





























































































































































































































































































































































































































































































































































































































































































## Everyday Scripting with Ruby

### [Pragmatic Home](#)

#### NEW BOOKS NOTICES

[more info](#)

#### Titles

[Pragmatic Fridays Series](#)

[The Pragmatic Programmer](#)

[Pragmatic Starter Kit Series](#)

[Programming Ruby](#)

[All Books](#)

#### Sections

[Books](#)

[Articles](#)

[Blogs & Mailing Lists](#)

[FAQ and Answers](#)

[News and Reviews](#)

[About Us](#)

#### Working with us

[For Authors](#)

[For Reviewers](#)

[For Booksellers](#)

[Press Releases](#)

#### Shopping

[Catalog](#)

[View Cart](#)



Sign up for [Rails Training](#)

#### Search

Search pragmaticprogrammer.  
com      Search WWW

Visit our bookstore:



*In stock and shipping*

### Everyday Scripting with Ruby

*For Teams, Testers, and You*

Brian Marick

Pages: 320

ISBN: 0-9776166-1-4

Date: January 2007

Resources: [Code](#)

[Errata and Feedback](#)

Post link to: [del.icio.us](#)

#### Purchase

 [Buy paper book](#)

[Buy PDF now](#) ([PDF FAQ](#))

 + [Buy book + PDF combo now](#)

([FAQ](#))

#### Excerpts

- [Table of Contents](#)
- [Introduction](#)
- [Chapter 7 \(excerpt\)](#)

### Everyday Scripting with Ruby

Are you a **tester** who spends more time manually creating complex test data than using it? A **business analyst** who seemingly went to college all those years so you can spend your days copying data from reports into spreadsheets? A **programmer** who can't finish each day's task without having to scan through version control system output, looking for the file you want?

If so, you're wasting that computer on your desk. **Offload the drudgery** to where it belongs, and free yourself to do what you should be doing: thinking. All you need is a scripting language (*free!*), this book (*cheap!*), and the dedication to work through the examples and exercises.

#### About This Book

*Everyday Scripting with Ruby* is divided into four parts. In the first, you'll learn the basics of the Ruby scripting language. In the second, you'll see how to create scripts in a steady, controlled way using test-driven design. The third part is about finding, understanding, and using the work of others---and about preparing your scripts for others to use. The fourth part, more advanced, is about saving even more time by using application

frameworks.

Throughout, you'll also see how to cope with common mistakes. You'll learn how to recognize that you're in a blind alley and recover from it. You'll even see examples of the most common typos, so that you'll recognize the symptoms when you see them.

- Learn to **automate** rote tasks
- Gain a detailed understanding of **working, finished scripts** that you can apply directly to your job
- Understand programming **terminology and concepts**
- **Exploit** the unpaid labor of others
- **Communicate** more efficiently and effectively with teammates

Read the [online reviews](#) around the web for our books.

## ***Reader Comments***

Here's what other readers have said about *Everyday Scripting with Ruby*:

*A fantastic type-along-with-me introduction to a powerful scripting language that starts in the shallows and then moves into the depths turning the reader into an accomplished Ruby scripter, almost without them noticing it!*

**Erik Petersen**  
**Emprove**

*Finally a hands-on book that is filled with gems of wisdom for the testing community.*

**Gunjan Doshi**  
**VP of Product Development and Process Excellence**  
**Community Connect, Inc**

*What a wondrous collection of recipes, guidelines, warnings, comprehensive examples, metaphors, exercises, and questions! It's a terrific value to software testing practitioners who want to get the most from their test automation effort.*

**Grigori Melnik**  
**Lecturer**  
**University of Calgary**

*Marick explains the Ruby language using a series of short, practical examples. Watir users and other testers who want to learn Ruby will find it very accessible.*

**Bret Pettichord**  
**Lead Developer**  
**Watir**

*The book is an excellent read, is very informative, and covers a lot of ground in a relatively slim book. I think this is always a good idea. I have a lot of 800+ page tech books that I've read about the first half or two thirds of, because they are padded toward the end with very esoteric information. This book held my interest throughout.*

**Paddy Healey**  
**Enterprise Systems Engineer**  
**Aventail Corporation**

## ***About The Author***

Brian Marick graduated in 1981 with one degree in Mathematics and Computer Science

and another one in English Literature. In his early career, he continued to be confused about his identity. Sometimes he was a programmer. Sometimes he was a tester. He became better at each because he understood the other.

Brian is the author of *The Craft of Software Testing*, a book about test design techniques for "programming-in-the-medium". He was an author of the Manifesto for Agile Software Development and was the second chair of the Agile Alliance board of directors. Because the Agile methods encourage cross-disciplinary work, he's once again without a fixed identity. When consulting, he sticks his nose into everything.

[Home](#)[All Books](#)[Privacy](#)[Contact us](#)[Phone Orders: 800-699-7764](#)

Copyright © 1999-2007

The Pragmatic Programmers, LLC

All Rights Reserved

```
#--- # Excerpted from "Everyday Scripting in Ruby" # We make no guarantees that this code is fit for
any purpose. # Visit http://www.pragmaticprogrammer.com/titles/bmsft for more book information. #---
if $0 == __FILE__ #(1) subsystem_names = ['audit', 'fulfillment', 'persistence', #(2) 'ui', 'util', 'inventory']
start_date = month_before(Time.now) #(3) puts header(start_date) #(4) subsystem_names.each do |
name | puts subsystem_line(name, change_count_for(name)) #(5) end end
```

## Watir

Web App Testing in Ruby

- . Overview
- . Examples
  - . Install
- . Documentation
  - . Platforms
- . Community
  - . Source

## Automated testing that doesn't hurt

Watir is a simple open-source library for automating web browsers. It allows you to write tests that are easy to read and easy to maintain. It is optimized for simplicity and flexibility.

Watir drives browsers the same way people do. It clicks links, fills in forms, presses buttons. Watir also checks results, such as whether expected text appears on the page.

Watir is a Ruby library that works with Internet Explorer on Windows. Watir is currently being ported to support **Firefox and Safari**.

Like other programming languages, Ruby gives you the power to connect to databases, read data files, export XML and structure your code into reusable libraries. Unlike other programming languages, Ruby is concise and often a joy to read.

Watir stands for "Web Application Testing in Ruby". It is pronounced *water*.

## Why Watir?

- It's a free **Open Source** tool. There are no costs to use the tool.
- There's a very active and growing **community** behind it.
- It uses **Ruby**, a full-featured modern scripting language, rather than a **proprietary vendorscript**.
- It is powerful and easy to use.
- Don't just take our word for it, **read what our users are saying**.

## Quotes

"Watir Rocks! Truly awesome!!"—Shashank Date

"Watir is the most compelling alternative [to Fit] for filling the automated acceptance testing need."—Ward Cunningham

"I have limited scripting experience and none with Ruby, but I was able to follow your examples and be productive in short order."—Kevin Sheehy

"I've been trying to find the Holy Grail of Automated Web UI Testing... And the one I'm currently enamored with is Watir."—Scott Hanselman

"I wanted to run around my office dancing and celebrating."—Beth Ferguson

## Install Watir Now

Watir is released under a **BSD license** | Hosted by **RubyForge** and **OpenQA**

Logo by Jacinda Scott | Web design by **webgen** and **Andreas Viklund**

## Watir

Web App Testing in Ruby

- . Overview
- . Examples
- . Install
- . Documentation
- . Platforms
- . Community
- . Source

## Watir 1.5

The [new users guide](#) provides a quick introduction to using Watir. The [cheat sheet](#) provides a handy reference.

View this [succinct summary of the Watir API](#) and the provided methods and attributes. For more detailed API documentation, refer to Watir's *rdoc* —the standard format for documenting Ruby gems. To view the **Watir rdoc**:

1. Go to the Start menu, and select Ruby → RubyGems → Start RubyGems Rdoc Server.
2. Select Ruby → RubyGems → View Installed Gems Rdoc.
3. Select the Watir rdoc in the list that appears.

The [FAQ](#) provides answers to frequently asked questions.

There are also several [articles](#) about Watir.

## Watir 1.4

The [old users guide](#) provides a broad, comprehensive introduction, but is out of date in places.



The **old rdoc** is online and easy to browse.

Watir is released under a [BSD license](#) | Hosted by [RubyForge](#) and [OpenQA](#)

Logo by Jacinda Scott | Web design by [webgen](#) and [Andreas Viklund](#)

## Watir

Web App Testing in Ruby

. Overview

. Examples

. Install

. Documentation

. Platforms

. Community

. Source

## Examples

View this detailed example of how to automate a [search on Google](#).

Here are several simple [examples](#). Many people also find our [unit tests](#) to be useful examples of the different Watir commands. These files are also conveniently packaged in a [bonus-zip](#) that you can download.

For more extensive examples, see these [test suites for open source projects](#).

Watir is released under a [BSD license](#) | Hosted by [RubyForge](#) and [OpenQA](#)

Logo by Jacinda Scott | Web design by [webgen](#) and [Andreas Viklund](#)

# WATIR Example Test Case

## Google Test Search

This document walks through a very simple test case: **googleSearch.rb**. To begin, from the **examples** folder in your Watir installation directory, open the googleSearch.rb file in a text editor. To open it with SciTE, simply right-click the file in Windows Explorer, and select **Edit** from the popup menu.

The format of this example will be to show the Ruby test case scripting code in a box as you would see it in a text editor, with an explanation after it.

## Getting Started

Open Internet Explorer, and try out the test case manually on your own:

### Steps:

1. go to the Google home page - <http://www.google.com>
2. enter "pickaxe" in the search text field
3. click the "Google Search" button

### Expected Result:

- a Google page with results should be shown. "Pragmatic Programmers LLC" should be high on the list.

Once you have tried the test case out manually, it's time to automate the test case using Watir. Return to the Google home page <http://www.google.com> and view the page source: right mouse click > View Source. Now you can follow along and see how to automate this test with Watir based on the HTML tags in the Google search web application.

## Section 1: Comments

Test cases should be commented, just as program code should be commented. In Ruby, any text on a single line that follows a **#** is a comment, and is ignored by the Ruby interpreter at run time.

What you see in the text editor:

```
#-----#
# demo test for the WATIR controller
#
# Simple Google test written by Jonathan Kohl 10/10/04
# Purpose: to demonstrate the following WATIR functionality:
# * entering text into a text field
# * clicking a button
# * checking to see if a page contains text.
# Test will search Google for the "pickaxe" Ruby book
#
#-----#
```

Styles differ with comments, but they can really help with test case maintenance. In this case, the author has provided their name, a date and a purpose of the test case. This is here to improve readability, and to help others who may be using the test understand what it does.

## Section 2: Includes

To use Watir, or any other library in our test case requires us to tell the program where to find the library.

What you see in the text editor:

```
#includes:
require 'watir' # the watir controller
include Watir
```

When we run our test script, the Watir library is loaded so that our test cases can use it.

## Section 3: Declare Variables

If we are going to use something in our script more than once, or something that could change, we can declare it as a variable and reuse it throughout the script. Some objects we can use for testing tend to change, such as URLs for applications we are testing. In this script, we assign the test URL as a variable. If it changes, we only have to change it in one place.

What you see in the text editor:

```
#variables:
test_site = 'http://www.google.com'
```

The test case author has chosen to assign the URL to a variable called **test\_site**. It may not be much of an issue in this test case, but using variables for test URLs is often a good practice.

## Section 4: Open an Internet Explorer Browser

To begin driving Internet Explorer, we need to tell Watir to open an instance for testing.

What you see in the text editor:

```
#open the IE browser
ie = IE.new
```

To explain what this is doing, we can start from the right of the "=" operator. We send a message "new" to the "IE" (Internet Explorer) class, and assign it to a variable called "ie". The "ie" variable is a local variable (like "test\_site"). This means it can be accessed from our script, but not from other Ruby functions or methods.

## Section 5: Interacting With Google

Now we are ready to start automating the steps we ran manually using Watir.

### Beginning the test case

What you see in the text editor:

```
puts "## Beginning of test: google search"
puts " "
```

*Puts* is a reserved word in the Ruby language that tells the Ruby Interpreter to print whatever comes after it contained in quotes to the screen. We could just as easily write this information to a file. These *puts* statements are in this test case to make it more self-explanatory. Not only to they tell the user what is happening in the test

case, they double as comments. Printing what we are doing when we automate the test case is useful for debugging when developing test cases, and for quickly repeating failures for bug reports when the test case doesn't pass.

### Step 1: Go to the Google site

This test case follows a pattern of printing out what we intend Watir to do on a web application, followed by the Watir scripting code to carry out that action, and finally printing out the action that Watir just executed. This is a style of test case development that is useful for tracking down test case failures quickly.

What you see in the text editor:

```
puts "Step 1: go to the test site: " + test_site
ie.goto(test_site)
puts "Action: entered " + test_site + "in the address bar."
```

The first line uses a *puts* statement to print out the test case step we are attempting to the screen. The second line uses the Watir method "goto" to direct the test case to the test site: <http://www.google.com> (stored in the variable **test\_site**). The third line prints the action that Watir just executed to the screen. When we print out the variable, we concatenate it to the string (the part in quotes) by using the + sign.

### Step 2: Enter Search Term "pickaxe" in the search field

We need to enter the term to search in the text field on the Google home page.

What you see in the web browser:

What you see in the text editor:

```
puts "Step 2: enter 'pickaxe' in the search text field"
ie.text_field(:name, "q").set("pickaxe") # q is the name of the
search field
puts " Action: entered pickaxe in the search field"
```

The first line prints the step we are on to the screen. The second line enters the text "pickaxe" in the text field named **q**. The third line prints what Watir just executed to the screen. The comment telling the user that "q" is the name of the text field is optional.

This is the tag in the HTML source with the *name* attribute we used highlighted:

```
<input maxLength=256 size=55 name=q value="">
```

The text field has a *name* attribute **q**, so we use that to tell Watir what object to interact with.

### Step 3: Click the "Google Search" button

We need to click the search button to activate the Google Search functionality.

What you see in the web browser:

What you see in the text editor:

```
puts "Step 3: click the 'Google Search' button"
ie.button(:name, "btnG").click # "btnG" is the name of the button
puts " Action: clicked the Google Search button."
```

The first line prints the step we are on to the screen. The second line clicks the Google Search button. The third line prints what Watir just executed to the screen.

This is the tag in the HTML source with the *name* attribute we used highlighted:

```
<input type=submit value="Google Search" name=btnG>
```

## Section 6: Evaluating the Results

### The Expected Result

This test case prints out what the Expected Result should be prior to the test case using Watir to evaluate the results.

What you see in the text editor:

```
puts "Expected Result: "
puts " - a Google page with results should be shown. 'Programming Ruby' should be high on the list."
```

These two statements simply print the Expected Result of the test to the screen.

### Verify Results

Using Watir and a little Ruby, we can evaluate the results to verify whether the test case passed or failed.

```
puts "Actual Result: Check that the 'Programming Ruby' link appears on the search results page"
if ie.contains_text("Programming Ruby")
  puts "Test Passed. Found the test string: Programming Ruby. Actual Results match Expected Results."
else
  puts "Test Failed! Could not find: Programming Ruby"
end
```

There is a lot more in this section, but we can break it down.

- The first line prints out the "Actual Result" heading to the screen.
- The second line calls the "contains\_text" method, and asks it to verify that *Programming Ruby* appears on the first page.
- Using an *if* statement, we evaluate whether the "contains\_text" method was True or False.

- If `pageContainsText` returns false, the text *Programming Ruby* does not appear. The test case fails and we print the "Test Failed!" message.
- Else, if `pageContainsText` returns true (or actually anything but false), the text *Programming Ruby* appears. The test case passes and we print the "Test Passed." message.
- We close the conditional *if* section with an *end* statement.

## End of Test Case

What you see in the text editor:

```
puts " "  
puts "##End of test: Google search"  
puts # -end of simple Google search test
```

The first line prints a blank line for formatting, and a the second line prints message to the user that the test case has ended. The final line in green is a comment that shows the end of the test case.

## Watir

Web App Testing in Ruby

- Overview
- Examples
- Install
- Documentation
- Platforms
- Community
- Source

## Internet Explorer and Windows

This version of Watir works with Internet Explorer and Windows. It works with IE 5.5, 6 and 7 and Windows 2000, XP, 2003 Server and Vista.

Support for other platforms is currently provided by separate ports. We have plans to eventually roll together the following Watir ports into a single package.

