A Little Ruby, A Lot of Objects

Preface

Welcome to my little book. In it, my goal is to teach you a way to think about computation, to show you how far you can take a simple idea: that all computation consists of sending messages to objects. Object-oriented programming is no longer unusual, but taking it to the extreme – making *everything* an object – is still supported by only a few programming languages.

Can I justify this book in practical terms? Will reading it make you a better programmer, even if you never use "call with current continuation" or indulge in "metaclass hackery"? I think it might, but perhaps only if you're the sort of person who would read this sort of book even if it had no practical value.

The real reason for reading this book is that the ideas in it are *neat*. There's an intellectual heritage here, a history of people building idea upon idea. It's an academic heritage, but not in the fussy sense. It's more a joyous heritage of tinkerers, of people buttonholing their friends and saying, "You know, if I take *that* and think about it like *this*, look what I can do!"

Prerequisites

With effort, someone who didn't know programming could read this book. I expect that most readers will know at least one programming language, not necessarily an object-oriented one.

I use a few simple mathematical ideas in some of the examples. The factorial function is the most complex, and I explain a simplified form of it, rather than assume you know what it is. I don't think the book requires any particular mathematical inclination, so don't be scared off at the first sight of *factorial*.

Using the book

This book is written as a dialogue between two people, one who knows objects well, and one who doesn't. The text builds cumulatively. If you don't understand something about one chapter, you'll likely understand the next chapter even less. So I recommend you read slowly. The characters in the book take frequent breaks. I think that's a good idea.

This book uses Ruby, a freely available language developed by Yukihiro Matsumoto, but it is not a book about Ruby. Ruby constructs are introduced gradually, as they're needed, rather than in any systematic order. They're described only enough to allow you to understand code that contains them.

If you want to try variants of the examples, you may need a little more Ruby knowledge. The example files (see below) define new constructs a little more completely. However,

even with the examples, this book is not a Ruby tutorial. If you want to use Ruby for general-purpose programming – and you should, since it's a wonderful rapid-development language for many types of applications - the book to read is *Programming Ruby*, by David Thomas and Andrew Hunt (available online at www.rubycentral.com/book/index.html). You'll find that Ruby has many more features than this book describes.

Notation

Ruby text and values printed by the Ruby interpreter are in *italic font*. Everything else is in normal font. Important terms are in **bold** when they're defined.

Sometimes, one participant will show a partially completed snippet of Ruby code. The unfinished part is indicated with ***???***:

```
def finish_this
  ???
end
```

Bold italic font is used to draw your attention to a part of some Ruby code

```
class Something
  def some_function
    "look here"
  end
end
```

Running the examples

I recommend you play with the examples as you read.

As of this writing, Ruby works on Unix and Windows. It is available from www.ruby-lang.org. The Windows download comes from www.rubycentral.com, at www.rubycentral.com/downloads/ruby-install.html.

I recommend you use the Ruby interpreter `irb`. Here's an example:

```
> irb
irb(main):001:0> 1 + 1
2
irb(main):002:0>
```

All of the examples in the book are available from www.visibleworkings.com/little-ruby/source. At the points in the text where an example is complete, a marginal note names the example's file:

Exactly. What do you suppose this Ruby function does?

```
def factorial(n)
  if n == 1
    n
  else
    n * factorial(n-1)
  end
end
```

ch1-factorial.rb

The name tells me it computes factorial, but I'm not sure how.

You can either cut and paste the example into `irb`, or load the example into Ruby like this:

```
> irb
irb(main):001:0> load 'ch1-factorial.rb'
true
irb(main):002:0>
```

(This assumes you're running `irb` in the directories where the examples live.) Thereafter, you can type things like this:

```
irb(main):002:0> factorial 5
120
irb(main):003:0>
```

Acknowledgements

This book was inspired by *The Little Lisper*, by Daniel P. Friedman and Mattthias Felleisen. I fell in love with their book around 1984. The fourth edition, titled *The Little Schemer*, is still available. If you like this book, you'll like that one too, especially because it treats computation from a different perspective.

The Little Lisper is famous for using food in its examples. As the authors say, "[it] is not a good book to read while dieting." As an ironic homage to my inspiration, one of the characters here is an exercise freak. In that way, this is a very different book.

I received help and encouragement from people who read drafts: Al Chou, Mikkel Damsgaard, Joani DiSilvestro, Pat Eyler, Darrell Ferguson, Tammo Freese, Hal E. Fulton, Ned Konz, Dragos A. Manolescu, Dawn Marick, Pete McBreen, Nat Pryce, Christopher Sawtell, Kevin Smith, Dave Thomas, David Tillman, and Eugene Wallingford.