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, …)

• 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
Write code.
At least a little.
Every day.
Basic Perl Syntax
(in 12 slides)
Perl Template

#!/usr/bin/perl

use strict;
use warnings;

# comments!

... code starts here ...
**Numbers**

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

<table>
<thead>
<tr>
<th>Expression</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>4 + 7</td>
<td>=&gt; 11</td>
</tr>
<tr>
<td>17.8 − 3.5</td>
<td>=&gt; 14.3</td>
</tr>
<tr>
<td>16 * 0x10</td>
<td>=&gt; 256</td>
</tr>
<tr>
<td>2 ** 8</td>
<td>=&gt; 256</td>
</tr>
<tr>
<td>10 / 3</td>
<td>=&gt; 3.333333…</td>
</tr>
<tr>
<td>10 % 3</td>
<td>=&gt; 1</td>
</tr>
<tr>
<td>(2 + 3) * 4</td>
<td>=&gt; 20</td>
</tr>
</tbody>
</table>
Strings

• literals: '...' and ""
• escapes: \n, \t, \x7f, \\, \
• operators: .  
• conversions

'foo\tbar'  => foo\tbar [literally]
"foo\tbar"   => foo   bar
'foo' . "\n"  => foo\n [with newline]
'x' x 6      => x x x x x x
6 * '5'      => 30
Simple Variables

- prefix with $
- on first use, declare with my
- operators:  =  +=  -=  ...=  ++  ––
- statements end with ;

```perl
my $name = 'Tim';
my $counter = 0;
my $odd_value_1 = $counter + 7;
$counter += 2;
$name .= ' Cartwright';
$counter++; 
```
Basic Output

- **print** (by default, to standard output)
  - can do variable interpolation in "..."
  - works with or without ()

```perl
print 42;
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>);
```
## 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>=&gt; true</td>
</tr>
<tr>
<td>2 != 2</td>
<td>=&gt; false</td>
</tr>
<tr>
<td>1 + 1 == 2</td>
<td>=&gt; true</td>
</tr>
<tr>
<td>2 == 2.0</td>
<td>=&gt; true</td>
</tr>
<tr>
<td>'a' == 'abc'</td>
<td>=&gt; false</td>
</tr>
<tr>
<td>'2' == '2.0'</td>
<td>=&gt; true! huh?</td>
</tr>
<tr>
<td>'2' eq '2.0'</td>
<td>=&gt; false</td>
</tr>
</tbody>
</table>
| 'Tim' gt 'Nick' | => true [ha!]
| 'Tim' gt 'nick' | => false|
true, false, and undef

- **undef:** “not defined”; like null/nil
- **false:** \(0\), `'`, empty array/hash, **undef**
- **true:** everything else

```
2 == 2 => 1
2 != 2 => ''
defined(undef) => ''
defined('') => 1
defined(0) => 1
```
## More 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 (binary)</td>
</tr>
<tr>
<td>➔</td>
<td>+  -  .</td>
</tr>
<tr>
<td>➔</td>
<td>&lt;&lt;  &gt;&gt;</td>
</tr>
<tr>
<td>➔</td>
<td>&lt;  &lt;=  &gt;  &gt;= lt le gt ge</td>
</tr>
<tr>
<td>➔</td>
<td>==  !=  eq  ne</td>
</tr>
</tbody>
</table>
| ➔ | &
| ➔ | |  ^ |
| ➔ | && |
| ➔ | || |
| ➔ | ?: |
| ➔ | =  +=  -=  ...= |
| ➔ | not |
| ➔ | and |
| ➔ | or  xor |

```perl
my $x = 6 * 3 - 2 & 0xF ? '' : 'b';
$x ||= 'c';
```
Conditionals

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

```perl
if (defined($target) and ($ammo > 0)) {
    print "There you are!\n";
} else {
    print "Goodnight.\n";
}

unless (defined($all_is_well)) {
    print "Shutting down!\n";
}
```
Basic Loops

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

```perl
while ($foo) {
    for (my $i = 0; $i < $foo; $i++) {
        if ($i > $max) {
            $foo--;  
            last;
        }
    # blah
    }
}
```
Statement Modifiers

- Can modify a statement using a loop or conditional
  - `if`, `unless`
  - `while`, `until`, `foreach`
- `()` around conditional are optional
- TIP: Use **only** when clear and natural

```perl
$foo = $MAX if $foo > $MAX;
die("I don't blame you.") if $at_fault;
print($i++ . "\n") while $i <= 10;
```
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!!!

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

# Homework for CS 368-1
# Assigned on Day 02, 2010-07-13
# Written by Your Name Here

use strict;
use warnings;
```