Programming Ruby
Keith Alcock, TCS Member
tcs@keithalcock.com

Ruby is an open source, cross-platform, object-oriented programming language especially suitable for, but not limited to, scripting tasks. This book, known as the PickAxe for its cover illustration, could easily guide an aspiring programmer to mastery of that first programming language. At the same time it provides an experienced programmer with nearly every language detail short of the interpreter's source code. The book has enabled me to satisfy a yearly resolution to learn a new programming language and has done so in record time.

Ruby shares with Objective-C a quality seldom found in current popular programming languages: substantial Smalltalk influence. Along with a minor amount of Smalltalk syntax, Ruby borrows many design ideas such as dynamic typing, internal iterators, blocks, and an object-oriented representation of even "primitive" data types. It melds these with a more popular C-style syntax, especially for control flow and argument passing, to achieve what its practitioners must consider the best of both (and maybe all) worlds. Objective-C is more schizophrenic, including all the syntax of C in addition to Smalltalk constructs offset by brackets. It requires little new syntax, just a mixing of old. A simple example illustrates the comparison.

Smalltalk:

```
1 to: 10 do: [ :index |
        self setTemperatureTo: 0.0 forHeaterIndex: index.
].
```

Ruby:

```
(0..9).each do | index |
        self.set_temperature_to_for_heater_index(0.0,index)
end
```

Objective-C:

```
int index;
for (index=0;index<10;index++)
        [ self setTemperatureTo: 0.0f forHeaterIndex: index ];
```

C:

```
int index;
for (index=0;index<10;index++)
        setTemperatureToForHeaterIndex(furnace,0.0f,index);
```

Programming Ruby is divided into four main sections: "Facets of Ruby" containing the Ruby tutorial, "Ruby in Its Setting" explaining how to use Ruby for various tasks including web integration and GUI development, "Ruby Crystallized" detailing language specifics, and the "Ruby Library Reference" documenting the built-in classes and

modules as well as the standard library from Abbrev to Zlib. A fifth section provides appendices and a comprehensive index. The book guides the reader from initial installation on day one to extending Ruby with C just before graduation, and then provides a comprehensive reference section just in case anything gets forgotten. It's a book that won't be confined to a shelf, but will find an important place opened on a desk.

In covering a topic, the author does an excellent job illuminating the big picture along with the details. For example, not only is iteration explained, but also the iterator; not only coercion, but double dispatch; not only classes, but types and dynamic typing; not only debugging, but unit testing. Ruby variables are flagged with $, @, or @@ to indicate scope (global, instance, class), which is enforced by the interpreter, but it is also useful to know that Ruby programmers use two spaces for indentation, which is only enforced by the community, as are variable and class naming conventions. Inclusion of this information ensures that Ruby programmers start off on the right foot and needn't unlearn frowned upon habits. Not everything is so serious, though. A fair amount of subtle humor is presented and I actually appreciated being associated with the "hippie-freak dynamic typing crowd" as the author fondly calls it.

Nothing is perfect, of course, and the book could be improved in some ways. A prominent and comprehensive class hierarchy diagram is lacking. Many methods accept a single, unnamed, optional block argument, but I missed a discussion of how to best handle cases when two blocks are required. Programmers are renowned for their insistent use of unnecessary abbreviations and the book contains examples such as hsh for hash, prc for proc for procedure, and thr for thread. Some documented methods return strange values which leave you wanting a more complete explanation. Two such examples are float.infinite? and stat.size? which would normally return Booleans but don't. These are obviously very minor details.

So why should you learn a new programming language this year (or every year)? According to the Pragmatic Programmers (one being author of this book), you can "broaden your thinking and avoid getting stuck in a rut." Ruby is filled with good ideas including some not yet mentioned like safe levels, freezing objects, and in place or immediate operations. If one isn't planning to use the language directly, but simply borrow its ideas, many can be applied to other languages. I added a Range class to my Smalltalk library, for instance, based on examples in this book. I highly recommend it.

Title: Programming Ruby
Author: Dave Thomas
Publisher: Pragmatic Bookshelf
Distributor: O'Reilly Media, Inc.
Pages: 862
Price: US $44.95
ISBN: 0-9745140-5-5
Website: http://www.pragmaticprogrammer.com/title/ruby