www.it-ebooks.info

Appendix A
Possible Solutions

Since the first edition of this book, the single question I have been asked the most is “Where are the answers to the exercises?”

My reluctance centered around the first occurrence of the word *the* in that question.

The answers? There’s more than one right answer, of course. Many, many more. These aren’t math problems. Even the first exercises, which are sort of like math problems, have many possible solutions. If, instead of writing a program about orange trees or the minutes in a decade, you were asked to write a poem about them, it would be silly (if not downright harmful) to include “the answers.”

That was my reasoning, anyway. Kind of stupid, in retrospect—while these aren’t math problems, neither are they poems.

Still, I’m really attached to the idea that there’s no one right answer here, so I did a few things to drive that point home. First, notice the title to this appendix: *possible* solutions, not *the* solutions.

Then I went through and did each exercise twice. Yes, seriously. The first time is to show just one possible way that you *could* have done it, given what you have learned up to that point in the book. The second time is to show you how I would do it, using whatever techniques tickled my fancy. Some of these techniques are not covered in this book, so it’s OK if you don’t understand exactly what’s going on. These programs tend to be more complex but also shorter (sometimes *much* shorter) and sometimes more correct or robust. Often cuter. (I like cute code.)

No more complaining about how hard the exercises were, OK? At least you had to do them only once.

Ignore them or study them as you prefer.

www.it-ebooks.info

A.1 Exercises from Chapter 2

Hours in a Year

(from on page 12)

How you could do it:

```
puts 24*365
```

```
8760
```

How I would do it:

```
#  depends on if it's a leap year
puts 24*365
puts "(or #{24*366} on occasion)"
```

```
8760
(or 8784 on occasion)
```

Minutes in a Decade

(from on page 12)

How you could do it:

```
puts 60*24*(365*10 + 2)
```

```
5258880
```

How I would do it:

```
#  depends on how many leap years in that decade
puts "#{60*24*(365*10 + 2)} or #{60*24*(365*10 + 3)}"
```

```
5258880 or 5260320
```

Your Age in Seconds

(from on page 13)

How you could do it:

www.it-ebooks.info

```
puts 60*60*24*(365*32 + 9)
```

```
1009929600
```

How I would do it:

```
puts(Time.new - Time.gm(1976, 8, 3, 13, 31))
```

```
1040353874.92412
```

Our Dear Author's Age

(from on page 13)

How you could do it:

```
puts 1025000000/(60*60*24*365)
```

```
32
```

And that's pretty much how I would do it, too. :)

A.2 Exercises from Chapter 5

Full Name Greeting

(from on page 28)

How you could do it:

```
puts 'What is your first name?'
f_name = gets.chomp
puts 'What is your middle name?'
m_name = gets.chomp
puts 'What is your last name?'
l_name = gets.chomp

full_name = f_name + ' ' + m_name + ' ' + l_name

puts 'Hello, ' + full_name + '!'
```

```
What is your first name?
Sam
What is your middle name?
I
What is your last name?
Am
Hello, Sam I Am!
```

How I would do it:

```
puts "What's your first name?"
f_name = gets.chomp
puts "What's your middle name?"
m_name = gets.chomp
puts "What's your last name?"
l_name = gets.chomp

puts "Chris Pine is cooler than you, #{f_name} #{m_name} #{l_name}."
```

```
What's your first name?
Marvin
What's your middle name?
K.
What's your last name?
Mooney
Chris Pine is cooler than you, Marvin K. Mooney.
```

Bigger, Better Favorite Number

(from on page 28)

How you could do it:

```
puts 'Hey!  What\'s your favorite number?'
fav_num = gets.chomp.to_i
better_num = fav_num + 1
puts 'That\'s ok, I guess, but isn\'t '+better_num.to_s+' just a bit better?'
```

```
Hey!  What's your favorite number?
5
That's ok, I guess, but isn't 6 just a bit better?
```

How I would do it:

```
puts "Hey!  What's your favorite number?"
fav_num = gets.chomp.to_i
puts "That's ok, I guess, but isn't #{fav_num + 1} just a bit better?"
```

```
Hey!  What's your favorite number?
5
That's ok, I guess, but isn't 6 just a bit better?
```

A.3 Exercises from Chapter 6

Angry Boss

(from on page 36)

How you could do it:

```
puts 'CAN\'T YOU SEE I\'M BUSY?!  MAKE IT FAST, JOHNSON!'
request = gets.chomp
puts 'WHADDAYA MEAN "' + request.upcase + '"?!? YOU\'RE FIRED!!'
```

```
CAN'T YOU SEE I'M BUSY?!  MAKE IT FAST, JOHNSON!
I want a raise
WHADDAYA MEAN "I WANT A RAISE"?!? YOU'RE FIRED!!
```

How I would do it:

```
names = ['johnson', 'smith', 'weinberg', 'filmore']
puts "CAN'T YOU SEE I'M BUSY?!  MAKE IT FAST, #{names[rand(4)].upcase}!"
request = gets.chomp
puts "WHADDAYA MEAN \"#{request.upcase}\"?!? YOU'RE FIRED!!"
```

```
CAN'T YOU SEE I'M BUSY?!  MAKE IT FAST, WEINBERG!
I quit
WHADDAYA MEAN "I QUIT"?!? YOU'RE FIRED!!
```

www.it-ebooks.info

Table of Contents

(from on page 36)

How you could do it:

```
title  = 'Table of Contents'.center(50)
chap_1 = 'Chapter 1:  Getting Started'.ljust(30) + 'page  1'.rjust(20)
chap_2 = 'Chapter 2:  Numbers'.ljust(30)         + 'page  9'.rjust(20)
chap_3 = 'Chapter 3:  Letters'.ljust(30)         + 'page 13'.rjust(20)

puts title
puts
puts chap_1
puts chap_2
puts chap_3
```

```
                Table of Contents

Chapter 1:  Getting Started                page  1
Chapter 2:  Numbers                        page  9
Chapter 3:  Letters                        page 13
```

And how would I do it? Well, that was a different exercise (at the end of Chapter 8).

A.4 Exercises from Chapter 7

“99 Bottles of Beer on the Wall”

(from on page 57)

How you could do it:

```
num_at_start = 5  #  change to 99 if you want

num_now = num_at_start

while num_now > 2
  puts num_now.to_s + ' bottles of beer on the wall, ' +
       num_now.to_s + ' bottles of beer!'
  num_now = num_now - 1
  puts 'Take one down, pass it around, ' +
       num_now.to_s + ' bottles of beer on the wall!'
end
```

www.it-ebooks.info

```
puts "2 bottles of beer on the wall, 2 bottles of beer!"
puts "Take one down, pass it around, 1 bottle of beer on the wall!"
puts "1 bottle of beer on the wall, 1 bottle of beer!"
puts "Take one down, pass it around, no more bottles of beer on the wall!"
```

```
5 bottles of beer on the wall, 5 bottles of beer!
Take one down, pass it around, 4 bottles of beer on the wall!
4 bottles of beer on the wall, 4 bottles of beer!
Take one down, pass it around, 3 bottles of beer on the wall!
3 bottles of beer on the wall, 3 bottles of beer!
Take one down, pass it around, 2 bottles of beer on the wall!
2 bottles of beer on the wall, 2 bottles of beer!
Take one down, pass it around, 1 bottle of beer on the wall!
1 bottle of beer on the wall, 1 bottle of beer!
Take one down, pass it around, no more bottles of beer on the wall!
```

How I would do it:

```
num_at_start = 5  #  change to 99 if you want

num_bot = proc { |n| "#{n} bottle#{n == 1 ? '' : 's'}" }

num_at_start.downto(2) do |num|
  puts "#{num_bot[num]} of beer on the wall, #{num_bot[num]} of beer!"
  puts "Take one down, pass it around, #{num_bot[num-1]} of beer on the wall!"
end

puts "#{num_bot[1]} of beer on the wall, #{num_bot[1]} of beer!"
puts "Take one down, pass it around, no more bottles of beer on the wall!"
```

```
5 bottles of beer on the wall, 5 bottles of beer!
Take one down, pass it around, 4 bottles of beer on the wall!
4 bottles of beer on the wall, 4 bottles of beer!
Take one down, pass it around, 3 bottles of beer on the wall!
3 bottles of beer on the wall, 3 bottles of beer!
Take one down, pass it around, 2 bottles of beer on the wall!
2 bottles of beer on the wall, 2 bottles of beer!
Take one down, pass it around, 1 bottle of beer on the wall!
1 bottle of beer on the wall, 1 bottle of beer!
Take one down, pass it around, no more bottles of beer on the wall!
```

www.it-ebooks.info

Deaf Grandma

(from on page 57)

