Perl Programming / Scripting

(Duration: 64 Hours)

Introduction To The Perl Programming Language

- Determine your version of Perl
- Identify the default directories searched for Perl library files
- Create a command-line script that prints a simple message
- Create a simple script that prints a simple message
- Test a script's syntax without executing the script
- Using an eclipse IDE to run your Perl script

Perl Development Using Eclipse

- Features of the Perl Plugin
- Creating a Perl Project
- Running a Perl Program
- Preferences - EPIC
- Preferences - Editor
- Preferences - Content Assist
- Preferences - Folding
- Preferences - Mark Occurrences
- Preferences - Templates
- Templates
- Preferences - Source Formatter
- Preferences - Task Tags
- Accessing Perl Documentation
- Project Properties
- Testing Regular Expressions

Debugging

- 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
- Highlight updated variables
- Show addresses of variables
- Show Perl Internal Variables
- Show Global Variables
- Show Local Variables
- Quitting the Debugger

Scalars

- Define and describe numeric and string scalar data
- Create string and numeric scalar variables
- Modify scalar variables using operators
- Print scalar values using single and double quotes
- Remove \n from user input using the chomp command
- Calculate a value using scalar operators based on user input
- Define uses and operation of the Perl interpreter, including but not limited to: basic scripting, print function, variables.

Control Structures

- Print a message using an if statement
- Print a message using an if/else statement

- Print a message using a compound if/elsif/else statement
- Use a for loop to display a list numbers
- Use a foreach loop to perform calculations on a list of numbers
- Use while loops to repeatedly perform logical tests using an if/elsif/else statement
- Print formatted text with a Here document
- Exit from a loop using loop controls and statement modifiers
- Use a switch construct in Perl to print a message
- Use smart match operator

Arrays

- Create an array variable and assign scalar values to the array
- Determine the length of an array using an array operator
- Use array slices to assign new values to an array
- Determine the length of an array using a scalar variable
- Reverse the contents of an array
- Array as Stack
- Array as Queue
- Array as Linked List
- Sort an array in lexicographical order
- Sort an array of numbers in ascending and descending order
- Create an array from a scalar using split
- Process the values passed in the command-line array, @ARGV
- Read and write an array into a text file
- Use splice to dynamically add and remove elements anywhere
- Sorting arrays
- Writing array to a file

Hashes

- Print a hash using a foreach statement
- Access, add, and delete keys and values from a hash

- Print a hash using a while loop and the each function
- Determine the number of occurrences of a string in an array using a hash
- Determine if a key and value exist in a hash
- Writing and reading a hash to a file
- Removing (deleting) all elements of a hash at once
- The ENV hash
- Sorting keys in a hash
- Sorting values in a hash

Perl functions, strings and sorting

- Random number generation
- Formating data
- Formating currency
- Handling time
- The eval function
- Warn and die functions
- Packing strings
- Finding a substring with index
- Manipulating a substring with substr
- Formating data with sprintf
- Advanced sorting
- Sorting by multiple Keys
- Using Array slice, Hash slice
- More advanced error handling
- Picking items from a list with grep
- Transforming items from a list with map

Regular Expressions

- Test for a word or phrase in a file using regular expressions
- Using flip flop operator to extract a range of lines

- Use Unicode properties
- Use Metacharacters
- Use simple quantifiers
- Grouping in patterns
- Use anchors and character classes in regular expressions
- Character class shortcuts
- Negating the shortcuts
- Use alternation in regular expressions
- Matching with m//
- Case insensitive matching, matching any character, adding whitespace
- Combining option modifiers
- Use variable interpolation to define regular expressions
- Binding operator
- Extract parts of strings using regular expressions
- Use binding operator
- Use substitutions with s///
- Use transliterations with tr///
- Greedy and non greedy (lazy) matches
- Named subexpressions aka Named captures
- Positive Lookahead anchors
- Negative Lookahead anchors
- Positive Lookbehind anchors
- Negative Lookbehind anchors
- Matching multiple line texts
- Updating many files
- In-Place editing from the command line
- Writing your own pattern test program

