

This 3 day course covers the fundamental components of the Ruby Programming Language. Emphasis is placed on the object oriented aspects of Ruby. Topics include arrays, hashes, regular expressions, io, exceptions, modules, and applications areas.

**Course Objectives:**

- Distinguish and use various Ruby databases.
- Master the use of arrays and hashes.
- Build home grown classes.
- Use the extensive pre-bundled classes.
- Use the I/O facilities of Ruby to read and write binary and text files.
- Master the use of Iterators to loop through the various data structures.
- Use Exceptions in handling various run time errors.
- Create Ruby modules.
- Use the wide variety of Ruby Modules that come with the Ruby distribution.

**Audience:** Programmers who have programmed languages such as, but not limited to, C, C++, Java, or Perl.

**Prerequisites:** Students should have at least six months of programming experience in at least one programming language.

**Number of Days: 3 days**

<p><b>1 An Introduction to Ruby</b>          What is Ruby?          Installing Ruby          Executing Ruby Code          Getting Help          Dynamic Types          Ruby Reserved Words          Naming Conventions</p> <p><b>2 Standard Ruby Data Types</b>          Numbers          Strings          Simple Input and Output          Converting String Input          Regular Expressions          Time Methods</p> <p><b>3 Language Components</b>          The if Statement          The case Construct          Loops          Iterators</p>	<p>Numeric Iterators          String Iterators          Methods          Odds and Ends</p> <p><b>4 Collections</b>          Arrays          Array Operator Methods          Array Equality Operator          Arrays as Stacks and Queues          Higher Dimensional Arrays          Other Useful Arrays Methods          Command Line Arguments          Hashes          Common Hash Methods          Sorting Hashes          Iterators with Arrays and Hashes          Arrays and Methods          Hashes and Methods          Named Parameters          Symbols</p>
--	--

	Procs		Creating Your Own Exceptions
	Closures		catch and throw
<b>5</b>	<b>Classes</b>	<b>8</b>	<b>Modules</b>
	Objects		Introduction
	Brief History of OOP		Using Core Ruby Classes
	OOP Vocabulary		Ruby Standard Library
	Creating a New Class		require
	Using Objects		Search Path
	Defining Operator Methods		File Organization
	Inheritance		load
	Ancestors		Modules
	self		include
	Access Levels - public		Mixins
	Access Levels – private		Using the Comparable Module
	Access Levels - protected		Collection Classes
	Access Levels - Specification		yield
	Class Data and Class Methods		Using the Enumerable Module
	Adding Methods to Classes and Objects	<b>9</b>	<b>Odds and Ends</b>
	Special Global Variables		Ruby Conventions
	Scope of Variables		Bit Manipulation
	Built-in Classes		Substituting
	The Math Class		Marshalling
	The NilClass Class		Reflection
	TrueClass and FalseClass		grep
	Built-in Class Hierarchy		Classes are Objects
<b>6</b>	<b>Input and Output</b>		Aliasing
	Introduction		Testing
	Reading from the Standard Input		Test::Unit::TestCase
	Reading a Character at a Time		Testing Your Own Classes
	Writing to the Standard Output		Freezing Objects
	Reading and Writing Disk Files		Object Equality
	Reading Files Using Iterators		
	I/O With Command Line Commands		
	Seeking About Files		
	tell		
	Capturing Data About Files		
	Processing Directories		
<b>7</b>	<b>Exceptions</b>		
	Introduction		
	Exception Hierarchy		
	Handling Exceptions		
	Multiple Rescue Clauses		
	Exceptions are Classes		
	ensure		
	retry		
	raise		