Perl Scripting

Duration: 4 Days *(Face-to-Face & Remote-Live)*, or 28 Hours *(On-Demand)*

Price: CDN$2,775 *(Face-to-Face & Remote-Live)*, or CDN$1,995 *(On-Demand)*

Discounts: We offer multiple discount options. Click here for more info.

Delivery Options: Attend face-to-face in the classroom, remote-live or on-demand training.

Students Will Learn

- Writing Perl scripts
- Using Perl to read from and write to files
- Using arrays, hashes and scalars to create robust Perl programs
- Using arithmetic, assignment, comparison and logical operators in Perl
- Writing clear and concise regular expressions
- Creating subroutines to promote cleaner code
- Controlling Perl program flow using conditional constructs and loops
- Creating hard and anonymous references
- Manipulating lists
- Debugging Perl scripts for optimum performance
- Writing Perl scripts that process script arguments
- Implementing pattern matching in Perl scripts using regular expressions
- Writing Perl Programs that access a Relational Database

Course Description

This hands-on Perl Programming course presents a thorough introduction to the Perl scripting language emphasizing the rapid development of portable and modular Perl programs and scripts. Students are introduced to all major language elements including built-in data types, powerful operators, flow control and robust built-in functions. The course also covers the use of command line processing, file and directory I/O to create flexible and user friendly programs. Attendees will also be introduced to object-oriented programming in Perl as well as how to use pattern matching with Regular Expressions and string handling functions to manipulate files and data. Students will learn to create reusable code using subroutines, modules, and Perl's object-oriented architecture to deploy business logic across many programs and scripts to enhance maintainability and scalability.

Students will also learn to use the DBI Perl module to write programs that provide a consistent database interface independent of the actual database being used. Comprehensive hands on exercises will be completed throughout the course to reinforce

key concepts and practice debugging techniques. Students are shown how to extend Perl's basic functionality with packages and loadable modules.

Course Prerequisites

Prior scripting experience or knowledge of fundamental programming concepts.

Course Overview

Introduction to Perl

- Origin and Design Goals of Perl
- Overview of Perl Features
- Getting and Installing Perl
- Accessing Documentation via `perldoc`
- HTML-Format Reference Documentation
- Perl Strengths and Limitations

Getting Started With Perl

- Explicit Invocation of the Perl Interpreter
  - Running Perl on UNIX vs. Windows
  - Running Perl from the Command Line
  - Using Command Line Options
  - Using Debug Mode
- Implicit Invocation of the Perl Interpreter
- Running and Debugging Perl Scripts
- Simple and Compound Statements
- Fundamental Input Techniques
- Using the `print` Function to Generate Standard Output

Using Variables

- Scalar Variables
- Introduction to Standard Data Types
- Retrieving Standard Input Using the Default Variable `$_`
- Reserved Scalar Variables
- Assigning Strings and Numbers to Scalar Variables
- Declaring Constants for Persistent Values
- Using `strict` to Declare Variables

Pattern Matching in Perl

- Regular Expressions in Perl
- Using Pattern Matching Operators
- Altering Data with Substitutions in Regular Expressions
- Using Backreferences to Capture Data from Regular Expression Matching
- Global and Case-Insensitive Matches
- Altering Data with Character Translation
- Using Variables in Patterns

Operators

- Introduction to Fundamental Operators
- Operator Precedence and Associativity
- Using the Ternary Operator `?:` as a Shortcut for the if Statement
- Using `<FILEHANDLE>` and `<>` File I/O Operators for Standard Input/Output
- Using the Shortcut Operators `+=`, `-=`, `*=`, `/=`

String Manipulation

- String Comparison
- String Relations
- Concatenation
- Substring Manipulation
- Using `chomp` and `chop` to Eliminate EOL Characters
- Escape Characters for Formatting
- String Manipulation Functions

Flow Control: Conditional Statements and Looping

Subroutines and Parameters

- Conditional Expressions and Logical Operators
- `if/else/elsif` and `unless`
- Constructing switch/case Equivalent Expressions
- `while` Loops and `do` Loops
- `for` and `foreach` Loops
- Labels
- Altering Program Flow with `next`, `last`, and `redo`
- Trapping Errors with the `eval` Function
- Terminating a Script with `exit`