How you could do it:

```
puts 'HEY THERE, SONNY!  GIVE GRANDMA A KISS!'

while true
  said = gets.chomp
  if said == "BYE"
    puts 'BYE SWEETIE!'
    break
  end

  if said != said.upcase
    puts 'HUH?!  SPEAK UP, SONNY!'
  else
    random_year = 1930 + rand(21)
    puts 'NO, NOT SINCE ' + random_year.to_s + '!'
  end
end
```

```
HEY THERE, SONNY!  GIVE GRANDMA A KISS!
hi, grandma
HUH?!  SPEAK UP, SONNY!
HI, GRANDMA!
NO, NOT SINCE 1946!
HOW YOU DOING?
NO, NOT SINCE 1934!
I SAID, HOW YOU DOING?
NO, NOT SINCE 1937!
OK
NO, NOT SINCE 1946!
BYE
BYE SWEETIE!
```

How I would do it:

```
puts 'HEY THERE, SONNY!  GIVE GRANDMA A KISS!'

while true
  said = gets.chomp
  break if said == "BYE"
  response = if said != said.upcase
    'HUH?!  SPEAK UP, SONNY!'
  else
    "NO, NOT SINCE #{1930 + rand(21)}!"
  end

  puts response
end

puts 'BYE SWEETIE!'
```

```
HEY THERE, SONNY!  GIVE GRANDMA A KISS!
hi, grandma
HUH?!  SPEAK UP, SONNY!
HI, GRANDMA!
NO, NOT SINCE 1934!
HOW YOU DOING?
NO, NOT SINCE 1942!
I SAID, HOW YOU DOING?
NO, NOT SINCE 1941!
OK
NO, NOT SINCE 1938!
BYE
BYE SWEETIE!
```

Deaf Grandma Extended

(from on page 57)

How you could do it:

```
puts 'HEY THERE, PEACHES!  GIVE GRANDMA A KISS!'
bye_count = 0

while true
  said = gets.chomp
  if said == 'BYE'
    bye_count = bye_count + 1
  else
    bye_count = 0
  end
```

www.it-ebooks.info

```
  if bye_count >= 3
    puts 'BYE-BYE CUPCAKE!'
    break
  end

  if said != said.upcase
    puts 'HUH?!  SPEAK UP, SONNY!'
  else
    random_year = 1930 + rand(21)
    puts 'NO, NOT SINCE ' + random_year.to_s + '!'
  end
end
```

```
HEY THERE, PEACHES!  GIVE GRANDMA A KISS!
HI, GRANDMA!
NO, NOT SINCE 1937!
BYE
NO, NOT SINCE 1937!
BYE
NO, NOT SINCE 1947!
ADIOS, MUCHACHA!
NO, NOT SINCE 1938!
BYE
NO, NOT SINCE 1935!
BYE
NO, NOT SINCE 1945!
BYE
BYE-BYE CUPCAKE!
```

How I would do it:

```
puts 'HEY THERE, PEACHES!  GIVE GRANDMA A KISS!'
bye_count = 0

while true
  said = gets.chomp
  if said == 'BYE'
    bye_count += 1
  else
    bye_count  = 0
  end
```

www.it-ebooks.info

```
  break if bye_count >= 3

  response = if said != said.upcase
    'HUH?!  SPEAK UP, SONNY!'
  else
    "NO, NOT SINCE #{1930 + rand(21)}!"
  end

  puts response
end

puts 'BYE-BYE CUPCAKE!'
```

```
HEY THERE, PEACHES!  GIVE GRANDMA A KISS!
HI, GRANDMA!
NO, NOT SINCE 1932!
BYE
NO, NOT SINCE 1935!
BYE
NO, NOT SINCE 1931!
ADIOS, MUCHACHA!
NO, NOT SINCE 1933!
BYE
NO, NOT SINCE 1930!
BYE
NO, NOT SINCE 1942!
BYE
BYE-BYE CUPCAKE!
```

Leap Years

(from on page 58)

How you could do it:

```
puts 'Pick a starting year (like 1973 or something):'
starting = gets.chomp.to_i

puts 'Now pick an ending year:'
ending   = gets.chomp.to_i

puts 'Check it out... these years are leap years:'
```

www.it-ebooks.info

```ruby
year = starting

while year <= ending
  if year%4 == 0
    if year%100 != 0 || year%400 == 0
      puts year
    end
  end

  year = year + 1
end
```

```
Pick a starting year (like 1973 or something):
1973
Now pick an ending year:
1977
Check it out... these years are leap years:
1976
```

How I would do it:

```ruby
puts 'Pick a starting year (like 1973 or something):'
starting = gets.chomp.to_i

puts 'Now pick an ending year:'
ending   = gets.chomp.to_i

puts 'Check it out... these years are leap years:'

(starting..ending).each do |year|
  next if year%4   != 0
  next if year%100 == 0 && year%400 != 0
  puts year
end
```

www.it-ebooks.info

```
Pick a starting year (like 1973 or something):
1973
Now pick an ending year:
1977
Check it out... these years are leap years:
1976
```

A.5 Exercises from Chapter 8

Building and Sorting an Array

(from on page 65)

How you could do it:

```
puts 'Give me some words, and I will sort them:'
words = []

while true
  word = gets.chomp
  if word == ''
    break
  end

  words.push word
end

puts 'Sweet!  Here they are, sorted:'
puts words.sort
```

```
Give me some words, and I will sort them:
banana
apple
cherry

Sweet!  Here they are, sorted:
apple
banana
cherry
```

www.it-ebooks.info

How I would do it:

```
puts 'Give me some words, and I will sort them:'
words = []

while true
  word = gets.chomp
  break if word.empty?

  words << word
end

puts 'Sweet!  Here they are, sorted:'
puts words.sort
```

```
Give me some words, and I will sort them:
banana
apple
cherry

Sweet!  Here they are, sorted:
apple
banana
cherry
```

Table of Contents, Revisited

(from on page 66)

How you could do it:

```
title    = 'Table of Contents'

chapters = [['Getting Started',  1],
            ['Numbers',          9],
            ['Letters',         13]]

puts title.center(50)
puts

chap_num = 1

chapters.each do |chap|
  name = chap[0]
  page = chap[1]

  beginning = 'Chapter ' + chap_num.to_s + ':  ' + name
  ending    = 'page ' + page.to_s

  puts beginning.ljust(30) + ending.rjust(20)
  chap_num = chap_num + 1
end
```

```
                 Table of Contents

Chapter 1:  Getting Started                 page 1
Chapter 2:  Numbers                         page 9
Chapter 3:  Letters                        page 13
```

How I would do it:

```
title    = 'Table of Contents'

chapters = [['Getting Started',  1],
            ['Numbers',          9],
            ['Letters',         13]]

puts title.center(50)
puts
chapters.each_with_index do |chap, idx|
  name, page = chap
  chap_num   = idx + 1

  beginning = "Chapter #{chap_num}:  #{name}"
  ending    = "page #{page}"

  puts beginning.ljust(30) + ending.rjust(20)
end
```

www.it-ebooks.info

```
                   Table of Contents

Chapter 1:  Getting Started                  page  1
Chapter 2:  Numbers                          page  9
Chapter 3:  Letters                          page 13
```

A.6 Exercises from Chapter 9

Improved ask Method

(from on page 80)

How you could do it:

```
def ask question
  while true
    puts question
    reply = gets.chomp.downcase

    if reply == 'yes'
      return true
    end
    if reply == 'no'
      return false
    end

    #  If we got this far, then we're going to loop
    #  and ask the question again.
    puts 'Please answer "yes" or "no".'
  end

  answer  #  This is what we return (true or false).
end

likes_it = ask 'Do you like eating tacos?'

puts likes_it
```

```
Do you like eating tacos?
yes
true
```

www.it-ebooks.info

How I would do it:

```
def ask question
  while true
    puts question
    reply = gets.chomp.downcase

    return true  if reply == 'yes'
    return false if reply == 'no'

    puts 'Please answer "yes" or "no".'
  end

  answer  #  This is what we return (true or false).
end

puts(ask('Do you like eating tacos?'))
```

```
Do you like eating tacos?
yes
true
```