## Firefox

**FireWatir** is the Watir port for Firefox. It works on Windows, Linux and Mac.

## Safari and OSX

**SafariWatir** is a Watir port for Safari on the Mac.

## Other Browsers



Selenium is a popular open-source testing tool that supports most browsers, including Opera, Camino and Konqueror. The Selenium-RC library includes Ruby support. We have demonstrated **a Watir interface that uses Selenium** to drive a browser, and expect to eventually grow this into a full-blown port.

Watir is released under a **BSD license** | Hosted by **RubyForge** and **OpenQA**

Logo by Jacinda Scott | Web design by **webgen** and **Andreas Viklund**

## Watir

Web App Testing in Ruby

- . Overview
- . Examples
- . Install
- . Documentation
- . Platforms
- . Community
- . Source

## Mailing Lists

**Watir general** is a Google group for anyone using Watir for testing. You can subscribe right here.

Email:

You also may want to search the **archives** of the list when it was hosted at Rubyforge.

## Wiki

The **Watir Wiki** hosts a growing collection of documentation and **contributions**.

## Users

People **all over the world** are using Watir and **loving it**.

## Core team

Watir is the creation of people committed to making testing easier.

- [Bret Pettichord](#)
- [Paul Rogers](#)
- [Jonathan Kohl](#)
- [Angrez Singh](#)
- [Charley Baker](#)
- [Zeljko Filipin](#)

Watir is released under a [BSD license](#) | Hosted by [RubyForge](#) and [OpenQA](#)

Logo by Jacinda Scott | Web design by [webgen](#) and [Andreas Viklund](#)

## Watir

Web App Testing in Ruby

- . Overview
- . Examples
- . Install
- . Documentation
- . Platforms
- . Community
- . Source

## Source

The Watir source code is hosted in an Subversion repository on OpenQA. You can [browse the code](#), or you can check out the source from

<https://svn.openqa.org/svn/watir/trunk/watir>

You can check out this code anonymously, or using your OpenQA username and password if you are a committer.

Subscribe to this [RSS feed](#) to receive notification of code changes.

Watir is released under a [BSD license](#) | Hosted by [RubyForge](#) and [OpenQA](#)