Filehandles and Files

- Use the <> operator to read a file specified on the command line, line-by-line
- Use the printf command to format the output of the script

- Using the predefined file handles such as STDIN, STDOUT, STDERR, DATA, ARGV, ARGVOUT
- Passing filehandles to subroutines
- Use filehandles to open a file for read, write and append
- Use die to display an error message if there is an error accessing a file
- Append data to the end of a file
- Using select
- Reading character by character from a file
- Reading a binary file
- Use seek, tell, flock and eof
- Stripping or adding carriage returns from DOS to Unix and back again.
- Use a filehandle to read the output of a program
- Autodie

Subroutines

- Create script that uses the strict pragma
- Create subroutines that accept passed parameters and return desired results based on the values passed
- Include a subroutine that uses the my operator to create private variables
- Pass file handles to subroutines
- Argument handling
- Multiple arguments
- Named arguments
- Aliasing
- Passing a list, hash or hashref?
- Returning data
- Returning true/false
- Returning single and multiple values
- Wantarray
- Existing subroutines
- Anonymous subroutines
- Closures
- Prototypes

- Argument coercion
- More prototype tricks
- Mimicking built-ins
- Forward declarations
- Recursion
- Write a menu driven program to demonstrate the usage of subroutines and hashes

Packages and Modules

- Use a Perl library file in your script
- Use a Perl package in your script
- Create a Perl module and call it from a script
- Use packages and modules to organize, reuse and export program code.
- Use vs Require
- Version numbers
- Package variables
- Finding modules
- Installing modules
- Using only some functions from a module
- Subroutines in other packages
- Exporting
- Naming conventions
- BEGIN blocks
- END blocks
- INIT, CHECK, and UNITCHECK Blocks
- Reporting errors within modules
- Carp function
- Cluck function
- Croak function
- Confess function

File and Directory Operations

- Use file operators to determine the characteristics of a file
- Display the contents of a directory using chdir and globbing
- Display the contents of a directory using directory handles such as opendir, readdir, telldir, seekdir, rewinddir, chdir, mkdir and rmdir
- Rename and unlink files
- Create symbolic links to files
- Display all symbolic links in a directory
- Set file permissions for files based on their extensions

References and Complex Data Structures

- Introduction to references
- Use Scalar references
- Use Array references
- Use Hash references
- Passing by value and passing by reference
- Using anonymous arrays and hashes
- Array of hashes
- Hashes of arrays
- Anonymous array of anonymous hashes
- Array of anonymous hashes
- Hash of anonymous arrays
- Hash of anonymous hashes
- Hashes of hashes
- Array of anonymous arrays (retrieving columns)
- Array of anonymous arrays (retrieving rows)
- Using complex data structures
- Insert column to matrix
- Append column to matrix
- Insert row to matrix
- Append row to matrix

Testing

- Basic tests
- Using Test::More
- Write your own tests
- Understanding the prove utility
- Understanding Test::More test functions
- Using ok
- Using is
- Using like
- Using is_deeply
- Using SKIP
- Using TODO
- Using eval{}
- Using use_ok and require_ok
- Working with miscellaneous test functions
- Using other testing modules
- End testing on failure
- Using die_on_fail/restore_fail
- Using bail_on_fail
- Using Test::Differences
- Using Test::Exception
- Using Test::Warn
- Using Test::Most
- Using Test::Class
- A basic Test class
- Extending a test class
- Using test control methods
- Calling Parent test control methods

Code coverage

- Using Devel Cover for Code Coverage
- Statement coverage
- Branch coverage
- Path coverage
- Condition coverage
- Using Perl Critic
- Using Perl Tidy
- Profiling

POD and CPAN

- Creating POD
- Documentation structure
- Headings
- Paragraphs
- Lists
- Verbatim
- Miscellaneous
- Finding and evaluating modules
- Downloading CPAN modules
- Creating your own CPAN module
- Using PPM
- Documenting your module

Object Oriented Programming

- Introduction to Object Oriented Programming
- Introduction to method invocation arrow
- The extra parameter of method invocation
- @ISA
- Overriding the methods
- Super way of doing things