Arrays and Hashes

- Defining Numeric Index Arrays
- Defining Associative Arrays
- Sorting Arrays with the sort Function
- Adding and Deleting Items Using `push`, `pop`, `shift`, and `unshift`
- Using `slice`, `splice`, and `reverse`
- Other Array Manipulation Techniques
- Looping through an Array
- Merging Arrays
- Associative Array Manipulation Functions
- Introduction to Hashes
- Preallocating Memory to Optimize Hash Performance

File and Directory I/O

- Using `open` and `close`
- File Open Modes
- Reading Files into Arrays
- Retrieving File Metadata
- Built-in File Management Functions
- Using `print` and `write`
- File Test Operators
- Directory Manipulation Using `opendir`, `closedir`, `readdir`, `chdir`, `mkdir` and `rmdir`

Implementing Command Line Arguments

- Reading Command Line Arguments from `@ARGV`
- Read Files Explicitly with `<ARGV>` and Implicitly with `<>`
- Manipulating Positional Parameters with `push`, `pop`, `shift`
- Process Lists of Files
- Processing Command Line Options
- Simplifying Scripts with Subroutines
- Defining and Calling a Subroutine
- Passing Arguments by Value
- Passing Arguments by Reference
- Using `return` to Return a Value
- Controlling Variable Scope using `my` and `local` Keywords

Packages and Modules

- The Power of Packages and Modules
- Introduction to Standard Modules
- Where to Find Modules on the Internet
- Installing a Module on UNIX or Windows
- Creating Packages for Portability
- Using Packages to Create Isolated Namespaces and to Separate Code
- Creating Modules
- Creating and Using Symbols in a Module
- Using the Exporter to Export Symbols from a Perl Module

Input/Output Processing

- Parsing Input
- Using Standard Input, Standard Output, and Standard Error
- String and Field Processing
- Using Streams and Pipes
- Using `die` to Quit with an Error
- Redirecting Standard Output and Standard Error to a File
- Getting Standard Input from a File

Perl Report Formatting

- Defining Report Formats
- Justifying Text (Left, Right, Center)
- Using `write` to Generate Reports
- Defining `here` Documents for Report Customization
- Creating Report Headers
- Using Built-in Variables to Control Report Appearance
- Printing Line Numbers on a Report

with `getopt` or `getopts`
- Analyzing Command Line Argument Values with the `Getopt::Std` and `Getopt::Long` Modules
- Reserved Variables
- Manipulating Identifiable Options Using `GetOptions`
- Formatting Multi-Line Output
- Writing Formatted Text to a File

Debugging In Perl

- Using the Built-in Perl Debugger
- Starting the Debugger
- Debugger Command Syntax
- Checking for Script Syntax Errors
- Solving Compile-Time Errors
- Single-Stepping through a Script
- Executing to Breakpoints
- Setting Global Watches
- Printing Values of Variables
- Listing All Variables Used in the Script
- Using `Strict` Error Checking
- Quitting the Debugger

References

- Life Cycle of a Reference
- Hard References and Anonymous References
- Use of References to Create Complex Data Structures
- Creating Hard and Anonymous References
- Modifying References
- Dereferencing a Reference
- The Arrow Operator `->`
- Building Complex Data Structures with Multi-Dimensional Arrays and Hashes

Accessing a Database Using Perl DBI

- Database Access Life Cycle
- Using DBI and DBD to Connect to a Database
- Fundamental Data Storage and Retrieval Strategies
- DBI Query Syntax
- Using DBI Methods to Retrieve Database Information
  - Preparing Queries to be Executed
  - Creating Parameterized Queries
  - Executing Queries Using `execute` and `do`
- Fetching the Result Set to Achieve Workable Data in the Perl Script
  - Extracting Data Using an Array
  - Extracting Data Using a Hash
- Useful Utilities to Aid in Database Development
- Using Other Modules to Access Databases on the Web
- Extracting Data Using a Hash
- Displaying Results from Queries in a Report
- Releasing Database Resources

Perl Object Oriented Programming

- Object Oriented Programming Concepts
- Object Oriented Programming Terminology
- How Perl Implements Object Oriented Programming
- Modeling Software Objects Using Classes and Base Classes
- Creating Classes, Objects, Methods and Attributes
- Writing Constructors to Initialize of Objects
- Using `bless` to Turn References into Objects
- Creating Class Hierarchies through Inheritance

14 Fletcher Street
Chelmsford, MA 01824