

# CS100: Introduction to Computer Science

Lecture 14: Programming Language: Perl (I)

## Review: Images

- n Image in HTML
  - q Background of your web page
  - q Insert images
  - q Make a hyperlink of an image
- n Image representations & compression
  - q GIF and JPEG
  - q Pixels, image size, file size, image resolution

## Review: Editing images with Photoshop

- n Crop the image
- n Resize the image
- n Save files as GIF or JPEG
- n Create images using Photoshop elements
- n Change the background of a picture
  - q <http://www.mtholyoke.edu/~xli>
- n Deal with imperfections in images

## Programming Languages

- n **Programming languages** are used to facilitate communication about the task of organizing and manipulating information, and to express algorithms precisely.
- n Many programming languages exist
  - q Basic, Fortran, C, C++, JAVA, Perl
- n Computers actually speak only one language, machine language.

## Creating and Executing Programs

- n Most programs are not written in machine language.
  - q General editor for creating a program
  - q ".pl" for a program in perl
  - q ".c", for a program in C, ".cpp" in C++
- n A **compiler** is used to translate a program from one programming language into machine language.
- n The resulting program (the object code) then is ready to run on your computer.
  - q ".exe" for "executable" in the Windows operating systems

## What is an Interpreter?

- n An **Interpreter** is a *program* that implements or simulates a *virtual machine* using the base set of instructions of a *programming language* as its *machine language*.
- n You can also think of an **Interpreter** as a *program* that implements a library containing the implementation of the basic instruction set of a *programming language* in *machine language*.
- n An **Interpreter** reads the statements of a program, analyzes them and then executes them on the virtual machine or calls the corresponding instructions of the library

## Compiler versus Interpreter

- n Using a *compiler* separates *translation* and *execution* of a program. The source code is translated only once.
- n The *compiled* program is machine-dependent, it can only be executed on a machine for which it has been compiled, whereas an *interpreted* program is not *machine-dependent*
- n An interpreted program runs much slower than if it had been compiled. But it can take less time to interpret it than the total time required to compile and run it.

## Scripting Languages: Perl

- n **Scripting programming languages** or **script languages** are computer programming languages that are typically interpreted and can be typed directly from a keyboard
- n A Perl program is a text file containing a list of perl commands. (Perl script)
- n Then we must start the Perl interpreter and tell it to run the Perl script.
  - q On Unix, type **perl myperl.pl**
  - q On Windows, double clicking the script file.

## A Simple Perl Program

```
#!/usr/local/bin/perl  
print "This is My First Perl Program!\n";
```

- Line 1 tells the script where to look for perl Interpreter.
- Line 2 is the program. print is a perl command to output text.
  - Note that: The string outputted by print is in double quotes;
  - Every perl command is terminated with a semi-colon;
  - Leading white space is ignored;

## Basic Concepts: Variables and Values

- n Each variable acts as named storage location where a value is stored.
- n Legal names for variables in Perl
  - q Begin with \$ (for scalar variables that can hold only one value)
  - q Combinations of letters, numbers and a few characters including hyphen (-) and underscore (\_) characters.
  - q Variables are case sensitive.
- n The value stored in a variable can be retrieved and changed at any time, and we can perform operations on it.

## Data Typing

- n A variable can contain more than just numbers
  - q An integer number: 7, 100, or 98
  - q A real number: 28.96
  - q A character string: "Hello world"
- n In many languages, you must declare what kinds of values a variable can contain before using it.
  - q C, C++,
- n In Perl, a variable can contain any data at any time. But decide
  - q whether the variable can hold one or more values
  - q Scalar variables, arrays, hashes

## Examples: a Perl script uses a single variable to store several types of data (assignment operator "=")

The program

```
$my_variable = 7;  
print ("my variable contains the value: $my_variable \n");  
$my_variable = 7.3217;  
print ("my variable contains the value: $my_variable \n");  
$my_variable = "apple";  
print ("my variable contains the value: $my_variable \n");
```