Old-School Roman Numerals

(from on page 81)

How you could do it:

```
def old_roman_numeral num
  roman = ''

  roman = roman + 'M' * (num        / 1000)
  roman = roman + 'D' * (num % 1000 /  500)
  roman = roman + 'C' * (num %  500 /  100)
  roman = roman + 'L' * (num %  100 /   50)
  roman = roman + 'X' * (num %   50 /   10)
  roman = roman + 'V' * (num %   10 /    5)
  roman = roman + 'I' * (num %    5 /    1)
  roman
end

puts(old_roman_numeral(1999))
```

www.it-ebooks.info

```
MDCCCCLXXXXVIIII
```

How I would do it:

```
def old_roman_numeral num
  raise 'Must use positive integer' if num <= 0

  roman = ''

  roman << 'M' * (num        / 1000)
  roman << 'D' * (num % 1000 /  500)
  roman << 'C' * (num %  500 /  100)
  roman << 'L' * (num %  100 /   50)
  roman << 'X' * (num %   50 /   10)
  roman << 'V' * (num %   10 /    5)
  roman << 'I' * (num %    5 /    1)

  roman
end

puts(old_roman_numeral(1999))
```

```
MDCCCCLXXXXVIIII
```

“Modern” Roman Numerals

(from on page 81)

How you could do it:

```
def roman_numeral num
  thous = (num        / 1000)
  hunds = (num % 1000 /  100)
  tens  = (num %  100 /   10)
  ones  = (num %   10       )

  roman = 'M' * thous

  if hunds == 9
    roman = roman + 'CM'
  elsif hunds == 4
    roman = roman + 'CD'
```

www.it-ebooks.info

```
  else
    roman = roman + 'D' * (num % 1000 / 500)
    roman = roman + 'C' * (num %  500 / 100)
  end

  if tens == 9
    roman = roman + 'XC'
  elsif tens == 4
    roman = roman + 'XL'
  else
    roman = roman + 'L' * (num %  100 /  50)
    roman = roman + 'X' * (num %   50 /  10)
  end

  if ones == 9
    roman = roman + 'IX'
  elsif ones == 4
    roman = roman + 'IV'
  else
    roman = roman + 'V' * (num %   10 /   5)
    roman = roman + 'I' * (num %    5 /   1)
  end

  roman
end

puts(roman_numeral(1999))
```

```
MCMXCIX
```

How I would do it:

```
def roman_numeral num
  raise 'Must use positive integer' if num <= 0

  digit_vals = [['I',    5,    1],
               ['V',   10,    5],
               ['X',   50,   10],
               ['L',  100,   50],
               ['C',  500,  100],
```

www.it-ebooks.info

```
                ['D', 1000,  500],
                ['M',  nil, 1000]]

  roman = ''
  remaining = nil

  # Build string "roman" in reverse.
  build_rev = proc do |l,m,n|
    num_l = m ? (num % m / n) : (num / n)
    full  = m && (num_l == (m/n - 1))

    if full && (num_l>1 || remaining)
      # must carry
      remaining ||= l # carry l if not already carrying
    else
      if remaining
        roman << l + remaining
        remaining = nil
      end

      roman << l * num_l
    end
  end

  digit_vals.each {|l,m,n| build_rev[l,m,n]}

  roman.reverse
end

puts(roman_numeral(1999))
```

```
MIM
```

A.7 Exercises from Chapter 10

Rite of Passage: Sorting

(from on page 88)

How you could do it:

www.it-ebooks.info

```
def sort arr
  rec_sort arr, []
end

def rec_sort unsorted, sorted
  if unsorted.length <= 0
    return sorted
  end

  #  So if we got here, then it means we still
  #  have work to do.
  smallest       = unsorted.pop
  still_unsorted = []

  unsorted.each do |tested_object|
    if tested_object < smallest
      still_unsorted.push smallest
      smallest = tested_object
    else
      still_unsorted.push tested_object
    end
  end

  #  Now "smallest" really does point to the
  #  smallest element that "unsorted" contained,
  #  and all the rest of it is in "still_unsorted".
  sorted.push smallest

  rec_sort still_unsorted, sorted
end

puts(sort(['can','feel','singing','like','a','can']))
```

```
a
can
can
feel
like
singing
```

www.it-ebooks.info

How I would do it (well, aside from just using the built-in sort method):

```
#  The well-known quicksort algorithm.
def sort arr
  return arr if arr.length <= 1

  middle = arr.pop
  less   = arr.select{|x| x <  middle}
  more   = arr.select{|x| x >= middle}

  sort(less) + [middle] + sort(more)
end

p(sort(['can','feel','singing','like','a','can']))
```

```
["a", "can", "can", "feel", "like", "singing"]
```

Shuffle

(from on page 90)

How you could do it:

```
def shuffle arr
  shuf = []

  while arr.length > 0
    # Randomly pick one element of the array.
    rand_index = rand(arr.length)

    # Now go through each item in the array,
    # putting them all into new_arr except for the
    # randomly chosen one, which goes into shuf.
    curr_index = 0
    new_arr = []

    arr.each do |item|
      if curr_index == rand_index
        shuf.push item
      else
        new_arr.push item
      end
```

www.it-ebooks.info

```
      curr_index = curr_index + 1
    end

    # Replace the original array with the new,
    # smaller array.
    arr = new_arr
  end

  shuf
end

puts(shuffle([1,2,3,4,5,6,7,8,9]))
```

```
1
5
4
8
7
9
6
2
3
```

How I would do it:

```
def shuffle arr
  arr.sort_by(&:rand)
end

p(shuffle([1,2,3,4,5,6,7,8,9]))
```

```
#<TypeError: wrong argument type Symbol (expected Proc)>
```

www.it-ebooks.info

Dictionary Sort

(from on page 90)

How you could do it:

```
def dictionary_sort arr
  rec_dict_sort arr, []
end

def rec_dict_sort unsorted, sorted
  if unsorted.length <= 0
    return sorted
  end

  #  So if we got here, then it means we still
  #  have work to do.
  smallest       = unsorted.pop
  still_unsorted = []

  unsorted.each do |tested_object|
    if tested_object.downcase < smallest.downcase
      still_unsorted.push smallest
      smallest = tested_object
    else
      still_unsorted.push tested_object
    end
  end

  #  Now "smallest" really does point to the
  #  smallest element that "unsorted" contained,
  #  and all the rest of it is in "still_unsorted".
  sorted.push smallest

  rec_dict_sort still_unsorted, sorted
end

puts(dictionary_sort(['can','feel','singing.','like','A','can']))
```

```
A
can
can
feel
like
singing.
```

How I would do it:

```ruby
#  The well-known quicksort algorithm.
def dictionary_sort arr
  return arr if arr.length <= 1

  middle = arr.pop
  less   = arr.select{|x| x.downcase <  middle.downcase}
  more   = arr.select{|x| x.downcase >= middle.downcase}

  sort(less) + [middle] + sort(more)
end

words = ['can','feel','singing.','like','A','can']
puts(dictionary_sort(words).join(' '))
```

```
A can can feel like singing.
```

Expanded english_number

(from on page 97)

