Day 14: Recipes III

Miscellaneous

Suggested Reading:
Perl Cookbook (2nd Ed.)

Chapter 1: Strings
Chapter 4: Arrays
Chapter 5: Hashes
CS 368, Fall 2012 · 8 Weeks, 22 Oct – 14 Dec

• Section 3
  – Introduction to Python
  – Alan De Smet (maybe)
  – Tuesdays & Thursdays, 9:55–10:45 a.m.
  – Computer Sciences 1263

• Section 4
  – Introduction to Scripting for CHTC
  – Tim Cartwright
  – Mondays & Thursdays, 1:20–2:10 p.m.
  – Grainger Hall 1180 (will change!)
Homework Review
Homework Preview
WEATHER REPORT

<table>
<thead>
<tr>
<th>DATE</th>
<th>HIGHS</th>
<th>LOWS (early next AM)</th>
</tr>
</thead>
<tbody>
<tr>
<td>2011-07-29 (Fri)</td>
<td>87-89 =&gt; 86.5 (below)</td>
<td>61-63 =&gt; 71.2 (above)</td>
</tr>
<tr>
<td>2011-07-30 (Sat)</td>
<td>87-89 =&gt; 88.1 (good)</td>
<td>67-69 =&gt; 72.1 (above)</td>
</tr>
<tr>
<td>2011-07-31 (Sun)</td>
<td>87-89 =&gt; 89.1 (good)</td>
<td>67-69 =&gt; 74.7 (above)</td>
</tr>
<tr>
<td>2011-08-01 (Mon)</td>
<td>89-91 =&gt; 86.9 (below)</td>
<td>71-73 =&gt; 76.0 (above)</td>
</tr>
<tr>
<td>2011-08-02 (Tue)</td>
<td>91-93 =&gt; 90.5 (below)</td>
<td>67-69 =&gt; 72.6 (above)</td>
</tr>
<tr>
<td>2011-08-03 (Wed)</td>
<td>only 7 weather observations</td>
<td>only 7 weather observations</td>
</tr>
<tr>
<td>2011-08-04 (Thu)</td>
<td>no weather observations</td>
<td>no weather observations</td>
</tr>
</tbody>
</table>

Of the 10 forecasts, the actual temperatures were below the forecast 3 times, above the forecast 5 times, and accurate 2 times. Overall accuracy was 20%.
Miscellaneous Tricks
Swapping Values

• Common approach:

```perl
my $x = 12;
my $y = 30;
...
my $temp = $x;
$x = $y;
$y = $temp;
```
Parallel Assignment

• The Perl way

```perl
my $x = 12;
my $y = 30;
...
($x, $y) = ($y, $x);
```

• Another example: Fibonacci iteration

```perl
my $x = 0; my $y = 1;
while ($y <= $max) {
    ($x, $y) = ($y, $x + $y);
}
```
Making Sure a Variable is Defined

• The obvious way:

```perl
my $foo = get_value();  # can return undef
if (not defined $foo) {
    $foo = default value here;
}
```

• A common Perl way:

```perl
my $foo = get_value();  # can return undef
$foo ||= default value here;
```

• Any problems with the Perl way?
Long String Literals

Use special Perl syntax called a “here-document”

my $long_string = "<<END;
This is the start of the string.
It can go on for many lines.
When done, terminate on the next line.
END"

safe_write($filename, "<<CONTENTS);
* File contents!
  - Formatting is preserved!!
* Interpolation even works: $foo!!
CONTENTS
Really Long String Literals

- Use `__DATA__` section at end of script
- **Pro:** Part of script… **Con:** Part of script

```perl
# Perl opens DATA filehandle automatically
while (<DATA>) {
    # do stuff with lines here
}

__DATA__
This is the start of a very long data block.
It must be the last part of the script file.
Perl will not run code after __DATA__.
No $variable interpolation here.
```
Subroutines: Scalar, List, or Void Context?

```perl
sub read_file {
    my $filename = shift;

    # Read file contents into array
    open(my $filehandle, '<', $filename) or die "Could not open '$filename': $!
"
    my @lines = <$filehandle>;
    close($filehandle);

    # Choose the correct return format
    return @lines if wantarray;
    return join('', @lines) if defined wantarray;
    return;
}
```
Subroutines: Scalar, List, or Void Context?

```
sub read_file { ... } # from previous slide

my $scalar_contents = read_file('test.txt');
=> "foo\nbar\n42\n"

my @array_contents = read_file('test.txt');
=> ("foo\n", "bar\n", "42\n");

read_file('test.txt');
=> undef
```
Collection Tricks
Randomize Order of List

