Day 2: Basic Syntax

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

Mondays, 2:30–3:30 p.m.
Friday, 10–11 a.m.

Computer Sciences 4265

Still best to email first
Write code.
At least a little.
Every day.
Play around!
Basic Perl Syntax
(in 13 slides)
#!/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 ⇒ °C Conversion*
Simple Variables

- Prefix name with $
- On first use, declare with my
- Assignment: = [ += -= etc.]
- Increment/decrement: ++ --

```perl
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 with "..."
- With or without ()

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

• User input (from terminal) is a bit tricky…
• For today’s homework, just use this:
• We will revisit input next Thursday

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

*Take input and use variables*
Conditionals

• `if (bool) {...} elsif (bool) {...} else {...}
• `unless`
• `{ and }` are required (unlike C)

```perl
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” (empty bucket); 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

```
"6" * '5' => 30
6 . 5    => '65'

'2' == '2.0' => true
'2' eq '2.0'  => false
'2' eq 2.0    => true
```
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...\n";
  $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>!  +  – (unary)</td>
</tr>
<tr>
<td>➔</td>
<td>*  /  %  x</td>
</tr>
<tr>
<td>➔</td>
<td>+  –  . (binary)</td>
</tr>
<tr>
<td>➔</td>
<td>&lt;&lt;  &gt;&gt;</td>
</tr>
<tr>
<td>➦</td>
<td>&lt;  &lt;=  &gt;  &gt;= lt le lt ge</td>
</tr>
<tr>
<td>➦</td>
<td>==  !=  eq  ne</td>
</tr>
<tr>
<td>➔</td>
<td>&amp;</td>
</tr>
<tr>
<td>➔</td>
<td></td>
</tr>
<tr>
<td>➔</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>not</td>
</tr>
<tr>
<td>➔</td>
<td>and</td>
</tr>
<tr>
<td>➔</td>
<td>or  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 (~30–40 lines)
  – Turn in your *code*, not your output

• **BE SURE TO LABEL YOUR PRINTOUT!!!**

```perl
#!/usr/bin/perl
# Homework for CS 368-3
# Assigned on Day 02, 2012-06-21
# Written by *Your Name Here*

use strict;
use warnings;
```