How you could do it:

```ruby
def english_number number
  if number < 0  #  No negative numbers.
    return 'Please enter a number that isn\'t negative.'
  end
  if number == 0
    return 'zero'
  end

  #  No more special cases!  No more returns!
```

www.it-ebooks.info

```
num_string = ''  #  This is the string we will return.

ones_place = ['one',       'two',      'three',
              'four',      'five',     'six',
              'seven',     'eight',    'nine']
tens_place = ['ten',       'twenty',   'thirty',
              'forty',     'fifty',    'sixty',
              'seventy',   'eighty',   'ninety']
teenagers  = ['eleven',    'twelve',   'thirteen',
              'fourteen',  'fifteen',  'sixteen',
              'seventeen', 'eighteen', 'nineteen']

zillions = [['hundred',            2],
            ['thousand',           3],
            ['million',            6],
            ['billion',            9],
            ['trillion',          12],
            ['quadrillion',       15],
            ['quintillion',       18],
            ['sextillion',        21],
            ['septillion',        24],
            ['octillion',         27],
            ['nonillion',         30],
            ['decillion',         33],
            ['undecillion',       36],
            ['duodecillion',      39],
            ['tredecillion',      42],
            ['quattuordecillion', 45],
            ['quindecillion',     48],
            ['sexdecillion',      51],
            ['septendecillion',   54],
            ['octodecillion',     57],
            ['novemdecillion',    60],
            ['vigintillion',      63],
            ['googol',           100]]

#  "left" is how much of the number
#         we still have left to write out.
#  "write" is the part we are
#          writing out right now.
#  write and left...get it?  :)
```

```
left  = number

while zillions.length > 0
  zil_pair = zillions.pop
  zil_name =       zil_pair[0]
  zil_base = 10 ** zil_pair[1]

  write = left/zil_base          #  How many zillions left?
  left  = left - write*zil_base  #  Subtract off those zillions.

  if write > 0
    #  Now here's the recursion:
    prefix = english_number write

    num_string = num_string + prefix + ' ' + zil_name

    if left > 0
      #  So we don't write 'two billionfifty-one'...
      num_string = num_string + ' '
    end
  end
end

write = left/10          #  How many tens left?
left  = left - write*10  #  Subtract off those tens.

if write > 0
  if ((write == 1) and (left > 0))
    #  Since we can't write "tenty-two" instead of
    #  "twelve", we have to make a special exception
    #  for these.
    num_string = num_string + teenagers[left-1]
    #  The "-1" is because teenagers[3] is
    #  'fourteen', not 'thirteen'.

    #  Since we took care of the digit in the
    #  ones place already, we have nothing left to write.
    left = 0
  else
    num_string = num_string + tens_place[write-1]
    #  The "-1" is because tens_place[3] is
```

www.it-ebooks.info

```
      #  'forty', not 'thirty'.
    end

    if left > 0
      #  So we don't write 'sixtyfour'...
      num_string = num_string + '-'
    end
  end

  write = left  #  How many ones left to write out?
  left  = 0     #  Subtract off those ones.

  if write > 0
    num_string = num_string + ones_place[write-1]
    #  The "-1" is because ones_place[3] is
    #  'four', not 'three'.
  end

  #  Now we just return "num_string"...
  num_string
end

puts english_number(  0)
puts english_number(  9)
puts english_number( 10)
puts english_number( 11)
puts english_number( 17)
puts english_number( 32)
puts english_number( 88)
puts english_number( 99)
puts english_number(100)
puts english_number(101)
puts english_number(234)
puts english_number(3211)
puts english_number(999999)
puts english_number(1000000000000)
puts english_number(109238745102938560129834709285360238475982374561034)
```

```
zero
nine
```

www.it-ebooks.info

```
ten
eleven
seventeen
thirty-two
eighty-eight
ninety-nine
one hundred
one hundred one
two hundred thirty-four
three thousand two hundred eleven
nine hundred ninety-nine thousand nine hundred ninety-nine
one trillion
one hundred nine quindecillion two hundred
   thirty-eight quattuordecillion seven hundred forty-five ...
```

And that's just about how I would do it, too.

Wedding Number

(from on page 97)

I *told* you I didn't do this one. It was a joke! Move on!

"Ninety-nine Bottles of Beer on the Wall."

(from on page 97)

How you could do it:

```ruby
# english_number as above, plus this:
num_at_start = 5  #  change to 9999 if you want

num_now = num_at_start

while num_now > 2
  puts english_number(num_now).capitalize + ' bottles of beer on the wall, ' +
       english_number(num_now) + ' bottles of beer!'
  num_now = num_now - 1
  puts 'Take one down, pass it around, ' +
       english_number(num_now) + ' bottles of beer on the wall!'
end

puts "Two bottles of beer on the wall, two bottles of beer!"
puts "Take one down, pass it around, one bottle of beer on the wall!"
puts "One bottle of beer on the wall, one bottle of beer!"
puts "Take one down, pass it around, no more bottles of beer on the wall!"
```

www.it-ebooks.info

```
Five bottles of beer on the wall, five bottles of beer!
Take one down, pass it around, four bottles of beer on the wall!
Four bottles of beer on the wall, four bottles of beer!
Take one down, pass it around, three bottles of beer on the wall!
Three bottles of beer on the wall, three bottles of beer!
Take one down, pass it around, two bottles of beer on the wall!
Two bottles of beer on the wall, two bottles of beer!
Take one down, pass it around, one bottle of beer on the wall!
One bottle of beer on the wall, one bottle of beer!
Take one down, pass it around, no more bottles of beer on the wall!
```

How I would do it:

```
# english_number as above, plus this:

num_at_start = 5  #  change to 9999 if you want

num_bot = proc { |n| "#{english_number n} bottle#{n == 1 ? '' : 's'}" }

num_at_start.downto(2) do |num|
  bottles =
  puts "#{num_bot[num]} of beer on the wall, #{num_bot[num]} of beer!".capitalize
  puts "Take one down, pass it around, #{num_bot[num-1]} of beer on the wall!"
end
puts "#{num_bot[1]} of beer on the wall, #{num_bot[1]} of beer!".capitalize
puts "Take one down, pass it around, no more bottles of beer on the wall!"
```

```
Five bottles of beer on the wall, five bottles of beer!
Take one down, pass it around, four bottles of beer on the wall!
Four bottles of beer on the wall, four bottles of beer!
Take one down, pass it around, three bottles of beer on the wall!
Three bottles of beer on the wall, three bottles of beer!
Take one down, pass it around, two bottles of beer on the wall!
Two bottles of beer on the wall, two bottles of beer!
Take one down, pass it around, one bottle of beer on the wall!
One bottle of beer on the wall, one bottle of beer!
Take one down, pass it around, no more bottles of beer on the wall!
```

www.it-ebooks.info

A.8 Exercises from Chapter 11

Safer Picture Downloading

(from on page 110)

Well, since I was asking you to adapt it to *your* computer, I can't really show you how to do it. I will show you the program I *actually* wrote, though.

It's a bit more complex that the other examples here, partly because it's a real, working tool.

```
#  For Katy, with love.

###  Download pictures from camera card.


require 'win32ole'

STDOUT.sync = true
Thread.abort_on_exception = true

Dir.chdir 'C:\Documents and Settings\Chris\Desktop\pictureinbox'

#  Always look here for pics.
pic_names = Dir['!undated/**/*.{jpg,avi}']
thm_names = Dir['!undated/**/*.{thm}'    ]

#  Scan for memory cards in the card reader.
WIN32OLE.new("Scripting.FileSystemObject").Drives.each() do |x|
  #driveType 1 is removable disk
  if x.DriveType == 1 && x.IsReady
    pic_names += Dir[x.DriveLetter+':/**/*.{jpg,avi}']
    thm_names += Dir[x.DriveLetter+':/**/*.{thm}'    ]
  end
end

months = %w(jan feb mar apr may jun jul aug sep oct nov dec)

encountered_error = false

print "Downloading #{pic_names.size} files:  "
```

www.it-ebooks.info

```
pic_names.each do |name|
  print '.'
  is_movie = (name[-3..-1].downcase == 'avi')

  if is_movie
    orientation = 0
    new_name = File.open(name) do |f|
      f.seek(0x144,IO::SEEK_SET)
      f.read(20)
    end

    new_name[0...3] = '%.2d' % (1 + months.index(new_name[0...3].downcase))
    new_name = new_name[-4..-1] + ' ' + new_name[0...-5]
  else
    new_name, orientation = File.open(name) do |f|
      f.seek(0x36, IO::SEEK_SET)
      orientation_ = f.read(1)[0]
      f.seek(0xbc, IO::SEEK_SET)
      new_name_ = f.read(19)
      [new_name_, orientation_]
    end
  end

  [4,7,10,13,16].each {|n| new_name[n] = '.'}

  if new_name[0] != '2'[0]
    encountered_error = true
    puts "\n"+'ERROR:  Could not process "'+name+
       '" because it\'s not in the proper format!'
    next
  end

  save_name = new_name + (is_movie ? '.orig.avi' : '.jpg')

  #  Make sure we don't save over another file!!
  while FileTest.exist? save_name
    new_name += 'a'
    save_name = new_name + (is_movie ? '.orig.avi' : '.jpg')
  end
```

www.it-ebooks.info

```
  case orientation
    when 6
      ‘convert "#{name}" -rotate "90>"  "#{save_name}"‘
      File.delete name
    when 8
      ‘convert "#{name}" -rotate "-90>" "#{save_name}"‘
      File.delete name
    else
      File.rename name, save_name
  end
end

print "\nDeleting #{thm_names.size} THM files:  "

thm_names.each do |name|
  print '.'
  File.delete name
end

#  If something bad happened, make sure she
#  sees the error message before the window closes.
if encountered_error
  puts
  puts "Press [Enter] to finish."
  puts
  gets
end
```

Build Your Own Playlist

(from on page 110)

How you could do it:

```
# using the shuffle method as defined above
all_oggs = shuffle(Dir['**/*.ogg'])

File.open 'playlist.m3u', 'w' do |f|
  all_oggs.each do |ogg|
    f.write ogg+"\n"
  end
end
puts 'Done!'
```

www.it-ebooks.info

And that's exactly how I'd do it, too.

