Perl Training Notes

**TODO**

1. Perl Installation Directory Structure
2. Arrange the table below to include all the perl scripts

|  |  |
| --- | --- |
| Perl Scripts Notes | |
| 001-HelloWorld.pl | * Explain the first line in perl – Shebang * Explain similarities with C – no main, same indentation like C, change the variable names, put braces, # instead of ! as comments, remove the datatypes and a C program becomes a perl program |
| 017-IO-ConsoleInput.pl | * Explain STDIN stream * Explain chomp * Explain the interpretation of $ symbol inside quotes |
| 002-Variables.pl | * Explain integers, float, octal, hexadecimal and string variable types * Discuss scalars in detail during Perl Data Structure |
| 003-StatementBlock.pl | * Explain the block and its scope * Explain the last semi-colon is optional |
| 004-ControlStructure-IfElse.pl | * Explain if * Explain if else * Explain if-elseif-else * Explain Nesting of if * Explain that braces are always mandatory * Explain the case when 2 else blocks matches |
| 005-ControlStructure-Unless.pl | * Only else part * Unless gets executed when the condition is false. |
| 006-LoopStructure-For.pl | * Explain that braces are mandatory |
| 007-LoopStructure-While.pl |  |
| 008-LoopStructure-Until.pl |  |
| 009-LoopStructure-DoWhile.pl |  |
| 010-LoopStructure-DoUntil.pl |  |
| 011-LoopStructure-Foreach.pl | * Explain an array example * Explain foreach with numbers * Explain foreach with strings * Explain with reverse (arrays) and show that originals are restored. * Explain $\_ as the special variable * Explain that if variables are changed then loop gets affected. |
| 012-ComplexForEachExample.pl | * One typical example of foreach. Give as a puzzle. |
| 013-CommandLineArguments.pl | * Explain the special array variable @ARGV * Explain $#ARGV to check the number of arguments * How to access individual array element like $ARGV[0] |
| 014-Functions.pl | * How to define a function * How to return a value * How to return multiple values * How to collect multiple values * How to invoke a function * How to invoke a function from an expression * How to invoke a subroutine with arguments * How to collect arguments inside the function * How to create private variables inside a function using my * Explain semi-private variables - local |
| 015-Operators-Scalars.pl | * Explain all mathematical operators |
| 016-Operators-Strings.pl | * Explain concatenation, comparison and repetition operators |
| 018-IO-STDIN.pl | * Explain $a = <STDIN> and @a = <STDIN> * Explain the termination is by Ctrl-Z * Explain while (defined($line\_ = <STDIN>)) loop * Explain while (<STDIN>) * Explain while (<>) |
| 019-IO-STDOUT.pl | * Explain $a = print("hello ", "world", "\n"); * Explain print (2+3),"hello\n"; * Explain print ((2+3),"hello\n"); * Explain print 2+3,"hello\n"; * Explain printf "%15s %5d %10.2f\n", $string, $number, $real; |
| 021-FileIO-Writing.pl | * Explain file modes: overwrite and append * Explain 3 ways of closing the file–termination, reopening, close call |
| 020-FileIO-Reading.pl | * How to handle error |
| 022-FileIO-Appending.pl | * How to append to a file |
| 023-FileIO-FileTests.pl | * Explain the file tests from the slide |
| 024-FileIO-ReadingDirectory.pl | * Opening the directory by changing paths * Reading the files in array context * Reading the files in scalar context |
| 025-FileIO-ManipulatingDirectory.pl | * Explain mkdir * Explain rmdir |
| 026-FileIO-FormattedOutput.pl | * It needs 026-Addresses * Just explain the program as an introduction |
| 100-Library-MathModule.pl | * How to use built-in modules |
| 029-DS-Scalars.pl | * Integer, Float, Octal and Hexadecimal |
| 030-DS-Scalars-Strings.pl | * Explain possible ways of creating a string * Explain variable interpolations |
| 028-DS-NamingRules.pl | * Explain the rules to name variables. |
| 031-DS-Scalars-BuiltInFunctions.pl | * Chop – Removes last character of the string and return that char * Chomp – Removes last new line character and return it * Chomp is the safer version * Chomp & chop does nothing to an empty string * Chr(65) * Index (string, substring, position) * Add mores… * Extrapolation of scalars |
| 032-DS-Lists-Creation.pl | * Define List and Arrays * Explain Scalar context and list context * Explain that different names are in different namespace * Explain all the examples from the PPT |
| 033-DS-Lists-Manipulation.pl | * Explain all the examples in the PPT |
| 034-DS-Lists-Accession.pl | * Explain all the examples in the PPT |
| 035-DS-Lists-BuiltInFunctions.pl | * Explain all the examples in the PPT |
| 036-DS-Lists-ListFromUserInput.pl | * Explain how to take input from console |
| 037-DS-Lists-Interpolation.pl | * “$array1[0]” * $array1[$index-2] * @array1 * @array1[2,3] * $array1\[1] |
| 038-DS-Hash-Creation.pl | * Explain Hashes creation with numbers and strings * Convert arrays to hash * Convert hashes to arrays * Convert hash to another hash * Explain reverse |
| 039-DS-Hash-BuiltInFunctions.pl | * Explain keys * Explain values * Explain each functions * Explain delete functions * Explain how to merge two hashes into one |
| 040-DS-HashOfArrays.pl | * Explain how to create an hash of an arrays |
| 041-RegEx-Search.pl  043-RegEx-Substitution.pl  044-RegEx-Patterns.pl | * Explain how to search a string * Explain $\_ * Explain how to use a variable instead of $\_ ($string1 =~ /abc/ ) * Replace the pattern with a variable * Replace the pattern with a part variable * Explain the searching by ignoring case |
| 042-RegEx-SearchInFile.pl | * Explain the example of searching in a file * Explain few examples on the run |
| Day 3 | |
| 054-Debug-TracingSteps.pl | * How to explore a debugger using perl -d -e 0 * Starts the debugger and waits at the first executable statement * @order = ( 1, 'cherry', 3.14 ) * Gives error when given wrong perl command $y = * Check the variable content using x @order   %hash = ( one => 1, two => 2, three => 3 )  x %hash  $pi = 'apple'  x $pi  $n = 3; $m = 4; $o = 6  x @order, %hash  p @order, %hash  V main order hash  @pi = (1,2,3)  V main pi  V main  x $vikash (Some undefined variables)   * Now run the debugger with the 053-Debug-SampleCode.pl   x $x  s  l 1-5  b 4 |