing code, chances are the complexity of the code is beyond your control. Try guessing around to get computers to work as instructed (by your code) is highly unlikely. In this case, it is better to restart from scratch. For most of exercises in this book, solutions are less than 20 lines of code.

1.7 Interactive Ruby Shell (IRB)

IRB is a tool that allows the execution of Ruby commands with immediate response, which can be very helpful for learning Ruby syntaxes and debugging code errors. IRB comes with Ruby and it is launched from a command line. To run IRB, just run `irb` from a command line window then try ruby code there.



```
C:\rubycode>irb
irb(main):001:0> puts 1 + 2+ 3 + 4
10
=> nil
irb(main):002:0> my_name = "Courtney Zhan"
=> "Courtney Zhan"
irb(main):003:0> puts "Hello " + my_name
Hello Courtney Zhan
=> nil
irb(main):004:0> _
```

In the screenshot above, the commands in the green boxes are what I entered. The rest were the responses returned from the commands.

In Appendix 1 (‘Ruby in Nutshell’) I summarized the core Ruby syntax and usage in examples which you can conveniently run in IRB.

2. Printing Shapes

Printing out asterisk characters (*) in shapes is often used as beginner's programming exercises, as they are simple, easy to understand and visually interesting.

2.1 Print out Triangle

Write a program to print out asterisks like the triangle below:

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

Purpose

- Develop ability to analyse patterns
- Variables
- Use of looping

Analyse

Row	The number of stars
1	1
2	2
3	3
...	...
n	n

Hints

Print out text.

```
puts '*'
puts "**"
```

For small ruby code fragments, you can try them out quickly in IRB.

Generate multiple occurrences of the same character. Please note that the star symbol and Math's multiply symbol are the same in programming languages. However, it is quite easy to distinguish: star symbol is quoted like this `"*"`.

```
'$' * 3  # => '$$$'
```

Because Math's times operator '×' is easy to get confused with the letter 'x' in code , `*` is commonly used as the multiplication operator in most programming languages.

Code Comment

Comments are annotations used in code for human to read, computers will ignore them. In Ruby, statements after `#` are comments.

```
# comment: the code below prints out 10 $ signs.
puts '$' * 10
```

Besides writing down your notes in code as comments, you may also comment out some code fragments when you are not sure to remove them yet.

Using a variable to store an integer.

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

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

Variables

You can think of a variable is a 'labeled box' in computers to store data, its data can be changed. The naming convention for Ruby variables is in lower case separated b underscores, for example, `my_birth_date`.

Print out the same text multiple times in a loop (fixed number of times)

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

The `do` … `end` mark the beginning and end of loop respectively,

Working out the solution on your computer

Make sure you understand the *Analyse* and *Hints* parts before you start.

Courtney's version

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

2.2 Print out a half diamond

Write a program that prints out half of the diamond shape using asterisks.

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

Purpose

- Decrement count in loops

Analyse

The key to this problem is to determine the number of stars for the corresponding rows.

```
row 1 to  8: the same as row number
row 9 to 15: 16 - row
```

Hints

Control flows using if … else

Code, in its simplest form, is executed from top to bottom. But if there are `if` conditions and loops (and later methods and classes), it will change the flow of execution. The conditional expressions (if-then-else statements) run different code statements depending on a boolean condition (`true` or `false`).

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

Output:

```
Pass!
```

If you change the `score = 59` and run again, you will get `Failed!`.

Boolean condition

The statement `score < 60` after `if` is called a boolean condition. Its value can only be either `true` or `false` (which are called boolean values).

Common comparison operators in Ruby

==	equal to
!=	not equal to
<	less than
<=	less than or equal to
>	greater than
>=	greater than or equal to

Examples:

```
2 > 1   # => true
2 == 1  # => false  (equal to)
2 != 1  # => true   (not equal to)
2 <= 2  # true
```

Equal sign = and Double equal sign ==

The equal sign (=) is the "assignment operator", it assigns a value to a variable on the left.

```
a = 1 + 2  # assign 3 to a
```

Please note the "assignment operator" is different from the "equality symbol" in Math. For example, the statement below increases the value of `a` by `1` (*assign a new value to* `a`) in programming code. The same equation in Math is invalid.

```
a = a + 1  # increment a by 1
```

The double equal signs (`==`) is a comparison operator, it compares two values for equality (returns `true` if `a` is equal to `b`, `false` otherwise).

```
if a == b
  puts "Found a match!"
end
```

Incorrect use of **=** for **==** is one of the most common mistakes in programming[1].

[1]http://www.cprogramming.com/tutorial/common.html

Courtney's version

```
count = 0
8.times do
  count += 1          # this is equivalent to count = count + 1
  stars = "*" * count
  puts stars
end
count = 10
8.times do
  count -= 1
  stars = "*" * count
  puts stars
end
```

Courtney's version loops 16 times (8 + 8), but prints out OK (15 lines). This is because when `count` is decrement to 0, an empty line is printed out instead.

Courtney uses two loops, which is fine and quite logical for beginners.

2.3 Print out diamond shape

Print 7 rows of '`*`' in a diamond shape as below:

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

Purpose

- Analyze more complex patterns

Analyse

Below are formulas to calculate the number of star; where `row_number` represents the row number and `total_rows` represents the total number of rows,

1. The number of stars for the rows before the middle one is `(row_number - 1) * 2 + 1`.
2. the number of stars for the rows after the middle one is `(total_rows - row_number) * 2 + 1`

Think about the spaces in front of each row, except for the 4th row (the longest middle one).

