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
  - cannot help (homework, office hrs, …)
- CSL accounts
  - old accounts may still be active
  - otherwise: **`newuser`**/**`NewUser09`**
  - problems? stop by CS 2350 (the CSL) or email **`lab@cs.wisc.edu`**

Write code.

At least a little.

Every day.

Numbers

- 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**, ****, **\"**
- operators: **.** **x**
- conversions

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

Simple Variables

- prefix with **$**
- declare at first use with **my**
- operators: **=** **+=** **-=** **…=** **++** **--**
- statements end with **;**

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

Basic I/O

- **print** (stdout by default)
  - can do variable interpolation in "…"
  - works with or without **()**

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

- idiom for user input

```
chomp(my $user_input = <STDIN>);
```

LIVE CODING EXAMPLE

Comparisons

numeric	==	!=	<	>	<=	>=
string	eq	ne	lt	gt	le	ge

```
2 == 2                => true
2 != 2                => false
1 + 1 == 2            => true
2 == 2.0              => true
'a' == 'abc'          => false
'2' == '2.0'          => true! huh?
'2' eq '2.0'          => false
'Tim' gt 'Alan'       => true [ha!]
'Tim' gt 'alan'       => false
```

true, false, and undef

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

```
2 == 2              => 1
2 != 2              => ''
defined(undef)      => ''
defined('')         => 1
defined(0)          => 1
```

More Operators & Precedence

→	`()`
—	`++ --`
←	`**`
←	`! + -` (unary)
→	`* / % x`
→	`+ - .` (binary)
→	`<< >>`
—	`< <= > >= lt le gt ge`
—	`== != eq ne`

→	`&`
→	`| ^`
→	`&&`
→	`||`
←	`?:`
←	`= += -= …=`
←	`not`
→	`and`
→	`or xor`

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

Conditionals

- **`if () {} elsif () {} else {}`**
- **`unless`**
- Always use **`{`** and **`}`** (unlike C)

```
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`

```
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 expression
  - **`if`**, **`unless`**
  - **`while`**, **`until`**, **`foreach`**
- **`()`** around conditional are optional
- Use ONLY when clear and natural

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

Other Scripting Languages

The DVD player metaphor

Other Scripting Languages

- C-like literals & operators
- "Extra" operators & syntax
  - Perl **->**; Python **"""**; Ruby **|x|**
  - here-docs, case statements, boolean literals, …
- Quotes & interpolation
  - PHP **"hi, $name"**; Ruby **"hi, #{name}"**
- Block syntax & delimiters
  - PHP, JS, AS; Ruby **{}** + *…**end**; Python indents
  - blocks as first-class objects
- Object syntax: **foo.bar** & **foo.baz()**