Logo by Jacinda Scott | Web design by [webgen](#) and [Andreas Viklund](#)

```
#--- # Excerpted from "Everyday Scripting in Ruby" # We make no guarantees that this code is fit for
any purpose. # Visit http://www.pragmaticprogrammer.com/titles/bmsft for more book information. #---
old_inventory = File.open('old-inventory.txt').readlines new_inventory = File.open('new-inventory.txt').
readlines puts "The following files have been added:" puts new_inventory - old_inventory puts "" puts
"The following files have been deleted:" puts old_inventory - new_inventory
```

```
#--- # Excerpted from "Everyday Scripting in Ruby" # We make no guarantees that this code is fit for
any purpose. # Visit http://www.pragmaticprogrammer.com/titles/bmsft for more book information. #---
unless ARGV.length == 2 puts "Usage: differences.rb old-inventory new-inventory" exit end def
inventory_from(filename) #(1) File.open(filename).collect do | line | line.downcase end end
old_inventory = inventory_from(ARGV[0]) #(2) new_inventory = inventory_from(ARGV[1]) #(3) puts
"The following files have been added:" puts new_inventory - old_inventory puts "" puts "The following
files have been deleted:" puts old_inventory - new_inventory
```

```
#--- # Excerpted from "Everyday Scripting in Ruby" # We make no guarantees that this code is fit for
any purpose. # Visit http://www.pragmaticprogrammer.com/titles/bmsft for more book information. #---
require 'test/unit' # (1) require 'X' # (2) class X < Test::Unit::TestCase # (3) def test_X # (4) assert_equal
('expected', 'actual') # (5) end end
```

```
#--- # Excerpted from "Everyday Scripting in Ruby" # We make no guarantees that this code is fit for
any purpose. # Visit http://www.pragmaticprogrammer.com/titles/bmsft for more book information. #---
require 'test/unit' require 'churn' class ChurnTests < Test::Unit::TestCase def
test_month_before_is_28_days assert_equal(Time.local(2005, 1, 1), month_before(Time.local(2005, 1,
29))) end end
```



```
#--- # Excerpted from "Everyday Scripting in Ruby" # We make no guarantees that this code is fit for
any purpose. # Visit http://www.pragmaticprogrammer.com/titles/bmsft for more book information. #---
def month_before(a_time) a_time - 28 * 24 * 60 * 60 end def header(a_time) a_time.strftime("Changes
since %Y-%m-%d:") end def subsystem_line(subsystem_name, change_count) asterisks = asterisks_for
(change_count) "#{subsystem_name.rjust(14)} #{asterisks} (#{change_count})" end def asterisks_for
(an_integer) '*' * (an_integer / 5.0).round end def change_count_for(name, start_date)
extract_change_count_from(svn_log(name, start_date)) end def extract_change_count_from(log_text)
lines = log_text.split("\n") #(1) dashed_lines = lines.find_all do | line | #(2) line.include?('-----') #(3)
end dashed_lines.length - 1 #(4) end def svn_log(subsystem, start_date) timespan = "--revision 'HEAD:
#{start_date}'" root = "svn://rubyforge.org/var/svn/churn-demo" `svn log #{timespan} #{root}/#
{subsystem}` end if $0 == __FILE__ subsystem_names = ['audit', 'fulfillment', 'persistence', 'ui', 'util',
'inventory'] start_date = month_before(Time.now) puts header(start_date) subsystem_names.each do |
name | puts subsystem_line(name, change_count_for(name, start_date)) end end
```

```
#--- # Excerpted from "Everyday Scripting in Ruby" # We make no guarantees that this code is fit for
any purpose. # Visit http://www.pragmaticprogrammer.com/titles/bmsft for more book information. #---
require 'test/unit' require 'churn' class ChurnTests < Test::Unit::TestCase def
test_month_before_is_28_days assert_equal(Time.local(2005, 1, 1), month_before(Time.local(2005, 1,
29))) end def test_header_format assert_equal("Changes since 2005-08-05:", header(month_before(Time.
local(2005, 9, 2)))) end def test_normal_subsystem_line_format assert_equal(' audit ***** (45)',
subsystem_line("audit", 45)) end def test_asterisks_for_divides_by_five assert_equal('****',
asterisks_for(20)) end end
```

```
#--- # Excerpted from "Everyday Scripting in Ruby" # We make no guarantees that this code is fit for
any purpose. # Visit http://www.pragmaticprogrammer.com/titles/bmsft for more book information. #---
def month_before(a_time) a_time - 28 * 24 * 60 * 60 end def header(a_time) a_time.strftime("Changes
since %Y-%m-%d:") end def subsystem_line(subsystem_name, change_count) asterisks = asterisks_for
(change_count) "#{subsystem_name.rjust(14)} #{asterisks} (#{change_count})" end def asterisks_for
(an_integer) end if $0 == __FILE__ subsystem_names = ['audit', 'fulfillment', 'persistence', 'ui', 'util',
'inventory'] start_date = month_before(Time.now) puts header(start_date) subsystem_names.each do |
name | puts subsystem_line(name, change_count_for(name)) end end
```

```
#--- # Excerpted from "Everyday Scripting in Ruby" # We make no guarantees that this code is fit for
any purpose. # Visit http://www.pragmaticprogrammer.com/titles/bmsft for more book information. #---
def month_before(a_time) a_time - 28 * 24 * 60 * 60 end def header(a_time) a_time.strftime("Changes
since %Y-%m-%d:") end def subsystem_line(subsystem_name, change_count) asterisks = asterisks_for
(change_count) "#{subsystem_name.rjust(14)} #{asterisks} (#{change_count})" end def asterisks_for
(an_integer) '*' * (an_integer / 5) end if $0 == __FILE__ subsystem_names = ['audit', 'fulfillment',
'persistence', 'ui', 'util', 'inventory'] start_date = month_before(Time.now) puts header(start_date)
subsystem_names.each do | name | puts subsystem_line(name, change_count_for(name)) end end
```

```
#--- # Excerpted from "Everyday Scripting in Ruby" # We make no guarantees that this code is fit for
any purpose. # Visit http://www.pragmaticprogrammer.com/titles/bmsft for more book information. #---
require 'test/unit' require 'churn' class ChurnTests < Test::Unit::TestCase def
test_month_before_is_28_days assert_equal(Time.local(2005, 1, 1), month_before(Time.local(2005, 1,
29))) end def test_header_format assert_equal("Changes since 2005-08-05:", header(month_before(Time.
local(2005, 9, 2)))) end def test_normal_subsystem_line_format assert_equal(' audit ***** (45)',
subsystem_line("audit", 45)) end def test_asterisks_for_divides_by_five assert_equal('****',
asterisks_for(20)) end def test_asterisks_for_rounds_up_and_down assert_equal('****', asterisks_for
(18)) assert_equal('***', asterisks_for(17)) end end
```

```
#--- # Excerpted from "Everyday Scripting in Ruby" # We make no guarantees that this code is fit for
any purpose. # Visit http://www.pragmaticprogrammer.com/titles/bmsft for more book information. #---
def month_before(a_time) a_time - 28 * 24 * 60 * 60 end def header(a_time) a_time.strftime("Changes
since %Y-%m-%d:") end def subsystem_line(subsystem_name, change_count) asterisks = asterisks_for
(change_count) "#{subsystem_name.rjust(14)} #{asterisks} (#{change_count})" end def asterisks_for
(an_integer) '*' * (an_integer / 5.0).round end if $0 == __FILE__ subsystem_names = ['audit',
'fulfillment', 'persistence', 'ui', 'util', 'inventory'] start_date = month_before(Time.now) puts header
(start_date) subsystem_names.each do | name | puts subsystem_line(name, change_count_for(name))
end end
```

```
#--- # Excerpted from "Everyday Scripting in Ruby" # We make no guarantees that this code is fit for
any purpose. # Visit http://www.pragmaticprogrammer.com/titles/bmsft for more book information. #---
def month_before(a_time) a_time - 28 * 24 * 60 * 60 end def header(a_time) a_time.strftime("Changes
since %Y-%m-%d:") end def subsystem_line(subsystem_name, change_count) asterisks = asterisks_for
(change_count) "#{subsystem_name.rjust(14)} #{asterisks} (#{change_count})" end def asterisks_for
(an_integer) '*' * (an_integer / 5.0).round end def change_count_for(name) extract_change_count_from
(svn_log(name)) end def extract_change_count_from(log_text) lines = log_text.split("\n") #(1)
dashed_lines = lines.find_all do | line | #(2) line.include?('-----') #(3) end dashed_lines.length - 1 #(4)
end if $0 == __FILE__ subsystem_names = ['audit', 'fulfillment', 'persistence', 'ui', 'util', 'inventory']
start_date = month_before(Time.now) puts header(start_date) subsystem_names.each do | name | puts
subsystem_line(name, change_count_for(name)) end end
```

```
#--- # Excerpted from "Everyday Scripting in Ruby" # We make no guarantees that this code is fit for
any purpose. # Visit http://www.pragmaticprogrammer.com/titles/bmsft for more book information. #---
require 'test/unit' require 'churn' class ChurnTests < Test::Unit::TestCase def
test_month_before_is_28_days assert_equal(Time.local(2005, 1, 1), month_before(Time.local(2005, 1,
29))) end def test_header_format assert_equal("Changes since 2005-08-05:", header(month_before(Time.
local(2005, 9, 2)))) end def test_normal_subsystem_line_format assert_equal(' audit ***** (45)',
subsystem_line("audit", 45)) end def test_asterisks_for_divides_by_five assert_equal('****',
asterisks_for(20)) end def test_asterisks_for_rounds_up_and_down assert_equal('****', asterisks_for
(18)) assert_equal('***', asterisks_for(17)) end def test_subversion_log_can_have_no_changes
assert_equal(0, extract_change_count_from("-----\n")) end def test_subversion_log_with_changes assert_equal(2,
extract_change_count_from("-----\n\n-----
=-----\nr2531 | bem | 2005-07-01 01:1\ 1:44 -0500 (Fri, 01 Jul 2005) | 1 line\n\nrevisions up
through ch 3 exer\ cises\n-----
=-----\n\n- \nr2524 | bem | 2005-06-30 18:45:59 -0500 (Thu, 30 Jun 2005)
| 1 line\n\n\nresults of read-through; including renaming mistyping to snapshots\n---\n
-----\n")) end end
```



```
#--- # Excerpted from "Everyday Scripting in Ruby" # We make no guarantees that this code is fit for
any purpose. # Visit http://www.pragmaticprogrammer.com/titles/bmsft for more book information. #---
def month_before(a_time) a_time - 28 * 24 * 60 * 60 end def svn_date(a_time) a_time.strftime("%Y-%
m-%d") end def header(an_svn_date) "Changes since #{an_svn_date}:" end def subsystem_line
(subsystem_name, change_count) asterisks = asterisks_for(change_count) "#{subsystem_name.rjust
(14)} #{asterisks} (#{change_count})" end def asterisks_for(an_integer) '*' * (an_integer / 5.0).round
end def change_count_for(name, start_date) extract_change_count_from(svn_log(name, start_date)) end
def extract_change_count_from(log_text) lines = log_text.split("\n") dashed_lines = lines.find_all do |
line | line.include?('-----') end dashed_lines.length - 1 end def svn_log(subsystem, start_date) timespan
= "--revision 'HEAD:#{start_date}'" root = "svn://rubyforge.org/var/svn/churn-demo" `svn log #
{timespan} #{root}/#{subsystem}` end if $0 == __FILE__ subsystem_names = ['audit', 'fulfillment',
'persistence', 'ui', 'util', 'inventory'] start_date = svn_date(month_before(Time.now)) puts header
(start_date) subsystem_names.each do | name | puts subsystem_line(name, change_count_for(name,
start_date)) end end
```

```
#--- # Excerpted from "Everyday Scripting in Ruby" # We make no guarantees that this code is fit for
any purpose. # Visit http://www.pragmaticprogrammer.com/titles/bmsft for more book information. #---
require 'test/unit' require 'churn' class ChurnTests < Test::Unit::TestCase def
test_month_before_is_28_days assert_equal(Time.local(2005, 1, 1), month_before(Time.local(2005, 1,
29))) end def test_svn_date assert_equal('2005-03-04', svn_date(Time.local(2005, 3, 4))) end def
test_header_format head = header(svn_date(month_before(Time.local(2005, 9, 2)))) assert_equal
("Changes since 2005-08-05:", head) end def test_normal_subsystem_line_format assert_equal(' audit
***** (45)', subsystem_line("audit", 45)) end def test_asterisks_for_divides_by_five assert_equal
('****', asterisks_for(20)) end def test_asterisks_for_rounds_up_and_down assert_equal('****',
asterisks_for(18)) assert_equal('***', asterisks_for(17)) end def
test_subversion_log_can_have_no_changes assert_equal(0, extract_change_count_from
("-----\n")) end def
test_subversion_log_with_changes assert_equal(2, extract_change_count_from
("-----\nr2531 | bem | 2005-07-01 01:11:44 -0500
(Fri, 01 Jul 2005) | 1 line\n\nrevisions up through ch 3 exercises
\n-----\nr2524 | bem | 2005-06-30 18:45:59 -
0500 (Thu, 30 Jun 2005) | 1 line\n\nresults of read-through; including renaming mistyping to snapshots
\n-----\n")) end end
```

```
#---
# Excerpted from "Everyday Scripting in Ruby"
# We make no guarantees that this code is fit for any purpose.
# Visit http://www.pragmaticprogrammer.com/titles/bmsft for more book information.
#---

# This is the same as churn.v7.rb. It's a different file because
# of the mechanics of book production.

def month_before(a_time)
  a_time - 28 * 24 * 60 * 60
end

def header(a_time)
  a_time.strftime("Changes since %Y-%m-%d:")
end

def subsystem_line(subsystem_name, change_count)
  asterisks = asterisks_for(change_count)
  "#{subsystem_name.rjust(14)} #{asterisks} (#{change_count})"
end

def asterisks_for(an_integer)
  '*' * (an_integer / 5.0).round
end

def change_count_for(name, start_date)
  extract_change_count_from(svn_log(name, start_date))
end

def extract_change_count_from(log_text)
  lines = log_text.split("\n")
  dashed_lines = lines.find_all do | line |
    line.include?('-----')
  end
  dashed_lines.length - 1
end

def svn_log(subsystem, start_date)
  timespan = "--revision 'HEAD:#{start_date}'" # (1)
  root = "svn://rubyforge.org/var/svn/churn-demo"
```

```
`svn log #{timespan} #{root}/#{subsystem}`  
end  
  
if $0 == __FILE__  
  subsystem_names = ['audit', 'fulfillment', 'persistence',  
                    'ui', 'util', 'inventory']  
  start_date = month_before(Time.now) #(2)  
  
  puts header(start_date)  
  subsystem_names.each do | name |  
    puts subsystem_line(name, change_count_for(name, start_date)) #(3)  
  end  
end
```

```
#--- # Excerpted from "Everyday Scripting in Ruby" # We make no guarantees that this code is fit for
any purpose. # Visit http://www.pragmaticprogrammer.com/titles/bmsft for more book information. #---
def month_before(a_time) a_time - 28 * 24 * 60 * 60 end def svn_date(a_time) a_time.strftime("%Y-%
m-%d") end def header(an_svn_date) "Changes since #{an_svn_date}:" end def subsystem_line
(subsystem_name, change_count) asterisks = asterisks_for(change_count) "#{subsystem_name.rjust
(14)} #{asterisks} (#{change_count})" end def asterisks_for(an_integer) '*' * (an_integer / 5.0).round
end def change_count_for(name, start_date) extract_change_count_from(svn_log(name, start_date)) end
def extract_change_count_from(log_text) lines = log_text.split("\n") dashed_lines = lines.find_all do |
line | line.include?('-----') end dashed_lines.length - 1 end def svn_log(subsystem, start_date) timespan
= "--revision 'HEAD:#{start_date}'" root = "svn://rubyforge.org/var/svn/churn-demo" `svn log #
{timespan} #{root}/#{subsystem}` end def order_by_descending_change_count(lines) lines.sort do |
one, another | one_count = churn_line_to_int(one) #(1) another_count = churn_line_to_int(another) #(2)
- (one_count <=> another_count) #(3) end end def churn_line_to_int(line) /\((\d+)\)/.match(line)[1].to_i
end if $0 == __FILE__ subsystem_names = ['audit', 'fulfillment', 'persistence', 'ui', 'util', 'inventory']
start_date = svn_date(month_before(Time.now)) puts header(start_date) lines = subsystem_names.
collect do | name | #(4) subsystem_line(name, change_count_for(name, start_date)) end puts
order_by_descending_change_count(lines) #(5) end
```

```
#--- # Excerpted from "Everyday Scripting in Ruby" # We make no guarantees that this code is fit for
any purpose. # Visit http://www.pragmaticprogrammer.com/titles/bmsft for more book information. #---
require 'test/unit' require 'churn' class ChurnTests < Test::Unit::TestCase def
test_month_before_is_28_days assert_equal(Time.local(2005, 1, 1), month_before(Time.local(2005, 1,
29))) end def test_svn_date assert_equal('2005-03-04', svn_date(Time.local(2005, 3, 4))) end def
test_header_format assert_equal("Changes since 2005-08-05:", header(svn_date(month_before(Time.
local(2005, 9, 2))))) end def test_normal_subsystem_line_format assert_equal(' audit ***** (45)',
subsystem_line("audit", 45)) end def test_asterisks_for_divides_by_five assert_equal('****',
asterisks_for(20)) end def test_asterisks_for_rounds_up_and_down assert_equal('****', asterisks_for
(18)) assert_equal('***', asterisks_for(17)) end def test_subversion_log_can_have_no_changes
assert_equal(0, extract_change_count_from("-----
\n")) end def test_subversion_log_with_changes assert_equal(2, extract_change_count_from
("-----\nr2531 | bem | 2005-07-01 01:11:44 -0500
(Fri, 01 Jul 2005) | 1 line\n\nrevisions up through ch 3 exercises
\n-----\nr2524 | bem | 2005-06-30 18:45:59 -
0500 (Thu, 30 Jun 2005) | 1 line\n\nresults of read-through; including renaming mistyping to snapshots
