Day 2: Basic Syntax

Suggested reading:
Learning Perl (4th Ed.)
Chapter 1: Introduction
Chapter 2: Scalar Data
Turn In Homework
Housekeeping

- If you haven’t enrolled:
  - please consider enrolling or auditing
  - you may attend regardless
  - I cannot provide help (homework, office hours, …)
  - I can add you to the mailing list (email me)

- CSL accounts
  - old accounts may still be active
  - otherwise, see login screen on instructional machine
  - problems? stop by CS 2350 (the CSL)
    or email lab@cs.wisc.edu
Office Hours

Wednesdays, 10–11 a.m.
Thursdays, 2–3 p.m.

Computer Sciences 4265

Still best to email first
Write code.
At least a little.
Every day.
Have fun!
Basic Perl Syntax
(in 12 13 slides)
Hello World

#!/usr/bin/perl

use strict;
use warnings;

# Everyone's first Perl program
print "Hello, world!\n";
Numbers & Math

- literals: 42, 3.141, –6.5e9, 0377, 0xff
- operators: +  –  *  /  **  %  (  )

4 + 7  => 11
17.8 – 3.5  => 14.3
16 * 0x10  => 256
2 ** 8  => 256
10 / 3  => 3.333333...
10 % 3  => 1
(2 + 3) * 4  => 20
Strings

- literals: '...' and "...
- escapes: \n, \t, \x7f, \, \"
- operator: .

'foo\tbar'    => foo\tbar [literally]
"foo\tbar"    => foo bar
'foo' . "\n"    => foo\n [with newline]

'Don't let "quotes" confuse you!'
"Don't let "quotes\" confuse you!"

Don't let "quotes" confuse you!
Time to Write Code!

*Simple °F $\Rightarrow$ °C Conversion*
Simple Variables

• prefix with $
• on first use, declare with my
• assignment: = [ += -= etc.]
• increment/decrement: ++ --

my $name = 'Tim';
my $counter = 0;
my $odd_value_1 = $counter + 7;
$counter += 2;
$name .= ' Cartwright';
$counter++;
print() Revisited

- writes to standard output by default
- variable interpolation in "...
- with or without ()

```perl
my $name = 'Tim';
print $name;
print "Hello, $name!\n";
print($name . " is teaching\n");
```
Basic Input

• terminal input is a bit tricky…
• for today’s homework, just use this:

```perl
chomp(my $user_input = <STDIN>);
```
More Coding!

*Take input and use variables*
Conditionals

- `if (boolean expression) {...} elsif (boolean) {...} else {...}
- `unless
- `{ and } are required (unlike C)

```
if (boolean expression) {
    print "Condition is true!\n"
} else {
    print "Condition is false.\n"
}
```

```
unless (boolean expression) {
    print "Condition is false.\n"
}
```
## Comparisons

<table>
<thead>
<tr>
<th>numeric</th>
<th>==</th>
<th>!=</th>
<th>&lt;</th>
<th>&gt;</th>
<th>&lt;=</th>
<th>&gt;=</th>
</tr>
</thead>
<tbody>
<tr>
<td>string</td>
<td>eq</td>
<td>ne</td>
<td>lt</td>
<td>gt</td>
<td>le</td>
<td>ge</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Expression</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>2  ==  2</td>
<td>true</td>
</tr>
<tr>
<td>2  !=  2</td>
<td>false</td>
</tr>
<tr>
<td>1 + 1 &lt; 3</td>
<td>true</td>
</tr>
<tr>
<td>2  &lt;=  2.0</td>
<td>true</td>
</tr>
<tr>
<td>'yes' ne 'no'</td>
<td>true</td>
</tr>
<tr>
<td>'2' eq '2.0'</td>
<td>false</td>
</tr>
<tr>
<td>'Tim' gt 'Nick'</td>
<td>true   [ha!]</td>
</tr>
<tr>
<td>'Tim' gt 'nick'</td>
<td>false</td>
</tr>
</tbody>
</table>
Boolean-Like Values

- **undef**: "not defined"; like `null/nil`
- **false**: 0, '', empty array/hash, `undef`
- **true**: everything else

```perl
my $foo;
defined($foo)    #=> ""  [i.e., empty string]
defined(undef)   #=> ""
defined('')      #=> 1
defined(0)       #=> 1

2 == 2           #=> 1
2 != 2           #=> ""
```
Conversions

- Perl converts values to fit the environment
  - operators require specific types
  - boolean interpretation

<table>
<thead>
<tr>
<th>Expression</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;6&quot; * '5'</td>
<td>30</td>
</tr>
<tr>
<td>6 . 5</td>
<td>'65'</td>
</tr>
<tr>
<td>'2' == '2.0'</td>
<td>true</td>
</tr>
<tr>
<td>'2' eq '2.0'</td>
<td>false</td>
</tr>
<tr>
<td>'2' eq 2.0</td>
<td>true</td>
</tr>
</tbody>
</table>
Hey, Let’s Code!