Build a Better Playlist

(from on page 110)

How you could do it:

```
def music_shuffle filenames
  #  We don't want a perfectly random shuffle, so let's
  #  instead do a shuffle like card-shuffling.  Let's
  #  shuffle the "deck" twice, then cut it once.  That's
  #  not enough times to make a perfect shuffle, but it
  #  does mix things up a bit.

  #  Before we do anything, let's actually *sort* the
  #  input, since we don't know how shuffled it might
  #  already be, and we don't want it to be *too* random.
  filenames = filenames.sort
  len       = filenames.length

  #  Now we shuffle twice.
  2.times do
    l_idx = 0     # index of next card in left pile
    r_idx = len/2 # index of next card in right pile
    shuf  = []
    #  NOTE:  If we have an odd number of "cards",
    #         then the right pile will be larger.

    while shuf.length < len
      if shuf.length%2 == 0
        #  take card from right pile
        shuf.push(filenames[r_idx])
        r_idx = r_idx + 1
      else
        #  take card from left pile
        shuf.push(filenames[l_idx])
        l_idx = l_idx + 1
      end
    end

    filenames = shuf
  end
```

www.it-ebooks.info

```
  #  And cut the deck.
  arr = []
  cut = rand(len) # index of card to cut at
  idx = 0

  while idx < len
    arr.push(filenames[(idx+cut)%len])
    idx = idx + 1
  end

  arr
end

songs = ['aa/bbb',   'aa/ccc',   'aa/ddd',
         'AAA/xxxx', 'AAA/yyyy', 'AAA/zzzz', 'foo/bar']

puts(music_shuffle(songs))
```

```
foo/bar
AAA/yyyy
aa/bbb
aa/ddd
AAA/xxxx
AAA/zzzz
aa/ccc
```

Well, that's OK, I guess. It's not all that random, and maybe if you had a larger playlist you'd want to shuffle it three or four times...I don't really know.

A better way would be mix more carefully and on every level (genre, artist, album). For example, if I have a playlist that is two-thirds lounge and one-third jazz, I want a jazz song roughly every third song (and rarely two in a row and *never* three in a row). Further, if I had, among all the jazz songs, only two by Kurt Elling (travesty, I know), then one should be *somewhere* in the first half of the playlist, and the other should be *somewhere* in the last half. (But where in the respective halves they appear should be truly random.) And all these constraints must be met simultaneously.

www.it-ebooks.info

What I do is find similar songs (let's say songs on the same CD), mix them up, and spread them out as far away from each other as I can in the next grouping (say, songs by the same artist). Then I do the same for the next level up (say, genre). The nice thing is that this algorithm is recursive, so I can add levels for free if I want. For example, I have a Billie Holiday CD with multiple recordings of one of the songs. I like it, but I'd like those to be spread out as far from each other as possible in the playlist (while respecting all other constraints at higher levels). No problem—I just make a directory inside the CD directory and move the similar recordings all in there, and the recursion takes care of the rest!

Enough talk; here's how I would do it:

```
def music_shuffle filenames
  songs_and_paths = filenames.map do |s|
    [s, s.split('/')]  #  [song, path]
  end

  tree = {:root => []}

  #  put each song into the tree
  insert_into_tree = proc do |branch, song, path|
    if path.length == 0 # add to current branch
      branch[:root] << song
    else # delve deeper
      sub_branch = path[0]
      path.shift # like "pop", but pops off the front

      if !branch[sub_branch]
        branch[sub_branch] = {:root => []}
      end

      insert_into_tree[branch[sub_branch], song, path]
    end
  end

  songs_and_paths.each{|sp| insert_into_tree[tree, *sp]}

  #  recursively:
  #    - shuffle sub-branches (and root)
  #    - weight each sub-branch (and root)
  #    - merge (shuffle) these groups together
```

www.it-ebooks.info

```
  shuffle_branch = proc do |branch|
    shuffled_subs = []

    branch.each do |key, unshuffled|
      shuffled_subs << if key == :root
        unshuffled # At this level, these are all duplicates.
      else
        shuffle_branch[unshuffled]
      end
    end

    weighted_songs = []

    shuffled_subs.each do |shuffled_songs|
      shuffled_songs.each_with_index do |song, idx|
        num = shuffled_songs.length.to_f
        weight = (idx + rand) / num
        weighted_songs << [song, weight]
      end
    end

    weighted_songs.sort_by{|s,v| v}.map{|s,v| s}
  end

  shuffle_branch[tree]
end

songs = ['aa/bbb',   'aa/ccc',   'aa/ddd',
         'AAA/xxxx', 'AAA/yyyy', 'AAA/zzzz', 'foo/bar']

puts(music_shuffle(songs))
```

```
AAA/yyyy
aa/ccc
aa/bbb
foo/bar
AAA/zzzz
AAA/xxxx
aa/ddd
```

www.it-ebooks.info

It might be hard to tell with such a tiny playlist, but with 500 songs you really begin to appreciate how well this method works.

A.9 Exercises from Chapter 12

One Billion Seconds!

(from on page 115)

Well, I don't know your brithday, so I don't know how you'd do it, but here's how I would do it:

```
#  I don't know what second I was born.
puts(Time.gm(1976, 8, 3, 13, 31) + 10**9)

#  And yes, I had a party.  It was awesome
#  (at least the parts I remember).
```

```
Fri Apr 11 15:17:40 UTC 2008
```

Happy Birthday!

(from on page 115)

How you could do it:

```
puts 'What year were you born?'
b_year = gets.chomp.to_i

puts 'What month were you born?  (1-12)'
b_month = gets.chomp.to_i

puts 'What day of the month were you born?'
b_day = gets.chomp.to_i

b = Time.local(b_year, b_month, b_day)
t = Time.new

age = 1

while Time.local(b_year + age, b_month, b_day) <= t
  puts 'SPANK!'
  age = age + 1
end
```

```
What year were you born?
2002
What month were you born?  (1-12)
2
What day of the month were you born?
20th
SPANK!
SPANK!
SPANK!
SPANK!
SPANK!
SPANK!
SPANK!
```

How I would do it:

```ruby
puts 'Hey, when were you born?  (Please use YYYYMMDD format.)'
input = gets.chomp

b_year  = input[0..3].to_i
b_month = input[4..5].to_i
b_day   = input[6..7].to_i

t = Time.new

t_year  = t.year
t_month = t.month
t_day   = t.day

age = t_year - b_year

if t_month < b_month || (t_month == b_month && t_day < b_day)
  age -= 1
end

if t_month == b_month && t_day == b_day
  puts 'HAPPY BIRTHDAY!!'
end

age.times { puts 'SPANK!' }
```

www.it-ebooks.info

```
Hey, when were you born?  (Please use YYYYMMDD format.)
20020220
SPANK!
SPANK!
SPANK!
SPANK!
SPANK!
SPANK!
SPANK!
```

Party Like It's roman_to_integer mcmxcix!

(from on page 120)

How you could do it:

```
def roman_to_integer roman
  digit_vals = {'i' =>    1,
                'v' =>    5,
                'x' =>   10,
                'l' =>   50,
                'c' =>  100,
                'd' =>  500,
                'm' => 1000}

  total = 0
  prev  = 0
  index = roman.length - 1

  while index >= 0
    c = roman[index].chr.downcase
    index = index - 1
    val = digit_vals[c]

    if !val
      puts 'This is not a valid roman numeral!'
      return
    end

    if val < prev
      val = val * -1
    else
      prev = val
    end
```

www.it-ebooks.info

```
    total = total + val
  end

  total
end

puts(roman_to_integer('mcmxcix'))
puts(roman_to_integer('CCCLXV'))
```

```
1999
365
```

How I would do it:

```
def roman_to_integer roman
  digit_vals = {'i' =>    1,
                'v' =>    5,
                'x' =>   10,
                'l' =>   50,
                'c' =>  100,
                'd' =>  500,
                'm' => 1000}

  total = 0
  prev  = 0

  roman.reverse.each_char do |c_or_C|
    c   = c_or_C.downcase
    val = digit_vals[c]

    if !val
      puts 'This is not a valid roman numeral!'
      return
    end

    if val < prev
      val *= -1
    else
      prev = val
    end
    total += val
  end

  total
end

puts(roman_to_integer('mcmxcix'))
puts(roman_to_integer('CCCLXV'))
```

```
#<NoMethodError: undefined method ‘each_char' for "xicxmcm":String>
```

Birthday Helper!

