

## **Scripting, Object-Oriented, and Function Programming in Ruby**

(2 credits, TCCS 498) -- Prerequisite: TCCS360

Monday, 6:30 – 8:35

Professor Steve Hanks

Spring 2009

---

### **Overview**

Ruby (<http://www.ruby-lang.org/eng>) is a relatively new programming language that has quickly gained wide adoption in the industry. Ruby is in no sense a revolutionary language; rather it is a new packaging of ideas taken from several successful languages, programming paradigms, and practices. Studying the language provides both valuable practical experience, and a new exposure to key concepts and methodologies in program development. This course teaches both the Ruby language and the fundamental concepts from which the language was built.

---

### **Learning Objectives**

It is useful to describe the course learning objectives in terms of the conceptual foundations of the Ruby language

#### **Object-oriented programming**

Ruby is a truly object-oriented language (like Smalltalk but unlike Java). All entities are objects, and any object's behavior can be modified dynamically.

Students will learn to exploit this property of the language by

- learning and learning to use properties of "standard classes" like numbers, strings, and arrays
- modifying class behavior dynamically by "re-opening the class" and adding or changing methods
- building composite structures using mix-ins

#### **Functional programming**

Ruby supports functional programming in that it supports a procedural object along with lexical closures and continuations. Any method can accept a procedure (block) as a parameter. Iterator methods on the standard container classes (Arrays and Hashes) strongly support functional manipulations of those objects (unlike Java, which promotes imperative manipulation of objects). This leads to a fundamentally different way of programming, familiar to Lisp/Scheme programmers but foreign to C++/Java programmers. Students will learn to use this style of programming by

- mastering the standard iterator methods on standard containers
- building classes that accept procedural arguments
- programming using closures and continuations

### **Scripting and text processing**

The Ruby language borrows heavily from standard scripting/text languages like Perl and PHP. This leads to a style of programming heavily oriented toward processing text streams, and writing small program modules that are intended to be embedded in larger structures (web servers or processing pipes).

Students will learn to use these features by

- mastering standard text processing methods, in particular regular expression processing
- building small "filter-like" applications and embedding them in larger text-processing applications (e.g. processing HTML documents and embedding Ruby code in HTML)

### **Unit testing**

Ruby has a strong methodological commitment to unit testing and a testing-first methodology. The language ships with a unit testing package similar to JUnit. Students have been exposed briefly to unit testing, and this course will reinforce and deepen those skills by

- building unit tests for all code developed during the course lectures
- requiring all handed-in work to include unit tests

### **Open source and community development**

Ruby is an active open-source project, both in terms of developing the language, and in terms of developing public libraries that extend the language. Through the Gem system (similar to Perl modules and CPAN), the programmer can use, provide, and document libraries for use by the larger community. Being able to do this is a vital part of modern programming practice. Students will learn this practice by

- using existing gems in building code as part of the class
- learn to package their modules into gems and release them to the larger community

---

### **Required Text**

Thomas, "Programming Ruby: The Practical Programmer's Guide", Second Edition, The Pragmatic Bookshelf, 2005.

There is an online version of the text available.

Any other course material will be available on the Internet.

---

## **Deliverables and Grading**

Grading will be based on five assignments, assigned in weeks 2, 4, 6, 8, and 10. For each assignment, part of the class will work on solving the assignment; the other part will work on unit tests. Grading will be based on either on the submitted solution's ability to pass the student tests, or on the tests' ability to find incorrect solutions.

There will be no exams. 85% of the course grade will be based on assignment grades. The remaining 15% will be based on participation, both in class and in course discussion forums, ongoing discussion about current assignments, etc. We expect both the class and even work on the assignment to be very interactive and participatory.

---

## **Tentative Schedule**

Week 1 (3/30): Language fundamentals; development environment; language installation; unit testing framework.

Week 2 (4/6): Language fundamentals (cont.); primitive data types; basic control structures

Week 3 (4/13): Containers, iterators, blocks and closures (part 1)

Week 4 (4/20): Containers, iterators, blocks and closures (part 2); exception handling

Week 5 (4/27): Object-oriented programming (part 1)

Week 6 (5/4): Object-oriented programming (part 2); modules and mix-ins

Week 7 (5/11): Text processing and scripting; regular expressions

Week 8 (5/18): Text processing (cont.) ; embedding Ruby in web stacks, erb, HTML/XML processing etc.

Week 9 (5/25): No class

Week 10 (6/1): Ruby on Rails / ActiveRecord; the open source community and Rubyforge