• Don’t reinvent the wheel!
• Hard to implement correctly
• Use built-in List::Util::shuffle

```perl
use List::Util qw/shuffle/;
my @list = (1, 2, 3, 4, 5, 6, 7, 8, 9, 10);
my @random_list = shuffle(@list);
=> [e.g.:] (7, 3, 2, 4, 1, 5, 8, 10, 6, 9)
```

• List::Util has other useful list functions: first, max, maxstr, min, minstr, reduce, sum
Fetch Hash Keys in Insertion Order

- Keep and manage separate, parallel array
- Deletions are painful

```perl
my %hash;
my @hash_keys;
foreach my $name (<$input_fh>) {
    unless (exists $hash{$name}) {
        $hash{$name} = $something;
        push(@hash_keys, $name);
    }
}

foreach my $key (@hash_keys) { … }
Hash as Set

Keys are set elements; values are always 1:

```perl
map { $set{$_} => 1 } @elements_for_set;
```

Test for set membership:

```perl
if (exists $set{$element}) { ... }
```

Compute *union* of two sets:

```perl
my %u = map { $_ => 1 } keys %set_a, %set_b;
```

Compute *intersection* of two sets:

```perl
my %i = map { $_ => 1 }
  grep(exists $set_b{$_}, keys %set_a);
```
Output Formatting
sub commify {
    my $text = reverse $_[0];
    $text =~ s/\d{3}(?!\d*\.)/$1,/g;
    return scalar reverse $text;
}

my $num_with_commas = commify(1234567.8901);
=> '1,234,567.8901'

• (?=pattern) : zero-width positive look-ahead
• (?!pattern) : zero-width negative look-ahead
Wrapped Text

• Don’t reinvent the wheel!

my $string = "This is a lot of text. But it is not very well formatted yet. What can be done about it?\n";

use Text::Wrap;                      # Read perldoc for usage
$Text::Wrap::columns = 35;
print wrap('', '', $string);

This is a lot of text. But it is not very well formatted yet. What can be done about it?
Homework
The Final Report

<table>
<thead>
<tr>
<th>DATE</th>
<th>HIGHS</th>
<th>LOWS (early next AM)</th>
</tr>
</thead>
<tbody>
<tr>
<td>2011-07-29 (Fri)</td>
<td>87-89 =&gt; 86.5 (below)</td>
<td>61-63 =&gt; 71.2 (above)</td>
</tr>
<tr>
<td>2011-07-30 (Sat)</td>
<td>87-89 =&gt; 88.1 (good)</td>
<td>67-69 =&gt; 72.1 (above)</td>
</tr>
<tr>
<td>2011-07-31 (Sun)</td>
<td>87-89 =&gt; 89.1 (good)</td>
<td>67-69 =&gt; 74.7 (above)</td>
</tr>
<tr>
<td>2011-08-01 (Mon)</td>
<td>89-91 =&gt; 86.9 (below)</td>
<td>71-73 =&gt; 76.0 (above)</td>
</tr>
<tr>
<td>2011-08-02 (Tue)</td>
<td>91-93 =&gt; 90.5 (below)</td>
<td>67-69 =&gt; 72.6 (above)</td>
</tr>
<tr>
<td>2011-08-03 (Wed)</td>
<td>-------- only 7 weather observations --------</td>
<td></td>
</tr>
<tr>
<td>2011-08-04 (Thu)</td>
<td>-------- no weather observations --------</td>
<td></td>
</tr>
</tbody>
</table>

Of the 10 forecasts, the actual temperatures were below the forecast 3 times, above the forecast 5 times, and accurate 2 times. Overall accuracy was 20%.
Weather Analysis, Part III

• Compare forecasts to observations!

• What does “UPPER 80S” mean?

• Which days/hours to use for a given forecast?
  – Daily high, 4–6 p.m.; daily low, ~5 a.m. *(of next day!)*
  – So use \((date, \, 12 \, p.m.) – (date + 1, \, 11 \, a.m.)\)

• Report is in two parts
  – Format should follow sample on assignment page
  – Sample may not show every case!
  – Wrap second part to a reasonable width