\n-----\n")) end def
test_churn_line_to_int_extracts_parenthesized_change_count assert_equal(19, churn_line_to_int(" ui2
**** (19)")) assert_equal(9, churn_line_to_int(" ui ** (9)")) end def
test_order_by_descending_change_count original = [ "all that really matters is the number in parens -
(1)", " inventory (0)", " ui ** (12)" ] expected = [ " ui ** (12)", "all that really matters is the number in
parens - (1)", " inventory (0)" ] actual = order_by_descending_change_count(original) assert_equal
(expected, actual) end end
```

```
#--- # Excerpted from "Everyday Scripting in Ruby" # We make no guarantees that this code is fit for
any purpose. # Visit http://www.pragmaticprogrammer.com/titles/bmsft for more book information. #---
require 'test/unit' require 'exercise-3' class RearrangeTests < Test::Unit::TestCase def
test_rearrange_with_middle_name assert_equal("Dawn E. Marick", rearrange("Marick, Dawn Elaine"))
end end
```

```
#--- # Excerpted from "Everyday Scripting in Ruby" # We make no guarantees that this code is fit for
any purpose. # Visit http://www.pragmaticprogrammer.com/titles/bmsft for more book information. #---
require 'test/unit' require 'exercise-4' class RearrangeTests < Test::Unit::TestCase def
test_rearrange_with_middle_name assert_equal("Dawn E. Marick", rearrange("Marick, Dawn Elaine"))
end def test_rearrange_without_middle_name assert_equal("Paul Marick", rearrange("Marick, Paul"))
end end
```



```
#---  
# Excerpted from "Everyday Scripting in Ruby"  
# We make no guarantees that this code is fit for any purpose.  
# Visit http://www.pragmaticprogrammer.com/titles/bmsft for more book information.  
#---  
require 'test/unit'  
require 'exercise-1'
```

```
class ChurnTests < Test::Unit::TestCase
```

```
  # Note to book readers:  
  # The current implementation depends less on the fact documented  
  # in the following two tests than the previous one did. But, since  
  # they're still true, I'm leaving the tests.
```

```
  def test_subsystem_line_surrounds_asterisks_with_spaces  
    assert_match(/ \* \d{3}\/, subsystem_line("ui", 3))  
    assert_match(/ \* \d{10}\/, subsystem_line("ui", 10))  
  end
```

```
  def test_subsystem_line_surrounds_even_no_asterisks_with_spaces  
    # ... so interesting() can depend on this, if it needs to.  
    assert_match(/ \d{0}\/, subsystem_line("ui", 0))  
  end
```

```
  def test_interesting_lines_contain_at_least_one_asterisk  
    boring_line = subsystem_line("inventory", 0)  
    interesting_line = subsystem_line("ui", 3)  
    big_line = subsystem_line('util', 61)
```

```
    original = [interesting_line, boring_line, big_line]  
    expected = [interesting_line, big_line]
```

```
    assert_equal(expected, interesting(original))  
  end
```

```
  def test_interesting_lines_subsystem_can_have_asterisk_at_end  
    boring_line = subsystem_line('inventory*', 0)  
    interesting_line = subsystem_line('ui*', 3)
```

```
    original = [interesting_line, boring_line]
```

```
expected = [interesting_line]
```

```
  assert_equal(expected, interesting(original))  
end
```

```
def test_interesting_lines_subsystem_can_have_asterisk_anywhere  
  weird_but_boring = subsystem_line('+ and *** (3) and -', 0)
```

```
  original = [weird_but_boring]  
  expected = []
```

```
  assert_equal(expected, interesting(original))  
end
```

```
end
```

```
#--- # Excerpted from "Everyday Scripting in Ruby" # We make no guarantees that this code is fit for
any purpose. # Visit http://www.pragmaticprogrammer.com/titles/bmsft for more book information. #---
class SubversionRepository def date(a_time) a_time.strftime("%Y-%m-%d") end def change_count_for
(name, start_date) extract_change_count_from(log(name, start_date)) end def
extract_change_count_from(log_text) lines = log_text.split("\n") dashed_lines = lines.find_all do | line |
line.include?('-----') end dashed_lines.length - 1 end def log(subsystem, start_date) timespan = "--
revision 'HEAD:#{start_date}{'" root = "svn://rubyforge.org/var/svn/churn-demo" `svn log #
{timespan} #{root}/#{subsystem}` end end def month_before(a_time) a_time - 28 * 24 * 60 * 60 end
def header(a_date) "Changes since #{a_date}:" end def subsystem_line(subsystem_name,
change_count) asterisks = asterisks_for(change_count) "#{subsystem_name.rjust(14)} #{asterisks} (#
{change_count})" end def asterisks_for(an_integer) '*' * (an_integer / 5.0).round end def
order_by_descending_change_count(lines) lines.sort do | one, another | one_count = churn_line_to_int
(one) another_count = churn_line_to_int(another) - (one_count <=> another_count) end end def
churn_line_to_int(line) /\((\d+)\)/.match(line)[1].to_i end if $0 == __FILE__ subsystem_names =
['audit', 'fulfillment', 'persistence', 'ui', 'util', 'inventory'] repository = SubversionRepository.new #(1)
start_date = repository.date(month_before(Time.now)) #(2) puts header(start_date) lines =
subsystem_names.collect do | name | subsystem_line(name, repository.change_count_for(name,
start_date)) #(3) end puts order_by_descending_change_count(lines) end
```

```
#--- # Excerpted from "Everyday Scripting in Ruby" # We make no guarantees that this code is fit for
any purpose. # Visit http://www.pragmaticprogrammer.com/titles/bmsft for more book information. #---
class SubversionRepository def initialize(root) @root = root # (1) end def date(a_time) a_time.strftime
("%Y-%m-%d") end def change_count_for(name, start_date) extract_change_count_from(log(name,
start_date)) end def extract_change_count_from(log_text) lines = log_text.split("\n") dashed_lines =
lines.find_all do | line | line.include?('-----') end dashed_lines.length - 1 end def log(subsystem,
start_date) timespan = "--revision 'HEAD:#{start_date} '" `svn log #{timespan} #{@root}/#
{subsystem}` # (2) end end def month_before(a_time) a_time - 28 * 24 * 60 * 60 end def header(a_date)
"Changes since #{a_date}:" end def subsystem_line(subsystem_name, change_count) asterisks =
asterisks_for(change_count) "#{subsystem_name.rjust(14)} #{asterisks} (#{change_count})" end def
asterisks_for(an_integer) '*' * (an_integer / 5.0).round end def order_by_descending_change_count
(lines) lines.sort do | one, another | one_count = churn_line_to_int(one) another_count =
churn_line_to_int(another) - (one_count <=> another_count) end end def churn_line_to_int(line) /\((\d+
)\)/.match(line)[1].to_i end if $0 == __FILE__ subsystem_names = ['audit', 'fulfillment', 'persistence', 'ui',
'util', 'inventory'] root="svn://rubyforge.org//var/svn/churn-demo" repository = SubversionRepository.
new(root) start_date = repository.date(month_before(Time.now)) puts header(start_date) lines =
subsystem_names.collect do | name | subsystem_line(name, repository.change_count_for(name,
start_date)) end puts order_by_descending_change_count(lines) end
```

```
#--- # Excerpted from "Everyday Scripting in Ruby" # We make no guarantees that this code is fit for
any purpose. # Visit http://www.pragmaticprogrammer.com/titles/bmsft for more book information. #---
class SubversionRepository def initialize(root) @root = root end def date(a_time) a_time.strftime("%Y-
%m-%d") end def change_count_for(name, start_date) extract_change_count_from(log(name,
start_date)) end def extract_change_count_from(log_text) lines = log_text.split("\n") dashed_lines =
lines.find_all do | line | line.include?('-----') end dashed_lines.length - 1 end def log(subsystem,
start_date) timespan = "--revision 'HEAD:{#{start_date}}'" `svn log #{timespan} #{@root}/#
{subsystem}` end end class Formatter def header(a_date) "Changes since #{a_date}:" end def
subsystem_line(subsystem_name, change_count) asterisks = asterisks_for(change_count) "#
{subsystem_name.rjust(14)} #{asterisks} ({#{change_count}})" end def asterisks_for(an_integer) '*' *
(an_integer / 5.0).round end def order_by_descending_change_count(lines) lines.sort do | first, second |
first_count = churn_line_to_int(first) second_count = churn_line_to_int(second) - (first_count <=>
second_count) end end def churn_line_to_int(line) /\((\d+)\)/.match(line)[1].to_i end end def
month_before(a_time) a_time - 28 * 24 * 60 * 60 end if $0 == __FILE__ subsystem_names = ['audit',
'fulfillment', 'persistence', 'ui', 'util', 'inventory'] root="svn://rubyforge.org//var/svn/churn-demo"
repository = SubversionRepository.new(root) start_date = repository.date(month_before(Time.now))
formatter = Formatter.new puts formatter.header(start_date) lines = subsystem_names.collect do | name |
# (1) formatter.subsystem_line(name, repository.change_count_for(name, start_date)) end puts formatter.
order_by_descending_change_count(lines) end
```

```
#--- # Excerpted from "Everyday Scripting in Ruby" # We make no guarantees that this code is fit for
any purpose. # Visit http://www.pragmaticprogrammer.com/titles/bmsft for more book information. #---
class SubversionRepository def initialize(root) @root = root end def date(a_time) a_time.strftime("%Y-
%m-%d") end def change_count_for(name, start_date) extract_change_count_from(log(name,
start_date)) end def extract_change_count_from(log_text) lines = log_text.split("\n") dashed_lines =
lines.find_all do | line | line.include?('-----') end dashed_lines.length - 1 end def log(subsystem,
start_date) timespan = "--revision 'HEAD:{#{start_date}}'" `svn log #{timespan} #{@root}/#
{subsystem}` end end class Formatter # Public interface def use_date(date_string) @date_string =
date_string end # Helpers def header(a_date) "Changes since #{a_date}:" end def subsystem_line
(subsystem_name, change_count) asterisks = asterisks_for(change_count) "#{subsystem_name.rjust
(14)} #{asterisks} (#{change_count})" end def asterisks_for(an_integer) '*' * (an_integer / 5.0).round
end def order_by_descending_change_count(lines) lines.sort do | first, second | first_count =
churn_line_to_int(first) second_count = churn_line_to_int(second) - (first_count <=> second_count) end
end def churn_line_to_int(line) /\((\d+)\)/.match(line)[1].to_i end end def month_before(a_time) a_time
- 28 * 24 * 60 * 60 end if $0 == __FILE__ subsystem_names = ['audit', 'fulfillment', 'persistence', 'ui',
'util', 'inventory'] root="svn://rubyforge.org/var/svn/churn-demo" repository = SubversionRepository.
new(root) start_date = repository.date(month_before(Time.now)) formatter = Formatter.new formatter.
use_date(start_date) #(1) subsystem_names.each do | name | formatter.
use_subsystem_with_change_count( #(2) name, repository.change_count_for(name, start_date)) end
puts formatter.output #(3) end
```

```
#--- # Excerpted from "Everyday Scripting in Ruby" # We make no guarantees that this code is fit for
any purpose. # Visit http://www.pragmaticprogrammer.com/titles/bmsft for more book information. #---
class SubversionRepository def initialize(root) @root = root end def date(a_time) a_time.strftime("%Y-
%m-%d") end def change_count_for(name, start_date) extract_change_count_from(log(name,
start_date)) end def extract_change_count_from(log_text) lines = log_text.split("\n") dashed_lines =
lines.find_all do | line | line.include?('-----') end dashed_lines.length - 1 end def log(subsystem,
start_date) timespan = "--revision 'HEAD:#{start_date}{' `svn log #{timespan} #{@root}/#
{subsystem}` end end class Formatter # Public interface def initialize @lines = [] end def use_date
(date_string) @date_string = date_string end def use_subsystem_with_change_count(name, count)
@lines.push(subsystem_line(name, count)) end def output ordered_lines =
order_by_descending_change_count(@lines) #(1) output_array = [header(@date_string)] +
ordered_lines #(2) output_array.join("\n") end # Helpers def header(a_date) "Changes since #{a_date}:"
end def subsystem_line(subsystem_name, change_count) asterisks = asterisks_for(change_count) "#
{subsystem_name.rjust(14)} #{asterisks} (#{change_count})" end def asterisks_for(an_integer) '*' *
(an_integer / 5.0).round end def order_by_descending_change_count(lines) lines.sort do | first, second |
first_count = churn_line_to_int(first) second_count = churn_line_to_int(second) - (first_count <=>
second_count) end end def churn_line_to_int(line) /\((\d+)\)/.match(line)[1].to_i end end def
month_before(a_time) a_time - 28 * 24 * 60 * 60 end if $0 == __FILE__ subsystem_names = ['audit',
'fulfillment', 'persistence', 'ui', 'util', 'inventory'] root="svn://rubyforge.org//var/svn/churn-demo"
repository = SubversionRepository.new(root) start_date = repository.date(month_before(Time.now))
formatter = Formatter.new formatter.use_date(start_date) subsystem_names.each do | name | formatter.
use_subsystem_with_change_count( name, repository.change_count_for(name, start_date)) end puts
formatter.output end
```

```
#--- # Excerpted from "Everyday Scripting in Ruby" # We make no guarantees that this code is fit for
any purpose. # Visit http://www.pragmaticprogrammer.com/titles/bmsft for more book information. #---
class SubversionRepository def initialize(root) @root = root end def date(a_time) a_time.strftime("%Y-
%m-%d") end def change_count_for(name, a_time) extract_change_count_from(log(name, date
(a_time))) end def extract_change_count_from(log_text) lines = log_text.split("\n") dashed_lines = lines.
find_all do | line | line.include?('-----') end dashed_lines.length - 1 end def log(subsystem, start_date)
timespan = "--revision 'HEAD:{#{start_date}}'" `svn log #{timespan} #{@root}/#{subsystem}` end
end class Formatter # Public interface def initialize @lines = [] end def report_range(from, to) @from =
from @to = to end def use_subsystem_with_change_count(name, count) @lines.push(subsystem_line
(name, count)) end def output ([header] + lines_ordered_by_descending_change_count).join("\n") end #
Helpers def date(time) date_format = "%B %d, %Y" time.strftime(date_format).sub(' 0', ' ') end def
header "Changes between #{date(@from)}, and #{date(@to)}:" end def subsystem_line
(subsystem_name, change_count) asterisks = asterisks_for(change_count) "#{subsystem_name.rjust
(14)} #{asterisks} (#{change_count})" end def asterisks_for(an_integer) '*' * (an_integer / 5.0).round
end def lines_ordered_by_descending_change_count @lines.sort do | first, second | first_count =
churn_line_to_int(first) second_count = churn_line_to_int(second) - (first_count <=> second_count) end
end def churn_line_to_int(line) /\((\d+)\)/.match(line)[1].to_i end end def month_before(a_time) a_time
- 28 * 24 * 60 * 60 end if $0 == __FILE__ subsystem_names = ['audit', 'fulfillment', 'persistence', 'ui',
'util', 'inventory'] root="svn://rubyforge.org/var/svn/churn-demo" repository = SubversionRepository.
new(root) last_month = month_before(Time.now) #(1) formatter = Formatter.new formatter.report_range
(last_month, Time.now) #(2) subsystem_names.each do | name | formatter.
use_subsystem_with_change_count( name, repository.change_count_for(name, last_month)) #(3) end
puts formatter.output end
```



```
#---  
# Excerpted from "Everyday Scripting in Ruby"  
# We make no guarantees that this code is fit for any purpose.  
# Visit http://www.pragmaticprogrammer.com/titles/bmsft for more book information.  
#---
```

```
class Time  
  def self.right_now  
    new  
  end  
end
```

```
# I'm using "class Time" twice so the examples look better in the  
# book. Both methods could just as well be in the same class block.
```

```
class Time  
  def self.yesterday  
    right_now - 24 * 60 * 60  
  end  
end
```

```
#--- # Excerpted from "Everyday Scripting in Ruby" # We make no guarantees that this code is fit for  
any purpose. # Visit http://www.pragmaticprogrammer.com/titles/bmsft for more book information. #---  
class Counter def self.counted_new @count = 0 if @count.nil? @count += 1 new end def self.count  
@count end end
```

```
#---
# Excerpted from "Everyday Scripting in Ruby"
# We make no guarantees that this code is fit for any purpose.
# Visit http://www.pragmaticprogrammer.com/titles/bmsft for more book information.
#---
require 'uri'
require 'open-uri'
require 'csv'

### This does the bulk of the work

# This simple version has some problems, but is the
# framework for the more complete version.

def trip(url, steps=10) # (1)

  steps.times do # (2)
    page = fetch(url)
    book_info = scrape_book_info(page)
    puts format_output(book_info)

    next_book = scrape_affinity_list(page)[0]

    url = next_book[:url]
  end
end

# This more complete version replaces the simple version above.

def trip(url, steps=10)
  so_far = [] # (3)

  steps.times do
    page = fetch(url)
    book_info = scrape_book_info(page)
    so_far << book_info[:title]
    puts format_output(book_info)

    next_book = scrape_affinity_list(page).find do | possible | # (4)
      not so_far.include?(possible[:title])
    end
  end
end
```

```
    url = next_book[:url]
  end
end
```