The output

```
my variable contains the value: 7  
my variable contains the value: 7.3217  
my variable contains the value: apple
```

**Arithmetic Operators:** used to execute general arithmetic procedures

Operator	Example	Result	Definition
+	7 + 7	= 14	Addition
-	7 - 7	= 0	Subtraction
*	7 * 7	= 49	Multiplication
/	7 / 7	= 1	Division
**	7 ** 7	= 823543	Exponents
%	7 % 7	= 0	Modulus

**Assignment Operators:** perform an arithmetic operation and then assign the value to the existing variable.

Operator	Definition	Example
+=	Addition	(\$x += 10)
-=	Subtraction	(\$x -= 10)
*=	Multiplication	(\$x *= 10)
/=	Division	(\$x /= 10)
%=	Modulus	(\$x %= 10)
**=	Exponent	(\$x **= 10)

**Example:**

```
$a = 2;
$b = 6;
$c = $a + $b;
print "The value of c is $c \n";
$d = $b % $a;
print "The value of d is $d \n";
$e = $a % $b;
print "The value of e is $e \n";
```

**Working with strings**

- n Create a string using double quotes
- n Any variables inside the string are replaced with their contents

```
$first_name = "John";
$middle_initial = "Q.";
$last_name = "Programmer";

$full_name = "$first_name $middle_name $last_name";
```

**The String Operators (. and x)**

- n the concatenation (.) operator  
\$first\_name = "David";  
\$last\_name = "Marshall";  
\$full\_name = \$first\_name . " " . \$last\_name;
- n the repetition (x) operator  
\$first\_name = "David";  
\$david\_cubed = \$first\_name x 3;

**The chop() function**

- n Removes the last character of a given string and return the last character.
- n For example

```
$string = 'frog';
$chr = chop($string);
print "String: $string\n";
print "Char: $chr\n";
```

## Problems with strings

- n Working with variables and strings: append the letter y to a string variable

```
$name = "Mike";  
$nickname = "$name y";
```

```
$name = "Mike";  
$nickname = "$namey";
```

```
$name = "Mike";  
$nickname = "${name}y";
```

## Problems with strings

- n Print the following string to the screen, exactly as it appears below  
**His name is Mike, but people call him "Mikey".**
- n How to include double quotes in the string?
  - q **Escape sequences:** special character sequence initiated with a backslash.
  - q `\n, \t, \', \'`

```
print ("His name is Mike, but people call him \"Mikey\".\n");
```

## Conversion between numbers and strings

```
$a = 2;  
$b = 6;  
$c = $a + $b;  
print "The value of c is $c\n";  
$c = $a . $b;  
print "The value of c is $c\n";  
$d = $c / 2;  
print "The value of d is $d\n";
```

**Note** if a string contains any trailing non-number characters they will be ignored.  
i.e. "123.45abc" would get converted to 123.45 for numeric work.  
If no number is present in a string it is converted to 0.

## Input from keyboard

```
#!/usr/local/bin/perl  
print "Please enter your name\n";  
#accept keyboard input, read it into $name  
$name=<STDIN>;  
print "Hello $name!\n";
```

- n Line 3 is a comment
- n Line 4 accept keyboard input, read it into \$name. STDIN is a buffer that contains the characters inputted by the keyboard
- n Line 5 prints a greeting and the name inputted by the user (contained in \$name).

## Exercises 1:

- n Write a program that display the following shape on the screen.

```
#  
##  
###  
####  
#####
```

## Exercise 2:

- n Write a program that will take three numbers and output the sum of them.

Exercise 3: what is the output of the following program?

```
$x = "40";  
$y = "11";  
$z = $x + $y;  
print "the value of z is $z \n";  
$w = $x . $y;  
print "the value of w is $w \n";
```

Announcements:

- n Next lecture: perl (II)
- n Lab 3 this week. Write codes in perl.