- Introducing bless
- Objects with data
- Invoking an instance
- Accessing instance data
- Accessing class data
- Inheriting the constructor
- Making a method work with either classes or instances
- Getters and setters
- Object destruction
- Cleaning up
- Nested object destruction
- Using class variables
- UNIVERSAL methods
- AUTOLOAD as a last resort
- Creating private subroutines

Report Writing

- The Template
- Steps in defining the template
- Changing the file handle
- Top of page formatting
- Formating text (Left, Centered, Right, Numeric, Decimal Point)
- The select function
- Multli line fields
- Filling fields
- Dynamic report writing

Common Tasks

- Using CSV data
- Reading CSV data

- Writing CSV data
- Reading data from an Excel file
- Writing data to an Excel file
- Handling dates
- Using DateTime module
- Using Date::Tiny
- Using DateTime::Tiny
- File::Find
- File::Path
- File::Find::Rule
- Using Log::Log4Perl to record your program state

Database Operations

- Introduction to database
- Architecture of the DBI application
- Connect to the database using DBI
- Using DBD::mysql
- Perform basic operations such as create, select, update, insert and delete
- Read data using array
- Read data using array reference
- Read data using hash reference
- Using Bind parameters
- Use commit
- Define database programming, including but not limited to: use of modules and SQL to access external data.
- Handling errors
- Disconnect from the database

XML

- Difference between HTML and XML
- XML Basics

- XML elements vs attributes
- XML Schemas and DTD
- Use XML::Simple
- Working with DOM (Document Object Model)
- Working with SAX (Simple API for XML)
- Reading an XML document using XML::Simple
- Converting specific XML elements into upper case/lower case
- Navigating the DOM tree and count the number of elements
- Read a specific attribute from a DOM tree
- Using a SAX parser to read and manipulate a XML document
- Check whether an XML is well formed or not
- Converting data structures into XML

CGI Programming

- An introduction to CGI and its environment
- Understand how the server and browser communicate
- Send a Hypertext Markup Language (HTML) page to a browser using a Common Gateway Interface (CGI) script
- Decoding data sent to your CGI programs
- Using environment variables in your programs
- Use a Here document in a CGI script to send an HTML page to a browser
- Read an HTML form using the GET or POST method
- Read an HTML form with radiobuttons and checkboxes
- Read an HTML form using textarea and drop down menu
- Use CGI to display date and time on a web page
- Use CGI to display hit counter to a web page
- Write a discount calculator app for a product
- Write a simple music download app using password authentication
- Read HTML5 email, URL and phone controls
- Read HTML5 number and range controls
- Read HTML5 date field
- Validating mobile phone fields in a HTML5 form using regular expression

- Validating and submitting an HTML5 enquiry form using regular expression
- Storing HTML5 enquiry form data in a flat file
- Storing HTML5 feedback form data in a flat file
- Storing HTML5 feedback form data in a database

Process Management

- System functions
- Exec functions
- Back quotes
- Fork
- Signals

Multithreading

- What is a thread anyway?
- Threaded program models
- Boss/Worker
- Work Crew
- Pipeline
- Native Threads
- Thread basics
- Basic Thread Support
- Creating Threads
- Giving up control
- Waiting for a thread to exit
- Errors in threads
- Ignoring a thread
- Threads and data
- Shared and unshared data
- Thread pitfalls : Races
- Controlling access : lock()

- Thread pitfalls : Deadlocks
- Queues : Passing data around
- Threads and code
- Semaphores : Synchronizing data access
- Attributes : Restricting access to subroutines
- Subroutines locks
- Methods
- Locking a subroutine
- What thread am I in?
- Thread IDs
- Are these threads are same?
- What threads are running?

Sockets

- What is a socket?
- Socket programming in Perl
- Types of sockets
- Create a socket
- Connect to remote server
- Send data
- Receive reply
- Understand the steps for making a server
- Asynchronous IO

Mini Projects

- Mini project 1
- Mini project 2
- Mini project 3
- Mini project 4
- Mini project 5

*Kind Note : You will be working on 1 mini project out of 5 based on your choice