### Fetching Amazon book pages

```
def url_for(isbn)
  "http://www.amazon.com/gp/product/" + isbn
end
```

```
def fetch(url)
  open(url) { | response |
    response.read
  }
end
```

### Scraping information out of Amazon book pages

```
def scrape_book_info(html)
  retval = { }
  html = restrict(html,
    /<div.+class\s*=\s*"buying".*?>\s*<b/,
    %r{</div\s*>})
  retval[:title] = scrape_title(html)
  retval[:authors] = scrape_authors(html)
  retval
end
```

```
def scrape_title(html)
  %r{<b.*?>(.*?)</b\s*>}m =~ html
  clean_title($1)
end
```

```
def scrape_authors(html)
  author_anchor = %r{<a.*?href=".*?field-author-exact.*?".*?>(.*?)</a\s*>}m
```

```
html.scan(author_anchor).flatten.collect do | author |
  clean_author(author)
end
end

def restrict(html, starting_regexp, stopping_regexp)
  start = html.index(starting_regexp)
  stop = html.index(stopping_regexp, start)
  html[start..stop]
end

def clean_title(amazon_title)
  # The regexps below have duplication. It could be removed, but
  # regexps are hard enough to understand as it is.

  paper = /\(\\s*Paperback\\s*\\)\\s*$ /m
  hard = /\(\\s*Hardcover\\s*\\)\\s*$ /m
  amazon_title.gsub(paper, "").gsub(hard, "").strip.squeeze(' ')
end

def clean_author(amazon_author)
  amazon_author.squeeze(' ')
end

def scrape_affinity_list(html)
  result = []
  whole_list_matches = %r{also\\s+bought.*?</ul\\s*>}m
  one_element_matches = %r{<li.*?>.*?</li\\s*>}m

  html_affinity_list = html[whole_list_matches]
  html_affinity_list.scan(one_element_matches).collect do | item |
    { :url => /href\\s*=\\s*"(.*)" /.match(item)[1],
      :title => %r{<a.*?>(.*)</a\\s*>} .match(item)[1] }
  end
end

# How to print

def format_output(book_info)
  self.send(FORMAT_STYLE, book_info)
```