(from on page 120)

How you could do it:

```
#  First, load in the birthdates.
birth_dates = {}
File.read('birthdates.txt').each_line do |line|
  line = line.chomp
  #  Find the index of first comma,
  #  so we know where the name ends.
  first_comma = 0
  while line[first_comma].chr != ',' &&
        first_comma < line.length
    first_comma = first_comma + 1
  end

  name = line[0..(first_comma - 1)]
  date = line[-12..-1]

  birth_dates[name] = date
end

#  Now ask the user which one they want to know.
puts 'Whose birthday would you like to know?'
name = gets.chomp
date = birth_dates[name]

if date == nil
  puts "Oooh, I don't know that one..."
else
```

www.it-ebooks.info

```
  puts date[0..5]
end
```

```
Whose birthday would you like to know?
Christopher Plummer
Dec 13
```

How I would do it:

```
#  First, load in the birthdates.
birth_dates = {}

File.readlines('birthdates.txt').each do |line|
  name, date, year = line.split(',')
  birth_dates[name] = Time.gm(year, *(date.split))
end

#  Now ask the user which one they want to know.
puts 'Whose birthday would you like to know?'
name = gets.chomp
bday = birth_dates[name]

if bday == nil
  puts "Oooh, I don't know that one..."
else
  now = Time.new
  age = now.year - bday.year

  if now.month > bday.month || (now.month == bday.month && now.day > bday.day)
    age += 1
  end

  if now.month == bday.month && now.day == bday.day
    puts "#{name} turns #{age} TODAY!!"
  else
    date = bday.strftime "%b %d"
    puts "#{name} will be #{age} on #{date}."
  end
end
```

www.it-ebooks.info

```
Whose birthday would you like to know?
Christopher Pine
Christopher Pine will be 33 on Aug 03.
```

A.10 Exercises from Chapter 13

Extend the Built-in Classes

(from on page 123)

How you could do it:

```ruby
class Array
  def shuffle
    arr = self
    #  Now we can just copy the old shuffle method.

    shuf = []

    while arr.length > 0
      # Randomly pick one element of the array.
      rand_index = rand(arr.length)

      # Now go through each item in the array,
      # putting them all into new_arr except for
      # the randomly chosen one, which goes into
      # shuf.
      curr_index = 0
      new_arr = []

      arr.each do |item|
        if curr_index == rand_index
          shuf.push item
        else
          new_arr.push item
        end

        curr_index = curr_index + 1
      end

      # Replace the original array with the new,
      # smaller array.
      arr = new_arr
    end
```

www.it-ebooks.info

```
    shuf
  end
end

class Integer
  def factorial
    if self <= 1
      1
    else
      self * (self-1).factorial
    end
  end

  def to_roman
    #  I chose old-school roman numerals just to save space.
    roman = ''

    roman = roman + 'M' * (self        / 1000)
    roman = roman + 'D' * (self % 1000 /  500)
    roman = roman + 'C' * (self %  500 /  100)
    roman = roman + 'L' * (self %  100 /   50)
    roman = roman + 'X' * (self %   50 /   10)
    roman = roman + 'V' * (self %   10 /    5)
    roman = roman + 'I' * (self %    5 /    1)

    roman
  end
end

puts [1,2,3,4,5].shuffle
puts  7.factorial
puts 73.to_roman
```

```
3
5
4
1
2
5040
LXXIII
```

How I would do it:

```
class Array
  def shuffle
    sort_by(&:rand)  #  "self" is implied, remember?
  end
end

class Integer
  def factorial
    raise 'Must not use negative integer' if self < 0
    (self <= 1) ? 1 : self * (self-1).factorial
  end

  def to_roman
    #  I chose old-school roman numerals just to save space.
    raise 'Must use positive integer' if self <= 0

    roman = ''

    roman << 'M' * (self        / 1000)
    roman << 'D' * (self % 1000 /  500)
    roman << 'C' * (self %  500 /  100)
    roman << 'L' * (self %  100 /   50)
    roman << 'X' * (self %   50 /   10)
    roman << 'V' * (self %   10 /    5)
    roman << 'I' * (self %    5 /    1)

    roman
  end
end

#  Get ready for the pure awesome...
p 7.factorial.to_roman.split(//).shuffle
```

```
["X", "X", "M", "M", "M", "X", "M", "X", "M"]
```

Orange Tree

(from on page 133)

How you could do it:

```ruby
class OrangeTree
  def initialize
    @height       = 0
    @orange_count = 0
    @alive        = true
  end

  def height
    if @alive
      @height
    else
      'A dead tree is not very tall.  :('
    end
  end

  def count_the_oranges
    if @alive
      @orange_count
    else
      'A dead tree has no oranges.  :('
    end
  end

  def one_year_passes
    if @alive
      @height = @height + 0.4
      @orange_count = 0 # old oranges fall off

      if @height > 10 && rand(2) > 0
        # tree dies
        @alive = false
        'Oh, no!  The tree is too old, and has died.  :('
      elsif @height > 2
        # new oranges grow
        @orange_count = (@height * 15 - 25).to_i
        "This year your tree grew to #{@height}m tall," +
        " and produced #{@orange_count} oranges."
      else
        "This year your tree grew to #{@height}m tall," +
        " but is still too young to bear fruit."
      end
```

www.it-ebooks.info

```
    else
      'A year later, the tree is still dead.  :('
    end
  end

  def pick_an_orange
    if @alive
      if @orange_count > 0
        @orange_count = @orange_count - 1
        'You pick a juicy, delicious orange!'
      else
        'You search every branch, but find no oranges.'
      end
    else
      'A dead tree has nothing to pick.  :('
    end
  end
end

ot = OrangeTree.new
23.times do
  ot.one_year_passes
end
puts(ot.one_year_passes)
puts(ot.count_the_oranges)
puts(ot.height)
puts(ot.one_year_passes)
puts(ot.one_year_passes)
puts(ot.one_year_passes)
puts(ot.one_year_passes)
puts(ot.one_year_passes)
puts(ot.height)
puts(ot.count_the_oranges)
puts(ot.pick_an_orange)
```

```
This year your tree grew to 9.6m tall, and produced 119 oranges.
119
9.6
This year your tree grew to 10.0m tall, and produced 125 oranges.
Oh, no! The tree is too old, and has died. :(
```

www.it-ebooks.info

```
A year later, the tree is still dead. :(
A year later, the tree is still dead. :(
A year later, the tree is still dead. :(
A dead tree is not very tall. :(
A dead tree has no oranges. :(
A dead tree has nothing to pick. :(
```

That's pretty much how I would do it, too: clean and simple.

