Programming in Perl
Lecture 1 Examples

A full list of these BACKSLASH ESCAPEs may be found in *Learning Perl*, p. 34. Some examples:

```
"\a"     Alarm(Bell)
"\n"     Newline
"\t"     Tab

    Various Perl escape sequences can alter the case of subsequent
strings.

"\l"  The following character is converted to lowercase
"\u"  The following character is converted to uppercase
"\L"  All subsequent characters are converted to lowercase
      up to a \E character
"\U"  All subsequent characters are converted to uppercase
      up to a \E character
"\Q"  All non-alphanumeric characters are prepended with a backslash
"\E"  Used with \L, \U, and \Q
```

Precedence of operators discussed so far (full table in *Learning Perl*, pp. 38-39):

```
Assoc         Operator
-----         --------
Right         **       ( 2 ** 3 ** 2  # is 2 ** (3 ** 2) = 256)
Right         + - (unary)
Left          * / % x (multiplication)
Left          + - . (addition)
```

```
#!/usr/interp/perl -w
# PERL STRING INCREMENT
# By Darren Lim 8/30/97

# Description: This program is a demonstration of Perl string increments.
# Procedure: Various strings are incremented via the AutoIncrement operator.
# Each one is then printed to the screen with various results.
# Input: None
# Output: The various contents of incremented strings.


  $str1 = "Hello World";
  print ++$str1, "\n";          #CanCan not handle spaces; prints 1<newline>
  $str2 = "HelloWorld";
  print ++$str2, "\n";          #printsprints HelloWorle<newline>
  $str3 = "zz9";
  print ++$str3, "\n";          #printsprints aaa0
  $str4 = "aZ";
  print ++$str4, "\n";          #printsprints bA
```

```
#!/usr/interp/perl -w

# PERL VARIABLE INTERPOLATION EXAMPLE
# By Louis Ziantz

# Description: This program is a demonstration of scalar variable
# interpolation in Perl.
# Procedure: A number of variables are used inside double-quoted
# strings, and the results are printed.
# Input: None
# Output: Various messages using interpolated variables.

$num = 5;
$friend = "John Smith\n";
$month = "Octo";
print "$num\n"; # prints 5<newline>
print $friend;  # prints John Smith<newline>
print "Can I have \$$num?\n";
print "The month is $monthber.\n";
print "The month is ${month}ber.\n";
print 'The month is ${month}ber.\n';
```

Output:

```
% var.interp.pl
Identifier "main::test" used only once: possible typo at ./var.interp.pl line 19.
Identifier "main::monthber" used only once: possible typo at ./var.interp.pl line 10.
5
John Smith
Can I have $5?
Use of uninitialized value at ./var.interp.pl line 10.
The month is .
The month is October.
The month is ${month}ber.\n
```

```
#!/usr/interp/perl -w

# PERL EXAMPLE OF CASE SHIFT ESCAPES
# By Louis Ziantz

# Description: This program is a demonstration the use of
# case shift backslash escape character sequences.
# Procedure: Case shift escapes are applied to a number of
# strings, and the results are printed.
# Input: None
# Output: Various strings that have their case modified.

$fred = "fred";
$bigfred = "\U$fred";
print "$bigfred\n";
# prints FRED<newline>
$bigbarney = "BARNEY";
$barney = "\L$bigbarney";
$capbarney = "\u\L$bigbarney"; # or "\u$barney"
print "$barney $capbarney $bigbarney\n";
# prints barney Barney BARNEY<newline>
```

```
#!/usr/interp/perl -w

# PERL READING SCALARS FROM STDIN EXAMPLE
# By Louis Ziantz

# Description: This program is a demonstration reading scalar
# variables from STDIN.
# Procedure: Two values are read from STDIN and they are echoed.
# Input: Name (string) and age (number).
# Output: Input is echo printed.

print "Enter first name: ";
$name = <STDIN>;
print "Enter age: ";
$age = <STDIN>;
print "$name is your name and you are $age years old.\n";
```

Output:

```
% stdin.pl
Enter first name: Bart
Enter age: 10
Bart
 is your name and you are 10
 years old.
```

```
#!/usr/interp/perl -w

# METHODS OF ACCESSING LAST ARRAY ELEMENT IN PERL
# By Louis Ziantz

# Description: This program is a demonstration different ways
# to access the last element of an array.
# Procedure: The last element of an array is accessed using
# three different methods.
# Input: None
# Output: The size and last element of the an array.

@strings = ("first", "second", "last");

$size = @strings;          # returns size of array

print("There are $size elements in the \@strings array.\n");
print("The last element of the array is $strings[$size - 1].\n");
print("The last element of the array is $strings[$#strings].\n");
print("The last element of the array is $strings[-1].\n");
```

Output:

```
% ar.elem.pl
There are 3 elements in the @strings array.
The last element of the array is last.
The last element of the array is last.
The last element of the array is last.
```

Push vs. Unshift:

```
@list = (1, 2, 3);
push(@list, 4, 5);
# produces the same list as
push(@list, 4);
push(@list, 5);
# either way yields (1, 2, 3, 4, 5)
#
@list = (1, 2, 3);
unshift(@list, 4, 5);
# produces (4, 5, 1, 2, 3)
# while
unshift(@list, 4);
unshift(@list, 5);
# produces (5, 4, 1, 2, 3)
```