---

1. Mumble something

Inserting Commas

Our next example is a single subroutine that takes a numeral, such as 1234567.89, and returns a version that has commas inserted, as `"1,234,567.89"`. Before we look at the subroutine itself, we should note that this is a Frequently Asked Question; the solution given in the Perl FAQ list is:

```
s/(^[-+]?\d+?(?=(?>(?:\d{3})+)(?!\d))|\G\d{3}(?=\d))/$1,/g;
```

This is the kind of thing that gives Perl a bad reputation. Let's look at the code we have and see if we can't come up with something a little less ridiculous.

The original function, `conversion()`, is 30 lines long. The author told me he thought that it seemed bulky. I agree.

The first thing that jumps out at me is that `$pos` is an **array length variable**, so we can easily get rid of it, by replacing this:

```
      $section[$pos] = substr($number, 0, $remain);
      $pos++;
```

with this:

```
push @section, substr($number, 0, $remain);
```

and this:

```
    $section[$pos] = substr($number, $next, 3);
    $pos++;
```

with this:

```
push @section, substr($number, 0, 3);
```

The main loop now looks like this:

```
while ($next < $loop) {
  if ($remain > 0) {
    push @section, substr($number, 0, $remain);
    $next = $remain++;
```

```
          $remain = 0;
        }
        push @section, substr($number, $next, 3);
        $next = ($next + 3);
      }
```

The thing to notice here is the `$remain` variable. If the `if` block is executed, then `$remain` is set to zero, and since it is not modified anywhere else, it must remain zero until the loop completes. That means that the `if` block can execute at most once, and only on the first pass through the loop. So we can hoist it out:

```
     if ($remain > 0) {
        push @section, substr($number, 0, $remain);
        $next = $remain++;
        $remain = 0;
      }
      while ($next < $loop) {
        push @section, substr($number, $next, 3);
        $next = ($next + 3);
      }
```

Now instead of one complicated block with nested logic, we have two simple blocks.

Mumble something

The goal of the `while` loop is now apparent: it is collecting the three-character substrings of `$number`. Armed with this logic, we can make sense of the `if` block too: it's acquiring the leftover characters from the front of the numeral. In the example of 1234567, the `if` block acquires the 1 and the `while` loop acquires `234` and then `567`.

```
sub conversion
{
  $number = shift;
  $size = length($number);
  $result = ($size / 3);
  @commas = split (/\./, $result);
  $remain = ($size - ($commas[0] * 3));
  $pos = 0;
  $next = 0;
  $loop = ($size - $remain);
  while ($next < $loop)
  {
     if ($remain > 0)
     {
        $section[$pos] = substr($number, 0, $remain);
        $next = $remain++;
        $remain = 0;
        $pos++;
     }
     $section[$pos] = substr($number, $next, 3);
     $next = ($next + 3);
     $pos++;
  }
  $loop = 0;
  @con = ();
  foreach (@section)
  {
     $loop++;
```

```
      $cell++;
      @tens = split (/:/, $_);
      $con[$cell] = $tens[0];
      if ($loop == $pos)
      {
         last;
      }
      $cell++;
      $con[$cell] = ",";
   }
   return @con
```

Chapter 7 | TOP