end

```
def normal_string(book_info)
  book_info[:title] # omit authors
end
```

```
def csv_string(book_info)
  title = book_info[:title]
  authors = book_info[:authors].join(', ')
  CSV.generate_line([title, authors])
end
```

```
if $0 == __FILE__
```

```
  if ARGV[0] == '--csv'
    FORMAT_STYLE = :csv_string
    ARGV.shift
  else
    FORMAT_STYLE = :normal_string
  end
```

```
  starting_isbn = ARGV[0] || '0974514055'
  trip(url_for(starting_isbn))
end
```

```
#---
# Excerpted from "Everyday Scripting in Ruby"
# We make no guarantees that this code is fit for any purpose.
# Visit http://www.pragmaticprogrammer.com/titles/bmsft for more book information.
#---
require 'test/unit'
require 'rexml/document'
include REXML # You don't need to understand this to understand the test.
              # See the chapter on modules if you want to.

class XHTMLExample < Test::Unit::TestCase

  def test_xhtml_included_in_document
    page_text = IO.read("www.w3.org.html") # (1)
    document = Document.new(page_text)      # (2)
    topics = XPath.match(document, '//li/a/abbr') # (3)
    assert(topics.find { | topic | topic.text.strip == "XML" }) # (4)
  end

end
```

```
#---
# Excerpted from "Everyday Scripting in Ruby"
# We make no guarantees that this code is fit for any purpose.
# Visit http://www.pragmaticprogrammer.com/titles/bmsft for more book information.
#---
require 'test/unit'
require 'watir'
include Watir # You don't need to understand this to understand the test.
               # See the chapter on modules if you want to.

class WatirExample < Test::Unit::TestCase

  def test_marick_vanity
    ie = IE.new # Launch Internet Explorer

    ie.goto('http://www.google.com')

    # If you view the HTML source, you can see that Google
    # names the search field 'q'.
    ie.text_field(:name, "q").set("scripting for testers")

    # 'btnI' is the name of the "I'm Feeling Lucky" button.
    ie.button(:name, "btnI").click

    # Case-insensitive search for my name.
    assert(ie.contains_text(/marick/i))
  end
end

end
```



```
#---  
# Excerpted from "Everyday Scripting in Ruby"  
# We make no guarantees that this code is fit for any purpose.  
# Visit http://www.pragmaticprogrammer.com/titles/bmsft for more book information.  
#---  
require 'test/unit'  
require 'net/http'  
require 'cgi'  
include Net # You don't need to understand this to understand the test.  
            # See the chapter on modules if you want to.
```

```
class HttpExample < Test::Unit::TestCase  
  
  def test_echoing  
    HTTP.start('www.testing.com') do |server| # (1)  
      name = CGI.escape("Brian Marick") # (2)  
      count = CGI.escape("3")  
      params = "name=#{name}&count=#{count}" # (3)  
      response = server.post('/cgi-bin/post-example', params) # (4)  
      assert_match(/Brian Marick/, response.body) # (5)  
      assert_match(/count is 3/, response.body)  
    end  
  end  
  
end  
  
end
```

```
#--- # Excerpted from "Everyday Scripting in Ruby" # We make no guarantees that this code is fit for
any purpose. # Visit http://www.pragmaticprogrammer.com/titles/bmsft for more book information. #---
class String def center(field_width, padding=" ") left_width = (field_width - self.length) / 2 left_width =
0 if left_width < 0 right_width = field_width - left_width - self.length right_width = 0 if right_width < 0
left = (padding * left_width)[0, left_width] right = (padding * right_width)[0, right_width] left + self +
right end end
```

```
#---
# Excerpted from "Everyday Scripting in Ruby"
# We make no guarantees that this code is fit for any purpose.
# Visit http://www.pragmaticprogrammer.com/titles/bmsft for more book information.
#---
require 'cs-valid'

module MyModule
  class Viewer
    include CsValid    # A class-level inclusion

    def hello
      "I can see Reader: " + Reader.new.hello
    end
  end

  class Oblivious
    # "This class does not include CsValid."
    # "So the following method, if used, will fail:"
    def hello
      "I can see Reader too... or can I?" + Reader.new.hello
    end
  end
end

puts "Reader can be seen within Viewer:"
puts MyModule::Viewer::new.hello

puts "Reader can't be seen outside the class..."
puts "...so the following will fail if uncommented:"
# puts Reader.new.hello

puts "So will this class that didn't include CsValid:"
# puts MyModule::Oblivious.new.hello
```

```
#--- # Excerpted from "Everyday Scripting in Ruby" # We make no guarantees that this code is fit for  
any purpose. # Visit http://www.pragmaticprogrammer.com/titles/bmsft for more book information. #---  
module S4tUtils Version = '0.2.0' end
```

```
#--- # Excerpted from "Everyday Scripting in Ruby" # We make no guarantees that this code is fit for
any purpose. # Visit http://www.pragmaticprogrammer.com/titles/bmsft for more book information. #---
module SomeModule
  class SomeClass
    def hello "hi"
    end
    Constant = "I am a constant."
    def self.
      speak "I am a module method."
    end
    def speak "I am a mixin method."
    end
  end
end
module SomeModule
  module InnerModule
    Constant = "I am an inner constant."
    def self.speak "I am an inner module
      method."
    end
    def speak "I am an inner mixin method."
    end
    def SomeClass
      def hello "an inner hello"
      end
    end
  end
end
```

```
#!/usr/bin/ruby
```

```
### The following adjusts the load path so that the correct version of  
### a self-contained package is found, no matter where the script is  
### run from.
```

```
require 'pathname'
```

```
$.unshift((Pathname.new(__FILE__).parent.parent + 'lib').to_s)
```

```
require 'error-handling/third-party/s4t-utils/load-path-auto-adjuster'
```

```
require 's4t-utils'
```

```
include S4tUtils
```

```
require 'error-handling'
```

```
# You probably want to include your module as well, but I won't assume  
# that.
```

```
# include ErrorHandler
```

```
if $0 == __FILE__
```

```
#START:main
```

```
  with_pleasant_exceptions do
```

```
    # Your program here.
```

```
  end
```

```
#END:main
```

```
end
```

```
#!/usr/bin/ruby
#---
# Excerpted from "Everyday Scripting in Ruby"
# We make no guarantees that this code is fit for any purpose.
# Visit http://www.pragmaticprogrammer.com/titles/bmsft for more book information.
#---

### The following adjusts the load path so that the correct version of
### a self-contained package is found, no matter where the script is
### run from.
require 'pathname'
$:.unshift((Pathname.new(__FILE__).parent.parent + 'lib').to_s)
require 'error-handling/third-party/s4t-utils/load-path-auto-adjuster'

require 's4t-utils'
include S4tUtils

require 'error-handling'
# You probably want to include your module as well, but I won't assume
# that.
# include ErrorHandler

if $0 == __FILE__

  with_pleasant_exceptions do
    File.open("no-such-file")
  end

end
```

```
#--- # Excerpted from "Everyday Scripting in Ruby" # We make no guarantees that this code is fit for  
any purpose. # Visit http://www.pragmaticprogrammer.com/titles/bmsft for more book information. #---  
begin File.open("no-such-file") puts "You will never see me." rescue Exception => ex puts ex.message  
end puts "End of script"
```



```
#--- # Excerpted from "Everyday Scripting in Ruby" # We make no guarantees that this code is fit for
any purpose. # Visit http://www.pragmaticprogrammer.com/titles/bmsft for more book information. #---
def level3 File.open("no-such-file") end def level2 level3 end begin # line 11 level2 rescue Exception =>
ex # line 13 puts ex.message end
```

```
#--- # Excerpted from "Everyday Scripting in Ruby" # We make no guarantees that this code is fit for
any purpose. # Visit http://www.pragmaticprogrammer.com/titles/bmsft for more book information. #---
def convert_to_integer(string) unless /^-?\d+$/ =~ string raise RuntimeError.new("'#{string}' is not an
integer.") # (1) end string.to_i end begin raise RuntimeError.new("An argument is required.") unless
ARGV[0] puts convert_to_integer(ARGV[0]) rescue Exception => ex puts ex.message end
```

```
#---
# Excerpted from "Everyday Scripting in Ruby"
# We make no guarantees that this code is fit for any purpose.
# Visit http://www.pragmaticprogrammer.com/titles/bmsft for more book information.
#---

def log(owner, exception)
  File.open("exception.log", 'a') { |io| # "a" means append to the file.
    io.puts(Time.now)
    io.puts(exception.class)
    io.puts(owner + ': ' + exception.message)
    io.puts(exception.backtrace)
  }
end

def owned_open(owner, name)
  File.open(name)
rescue Errno::ENOENT => ex
  log(owner, ex)
  raise
end

begin
  f = owned_open('marick', "no-such-file")
  puts f.readlines
rescue Exception => ex
  puts ex.message
end
```

```
#--- # Excerpted from "Everyday Scripting in Ruby" # We make no guarantees that this code is fit for  
any purpose. # Visit http://www.pragmaticprogrammer.com/titles/bmsft for more book information. #---  
def simple yield puts 'block done.' end  
def with_arg(arg) yield arg end
```

```
#--- # Excerpted from "Everyday Scripting in Ruby" # We make no guarantees that this code is fit for
any purpose. # Visit http://www.pragmaticprogrammer.com/titles/bmsft for more book information. #---
def with_pleasant_exceptions begin yield rescue SystemExit #(1) raise rescue Exception => ex #(2)
$stderr.puts(ex.message) #(3) end end
```

```
#--- # Excerpted from "Everyday Scripting in Ruby" # We make no guarantees that this code is fit for
any purpose. # Visit http://www.pragmaticprogrammer.com/titles/bmsft for more book information. #---
class TestRunner def download(tests, machine) @tests = tests tests.each do | test | puts "Downloading #
{test} to #{machine.name}..." end end def run @tests.each do | test | puts "Running #{test}..." raise
"network down" if test == 'test3' end end end class Machine attr_reader :name def initialize(name)
@name = name raise "Could not reserve #{name}." if ARGV[0] end end class Reserver def reserve
(machine_name) machine = Machine.new(machine_name) puts "Reserved #{machine.name}." machine
end def release(machine) puts "Released #{machine.name}." end end reserver = Reserver.new
test_runner = TestRunner.new tests = ['test1', 'test2', 'test3'] # This works begin machine = reserver.
reserve('Mycroft') test_runner.download(tests, machine) test_runner.run rescue Exception => ex puts
"Test failure: #{ex.message}" ensure reserver.release(machine) if machine # (1) end # But I want this: #
reserver.reserve('Mycroft') do | machine | # test_runner.download(tests, machine) # test_runner.run # end
```

```
#--- # Excerpted from "Everyday Scripting in Ruby" # We make no guarantees that this code is fit for
any purpose. # Visit http://www.pragmaticprogrammer.com/titles/bmsft for more book information. #---
class TestRunner def download(tests, machine) @tests = tests tests.each do | test | puts "Downloading #{
test} to #{machine.name}..." end end def run @tests.each do | test | puts "Running #{test}..." raise
"network down" if test == 'test3' end end end class Machine attr_reader :name def initialize(name)
@name = name raise "Could not reserve #{name}." if ARGV[0] end end class Reserver def reserve
(machine_name) machine = Machine.new(machine_name) yield machine puts "Reserved #{machine.
name}." machine rescue Exception => ex puts "Test failure: #{ex.message}" ensure release(machine) if
machine end def release(machine) puts "Released #{machine.name}." end end reserver = Reserver.new
test_runner = TestRunner.new tests = ['test1', 'test2', 'test3'] reserver.reserve('Mycroft') do | machine |
test_runner.download(tests, machine) test_runner.run end
```

```
#--- # Excerpted from "Everyday Scripting in Ruby" # We make no guarantees that this code is fit for  
any purpose. # Visit http://www.pragmaticprogrammer.com/titles/bmsft for more book information. #---  
require 'test/unit' class Inheritor < Test::Unit::TestCase def test_announcer puts "This class inherits  
TestCase's behavior." end end
```



```
#--- # Excerpted from "Everyday Scripting in Ruby" # We make no guarantees that this code is fit for
any purpose. # Visit http://www.pragmaticprogrammer.com/titles/bmsft for more book information. #---
require 'test/unit' class MyNonTest < Test::Unit::TestCase def run(*ignore_all_arguments) puts
"Nothing will be run because I override my ancestor." end def test_not puts "I will not be run." end end
```

```
#--- # Excerpted from "Everyday Scripting in Ruby" # We make no guarantees that this code is fit for  
any purpose. # Visit http://www.pragmaticprogrammer.com/titles/bmsft for more book information. #---  
require 'test/unit' class MyExtendedTest < Test::Unit::TestCase def run(*args) puts "About to run."  
super puts "Done running." end def test_extension puts "I will be run verbosely." end end
```

```
#--- # Excerpted from "Everyday Scripting in Ruby" # We make no guarantees that this code is fit for
any purpose. # Visit http://www.pragmaticprogrammer.com/titles/bmsft for more book information. #---
class Parent
  def arglist_taker(one_arg)
    puts "Parent: I was given #{one_arg}."
  end
end
class Child < Parent
  def arglist_taker(one_arg, another)
    puts "Child: I have two arguments: #{one_arg} and #{another}."
  end
end
Child.new.arglist_taker(1, 2)
```

```
#--- # Excerpted from "Everyday Scripting in Ruby" # We make no guarantees that this code is fit for
any purpose. # Visit http://www.pragmaticprogrammer.com/titles/bmsft for more book information. #---
class Parent def arglist_taker(*args) puts "Parent: I was given #{args.inspect}." end end class Child <
Parent def arglist_taker(one_arg, another) puts "Child: I have two arguments: #{one_arg} and #
{another}." super() # (1) end end Child.new.arglist_taker(1, 2)
```

```
#--- # Excerpted from "Everyday Scripting in Ruby" # We make no guarantees that this code is fit for
any purpose. # Visit http://www.pragmaticprogrammer.com/titles/bmsft for more book information. #---
class NoteTaker attr_reader :commentary def initialize(title) # (1) @commentary = [title] end def note
(notations) # (2) @commentary << "Note: #{notations}" # (3) end end class TimingNoteTaker <
NoteTaker def timestamp # (4) note(boundary('-')) # (5) @commentary << Time.now.to_s # (6) end def
boundary(character) # (7) character * 20 end end child = TimingNoteTaker.new("May 1") # (8) child.
timestamp # (9) child.note("coffee") # (10) child.note("bagels") child.timestamp child.note('email') puts
child.commentary
```

```
#---
# Excerpted from "Everyday Scripting in Ruby"
# We make no guarantees that this code is fit for any purpose.
# Visit http://www.pragmaticprogrammer.com/titles/bmsft for more book information.
#---
require 'user-choices'
require 'pp'
require 'watchdog/site-defaults'
require 'watchdog/kennel'

module Watchdog
  include UserChoices

  class WatchdogCommand < Command

    def add_sources(builder)
      builder.add_source(PosixCommandLineChoices, :usage,
        "Usage: ruby #{ $0 } [options] program args...",
        "Site-wide defaults are noted below.",
        "Override them in the '#{ RC_FILE }' file in your home folder.")
      builder.add_source(XmlConfigFileChoices, :from_file, RC_FILE) # (1)
    end

    def add_choices(builder)
      builder.add_choice(:jabber,
        :type => :boolean,
        :default => DEFAULT_JABBER) { | command_line |
        command_line.uses_switch('-j', "--jabber",
          "Control IM notification.",
          "Defaults to #{ DEFAULT_JABBER }.")
      }

      builder.add_choice(:mail,
        :type => :boolean,
        :default => DEFAULT_MAIL) { | command_line |
        command_line.uses_switch('-m', "--mail",
          "Control mail notification.",
          "Defaults to #{ DEFAULT_MAIL }.")
      }
    end
  end
end
```

```
builder.add_choice(:command_line,  
  :type => :boolean,  
  :default => DEFAULT_COMMAND_LINE) { | command_line |  
  command_line.uses_switch("-t", "--terminal", '-s',  
    "Control display to terminal (standard output).",  
    "Defaults to #{DEFAULT_COMMAND_LINE}.")  
}
```

```
builder.add_choice(:mail_to,  
  :default => DEFAULT_MAIL_TO,  
  :type => [:string]) { | command_line |  
  command_line.uses_option("--mail-to RECIPIENTS",  
    "Recipients of mail.",  
    "This can be a comma-separated list.",  
    "Defaults to #{DEFAULT_MAIL_TO}."  
  )  
}
```

```
builder.add_choice(:mail_from,  
  :default => DEFAULT_MAIL_FROM) { | command_line |  
  command_line.uses_option("--mail-from SENDER",  
    "The sender of the mail (appears in From line).",  
    "Defaults to #{DEFAULT_MAIL_FROM}."  
  )  
}
```

```
builder.add_choice(:mail_server,  
  :default => DEFAULT_SMTP_SERVER) { | command_line |  
  command_line.uses_option("--mail-server HOSTNAME",  
    "SMTP server. Defaults to #{DEFAULT_SMTP_SERVER}."  
  )  
}
```

```
builder.add_choice(:mail_port,  
  :type => :integer,  
  :default => DEFAULT_SMTP_PORT) { | command_line |  
  command_line.uses_option("--mail-port NUMBER",  
    "Mail port on that server. (Defaults to #{DEFAULT_SMTP_PORT})."  
  )  
}
```

```
builder.add_choice(:mail_account,
```

```
        :default => DEFAULT_MAIL_ACCOUNT) { | command_line |
command_line.uses_option("--mail-account USERNAME",
        "Your account name on the SMTP server.",
        "Defaults to #{DEFAULT_MAIL_ACCOUNT}."
    )
}

builder.add_choice(:mail_from_domain,
    :default => DEFAULT_FROM_DOMAIN) { | command_line |
command_line.uses_option("--mail-from-domain HOSTNAME",
        "The server the mail supposedly comes from.",
        "(Not necessarily the SMTP server.)",
        "Defaults to #{DEFAULT_FROM_DOMAIN}."
    )
}

builder.add_choice(:mail_password,
    :default => DEFAULT_SMTP_PASSWORD) { | command_line |
command_line.uses_option("--mail-password HOSTNAME",
        "Your password on the SMTP server.",
        "Defaults to #{DEFAULT_SMTP_PASSWORD}."
    )
}

auth_types = ['plain', 'login', 'cram_md5']
builder.add_choice(:mail_authentication,
    :type => auth_types,
    :default => DEFAULT_SMTP_AUTH) { | command_line |
command_line.uses_option("--mail-authentication TYPE",
        "The kind of authentication your SMTP server uses.",
        "One of #{friendly_list('or', auth_types)}.",
        "Defaults to #{DEFAULT_SMTP_AUTH}."
    )
}

builder.add_choice(:jabber_to,
    :default => DEFAULT_JABBER_TO,
    :type => [:string]) { | command_line |
command_line.uses_option("--jabber-to RECIPIENTS",
        "Recipients of Jabber instant messages.",
        "This can be a comma-separated list.",
        "Defaults to #{DEFAULT_JABBER_TO}."
    )
}
```



```
)
}

builder.add_choice(:jabber_account,
  :default => DEFAULT_JABBER_ACCOUNT) { | command_line |
  command_line.uses_option("--jabber-account SENDER",
    "Your Jabber account.",
    "Note that this includes the Jabber server.",
    "Defaults to #{DEFAULT_JABBER_ACCOUNT}."
  )
}

builder.add_choice(:jabber_password,
  :default => DEFAULT_JABBER_PASSWORD) { | command_line |
  command_line.uses_option("--jabber-password PASSWORD",
    "Your Jabber password.",
    "Defaults to #{DEFAULT_JABBER_PASSWORD}."
  )
}
```

```
builder.add_choice(:command_to_watch) { | command_line |
  command_line.uses_arglist
}
```

```
builder.add_choice(:choices,
  :default => DEFAULT_SHOW_CHOICES,
  :type => :boolean) { | command_line |
  command_line.uses_switch('-c', "--choices",
    "Show all configuration choices.")
}
```