Interactive Baby Dragon

(from on page 133)

How you could do it:

```
#  using the Dragon class from the chapter

puts 'What would you like to name your baby dragon?'
name = gets.chomp
pet  = Dragon.new name

while true
  puts
  puts 'commands:  feed, toss, walk, rock, put to bed, exit'
  command = gets.chomp

  if command == 'exit'
    exit
  elsif command == 'feed'
    pet.feed
  elsif command == 'toss'
    pet.toss
  elsif command == 'walk'
    pet.walk
  elsif command == 'rock'
    pet.rock
  elsif command == 'put to bed'
    pet.put_to_bed
  else
    puts 'Huh?  Please type one of the commands.'
  end
end
```

www.it-ebooks.info

How I would do it:

```
#  using the Dragon class from the chapter

puts 'What would you like to name your baby dragon?'
name = gets.chomp
pet  = Dragon.new name
obj  = Object.new  #  just a blank, dummy object

while true
  puts
  puts 'commands:  feed, toss, walk, rock, put to bed, exit'
  command = gets.chomp

  if command == 'exit'
    exit
  elsif pet.respond_to?(command) && !obj.respond_to?(command)
    #  I only want to accept methods that dragons have,
    #  but that regular objects *don't* have.
    pet.send command
  else
    puts 'Huh?  Please type one of the commands.'
  end
end
```

A.11 Exercises from Chapter 14

Even Better Profiling

(from on page 142)

How you could do it:

```
def profile block_description, &block
  #  To turn profiling on/off, set this
  #  to true/false.
  profiling_on = false

  if profiling_on
    start_time = Time.new
    block.call
    duration = Time.new - start_time
```

www.it-ebooks.info

```
    puts "#{block_description}:  #{duration} seconds"
  else
    block.call
  end
end
```

How I would do it:

```
$OPT_PROFILING_ON = false

def profile block_description, &block
  if $OPT_PROFILING_ON
    start_time = Time.new
    block[]
    duration = Time.new - start_time
    puts "#{block_description}:  #{duration} seconds"
  else
    block[]
  end
end
```

Grandfather Clock

(from on page 142)

How you could do it:

```
def grandfather_clock &block
  hour = Time.new.hour

  if hour >= 13
    hour = hour - 12
  end

  if hour == 0
    hour = 12
  end

  hour.times do
    block.call
  end
end
```

www.it-ebooks.info

```
grandfather_clock do
  puts 'DONG!'
end
```

```
DONG!
DONG!
DONG!
DONG!
DONG!
DONG!
DONG!
DONG!
DONG!
DONG!
DONG!
```

How I would do it:

```
def grandfather_clock &block
  hour = (Time.new.hour + 11)%12 + 1

  hour.times(&block)
end

grandfather_clock { puts 'DONG!' }
```

```
DONG!
DONG!
DONG!
DONG!
DONG!
DONG!
DONG!
DONG!
DONG!
DONG!
DONG!
```

Program Logger

(from on page 143)

How you could do it:

www.it-ebooks.info

```
def log desc, &block
  puts 'Beginning "' + desc + '"...'
  result = block.call
  puts '..."' + desc + '" finished, returning:  ' + result.to_s
end
log 'outer block' do
  log 'some little block' do
    1**1 + 2**2
  end

  log 'yet another block' do
    '!doof iahT ekil I'.reverse
  end

  '0' == 0
end
```

```
Beginning "outer block"...
Beginning "some little block"...
..."some little block" finished, returning:  5
Beginning "yet another block"...
..."yet another block" finished, returning:  I like Thai food!
..."outer block" finished, returning:  false
```

How I would do it:

```
def log desc, &block
  puts "Beginning #{desc.inspect}..."
  result = block[]
  puts "...#{desc.inspect} finished, returning:  #{result}"
end

log 'outer block' do
  log 'some little block' do
    1**1 + 2**2
  end

  log 'yet another block' do
    '!doof iahT ekil I'.reverse
```

www.it-ebooks.info

```
  end

  '0' == 0
end
```

```
Beginning "outer block"...
Beginning "some little block"...
..."some little block" finished, returning:  5
Beginning "yet another block"...
..."yet another block" finished, returning:  I like Thai food!
..."outer block" finished, returning:  false
```

Better Program Logger

(from on page 143)

How you could do it:

```
$logger_depth = 0

def log desc, &block
  prefix = '  '*$logger_depth

  puts prefix + 'Beginning "' + desc + '"...'
  $logger_depth = $logger_depth + 1
  result = block.call
  $logger_depth = $logger_depth - 1
  puts prefix + '..."' + desc + '" finished, returning:  ' + result.to_s
end

log 'outer block' do
  log 'some little block' do
    log 'teeny-tiny block' do
      'lOtS oF lOVe'.downcase
    end

    7 * 3 * 2
  end

  log 'yet another block' do
    '!doof naidnI evol I'.reverse
  end
```

www.it-ebooks.info

```
  '0' == "0"
end
```

```
Beginning "outer block"...
  Beginning "some little block"...
    Beginning "teeny-tiny block"...
    ..."teeny-tiny block" finished, returning:  lots of love
  ..."some little block" finished, returning:  42
  Beginning "yet another block"...
  ..."yet another block" finished, returning:  I love Indian food!
..."outer block" finished, returning:  true
```

How I would do it:

```
$logger_depth = 0

def log desc, &block
  prefix = '  '*$logger_depth

  puts prefix+"Beginning #{desc.inspect}..."
  $logger_depth += 1
  result = block[]
  $logger_depth -= 1
  puts prefix+"...#{desc.inspect} finished, returning:  #{result}"
end

log 'outer block' do
  log 'some little block' do
    log 'teeny-tiny block' do
      'lOtS oF lOVe'.downcase
    end

    7 * 3 * 2
  end

  log 'yet another block' do
    '!doof naidnI evol I'.reverse
  end

  '0' == "0"
end
```

www.it-ebooks.info

```
Beginning "outer block"...
  Beginning "some little block"...
    Beginning "teeny-tiny block"...
    ..."teeny-tiny block" finished, returning:  lots of love
  ..."some little block" finished, returning:  42
  Beginning "yet another block"...
  ..."yet another block" finished, returning:  I love Indian food!
..."outer block" finished, returning:  true
```

www.it-ebooks.info

Index

www.it-ebooks.info

www.it-ebooks.info

www.it-ebooks.info

www.it-ebooks.info

The Pragmatic Bookshelf

Available in paperback and DRM-free PDF, our titles are here to help you stay on top of your game. The following are in print as of July 2009; be sure to check our website at pragprog.com *for newer titles.*

Title	**Year**	**ISBN**	**Pages**
Advanced Rails Recipes: 84 New Ways to Build Stunning Rails Apps	2008	9780978739225	464
Agile Retrospectives: Making Good Teams Great	2006	9780977616640	200
Agile Web Development with Rails, Third Edition	2009	9781934356166	784
Augmented Reality: A Practical Guide	2008	9781934356036	328
Behind Closed Doors: Secrets of Great Management	2005	9780976694021	192
Best of Ruby Quiz	2006	9780976694076	304
Core Animation for Mac OS X and the iPhone: Creating Compelling Dynamic User Interfaces	2008	9781934356104	200
Data Crunching: Solve Everyday Problems using Java, Python, and More	2005	9780974514079	208
Deploying Rails Applications: A Step-by-Step Guide	2008	9780978739201	280
Design Accessible Web Sites: 36 Keys to Creating Content for All Audiences and Platforms	2007	9781934356029	336
Desktop GIS: Mapping the Planet with Open Source Tools	2008	9781934356067	368
Developing Facebook Platform Applications with Rails	2008	9781934356128	200
Enterprise Integration with Ruby	2006	9780976694069	360
Enterprise Recipes with Ruby and Rails	2008	9781934356234	416
Everyday Scripting with Ruby: for Teams, Testers, and You	2007	9780977616619	320
FXRuby: Create Lean and Mean GUIs with Ruby	2008	9781934356074	240
From Java To Ruby: Things Every Manager Should Know	2006	9780976694090	160
GIS for Web Developers: Adding Where to Your Web Applications	2007	9780974514093	275
Google Maps API, V2: Adding Where to Your Applications	2006	PDF-Only	83
Groovy Recipes: Greasing the Wheels of Java	2008	9780978739294	264
Hello, Android: Introducing Google's Mobile Development Platform	2008	9781934356173	200
Interface Oriented Design	2006	9780976694052	240
Land the Tech Job You Love	2009	9781934356265	280

