COURSE NAME

Ruby Scripting

OVERVIEW

The goal of the training is to give each participant a solid foundation on which to quickly become a productive Ruby developer.

- Object-oriented techniques to make code well-organized and maintainable
- The advantages of dynamic typing and open classes
- How blocks add clarity to thought and source code
- Efficient error-handling with exceptions
- Flexible numeric libraries to achieve precision and avoid under- and overflow
- Manipulating text with regular expressions

DURATION

3 Days

TRAINEE PREREQUISITES

Participants should already be comfortable with one high-level programming language, such as Java, C#, C++, C, Python, Perl, etc.

LAB SETUP (TO BE ARRANGED BY THE CLIENT)

Each participant must be provided with a machine

1) Hardware requirement (RAM, HDD, any other.):

i5 Processor, 8 GB RAM, 120 GB Minium

2) Software requirement(Part-I): Local installation.-if any (e.g Oracle DB, informatica, etc.), if Yes need to provide Dumps (through CD, USB, HDD) in two weeks in advance.

* Ruby 2.3.1
- sublime Text2

3) Software requirement (Part-II): remote access. Any server accessible, if yes, need server IP, Ports etc, need details 2-3 weeks in advance to initiate internal firewall exceptions.

None

4)Any URL's required to access.

https://www.ruby-lang.org/en/downloads/

https://sublimetext.com/2

5) Bandwidth requirement to conduct training (Internet speed requirement in MB's).

16 mbps

6) Provide trainer Laptop details (Model & Sl no) to issue Guest pass.

Lenovo R8K3LV8

DAY WISE SYLLABUS

Day 1
---

* Introduction
  - What is Ruby?
  - What is it used for?
  - Where do I get and install Ruby?
  - Core facilities, built-in library and standard library.
  - Basic concepts - object orientation, regular expressions, arrays, hashes, etc.

* Basic Ruby Language Elements
  - Structure of statements and comments.
  - Variables and constants.
  - Operators. Assignments, calculations, etc. Integer, float and string formats.
  - Single and double quotes, here documents, general strings.

* Execute System commands from RUBY
  - Using execution operator

* Control Structures
  - Blocks and the if statement.
  - Writing conditions.
  - Comparative, boolean and range operators.
  - Conditionals - if, unless, case, etc.
  - Loops - while, for in, until, etc. break, next, retry and redo. defined? and ternary operators.

* Collections (Arrays and Hashes) in Ruby
  - What is a collection?
  - Arrays and hashes.
  - Constructing an array.
  - Nesting arrays. Hash keys, iterators, etc.

Day 2
---
* Regular Expressions
  - Making Matches
  - Match Groups
  - MatchData
  - Prematch and Postmatch
  - Greedy Matching
  - String Methods
  - File Operations

- Modifiers i, o, x and m.
- Pattern matching variables.

* Special Variables and Pseudo-Variables
- ARGV, $0 and friends - the command line.
- Other special variables from $: through $$ to $<.
- Environment variables.
- Pseudo-variables.
- Reserved words in Ruby.

* Functions
- Functions and Methods
- Returning Values
- Returning Multiple Values
- Default and Multiple Arguments
- Assignment and Parameter Passing
- Modifying Receivers and Yielding New Objects
- Potential Side Effects of Reliance on Argument Values
- Parallel Assignment

* Blocks, Procs, and Lambdas
- What Is a Block?
- Line Breaks Are Significant
- Nameless Functions
- Look Familiar?
- Blocks and Arrays
- Procs and Lambdas
- Block or Hash?
- What Is a Closure?
- yield
- Blocks Within Blocks

* Object Orientation: Individual Objects
- History - unstructured and structured code. Introduction to object oriented programming.
- classes and methods.
- Static and nonstatic.
- Instances, constructors and destructors.
- Accessing members of a class.
- Loading and using classes.
- Direct access to variables.
- Encouraging class use.

* Classes and Objects

- Objects, classes and methods.
- Constructors and attributes.
- Instance and class variables.
- Local and global variables.
- Class and object methods.
- Including files - load and require.

* More Classes and Objects
- Public, private and protected visibility.
- Singletons and defs.
- Inheritance mixins, and super.
- Destructors and garbage collection.
- Namespaces and modules.

- Calling methods with code blocks.
- Looking inside objects - reflection.

Day 3
---

* Exception Handling
- rescue: Execute Code When Error Occurs
- ensure: Execute Code Whether or Not an Error Occurs
- else: Execute Code When No Error Occurs
- Error Numbers
- retry: Attempt to Execute Code Again After an Error
- raise: Reactivate a Handled Error

Modules and Mixins
- A Module Is Like a Class
- Module Methods
- Modules as Namespaces
- Included Modules, or “Mixins”
- Name Conflicts
- Alias Methods
- Mix In with Care!
- Including Modules from Files

* Files and IO
- Opening and Closing Files
- Characters and Compatibility
- Files and Directories

- Copying Files
- Directory Inquiries
- A Discursion into Recursion
- Sorting by Size