

# *Perl Training*

Use of Strings and Print



# Agenda

- Print format and usage
  - Difference between single and double quotes
  - Special Characters
  - Quoting Functions
  - Here Documents
  - Putting two strings together
  - String Manipulation
  - String subtraction
  - String Transformations
  - Getting information about a string
  - Homework!!!
- 
-

# *Knowledge Assumptions*

- You have access to a perl installation  
(usually /usr/bin/perl or /usr/local/bin/perl)
  - All text after a # to the end of the line is a comment
  - Variables in Perl are created by  
    `$var_name = value;`
- 
-

# *Print format and usage*

## Basic Syntax:

```
print "This is my string\n";
```

print tells Perl to print something  
\n is a new line feed



# *Print format and usage (cont.)*

Hello World Program:

```
#!/usr/bin/perl
```

```
print "Hello World\n";
```

---

---

# *Difference between Single and Double Quotes*

- Single quotation marks do not interpret, but double quotation marks do

```
$tmp = "World";
```

```
print 'Hello $tmp\n';           # prints Hello $tmp  
print "Hello $tmp\n";         # prints Hello World
```

# Special Characters

- Some characters can't be typed in regular text

|                 |                     |
|-----------------|---------------------|
| <code>\n</code> | New Line            |
| <code>\r</code> | Carriage Return     |
| <code>\t</code> | Tab Character       |
| <code>\f</code> | Formfeed character  |
| <code>\b</code> | Backspace character |
| <code>\v</code> | Vertical Tab        |
| <code>\a</code> | Bell or beep        |
| <code>\e</code> | Escape character    |

`\n` and `\t` are used frequently. These may not have the same functionality on all systems.

---

---

# *Special Characters (cont.)*

- Examples

```
print "This is line one\n This is line two\n";  
print "This is line one\n\t This line is indented\n";  
print "How do you print \\\"s?\n";  
print "The total is \$5.73 please\n";
```

```
$shout = "Help Me";  
print "And then he shouted \"$shout\" very loudly\n";
```

# Functions for Quoting Text

- The `q//` function quotes with single quotes, the `qq//` function quotes with double quotes

```
$tmp = 'This isn\'t a Java class, I\'m sure.';
```

is the same as

```
$tmp = q/This isn't a Java class, I'm sure/;
```

Isn't that easier?

Note: `qx//` is closely related. It interprets any variables, then goes to the system and executes the command, returning the resulting text

So `$tmp = qx/whoami/; print "$tmp\n";` would print who you're logged in as

---

---

# Here Documents

- If you have a lot to say, it can get monotonous to say the following:

```
print "This is the first line\n";  
print "This is the second line\n";  
[...]  
print "This is the 400th line\n";
```

- Instead, use a Here document (called a here document based on old UNIX terminology):

```
print <<'EOL';  
This is the first line  
This is the second line  
[...]  
This is the 400th line  
EOL
```

Note: If you put tabs into the Here document, they're printed as tabs automatically

---

---

# Putting Strings together

- Concatenation operator is “.”

```
$tmp = “Hello”;  
$tmp2 = “World”;  
$tmp3 = $tmp . $tmp2;  
print “$tmp3\n”;
```

Why does this print out “HelloWorld” instead of “Hello  
World”?



# *Putting Strings together (cont.)*

- We forgot to add a space

```
$tmp = "Hello";  
$tmp2 = "World";  
$tmp3 = $tmp . " " . $tmp2 . "\n";  
print "$tmp3\n";
```

# *String Multiplication*

- Ever wanted to print the string “ha ha ha ha ha ha ha ha ha ha”?

```
$tmp = “ha”;  
print “$tmp”x10;  
print “\n”;
```

# String Subtraction

- Used to get rid of data at the end of the line (line feeds or characters)

```
$tmp = "Hellox";  
$value = chop($tmp);  
print "$tmp\n";           # prints "Hello"  
print "$value\n";        # prints "x" (the character removed from the  
                           string)
```

To remove line feeds, it is easier/better to use `chomp` because `chomp` knows how many characters to remove (some computers have a two-character line ending, some have one)

`Chomp` has the same syntax as `chop` (but only removes line feeds, not characters)

# String Transformations

- lc, uc, lcfirst and ucfirst functions

```
$name = "jAsOn";  
print lc($name);           # prints "jason"  
print uc($name);          # prints "JASON"  
print lcfirst($name);     # prints "jAsOn"  
print ucfirst($name);     # prints "JAsOn"
```

# Getting information from strings

- length

- `print length ("Hello World");` # prints 11

- Chr

- `print chr(35);` # prints the “#” symbol
  - `print chr(65);` # prints “A”

- ord

- `print ord('A');` # prints 65
  - `print ord('#');` # prints 35



# Homework Problems #1 & #2

- Create a program that will print out the current login id's of all current users of a system
  - Print the following strings:
    - The ' quotes are easy as the “ to use
    - \$1,000,000 is a lot of money
    - Countdown: 10... 9... 8... 7... 6... 5... 4... 3... 2... 1...  
Blastoff!!!
- 
-

## *Homework Problems #3, #4, #5*

- Print the following string backwards
    - I like perl
  - Print the length of 5 strings that you make up
  - Print a HTML document that says Hello World  
(Use a HERE document)
- 
-

# Bonus Homework Problem

- Create a program that displays the character tables (numbers 33-126)
  - Use the form 33=! 34=" 35=#
  - Create another table that does the opposite
    - != 33 " = 34 # = 35
  - For loop construct is as follows:

```
for($i = 0; $i <= 10; $i++) {  
    print "$i is $i\n";  
}
```

*Questions?*

