Day 14: Recipes IV

Miscellaneous

Suggested Reading:
Perl Cookbook (2nd Ed.)

Chapter 4: Arrays
Chapter 5: Hashes

Homework Review

Homework Preview

The Final Report, Version 1

```
On July 30, the forecast high was 81-83F and
the actual high was 82.2F (correct); the
forecast low was 61-63F and the actual low
was 64.2F (high).

On July 31, the forecast high was 83-85F and
the actual high was 86.2F (high); the
forecast low was 61-63F and the actual low
was 61.1F (correct).
```

The Final Report, Version 2

```
There are 5 days of data available between 31
July 2010 and 6 August 2010.  The forecast
high was accurate on 3 days (60%); the actual
high temperature was higher than forecast on 2
days by an average of 1.6 degrees Fahrenheit.
The forecast low was accurate on 2 days (40%);
the actual low temperature was lower than
forecast on 1 day by 2.0 degrees Fahrenheit,
and the actual low temperature was higher than
forecast on 2 days by an average of 1.8
degrees Fahrenheit.
```

Miscellaneous Tricks

Swapping Values

- Common approach:

```
my $x = 12;
my $y = 30;
...
my $temp = $x;
$x = $y;
$y = $temp;
```

Parallel Assignment

- The Perl approach

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

- Another example: Fibonacci iteration

```
my $x = 0; my $y = 1;
while ($y <= $max) {
    ($x, $y) = ($y, $x + $y);
}
```

Subroutines: Scalar, List, or Void Context?

```
sub slurp {
    my @lines;
    if (open(my $fh, $_[0])) {
        @lines = <$fh>;
        close($fh);
    } # need better way to handle error?
    if (wantarray()) {
        return @lines;
    } elsif (defined wantarray()) {
        return join('', @lines);
    } else {
        return;
    }
}
```

Subroutines: Scalar, List, or Void Context?

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

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

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

slurp('text.txt');
=> undef
```

Collection Tricks

Randomize Order of List

- Don't reinvent the wheel!
- Hard to implement correctly
- Use **`List::Util::shuffle`**

```
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

```
my %hash;
my @hash_keys;

foreach my $name (<$input_fh>) {
    $hash{$name} = 1;
    push(@hash_keys, $name);
}

foreach my $key (@hash_keys) { ... }
```

Set Operations: Union

```
my %A;   # A is a set, use only keys
my %B;   # B is a set, use only keys

# Verbose method
my %Un;
foreach my $key (keys %A, keys %B) {
    $Un{$key} = 1;
}

# Brief method
my %Un = map { $_ => 1 } keys %A, keys %B;
```

Set Operations: Intersection

```
my %A;   # A is a set, use only keys
my %B;   # B is a set, use only keys

# Verbose method
my %In;
foreach my $key (keys %A) {
    $In{$key} = 1 if exists $B{$key};
}

# Brief method
my %In = map { $_ => 1 }
             grep(exists $B{$_}, keys %A);
```

Set Operations: Complement

```
my %A;   # A is a set, use only keys
my %B;   # B is a set, use only keys

# Verbose method
my %A_not_B;
foreach my $key (keys %A) {
  $A_not_B{$key} = 1 unless exists $B{$key};
}

# Brief method
my %A_not_B = map { $_ => 1 }
                grep(!exists $B{$_}, keys %A);
```

Set Operations: Symmetric Difference

```
my %A;   # A is a set, use only keys
my %B;   # B is a set, use only keys

my (%union, %intersection, %diff);
$union{$_}++ foreach keys %A, keys %B;
foreach my $key (keys %union) {
  if ($union{$key} == 2) {
    $intersection{$key} = 1;
  } else {
    $diff{$key} = 1;
  }
}
```

Output Formatting

Large Numbers With Commas

```
sub commify {
   my $text = reverse $_[0];
   $text =~ s/(\d\d\d)(?=\d)(?!\d*\.)/$1,/g;
   return scalar reverse $text;
}

my $num_with_commas = commify(12345.6789);

=> '12,345.6789'
```

- **(?=*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;
$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

```
There are 5 days of data available between 31
July 2010 and 6 August 2010.  The forecast
high was accurate on 3 days (60%); the actual
high temperature was higher than forecast on 2
days by an average of 1.6 degrees Fahrenheit.
The forecast low was accurate on 2 days (40%);
the actual low temperature was lower than
forecast on 1 day by 2.0 degrees Fahrenheit,
and the actual low temperature was higher than
forecast on 2 days by an average of 1.8
degrees Fahrenheit.
```

Weather Analysis, Part IV

- Compare forecasts to actuals!
  - Make up data if *(and only if)* you have to
- Obey user settings (see Part II)
  - Limit analysis to given dates
  - Display temperatures (or errors) in given units
- Display results in a paragraph of text
  - **Choose only one output format!**
  - Format should *mostly* follow sample
  - Exact format is up to you
  - Wrap text to 70 columns (as if for email)