Add conditional conversions
Basic Loops

- `while (bool) {...} until (bool) {...}
- `for (init; condition; change) {...}
- `next, last

```perl
my $counter = 10;
while ($counter > 0) {
    print "$counter...
"
    $counter--;
}

my $sum = 0;
for (my $i = 1; $i <= 10; $i++) {
    $sum += $i;
}
```
Statement Modifiers

• Modify statement using condition or loop
  – if, unless
  – while, until, foreach
• () around condition optional
• Tim says: “Use **only** when clear and natural!”

```perl
$parameter = $MAX if $parameter > $MAX;
die("Invalid data.") if $no_input_found;
print($i-- . "\n") while $i > 0;
```
### Operators & Precedence

<table>
<thead>
<tr>
<th>( )</th>
<th>( )</th>
</tr>
</thead>
<tbody>
<tr>
<td>++</td>
<td>++</td>
</tr>
<tr>
<td>--</td>
<td>--</td>
</tr>
<tr>
<td>**</td>
<td>**</td>
</tr>
<tr>
<td>!</td>
<td>!</td>
</tr>
<tr>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>.</td>
<td>.</td>
</tr>
<tr>
<td>&lt;&lt;&lt;</td>
<td>&lt;&lt;&lt;</td>
</tr>
<tr>
<td>&gt;&gt;&gt;</td>
<td>&gt;&gt;&gt;</td>
</tr>
<tr>
<td>&lt;</td>
<td>&lt;</td>
</tr>
<tr>
<td>&lt;=</td>
<td>&lt;=</td>
</tr>
<tr>
<td>&gt;</td>
<td>&gt;</td>
</tr>
<tr>
<td>&gt;=</td>
<td>&gt;=</td>
</tr>
<tr>
<td>==</td>
<td>==</td>
</tr>
<tr>
<td>!=</td>
<td>!=</td>
</tr>
<tr>
<td>eq</td>
<td>eq</td>
</tr>
<tr>
<td>ne</td>
<td>ne</td>
</tr>
<tr>
<td>&amp;</td>
<td>&amp;</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>^</td>
<td>^</td>
</tr>
<tr>
<td>&amp;&amp;</td>
<td>&amp;&amp;</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>? :</td>
<td>? :</td>
</tr>
<tr>
<td>=</td>
<td>=</td>
</tr>
<tr>
<td>+=</td>
<td>+=</td>
</tr>
<tr>
<td>-=</td>
<td>-=</td>
</tr>
<tr>
<td>…=</td>
<td>…=</td>
</tr>
<tr>
<td>not</td>
<td>not</td>
</tr>
<tr>
<td>and</td>
<td>and</td>
</tr>
<tr>
<td>or</td>
<td>or</td>
</tr>
<tr>
<td>xor</td>
<td>xor</td>
</tr>
</tbody>
</table>

```perl
my $x = 6 * 3 - 2 & 0xF ? '' : 'b';
$x ||= 'c';
```
You Made It!
Other Scripting Languages

• The cellphone metaphor...

• Check for different or additional:
  – **Literals** (true/false, null-nil/None, 1_234, """)
  – **Operators** (===, =~, overloading)
  – **Conditionals** (elsif vs. else if)
  – **Loops** (syntax, object-oriented iterators)
  – **Block syntax** ({…} vs. do…end vs. indentation)
  – **Object syntax** (foo.bar, foo.bar())
Homework

• Simple number-guessing game
  – You pick the number & the computer guesses
  – Seek a straightforward solution

• BE SURE TO LABEL YOUR PRINTOUT!!!

#!/usr/bin/perl

# Homework for CS 368-1
# Assigned on Day 02, 2011-07-12
# Written by Your Name Here

use strict;
use warnings;