end

```
def postprocess_user_choices
  if @user_choices[:choices]
    puts "Choices gathered from all sources:"
    pp @user_choices
  end
end
```

```
puts "Looking for configuration information in:"
puts File.join($4tUtils.find_home, RC_FILE)
end
```

```
@user_choices[:mail_authentication] =
  @user_choices[:mail_authentication].to_sym
errors = []
```

```
# This is checked here, instead of giving a range to uses_arglist,
# so that the actual choices can be printed out with -c without
# having to give an argument.
if @user_choices[:command_to_watch].empty?
  errors << "No command to run was given."
end
```

```
@kennel = Kennel.new # (2)
Barker::Subclasses.each do | barker_class | # (3)
  barker = barker_class.new(@user_choices) # (4)
  errors += barker.errors # (5)
  barker.invite_into(@kennel) # (6)
end
```

```
raise errors.join("\n") unless errors.empty? # (7)
```

```
end
```

```
end
```

```
end
```

```
#!/opt/local/bin/ruby #--- # Excerpted from "Everyday Scripting in Ruby" # We make no guarantees
that this code is fit for any purpose. # Visit http://www.pragmaticprogrammer.com/titles/bmsft for more
book information. #--- require 'pathname' $:.unshift((Pathname.new(__FILE__).parent.parent + 'lib').
to_s) require 'watchdog/third-party/s4t-utils/load-path-auto-adjuster' require 'extensions/string' require
'open3' require 's4t-utils' include S4tUtils require 'watchdog' include Watchdog class
WatchdogCommand < Command def command_string(command_to_watch = @user_choices[:
command_to_watch]) command_to_watch.join(' ') end def command_name(command_to_watch =
@user_choices[:command_to_watch]) progame = if command_to_watch[0] == 'ruby'
command_to_watch[1] else command_to_watch[0] end File.basename(progame) end def message
(duration, text) [ "Duration: #{duration} seconds.", "Command: #{command_string}", "Output:", text.
indent(2), ].join("\n") end def execute # (1) duration, text = Watchdog.time { # (2) `#{self.
command_string} 2>&1` # (3) } title = "Program #{self.command_name} finished." @kennel.bark(title,
message(duration, text)) # (4) end end if $0 == __FILE__ with_pleasant_exceptions do
WatchdogCommand.new.execute # (5) end end
```

```
#--- # Excerpted from "Everyday Scripting in Ruby" # We make no guarantees that this code is fit for
any purpose. # Visit http://www.pragmaticprogrammer.com/titles/bmsft for more book information. #---
require 'net/smtp' require 'xmpp4r' require 'extensions/string' module Watchdog class Barker def bark
(subject, body); subclass_responsibility; end def name; subclass_responsibility; end def symbol; :
this_barker_can_never_be_chosen; end def validate; []; end attr_reader :errors def initialize
(user_choices = {}) @user_choices = user_choices @errors = self.validate end def invite_into(kennel)
return unless errors.empty? return unless wanted? kennel.add(self) end def wanted? @user_choices[self.
symbol] end Subclasses = [] def self.inherited(new_child_class) Subclasses << new_child_class end end
class StdoutBarker < Barker def name; "terminal"; end def symbol; :command_line; end def bark
(subject, body) all = [subject, body].join("\n") puts all.indent(2) end end class JabberBarker < Barker
include Jabber def name; "jabber"; end def symbol; :jabber; end def validate errors = [] unless
@user_choices[:jabber_account] errors << "There is no Jabber account to log in as. (jabber_account)"
end unless @user_choices[:jabber_password] errors << "There is no Jabber password.
(jabber_password)" end jabber_to = @user_choices[:jabber_to] if jabber_to == nil or jabber_to.empty?
errors << "There is no destination for Jabber messages. (jabber_to)" end errors end # This shows a
possible implementation for the book: def invite_into(kennel) kennel.add(self) if @user_choices[:jabber]
end # Use the superclass method for real. def invite_into(kennel); super; end def bark(subject, body)
my_jid = JID.new(@user_choices[:jabber_account]) cl = Client.new(my_jid, false) cl.connect cl.auth
(@user_choices[:jabber_password]) body = [subject, body].join("\n") @user_choices[:jabber_to].each
do | recipient | m = Message::new(recipient, body). set_type(:normal).set_id('1').set_subject(subject) cl.
send(m) end cl.close end end class SmtpBarker < Barker def name; 'mail'; end def symbol; :mail; end
def validate errors = [] mail_to = @user_choices[:mail_to] if mail_to == nil or mail_to.empty? errors <<
"There are no recipients. (mail_to)" end unless @user_choices[:mail_from] errors << "There is nothing
for the From line. (mail_from)" end unless @user_choices[:mail_server] errors << "There is no server to
send to. (mail_server)" end unless @user_choices[:mail_account] errors << "There is no SMTP account
to send mail through. (mail_account)" end unless @user_choices[:mail_password] errors << "There is
no password for the SMTP account. (mail_password)" end unless @user_choices[:mail_port] errors <<
"There is no port for the SMTP host. (mail_port)" end unless @user_choices[:mail_from_domain] errors
<< "There is no domain to use as the From domain. (mail_from_domain)" end unless @user_choices[:
mail_authentication] errors << "No authentication method was chosen. (mail_authentication)" end errors
end def bark(subject, message) from = @user_choices[:mail_from] to = @user_choices[:mail_to] Net::
SMTP.start(@user_choices[:mail_server], @user_choices[:mail_port], @user_choices[:
mail_from_domain], @user_choices[:mail_account], @user_choices[:mail_password], @user_choices[:
mail_authentication]) { | smtp | mail = " . From: #{from} . To: #{to.join(', ')} . Subject: [watchdog] #
{subject} . . #{message} ".trim('.') smtp.send_message(mail, from, to) } end end end
```

```
#---
# Excerpted from "Everyday Scripting in Ruby"
# We make no guarantees that this code is fit for any purpose.
# Visit http://www.pragmaticprogrammer.com/titles/bmsft for more book information.
#---
require 'watchdog/multi-exceptions'
require 'watchdog/barkers'

module Watchdog

  # A Kennel is a collection of barkers who all bark as one.
  class Kennel

    attr_reader :barkers
    def initialize
      @barkers = []
    end

    def add(*barkers)
      @barkers += barkers
    end

    def bark(subject, message)
      simultaneously do | barker |
        barker.bark(subject, message)
      end
    end

    def annotate(exception, barker)
      # Note: cannot raise exception.class because on Windows the
      # exception message for Net errors will contain duplicate errors.
      # This is annoying because with_pleasant_exceptions appends the
      # exception class. So we have to append it ourselves.
      annotated = StandardError.new("Complaint from #{barker.name}: #{exception.to_s} (#{exception.class})")
      annotated.set_backtrace(exception.backtrace)
      annotated
    end

    def simultaneously # execute block in thread
      Thread.abort_on_exception = false
      queue = Queue.new
      threads = @barkers.collect do | barker |
```

```
Thread.new(barker) do | barker |  
  begin  
    yield(barker)  
  rescue Exception => ex  
    queue << annotate(ex, barker)  
  end  
end  
end  
threads.each { | thread | thread.join }
```

```
# This peculiarness is the way to set the exception's  
# backtrace with the combination of the backtraces of  
# all saved exceptions (if any)  
MultiException.reraise_with_combined_backtrace {  
  raise MultiException.new(queue.to_a) if queue.length > 0  
}  
end  
end  
end
```

```
#--- # Excerpted from "Everyday Scripting in Ruby" # We make no guarantees that this code is fit for
any purpose. # Visit http://www.pragmaticprogrammer.com/titles/bmsft for more book information. #---
class Threer include Enumerable def each yield(1) yield(2) yield(3) end end
```

```
#--- # Excerpted from "Everyday Scripting in Ruby" # We make no guarantees that this code is fit for
any purpose. # Visit http://www.pragmaticprogrammer.com/titles/bmsft for more book information. #---
old_inventory = File.open('old-inventory.txt').readlines new_inventory = File.open('new-inventory.txt').
readlines puts "The following files have been added:" puts new_inventory - old_inventory puts "" puts
"The following files have been deleted:" puts old_inventory - new_inventory new_count =
(new_inventory - old_inventory).length deleted_count = (old_inventory - new_inventory).length
common_count = new_inventory.length - new_count puts "" puts "New files added:" puts new_count
puts "Old files deleted:" puts deleted_count puts "Files in common:" puts common_count
```



```
#--- # Excerpted from "Everyday Scripting in Ruby" # We make no guarantees that this code is fit for
any purpose. # Visit http://www.pragmaticprogrammer.com/titles/bmsft for more book information. #---
def check_usage if ARGV[0].nil? puts "Usage: differences.rb old-inventory new-inventory" exit end end
def boring?(line) line.chomp.split('/').include?('temp') or line.chomp.split('/').include?('recycler') end def
inventory_from(filename) inventory = File.open(filename) downcased = inventory.collect do | line | line.
downcase end downcased.reject do | line | boring?(line) end end def compare_inventory_files(old_file,
new_file) old_inventory = inventory_from(old_file) new_inventory = inventory_from(new_file) puts
"The following files have been added:" puts new_inventory - old_inventory puts "" puts "The following
files have been deleted:" puts old_inventory - new_inventory end if $0 == __FILE__ check_usage
compare_inventory_files(ARGV[0], ARGV[1]) end
```

```
#--- # Excerpted from "Everyday Scripting in Ruby" # We make no guarantees that this code is fit for
any purpose. # Visit http://www.pragmaticprogrammer.com/titles/bmsft for more book information. #---
def check_usage if ARGV[0].nil? puts "Usage: differences.rb old-inventory new-inventory" exit end end
def boring?(line) contains?(line, 'temp') or contains?(line, 'recycler') end def contains?(line,
a_boring_word) line.chomp.split('/').include?(a_boring_word) end def inventory_from(filename)
inventory = File.open(filename) downcased = inventory.collect do | line | line.downcase end downcased.
reject do | line | boring?(line) end end def compare_inventory_files(old_file, new_file) old_inventory =
inventory_from(old_file) new_inventory = inventory_from(new_file) puts "The following files have
been added:" puts new_inventory - old_inventory puts "" puts "The following files have been deleted:"
puts old_inventory - new_inventory end if $0 == __FILE__ check_usage compare_inventory_files
(ARGV[0], ARGV[1]) end
```

```
#--- # Excerpted from "Everyday Scripting in Ruby" # We make no guarantees that this code is fit for
any purpose. # Visit http://www.pragmaticprogrammer.com/titles/bmsft for more book information. #---
def check_usage if ARGV[0].nil? puts "Usage: differences.rb old-inventory new-inventory" exit end end
def boring?(line, boring_words) boring_words.any? do | a_boring_word | contains?(line,
a_boring_word) end end def contains?(line, a_boring_word) line.chomp.split('/').include?
(a_boring_word) end def inventory_from(filename) inventory = File.open(filename) downcased =
inventory.collect do | line | line.downcase end downcased.reject do | line | boring?(line) # (1) end end def
compare_inventory_files(old_file, new_file) old_inventory = inventory_from(old_file) new_inventory =
inventory_from(new_file) puts "The following files have been added:" puts new_inventory -
old_inventory puts "" puts "The following files have been deleted:" puts old_inventory - new_inventory
end if $0 == __FILE__ check_usage compare_inventory_files(ARGV[0], ARGV[1]) end
```

```
#--- # Excerpted from "Everyday Scripting in Ruby" # We make no guarantees that this code is fit for
any purpose. # Visit http://www.pragmaticprogrammer.com/titles/bmsft for more book information. #---
def check_usage # (1) if ARGV[0].nil? puts "Usage: differences.rb old-inventory new-inventory" exit
end end def boring?(line, boring_words) boring_words.any? do | a_boring_word | contains?(line,
a_boring_word) end end def contains?(line, a_boring_word) line.chomp.split('/').include?
(a_boring_word) end def inventory_from(filename) inventory = File.open(filename) downcased =
inventory.collect do | line | line.downcase end downcased.reject do | line | boring?(line, ['temp',
'recycler']) end end def compare_inventory_files(old_file, new_file) # (2) old_inventory =
inventory_from(old_file) new_inventory = inventory_from(new_file) puts "The following files have
been added:" puts new_inventory - old_inventory puts "" puts "The following files have been deleted:"
puts old_inventory - new_inventory end if $0 == __FILE__ # (3) check_usage compare_inventory_files
(ARGV[0], ARGV[1]) end
```

```
#--- # Excerpted from "Everyday Scripting in Ruby" # We make no guarantees that this code is fit for
any purpose. # Visit http://www.pragmaticprogrammer.com/titles/bmsft for more book information. #---
def check_usage if ARGV[0].nil? puts "Usage: differences.rb old-inventory new-inventory" exit end end
def boring?(line, boring_words) boring_words.any? do | a_boring_word | contains?(line,
a_boring_word) end end def contains?(line, a_boring_word) line.chomp.split('/').include?
(a_boring_word) end def inventory_from(filename, boring_words) #(1) inventory = File.open(filename)
downcased = inventory.collect do | line | line.downcase end downcased.reject do | line | boring?(line,
boring_words) #(2) end end def compare_inventory_files(old_file, new_file) old_inventory =
inventory_from(old_file) new_inventory = inventory_from(new_file) puts "The following files have
been added:" puts new_inventory - old_inventory puts "" puts "The following files have been deleted:"
puts old_inventory - new_inventory end if $0 == __FILE__ # (3) check_usage compare_inventory_files
(ARGV[0], ARGV[1]) end
```

```
#--- # Excerpted from "Everyday Scripting in Ruby" # We make no guarantees that this code is fit for
any purpose. # Visit http://www.pragmaticprogrammer.com/titles/bmsft for more book information. #---
require 'test/unit' require 'exercise-1' class ChurnTests < Test::Unit::TestCase def
test_month_before_is_28_days assert_equal(Time.local(2005, 1, 1), month_before(Time.local(2005, 1,
29))) end def test_svn_date assert_equal('2005-03-04', svn_date(Time.local(2005, 3, 4))) end def
test_header_format assert_equal("Changes between 2005-08-05 and 2006-12-30:", header(svn_date
(month_before(Time.local(2005, 9, 2))), svn_date(Time.local(2006, 12, 30)))) end def
test_normal_subsystem_line_format assert_equal(' audit ***** (45)', subsystem_line("audit", 45))
end def test_asterisks_for_divides_by_five assert_equal('****', asterisks_for(20)) end def
test_asterisks_for_rounds_up_and_down assert_equal('****', asterisks_for(18)) assert_equal('***',
asterisks_for(17)) end def test_subversion_log_can_have_no_changes assert_equal(0,
extract_change_count_from("-----\n")) end def
test_subversion_log_with_changes assert_equal(2, extract_change_count_from
("-----\nr2531 | bem | 2005-07-01 01:11:44 -0500
(Fri, 01 Jul 2005) | 1 line\n\nrevisions up through ch 3 exercises
\n-----\nr2524 | bem | 2005-06-30 18:45:59 -
0500 (Thu, 30 Jun 2005) | 1 line\n\nresults of read-through; including renaming mistyping to snapshots
\n-----\n")) end end
```

```
#---
# Excerpted from "Everyday Scripting in Ruby"
# We make no guarantees that this code is fit for any purpose.
# Visit http://www.pragmaticprogrammer.com/titles/bmsft for more book information.
#---
def month_before(a_time)
  a_time - 28 * 24 * 60 * 60
end

def svn_date(a_time)
  a_time.strftime("%Y-%m-%d")
end

def header(an_svn_date)
  "Changes between #{an_svn_date} and #{svn_date(Time.now)}:"
end

# Note: the previous definition will be overridden by the following.
# That's a side effect of the way code samples are included in the
# book. It's not what I'd normally do when changing a method.

def header(starting_svn_date, ending_svn_date)
  "Changes between #{starting_svn_date} and #{ending_svn_date}:"
end

def subsystem_line(subsystem_name, change_count)
  asterisks = asterisks_for(change_count)
  "#{subsystem_name.rjust(14)} #{asterisks} (#{change_count})"
end

def asterisks_for(an_integer)
  '*' * (an_integer / 5.0).round
end

def change_count_for(name, start_date)
  extract_change_count_from(svn_log(name, start_date))
end
```

```
def extract_change_count_from(log_text)
  lines = log_text.split("\n")
  dashed_lines = lines.find_all do | line |
    line.include?('-----')
  end
  dashed_lines.length - 1
end

def svn_log(subsystem, start_date)
  timespan = "--revision 'HEAD:{#{start_date}}'"
  root = "svn://rubyforge.org//var/svn/churn-demo"

  `svn log #{timespan} #{root}/#{subsystem}`
end

if $0 == __FILE__
  subsystem_names = ['audit', 'fulfillment', 'persistence',
                    'ui', 'util', 'inventory']

  start_date = svn_date(month_before(Time.now))

  puts header(start_date, svn_date(Time.now))

  subsystem_names.each do | name |
    puts subsystem_line(name, change_count_for(name, start_date))
  end
end
```



```
#--- # Excerpted from "Everyday Scripting in Ruby" # We make no guarantees that this code is fit for
any purpose. # Visit http://www.pragmaticprogrammer.com/titles/bmsft for more book information. #---
require 'test/unit' require 'exercise-2' class ChurnTests < Test::Unit::TestCase def
test_month_before_is_28_days assert_equal(Time.local(2005, 1, 1), month_before(Time.local(2005, 1,
29))) end def test_svn_date assert_equal('2005-03-04', svn_date(Time.local(2005, 3, 4))) end def
test_header_format assert_equal("Changes between 2005-08-05 and 2006-12-30:", header(svn_date
(month_before(Time.local(2005, 9, 2))), svn_date(Time.local(2006, 12, 30)))) end def
test_normal_subsystem_line_format assert_equal(' audit ***** (45)', subsystem_line("audit", 45))
end def test_asterisks_for_divides_by_five assert_equal('****', asterisks_for(20)) end def
test_asterisks_for_rounds_up_and_down assert_equal('****', asterisks_for(18)) assert_equal('***',
asterisks_for(17)) end def test_asterisks_for_zero_is_a_dash assert_equal('-', asterisks_for(0)) end def
test_asterisks_for_rounds_up_small_numbers assert_equal('*', asterisks_for(1)) assert_equal('*',
asterisks_for(2)) # Just in case, check nearby boundaries. assert_equal('*', asterisks_for(5)) assert_equal
('*', asterisks_for(7)) assert_equal('**', asterisks_for(8)) end def
test_subversion_log_can_have_no_changes assert_equal(0, extract_change_count_from
("-----\n")) end def
test_subversion_log_with_changes assert_equal(2, extract_change_count_from
("-----\nr2531 | bem | 2005-07-01 01:11:44 -0500
(Fri, 01 Jul 2005) | 1 line\n\nrevisions up through ch 3 exercises
\n-----\nr2524 | bem | 2005-06-30 18:45:59 -
0500 (Thu, 30 Jun 2005) | 1 line\n\nresults of read-through; including renaming mistyping to snapshots
\n-----\n")) end end
```

```
#--- # Excerpted from "Everyday Scripting in Ruby" # We make no guarantees that this code is fit for
any purpose. # Visit http://www.pragmaticprogrammer.com/titles/bmsft for more book information. #---
def month_before(a_time) a_time - 28 * 24 * 60 * 60 end def svn_date(a_time) a_time.strftime("%Y-%
m-%d") end def header(starting_svn_date, ending_svn_date) "Changes between #{starting_svn_date}
and #{ending_svn_date}:" end def subsystem_line(subsystem_name, change_count) asterisks =
asterisks_for(change_count) "#{subsystem_name.rjust(14)} #{asterisks} (#{change_count})" end def
asterisks_for(an_integer) if an_integer == 0 '-' elsif an_integer < 3 '*' else '*' * (an_integer / 5.0).round
end end def change_count_for(name, start_date) extract_change_count_from(svn_log(name, start_date))
end def extract_change_count_from(log_text) lines = log_text.split("\n") dashed_lines = lines.find_all
do | line | line.include?('-----') end dashed_lines.length - 1 end def svn_log(subsystem, start_date)
timespan = "--revision 'HEAD:#{start_date}'" root = "svn://rubyforge.org/var/svn/churn-demo" `svn
log #{timespan} #{root}/#{subsystem}` end if $0 == __FILE__ subsystem_names = ['audit',
'fulfillment', 'persistence', 'ui', 'util', 'inventory'] start_date = svn_date(month_before(Time.now)) puts
header(start_date, svn_date(Time.now)) subsystem_names.each do | name | puts subsystem_line(name,
change_count_for(name, start_date)) end end
```

```
#---
# Excerpted from "Everyday Scripting in Ruby"
# We make no guarantees that this code is fit for any purpose.
# Visit http://www.pragmaticprogrammer.com/titles/bmsft for more book information.
#---

def month_before(a_time)
  a_time - 28 * 24 * 60 * 60
end

def svn_date(a_time)
  a_time.strftime("%Y-%m-%d")
end

def header(starting_svn_date, ending_svn_date)
  "Changes between #{starting_svn_date} and #{ending_svn_date}:"
end

def subsystem_line(subsystem_name, change_count)
  name = subsystem_name.ljust(14)
  change_description = "(#{change_count} changes)".ljust(14)
  asterisks = asterisks_for(change_count)
  "#{name} #{change_description} #{asterisks}"
end

# Note: the previous definition will be overridden by the following.

def subsystem_line(subsystem_name, change_count)
  image = image_for(change_count)
  text = text_for(change_count)

  "#{subsystem_name.ljust(14)} #{text.ljust(14)} #{image}"
end

def text_for(an_integer)
  if an_integer == 0
    '-'
  else