Continued on next page

www.it-ebooks.info

Title	**Year**	**ISBN**	**Pages**
Learn to Program, 2nd Edition	2009	9781934356364	230
Manage It! Your Guide to Modern Pragmatic Project Management	2007	9780978739249	360
Mastering Dojo: JavaScript and Ajax Tools for Great Web Experiences	2008	9781934356111	568
Modular Java: Creating Flexible Applications with OSGi and Spring	2009	9781934356401	260
No Fluff Just Stuff 2006 Anthology	2006	9780977616664	240
No Fluff Just Stuff 2007 Anthology	2007	9780978739287	320
Practical Programming: An Introduction to Computer Science Using Python	2009	9781934356272	350
Practices of an Agile Developer	2006	9780974514086	208
Pragmatic Project Automation: How to Build, Deploy, and Monitor Java Applications	2004	9780974514031	176
Pragmatic Thinking and Learning: Refactor Your Wetware	2008	9781934356050	288
Pragmatic Unit Testing in C# with NUnit	2007	9780977616671	176
Pragmatic Unit Testing in Java with JUnit	2003	9780974514017	160
Pragmatic Version Control Using Git	2008	9781934356159	200
Pragmatic Version Control using CVS	2003	9780974514000	176
Pragmatic Version Control using Subversion	2006	9780977616657	248
Programming Clojure	2009	9781934356333	304
Programming Erlang: Software for a Concurrent World	2007	9781934356005	536
Programming Groovy: Dynamic Productivity for the Java Developer	2008	9781934356098	320
Programming Ruby: The Pragmatic Programmers' Guide, Second Edition	2004	9780974514055	864
Programming Ruby 1.9: The Pragmatic Programmers' Guide	2009	9781934356081	960
Programming Scala: Tackle Multi-Core Complexity on the Java Virtual Machine	2009	9781934356319	250
Prototype and script.aculo.us: You Never Knew JavaScript Could Do This!	2007	9781934356012	448
Rails Recipes	2006	9780977616602	350
Rails for .NET Developers	2008	9781934356203	300
Rails for Java Developers	2007	9780977616695	336
Rails for PHP Developers	2008	9781934356043	432
Rapid GUI Development with QtRuby	2005	PDF-Only	83
Release It! Design and Deploy Production-Ready Software	2007	9780978739218	368
Scripted GUI Testing with Ruby	2008	9781934356180	192
Ship it! A Practical Guide to Successful Software Projects	2005	9780974514048	224

Continued on next page

www.it-ebooks.info

Title	**Year**	**ISBN**	**Pages**
Stripes ...and Java Web Development Is Fun Again	2008	9781934356210	375
TextMate: Power Editing for the Mac	2007	9780978739232	208
The Definitive ANTLR Reference: Building Domain-Specific Languages	2007	9780978739256	384
The Passionate Programmer: Creating a Remarkable Career in Software Development	2009	9781934356340	200
ThoughtWorks Anthology	2008	9781934356142	240
Ubuntu Kung Fu: Tips, Tricks, Hints, and Hacks	2008	9781934356227	400

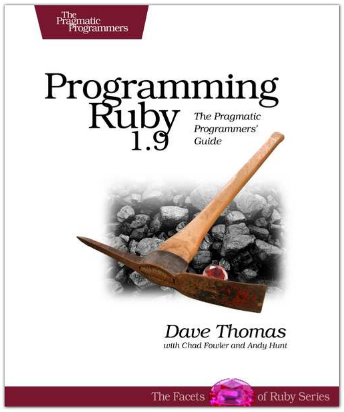
www.it-ebooks.info

All About Ruby

Programming Ruby 1.9 (The Pickaxe for 1.9)

The Pickaxe book, named for the tool on the cover, is the definitive reference to this highly-regarded language.

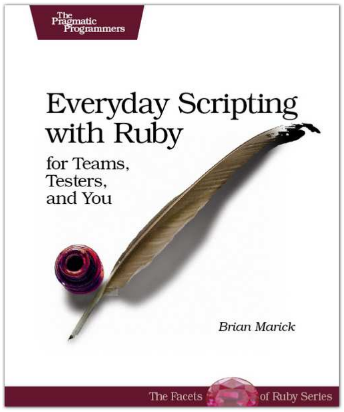
• Up-to-date and expanded for Ruby version 1.9 • Complete documentation of all the built-in classes, modules, and methods • Complete descriptions of all standard libraries • Learn more about Ruby's web tools, unit testing, and programming philosophy


The Pragmatic Programmers
Programming Ruby 1.9
The Pragmatic Programmers' Guide
Dave Thomas
with Chad Fowler and Andy Hunt
The Facets of Ruby Series

Programming Ruby 1.9: The Pragmatic Programmers' Guide
Dave Thomas with Chad Fowler and Andy Hunt
(992 pages) ISBN: 978-1-9343560-8-1. $49.95
http://pragprog.com/titles/ruby3

Everyday Scripting with Ruby

Don't waste that computer on your desk. Offload your daily 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. Learn the basics of the Ruby scripting language and see how to create scripts in a steady, controlled way using test-driven design.


The Pragmatic Programmers
Everyday Scripting with Ruby
for Teams, Testers, and You
Brian Marick
The Facets of Ruby Series

Everyday Scripting with Ruby: For Teams, Testers, and You
Brian Marick
(320 pages) ISBN: 0-9776166-1-4. $29.95
http://pragprog.com/titles/bmsft

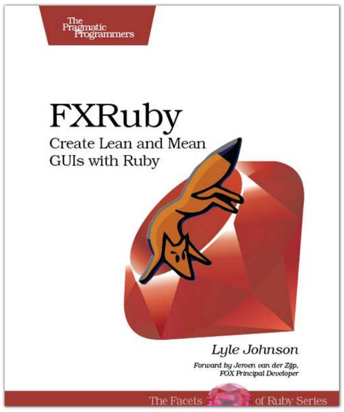
www.it-ebooks.info

Ruby & GUIs

FXRuby

Get started developing GUI applications using FXRuby. With a combination of tutorial exercises and focused, technical information, this book goes beyond the basics to equip you with proven, practical knowledge and techniques for developing real-world FXRuby applications. Learn directly from the lead developer of FXRuby, and you'll be writing powerful and sophisticated GUIs in your favorite programming language.

FXRuby Create Lean and Mean GUIs with Ruby
Lyle Johnson
(240 pages) ISBN: 978-1-9343560-7-4. $36.95
http://pragprog.com/titles/fxruby

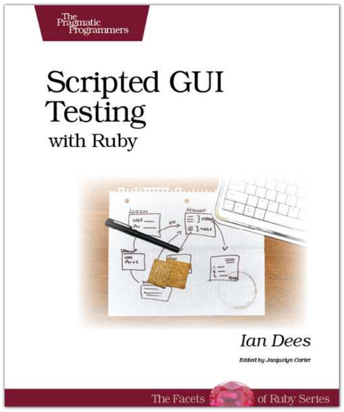

The Pragmatic Programmers
FXRuby
Create Lean and Mean GUIs with Ruby
Lyle Johnson
Forward by Jeroen van der Zijp, FOX Principal Developer
The Facets of Ruby Series

Scripted GUI Testing with Ruby

If you need to automatically test a user interface, this book is for you. Whether it's Windows, a Java platform (including Mac, Linux, and others) or a web app, you'll see how to test it reliably and repeatably.

This book is for people who want to get their hands dirty on examples from the real world—and who know that testing can be a joy when the tools don't get in the way. It starts with the mechanics of simulating button pushes and keystrokes, and builds up to writing clear code, organizing tests, and beyond.

Scripted GUI Testing with Ruby
Ian Dees
(192 pages) ISBN: 978-1-9343561-8-0. $34.95
http://pragprog.com/titles/idgtr


The Pragmatic Programmers
Scripted GUI Testing
with Ruby
Ian Dees
Edited by Jacquelyn Carter
The Facets of Ruby Series

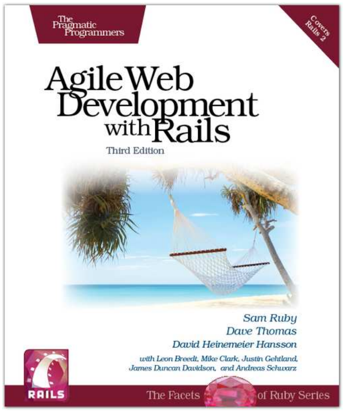
www.it-ebooks.info

Ruby on Rails

Agile Web Development with Rails

Rails is a full-stack, open-source web framework, with integrated support for unit, functional, and integration testing. It enforces good design principles, consistency of code across your team (and across your organization), and proper release management. This is the newly updated Third Edition, which goes beyond the award winning previous editions with new material covering the latest advances in Rails 2.0.

Agile Web Development with Rails: Third Edition
Sam Ruby, Dave Thomas, and David Heinemeier Hansson, et al.
(784 pages) ISBN: 978-1-9343561-6-6. $43.95
http://pragprog.com/titles/rails3



Advanced Rails Recipes

A collection of practical recipes for spicing up your web application without a lot of prep and cleanup. You'll learn how the pros have solved the tough problems using the most up-to-date Rails techniques (including Rails 2.0 features).

Advanced Rails Recipes
Mike Clark
(464 pages) ISBN: 978-0-9787392-2-5. $38.95
http://pragprog.com/titles/fr_arr