Hints

Write down the number of spaces and stars for each row.

```
row 1: print 3 spaces + 1 star
row 2: print 2 spaces + 3 stars
row 3: print 1 space  + 5 stars
row 4: print 0 space  + 7 stars
row 5: print 1 space  + 5 stars
row 6: print 2 spaces + 3 stars
row 7: print 3 spaces + 1 star
```

If you have difficulty, do it step by step. You may try to print out the top triangle first.

Courtney's version

```ruby
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
```

Courtney says:

I was stuck on the number of stars and number of spaces. I had to sit down with dad to work out the math formula. Also, the multiple variables makes it confusing. So remember to name variables properly and meaningfully. If you are stuck, you can print the variable you think may be the problem. This can help you understand what is going on and how to fix it.

Courtney uses variable `space` to represent a space string, which is a good practice.

2.4 Print big diamond, name your size

Ask the user for the size of diamond (based on the total number of rows) and then print out a diamond shape using asterisks '*'.

```
Enter the maximum number of rows (odd number): 9
            *
           ***
          *****
         *******
        *********
         *******
          *****
           ***
            *
```

Purpose

- Read user's input into a variable
- Convert string to integer
- Use variable use loop times

Analyse

The size of the diamond is not fixed, it depends on the number the user entered. The number the program asks the user to enter is the total number of rows, which can be stored in an variable.

If you divide the diamond into two parts (top and bottom), work out the number of rows for each part.

Hints

Read user's input

```
user_input = gets  # read user input to a string stored in user_input
```

The above `gets` accepts a line of text from keyboard and assigns the string typed and a “\n” character to variable `user_input`. “\n” is a new line character, which is added when the user press `Enter` key. Quite often, we want to remove this “\n” immediately. Here is code to do that:

```
user_input = gets.chomp
```

String and Fixnum

The `String` and `Fixnum` (number) are two most common data types in Ruby.

```
a = "12"
b = "3"
a + b       # => "123"

c = 12
d = 3
c + d       # => 15
```

Adding a `String` to an `Fixnum` will get an error.

```
a.class  # => String
c.class  # => Fixnum
a + c    # return errors TypeError: no implicit conversion of Fixnum into String
```

Convert a number string to integer

```
"6".to_i        # => to integer 6
"12".to_i  + 3  # => 15
```

Math divide operator in Ruby: “/”

```
 8 / 2                # => 4
 9 / 2                # => 4,  ignore the remainder
 (1 + 2) * 3 + 3 / 2  # => 10
```

Courtney's version

```
puts "Enter the maximum number of rows (odd number):"
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
```

Courtney says:

When I first tried replacing diamond size with user input variable, I forgot to change the variables and that made the outcome completely different. It is not hard to fix it though.

You must know it first, then instruct computers

Trying to enter a big number for the last program, say 99. It will print a big diamond, Wow!

This an important aspect of programming. Once you figure out the pattern and logic of a problem and translate it into computer understandable language (program), it can solve the similar problems at any scale. For example, the effort taken for computers to calculate `2 x 2` is not much different from `12343 x 35345`. In other words, we (human) must understand **how** to solve the problem first. Programming translates the how into instructions that computers can follow.

2.5 Exercises

Write code to print out the shapes below, the width of shape is changeable.

Rhombus

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

Hollow Square

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

Heart

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

Hints

The first three rows are static regardless of the size.

Resources

Solutions to exercises
http://zhimin.com/books/learn-ruby-programming-by-examples[2]

Username: `agileway`
Password: `SUPPORTWISE15`

Log in with the above, or scan QR Code to acess directly.

Code Editor

- **SciTE**[3], Windows and Linux Platform, Free.
- **TextWrangler**[4], Mac, Free.
- **TextMate**[5], Mac, 48.75 EUR.
- **Sublime Text**[6], Windows & Mac & Linux, $70.
- **Visual Studio Code**[7], Windows & Mac & Linux, Free.

Ruby Language

- **Ruby Installer for Windows**[8]

Ruby Tutorials

- **Ruby in Twenty Minutes**[9]

[2]http://zhimin.com/books/learn-ruby-programming-by-examples
[3]http://www.scintilla.org/SciTEDownload.html
[4]http://www.barebones.com/products/textwrangler
[5]http://http://macromates.com
[6]http://www.sublimetext.com/
[7]https://code.visualstudio.com/
[8]http://rubyinstaller.org/downloads
[9]https://www.ruby-lang.org/en/documentation/quickstart

Official Ruby tutorial, as its name suggests, it takes only about 20 minutes to go through.

- **Codecademy's "Introduction to Ruby" course** http://www.codecademy.com[10]

  Codecademy offers free interactive coding courses. One of them is "Introduction to Ruby". Beside explaining concepts, the course also have simple exercises that you can edit and submit code.

More Exercises

- **Ruby Quiz**[11]

  A programming challenge site for Ruby programmers, contains over 150+ quizzes. The quizzes are generally quite hard.

Test Automation

- **Practical Web Test Automation**[12] by Zhimin Zhan

  A practical guide to learn and master automated testing for web applications.

- **Selenium WebDriver Recipes in Ruby**[13] by Zhimin Zhan

  Over 200 automated test solutions in Selenium WebDriver, a popular automated test framework for web applications (used in FaceBook and Google).

Others

- **Learn Swift Programming by Examples**[14] by Zhimin Zhan

  Leverage similar exercises in this book to learn a hot new programming language Swift to develop your own iOS apps.

[10]http://www.codecademy.com
[11]http://rubyquiz.com/
[12]https://leanpub.com/practical-web-test-automation
[13]https://leanpub.com/selenium-recipes-in-ruby
[14]https://leanpub.com/learn-swift-programming-by-examples