```

```
"(#{an_integer} changes)"
end
end

def image_for(an_integer)
  if an_integer == 0
    '_'
  elsif an_integer < 3
    '*'
  else
    '*' * (an_integer / 5.0).round
  end
end

def change_count_for(name, start_date)
  extract_change_count_from(svn_log(name, start_date))
end

def extract_change_count_from(log_text)
  lines = log_text.split("\n")
  dashed_lines = lines.find_all do | line |
    line.include?('-----')
  end
  dashed_lines.length - 1
end

def svn_log(subsystem, start_date)
  timespan = "--revision 'HEAD:#{start_date}'"
  root = "svn://rubyforge.org//var/svn/churn-demo"

  `svn log #{timespan} #{root}/#{subsystem}`
end

if $0 == __FILE__
  subsystem_names = ['audit', 'fulfillment', 'persistence',
                    'ui', 'util', 'inventory']
  start_date = svn_date(month_before(Time.now))

  puts header(start_date, svn_date(Time.now))
```

```
subsystem_names.each do | name |  
  puts subsystem_line(name, change_count_for(name, start_date))  
end  
end
```

```
#--- # Excerpted from "Everyday Scripting in Ruby" # We make no guarantees that this code is fit for
any purpose. # Visit http://www.pragmaticprogrammer.com/titles/bmsft for more book information. #---
require 'test/unit' require 'exercise-3' class ChurnTests < Test::Unit::TestCase def
test_month_before_is_28_days assert_equal(Time.local(2005, 1, 1), month_before(Time.local(2005, 1,
29))) end def test_svn_date assert_equal('2005-03-04', svn_date(Time.local(2005, 3, 4))) end def
test_header_format assert_equal("Changes between 2005-08-05 and 2006-12-30:", header(svn_date
(month_before(Time.local(2005, 9, 2))), svn_date(Time.local(2006, 12, 30)))) end def
test_normal_subsystem_line_format assert_equal('audit (45 changes) *****', subsystem_line
("audit", 45)) end def test_subsystem_line_has_special_format_for_zero_changes assert_equal('data - -',
subsystem_line("data", 0)) end # Note: the texts claims the previous test was deleted. It would have
been, # except that the book's code snippets are generated from these files, so # I either had to leave it or
create a "part b" solution file. def test_normal_text_for_format assert_equal('(45 changes)', text_for(45))
end def test_special_text_for_no_changes assert_equal('-', text_for(0)) end def
test_image_for_divides_by_five assert_equal('****', image_for(20)) end def
test_image_for_rounds_up_and_down assert_equal('****', image_for(18)) assert_equal('***', image_for
(17)) end def test_image_for_zero_is_a_dash assert_equal('-', image_for(0)) end def
test_image_for_rounds_up_small_numbers assert_equal('*', image_for(1)) assert_equal('*', image_for
(2)) # Just in case, check nearby boundaries. assert_equal('*', image_for(5)) assert_equal('*', image_for
(7)) assert_equal('*', image_for(8)) end def test_subversion_log_can_have_no_changes assert_equal(0,
extract_change_count_from("-----\n")) end def
test_subversion_log_with_changes assert_equal(2, extract_change_count_from
("-----\nr2531 | bem | 2005-07-01 01:11:44 -0500
(Fri, 01 Jul 2005) | 1 line\n\nrevisions up through ch 3 exercises
\n-----\nr2524 | bem | 2005-06-30 18:45:59 -
0500 (Thu, 30 Jun 2005) | 1 line\n\nresults of read-through; including renaming mistyping to snapshots
\n-----\n")) end end
```

```
#--- # Excerpted from "Everyday Scripting in Ruby" # We make no guarantees that this code is fit for
any purpose. # Visit http://www.pragmaticprogrammer.com/titles/bmsft for more book information. #---
def month_before(a_time) a_time - 28 * 24 * 60 * 60 end def svn_date(a_time) a_time.strftime("%Y-%
m-%d") end def header(an_svn_date) "Changes since #{an_svn_date}:" end def subsystem_line
(subsystem_name, change_count) asterisks = asterisks_for(change_count) "#{subsystem_name.rjust
(14)} #{asterisks} (#{change_count})" end def asterisks_for(an_integer) '*' * (an_integer / 5.0).round
end def change_count_for(name, start_date) extract_change_count_from(svn_log(name, start_date)) end
def extract_change_count_from(log_text) lines = log_text.split("\n") dashed_lines = lines.find_all do |
line | line.include?('-----') end dashed_lines.length - 1 end def svn_log(subsystem, start_date) timespan
= "--revision 'HEAD:#{start_date}'" root = "svn://rubyforge.org/var/svn/churn-demo" `svn log #
{timespan} #{root}/#{subsystem}` end def interesting(array) array.find_all do | line | /\*/ =~ line end
end def order_by_descending_change_count(lines) lines.sort do | first, second | first_count =
churn_line_to_int(first) second_count = churn_line_to_int(second) - (first_count <=> second_count) end
end def churn_line_to_int(line) /\d+\.match(line)[1].to_i end if $0 == __FILE__ subsystem_names
= ['audit', 'fulfillment', 'persistence', 'ui', 'util', 'inventory'] start_date = svn_date(month_before(Time.
now)) puts header(start_date) lines = subsystem_names.collect do | name | subsystem_line(name,
change_count_for(name, start_date)) end puts order_by_descending_change_count(interesting(lines)) #
(1) end
```

```
#--- # Excerpted from "Everyday Scripting in Ruby" # We make no guarantees that this code is fit for
any purpose. # Visit http://www.pragmaticprogrammer.com/titles/bmsft for more book information. #---
require 'test/unit' require 'exercise-1' class ChurnTests < Test::Unit::TestCase def
test_month_before_is_28_days assert_equal(Time.local(2005, 1, 1), month_before(Time.local(2005, 1,
29))) end def test_svn_date assert_equal('2005-03-04', svn_date(Time.local(2005, 3, 4))) end def
test_header_format assert_equal("Changes since 2005-08-05:", header(svn_date(month_before(Time.
local(2005, 9, 2))))) end def test_normal_subsystem_line_format assert_equal(' audit ***** (45)',
subsystem_line("audit", 45)) end def test_asterisks_for_divides_by_five assert_equal('****',
asterisks_for(20)) end def test_asterisks_for_rounds_up_and_down assert_equal('****', asterisks_for
(18)) assert_equal('***', asterisks_for(17)) end def test_subversion_log_can_have_no_changes
assert_equal(0, extract_change_count_from("-----
\n")) end def test_subversion_log_with_changes assert_equal(2, extract_change_count_from
("-----\nr2531 | bem | 2005-07-01 01:11:44 -0500
(Fri, 01 Jul 2005) | 1 line\n\nrevisions up through ch 3 exercises
\n-----\nr2524 | bem | 2005-06-30 18:45:59 -
0500 (Thu, 30 Jun 2005) | 1 line\n\nresults of read-through; including renaming mistyping to snapshots
\n-----\n")) end def
test_churn_line_to_int_extracts_parenthesized_change_count assert_equal(19, churn_line_to_int(" ui2
**** (19)")) assert_equal(9, churn_line_to_int(" ui ** (9)")) end def
test_order_by_descending_change_count original = [ "all that really matters is the number in parens -
(1)", " inventory (0)", " ui ** (12)" ] expected = [ " ui ** (12)", "all that really matters is the number in
parens - (1)", " inventory (0)" ] actual = order_by_descending_change_count(original) assert_equal
(expected, actual) end def test_interesting_lines_contain_at_least_one_asterisk boring_line = " inventory
(0)" interesting_line = " ui * (3)" big_line = " util ***** (61)" original = [interesting_line,
boring_line, big_line] expected = [interesting_line, big_line] assert_equal(expected, interesting
(original)) end end
```



```
#--- # Excerpted from "Everyday Scripting in Ruby" # We make no guarantees that this code is fit for
any purpose. # Visit http://www.pragmaticprogrammer.com/titles/bmsft for more book information. #---
require 'test/unit' require 'exercise-2' class ChurnTests < Test::Unit::TestCase def
test_subsystem_line_surrounds_asterisks_with_spaces assert_match(/ \* \(3\)/, subsystem_line("ui", 3))
assert_match(/ \* \* \(10\)/, subsystem_line("ui", 10)) end def
test_subsystem_line_surrounds_even_no_asterisks_with_spaces # ... so interesting can depend on this, if
it needs to. assert_match(/ \(0\)/, subsystem_line("ui", 0)) end def
test_interesting_lines_contain_at_least_one_asterisk boring_line = subsystem_line("inventory", 0)
interesting_line = subsystem_line("ui", 3) big_line = subsystem_line('util', 61) original =
[interesting_line, boring_line, big_line] expected = [interesting_line, big_line] assert_equal(expected,
interesting(original)) end def test_interesting_lines_subsystem_can_have_asterisk_at_end boring_line =
subsystem_line('inventory*', 0) interesting_line = subsystem_line('ui*', 3) original = [interesting_line,
boring_line] expected = [interesting_line] assert_equal(expected, interesting(original)) end end
```

```
#--- # Excerpted from "Everyday Scripting in Ruby" # We make no guarantees that this code is fit for
any purpose. # Visit http://www.pragmaticprogrammer.com/titles/bmsft for more book information. #---
def month_before(a_time) a_time - 28 * 24 * 60 * 60 end def svn_date(a_time) a_time.strftime("%Y-%
m-%d") end def header(an_svn_date) "Changes since #{an_svn_date}:" end def subsystem_line
(subsystem_name, change_count) asterisks = asterisks_for(change_count) "#{subsystem_name.rjust
(14)} #{asterisks} (#{change_count})" end def asterisks_for(an_integer) '*' * (an_integer / 5.0).round
end def change_count_for(name, start_date) extract_change_count_from(svn_log(name, start_date)) end
def extract_change_count_from(log_text) lines = log_text.split("\n") dashed_lines = lines.find_all do |
line | line.include?('-----') end dashed_lines.length - 1 end def svn_log(subsystem, start_date) timespan
= "--revision 'HEAD:#{start_date}'" root = "svn://rubyforge.org/var/svn/churn-demo" `svn log #
{timespan} #{root}/#{subsystem}` end def interesting(array) array.find_all do | line | /\*+ / =~ line end
end def order_by_descending_change_count(lines) lines.sort do | first, second | first_count =
churn_line_to_int(first) second_count = churn_line_to_int(second) - (first_count <=> second_count) end
end def churn_line_to_int(line) /\d+\/.match(line)[1].to_i end if $0 == __FILE__ subsystem_names
= ['audit', 'fulfillment', 'persistence', 'ui', 'util', 'inventory'] start_date = svn_date(month_before(Time.
now)) puts header(start_date) lines = subsystem_names.collect do | name | subsystem_line(name,
change_count_for(name, start_date)) end puts order_by_descending_change_count(interesting(lines))
end
```

```
#--- # Excerpted from "Everyday Scripting in Ruby" # We make no guarantees that this code is fit for
any purpose. # Visit http://www.pragmaticprogrammer.com/titles/bmsft for more book information. #---
def rearrange(string) match = /(\w+), (\w+) (\w+)/.match(string) #(1) last_name = match[1] first_name =
match[2] middle_name = match[3] "#{first_name} #{middle_name[0,1]}. #{last_name}" #(2) end
```

```
#--- # Excerpted from "Everyday Scripting in Ruby" # We make no guarantees that this code is fit for
any purpose. # Visit http://www.pragmaticprogrammer.com/titles/bmsft for more book information. #---
def rearrange(string) has_middle_name = /(\w+), (\w+) (\w+)/.match(string) # (1) no_middle_name = /
(\w+), (\w+)/.match(string) # (2) if has_middle_name # (3) last_name = has_middle_name[1] first_name
= has_middle_name[2] middle_name = has_middle_name[3] "#{first_name} #{middle_name[0,1]}. #
{last_name}" elsif no_middle_name last_name = no_middle_name[1] first_name = no_middle_name[2]
"#{first_name} #{last_name}" end end
```

```
#--- # Excerpted from "Everyday Scripting in Ruby" # We make no guarantees that this code is fit for
any purpose. # Visit http://www.pragmaticprogrammer.com/titles/bmsft for more book information. #---
def month_before(a_time) a_time - 28 * 24 * 60 * 60 end def svn_date(a_time) a_time.strftime("%Y-%
m-%d") end def header(an_svn_date) "Changes since #{an_svn_date}:" end def subsystem_line
(subsystem_name, change_count) asterisks = asterisks_for(change_count) "#{subsystem_name.rjust
(14)} #{asterisks} (#{change_count})" end def asterisks_for(an_integer) '*' * (an_integer / 5.0).round
end def change_count_for(name, start_date) extract_change_count_from(svn_log(name, start_date)) end
def extract_change_count_from(log_text) lines = log_text.split("\n") dashed_lines = lines.find_all do |
line | line.include?('-----') end dashed_lines.length - 1 end def svn_log(subsystem, start_date) timespan
= "--revision 'HEAD:#{start_date}'" root = "svn://rubyforge.org/var/svn/churn-demo" `svn log #
{timespan} #{root}/#{subsystem}` end def interesting(array) array.find_all do | line | /\* (\d+)\$/ =~
line end end def order_by_descending_change_count(lines) lines.sort do | first, second | first_count =
churn_line_to_int(first) second_count = churn_line_to_int(second) - (first_count <=> second_count) end
end def churn_line_to_int(line) /\((\d+)\)/.match(line)[1].to_i end if $0 == __FILE__ subsystem_names
= ['audit', 'fulfillment', 'persistence', 'ui', 'util', 'inventory'] start_date = svn_date(month_before(Time.
now)) puts header(start_date) lines = subsystem_names.collect do | name | subsystem_line(name,
change_count_for(name, start_date)) end puts order_by_descending_change_count(interesting(lines))
end
```

```
#--- # Excerpted from "Everyday Scripting in Ruby" # We make no guarantees that this code is fit for
any purpose. # Visit http://www.pragmaticprogrammer.com/titles/bmsft for more book information. #---
def rearrange(name) match = /(\w+), (\w+)( \w+)?/.match(name) # (1) last_name = match[1] first_name
= match[2] if match[3] separator = "#{match[3][0,2]}." # (2) else separator = ' ' # (3) end "#{first_name}
#{separator}#{last_name}" # (4) end
```

```
#--- # Excerpted from "Everyday Scripting in Ruby" # We make no guarantees that this code is fit for
any purpose. # Visit http://www.pragmaticprogrammer.com/titles/bmsft for more book information. #---
require 'test/unit' require 'exercise-3' class SubversionRepositoryTests < Test::Unit::TestCase def setup
  @repository = SubversionRepository.new('root') end def test_date assert_equal('2005-03-04',
  @repository.date(Time.local(2005, 3, 4))) end def test_subversion_log_can_have_no_changes
  assert_equal(0, @repository.extract_change_count_from
  ("-----\n")) end def
  test_subversion_log_with_changes assert_equal(2, @repository.extract_change_count_from
  ("-----\nr2531 | bem | 2005-07-01 01:11:44 -0500
  (Fri, 01 Jul 2005) | 1 line\n\nrevisions up through ch 3 exercises
  \n-----\nr2524 | bem | 2005-06-30 18:45:59 -
  0500 (Thu, 30 Jun 2005) | 1 line\n\nresults of read-through; including renaming mistyping to snapshots
  \n-----\n")) end end class FormatterTests < Test::
  Unit::TestCase #(1) def setup #(2) @formatter = Formatter.new end def test_header_format assert_equal
  ("Changes since 2005-08-05:", @formatter.header('2005-08-05')) #(3) end def
  test_normal_subsystem_line_format assert_equal(' audit ***** (45)', @formatter.subsystem_line
  ("audit", 45)) end def test_asterisks_for_divides_by_five assert_equal('****', @formatter.asterisks_for
  (20)) end def test_asterisks_for_rounds_up_and_down assert_equal('****', @formatter.asterisks_for
  (18)) assert_equal('***', @formatter.asterisks_for(17)) end def
  test_churn_line_to_int_extracts_parenthesized_change_count assert_equal(19, @formatter.
  churn_line_to_int(" churn2 **** (19)")) assert_equal(9, @formatter.churn_line_to_int(" churn ** (9)"))
  end def test_order_by_descending_change_count original = [ "all that really matters is the number in
  parens - (1)", " inventory (0)", " churn ** (12)" ] expected = [ " churn ** (12)", "all that really matters is
  the number in parens - (1)", " inventory (0)" ] actual = @formatter.order_by_descending_change_count
  (original) assert_equal(expected, actual) end end class ChurnTests < Test::Unit::TestCase def
  test_month_before_is_28_days assert_equal(Time.local(2005, 1, 1), month_before(Time.local(2005, 1,
  29))) end end
```

```
#--- # Excerpted from "Everyday Scripting in Ruby" # We make no guarantees that this code is fit for
any purpose. # Visit http://www.pragmaticprogrammer.com/titles/bmsft for more book information. #---
require 'test/unit' require 'exercise-7' class SubversionRepositoryTests < Test::Unit::TestCase def setup
@repository = SubversionRepository.new('root') end def test_date assert_equal('2005-03-04',
@repository.date(Time.local(2005, 3, 4))) end def test_subversion_log_can_have_no_changes
assert_equal(0, @repository.extract_change_count_from
("-----\n")) end def
test_subversion_log_with_changes assert_equal(2, @repository.extract_change_count_from
("-----\nr2531 | bem | 2005-07-01 01:11:44 -0500
(Fri, 01 Jul 2005) | 1 line\n\nrevisions up through ch 3 exercises
\n-----\nr2524 | bem | 2005-06-30 18:45:59 -
0500 (Thu, 30 Jun 2005) | 1 line\n\nresults of read-through; including renaming mistyping to snapshots
\n-----\n")) end end class
FormatterNormalUseTests < Test::Unit::TestCase def setup formatter = Formatter.new formatter.
use_date('1960-02-19') formatter.add_subsystem_change_count('sub1', 30) formatter.
add_subsystem_change_count('sub2', 39) @output_lines = formatter.output.split("\n") #(1) end def
test_header_comes_before_subsystem_lines #(2) assert_match(/Changes since 1960-02-19/,
@output_lines[0]) end def test_both_lines_are_present_in_descending_change_count_order #(3)
assert_match(/sub2.*39/, @output_lines[1]) assert_match(/sub1.*30/, @output_lines[2]) end def
test_nothing_else_is_present #(4) assert_equal(3, @output_lines.size) end end class
FormatterPrivateMethodTests < Test::Unit::TestCase def setup @formatter = Formatter.new end def
test_header_format assert_equal("Changes since 2005-08-05:", @formatter.header('2005-08-05')) end
def test_normal_subsystem_line_format assert_equal(' audit ***** (45)', @formatter.
subsystem_line("audit", 45)) end def test_asterisks_for_divides_by_five assert_equal('****', @formatter.
asterisks_for(20)) end def test_asterisks_for_rounds_up_and_down assert_equal('****', @formatter.
asterisks_for(18)) assert_equal('***', @formatter.asterisks_for(17)) end def
test_churn_line_to_int_extractes_parenthesized_change_count assert_equal(19, @formatter.
churn_line_to_int(" churn2 **** (19)")) assert_equal(9, @formatter.churn_line_to_int(" churn ** (9)"))
end def test_order_by_descending_change_count original = [ "all that really matters is the number in
parens - (1)", " inventory (0)", " churn ** (12)" ] expected = [ " churn ** (12)", "all that really matters is
the number in parens - (1)", " inventory (0)" ] actual = @formatter.order_by_descending_change_count
(original) assert_equal(expected, actual) end end class ChurnTests < Test::Unit::TestCase def
test_month_before_is_28_days assert_equal(Time.local(2005, 1, 1), month_before(Time.local(2005, 1,
29))) end end
```



```
#--- # Excerpted from "Everyday Scripting in Ruby" # We make no guarantees that this code is fit for
any purpose. # Visit http://www.pragmaticprogrammer.com/titles/bmsft for more book information. #---
class SubversionRepository def initialize(root) @root = root end def date(a_time) a_time.strftime("%Y-
%m-%d") end def change_count_for(name, start_date) extract_change_count_from(log(name,
start_date)) end def extract_change_count_from(log_text) lines = log_text.split("\n") dashed_lines =
lines.find_all do | line | line.include?('-----') end dashed_lines.length - 1 end def log(subsystem,
start_date) timespan = "--revision '#{start_date}'" `svn log #{timespan} #{@root}/#
{subsystem}` end end class Formatter # Public interface def initialize @lines = [] end def use_date
(date_string) @date_string = date_string end def use_subsystem_with_change_count(name, count)
@lines.push(subsystem_line(name, count)) end def output ordered_lines =
order_by_descending_change_count(@lines) output_array = [header(@date_string)] + ordered_lines
output_array.join("\n") end # Helpers def header(a_date) "Changes since #{a_date}:" end def
subsystem_line(subsystem_name, change_count) asterisks = asterisks_for(change_count) "#
{subsystem_name.rjust(14)} #{asterisks} (#{change_count})" end def asterisks_for(an_integer) '*' *
(an_integer / 5.0).round end def order_by_descending_change_count(lines) lines.sort do | first, second |
first_count = churn_line_to_int(first) second_count = churn_line_to_int(second) - (first_count <=>
second_count) end end def churn_line_to_int(line) /\((\d+)\)/.match(line)[1].to_i end end def
month_before(a_time) a_time - 28 * 24 * 60 * 60 end if $0 == __FILE__ subsystem_names = ['audit',
'fulfillment', 'persistence', 'ui', 'util', 'inventory'] root="svn://rubyforge.org/var/svn/churn-demo"
repository = SubversionRepository.new(root) start_date = repository.date(month_before(Time.now))
formatter = Formatter.new formatter.use_date(start_date) subsystem_names.each do | name | formatter.
use_subsystem_with_change_count(name, repository.change_count_for(name, start_date)) end puts
formatter.output end
```

```
#--- # Excerpted from "Everyday Scripting in Ruby" # We make no guarantees that this code is fit for
any purpose. # Visit http://www.pragmaticprogrammer.com/titles/bmsft for more book information. #---
class SubversionRepository def initialize(root) @root = root end def date(a_time) a_time.strftime("%Y-
%m-%d") end def change_count_for(name, start_date) extract_change_count_from(log(name,
start_date)) end def extract_change_count_from(log_text) lines = log_text.split("\n") dashed_lines =
lines.find_all do | line | line.include?('-----') end dashed_lines.length - 1 end def log(subsystem,
start_date) timespan = "--revision '#{start_date}'" `svn log #{timespan} #{@root}/#
{subsystem}` end end class Formatter # Public interface def initialize @lines = [] end def use_date
(date_string) @date_string = date_string end def use_subsystem_with_change_count(name, count)
@lines.push(subsystem_line(name, count)) end def output ([header] +
lines_ordered_by_descending_change_count).join("\n") end # Helpers def header "Changes since #
{ @date_string}:" end def subsystem_line(subsystem_name, change_count) asterisks = asterisks_for
(change_count) "#{subsystem_name.rjust(14)} #{asterisks} (#{change_count})" end def asterisks_for
(an_integer) '*' * (an_integer / 5.0).round end def lines_ordered_by_descending_change_count @lines.
sort do | first, second | first_count = churn_line_to_int(first) second_count = churn_line_to_int(second) -
(first_count <=> second_count) end end def churn_line_to_int(line) /\((\d+)\)/.match(line)[1].to_i end
end def month_before(a_time) a_time - 28 * 24 * 60 * 60 end if $0 == __FILE__ subsystem_names =
['audit', 'fulfillment', 'persistence', 'ui', 'util', 'inventory'] root="svn://rubyforge.org//var/svn/churn-demo"
repository = SubversionRepository.new(root) start_date = repository.date(month_before(Time.now))
formatter = Formatter.new formatter.use_date(start_date) subsystem_names.each do | name | formatter.
use_subsystem_with_change_count(name, repository.change_count_for(name, start_date)) end puts
formatter.output end
```

```
#--- # Excerpted from "Everyday Scripting in Ruby" # We make no guarantees that this code is fit for
any purpose. # Visit http://www.pragmaticprogrammer.com/titles/bmsft for more book information. #---
require 'test/unit' require 'exercise-8' class SubversionRepositoryTests < Test::Unit::TestCase def setup
  @repository = SubversionRepository.new('root') end def test_date assert_equal('2005-03-04',
  @repository.date(Time.local(2005, 3, 4))) end def test_subversion_log_can_have_no_changes
  assert_equal(0, @repository.extract_change_count_from
  ("-----\n")) end def
  test_subversion_log_with_changes assert_equal(2, @repository.extract_change_count_from
  ("-----\nr2531 | bem | 2005-07-01 01:11:44 -0500
  (Fri, 01 Jul 2005) | 1 line\n\nrevisions up through ch 3 exercises
  \n-----\nr2524 | bem | 2005-06-30 18:45:59 -
  0500 (Thu, 30 Jun 2005) | 1 line\n\nresults of read-through; including renaming mistyping to snapshots
  \n-----\n")) end end class
  FormatterNormalUseTests < Test::Unit::TestCase def setup formatter = Formatter.new formatter.
  report_range(Time.local(2005, 1, 1), Time.local(2005, 2, 1)) formatter.
  use_subsystem_with_change_count('sub1', 30) formatter.use_subsystem_with_change_count('sub2', 39)
  @output_lines = formatter.output.split("\n") end def test_header_comes_before_subsystem_lines
  assert_match(/Changes between/, @output_lines[0]) end def
  test_both_lines_are_present_in_descending_change_count_order assert_match(/sub2.*39/,
  @output_lines[1]) assert_match(/sub1.*30/, @output_lines[2]) end def test_nothing_else_is_present
  assert_equal(3, @output_lines.size) end end class FormatterPrivateMethodTests < Test::Unit::TestCase
  def setup @formatter = Formatter.new end def test_header_format @formatter.report_range(Time.local
  (2001, 3, 3), Time.local(2002, 2, 2)) assert_equal("Changes between March 3, 2001, and February 2,
  2002:", @formatter.header) end def test_normal_subsystem_line_format assert_equal(' audit *****
  (45)', @formatter.subsystem_line("audit", 45)) end def test_asterisks_for_divides_by_five assert_equal
  ('****', @formatter.asterisks_for(20)) end def test_asterisks_for_rounds_up_and_down assert_equal
  ('****', @formatter.asterisks_for(18)) assert_equal('***', @formatter.asterisks_for(17)) end def
  test_churn_line_to_intextracts_parenthesized_change_count assert_equal(19, @formatter.
  churn_line_to_int(" churn2 **** (19)")) assert_equal(9, @formatter.churn_line_to_int(" churn ** (9)"))
  end def test_lines_are_ordered_by_descending_change_count @formatter.
  use_subsystem_with_change_count("a count matters for sorting, not a name", 1) @formatter.
  use_subsystem_with_change_count("inventory", 0) @formatter.use_subsystem_with_change_count
  ("churn", 12) expected = [ " churn ** (12)", "all that really matters is the number in parens - (1)", "
  inventory (0)" ] actual = @formatter.lines_ordered_by_descending_change_count assert_match(/churn/,
  actual[0]) assert_match(/a count matters/, actual[1]) assert_match(/inventory/, actual[2]) end end class
  ChurnTests < Test::Unit::TestCase def test_month_before_is_28_days assert_equal(Time.local(2005, 1,
  1), month_before(Time.local(2005, 1, 29))) end end
```

```
#--- # Excerpted from "Everyday Scripting in Ruby" # We make no guarantees that this code is fit for
any purpose. # Visit http://www.pragmaticprogrammer.com/titles/bmsft for more book information. #---
require "subversion-repository" require "formatter" def month_before(a_time) a_time - 28 * 24 * 60 *
60 end if $0 == __FILE__ subsystem_names = ['audit', 'fulfillment', 'persistence', 'ui', 'util', 'inventory']
root="svn://rubyforge.org/var/svn/churn-demo" repository = SubversionRepository.new(root)
last_month = month_before(Time.now) formatter = Formatter.new formatter.report_range(last_month,
Time.now) subsystem_names.each do | name | formatter.use_subsystem_with_change_count( name,
repository.change_count_for(name, last_month)) end puts formatter.output end
```

```
#--- # Excerpted from "Everyday Scripting in Ruby" # We make no guarantees that this code is fit for
any purpose. # Visit http://www.pragmaticprogrammer.com/titles/bmsft for more book information. #---
require 'churn-tests.rb' require 'formatter-tests.rb' require 'subversion-repository-tests.rb'
```

```
#--- # Excerpted from "Everyday Scripting in Ruby" # We make no guarantees that this code is fit for
any purpose. # Visit http://www.pragmaticprogrammer.com/titles/bmsft for more book information. #---
Dir.glob("*tests.rb").each do | testfile | puts testfile require testfile end
```

```
#--- # Excerpted from "Everyday Scripting in Ruby" # We make no guarantees that this code is fit for
any purpose. # Visit http://www.pragmaticprogrammer.com/titles/bmsft for more book information. #---
require 'test/unit' require 'exercise-1.v5' class CounterTests < Test::Unit::TestCase def
test_Counter_counts assert_equal(0, Counter.count) Counter.counted_new assert_equal(1, Counter.
count) Counter.counted_new assert_equal(2, Counter.count) end end
```

```
#--- # Excerpted from "Everyday Scripting in Ruby" # We make no guarantees that this code is fit for
any purpose. # Visit http://www.pragmaticprogrammer.com/titles/bmsft for more book information. #---
class Counter
  def self.counted_new
    @count = 0
    if @count.nil?
      @count += 1
    end
  end
  def self.count
    @count = 0
    if @count.nil?
      # (1)
    end
  end
end
```



```
#--- # Excerpted from "Everyday Scripting in Ruby" # We make no guarantees that this code is fit for
any purpose. # Visit http://www.pragmaticprogrammer.com/titles/bmsft for more book information. #---
class Counter
  def self.maybe_initialize # (1) @count = 0 if @count.nil? end
  def self.counted_new
  maybe_initialize # (2) @count += 1 new end
  def self.count
  maybe_initialize # (3) @count end
end
```

```
#--- # Excerpted from "Everyday Scripting in Ruby" # We make no guarantees that this code is fit for  
any purpose. # Visit http://www.pragmaticprogrammer.com/titles/bmsft for more book information. #---  
class Counter def self.counted_new @count = self.count + 1 #(1) new end def self.count @count = 0 if  
@count.nil? @count end end
```

```
#--- # Excerpted from "Everyday Scripting in Ruby" # We make no guarantees that this code is fit for
any purpose. # Visit http://www.pragmaticprogrammer.com/titles/bmsft for more book information. #---
class Counter def self.counted_new self.count += 1 new end def self.count @count = 0 if @count.nil?
@count end def self.count=(value) @count = value end end
```

```
#--- # Excerpted from "Everyday Scripting in Ruby" # We make no guarantees that this code is fit for  
any purpose. # Visit http://www.pragmaticprogrammer.com/titles/bmsft for more book information. #---  
class Counter def self.counted_new self.count += 1 new end def self.count @count = 0 if @count.nil?  
@count end def self.count=(value) @count = value end def self.reset self.count = nil end end
```

```
#--- # Excerpted from "Everyday Scripting in Ruby" # We make no guarantees that this code is fit for
any purpose. # Visit http://www.pragmaticprogrammer.com/titles/bmsft for more book information. #---
require 'test/unit' require 'exercise-2' class CounterTests < Test::Unit::TestCase def setup Counter.reset
end def test_Counter_counts assert_equal(0, Counter.count) Counter.counted_new assert_equal(1,
Counter.count) Counter.counted_new assert_equal(2, Counter.count) end end
```

```
#--- # Excerpted from "Everyday Scripting in Ruby" # We make no guarantees that this code is fit for
any purpose. # Visit http://www.pragmaticprogrammer.com/titles/bmsft for more book information. #---
class Counter # Class methods def self.counted_new self.count += 1 new end def self.count @count = 0
if @count.nil? @count end def self.count=(value) @count = value end def self.reset self.count = nil end
# Instance methods attr_accessor :birth_order end
```

```
#--- # Excerpted from "Everyday Scripting in Ruby" # We make no guarantees that this code is fit for
any purpose. # Visit http://www.pragmaticprogrammer.com/titles/bmsft for more book information. #---
require 'test/unit' require 'exercise-4' class CounterTests < Test::Unit::TestCase def setup Counter.reset
end def test_Counter_counts assert_equal(0, Counter.count) Counter.counted_new assert_equal(1,
Counter.count) Counter.counted_new assert_equal(2, Counter.count) end def test_birth_order
assert_equal(1, Counter.counted_new.birth_order) # Test another one, just for luck. assert_equal(2,
Counter.counted_new.birth_order) end end
```

```
#--- # Excerpted from "Everyday Scripting in Ruby" # We make no guarantees that this code is fit for
any purpose. # Visit http://www.pragmaticprogrammer.com/titles/bmsft for more book information. #---
class Counter # Class methods
  def self.counted_new
    self.count += 1
    new_counter = new
    new_counter.birth_order = self.count
  end
  def self.count
    @count = 0 if @count.nil?
    @count
  end
  def self.count=(value)
    @count = value
  end
  def self.reset
    self.count = nil
  end
# Instance methods
  attr_accessor :birth_order
end
```



```
#--- # Excerpted from "Everyday Scripting in Ruby" # We make no guarantees that this code is fit for
any purpose. # Visit http://www.pragmaticprogrammer.com/titles/bmsft for more book information. #---
def check_args(args) raise "Exactly one argument is required." unless args.length == 1 only_arg = args
[0] raise "'#{only_arg}' is not an integer." unless /\d+$/ =~ only_arg end if $0 == __FILE__ begin
check_args(ARGV) puts ARGV[0].to_i rescue Exception => ex puts ex.message end end
```

```
#--- # Excerpted from "Everyday Scripting in Ruby" # We make no guarantees that this code is fit for  
any purpose. # Visit http://www.pragmaticprogrammer.com/titles/bmsft for more book information. #---  
class Array  
  def my_each  
    index = 0  
    while index < self.length  
      yield self[index]  
      index += 1  
    end  
  end  
end
```