THE FIVE-MINUTE LISP TUTORIAL

People new to Lisp often find the language hard to read. This is mostly because Lisp has fewer syntactic constructs than most languages, but works those constructs harder to get an even richer vocabulary of features. To give you a reader's intuition, consider the following rules for converting code from Python to Lisp:

1. Write control constructs as if they were function calls. That is, `while x*y < z: q()` becomes `while (x*y < z, q())`. This is just a change in notation; it's still a while loop.
2. Write uses of infix operators as if they were also function calls, where the functions have odd-looking names. The expression `x*y < z` would become `<(*(x, y), z)`. Now any parentheses present only for grouping (as opposed to those that surround function arguments) are redundant; remove them.
3. Now everything looks like a function call, including control constructs and primitive operations. Move the opening parenthesis of every "call" from after the function name to before the function name, and drop the commas. For example, `f(x, y, z)` becomes `(f x y z)`. To continue the previous example, `<(*(x, y), z)` becomes `(< (* x y) z)`.

Taking all three rules together, the Python code `while x*y < z: q()` becomes the Lisp code `(while (< (* x y) z) (q))`. This is a proper Emacs Lisp expression, with the same meaning as the original Python statement.

There's more, obviously, but this is the essence of Lisp's syntax. The bulk of Lisp is simply a vocabulary of functions (such as < and *) and control structures (such as `while`), all used as shown here.

Here is the definition of an interactive Emacs command to count words in the currently selected region of text. If I explain that `"\\<"` is an Emacs regular expression matching the beginning of a word, you can probably read through it:

```
(defun count-region-words (start end)
  "Count the words in the selected region of text."
  (interactive "r")
  (save-excursion
    (let ((count 0))
      (goto-char start)
      (while (re-search-forward "\\<" end t)
        (setq count (+ count 1))
        (forward-char 1))
      (message "Region has %d words." count))))
```

There's an argument to be made that Lisp gives its human readers too few visual cues as to what's really going on in the code, and that it should abandon this bland syntax in favor of one that makes more apparent distinctions between primitive operations, function calls, control structures, and the like. However, some of Lisp's most important features (which I can't go into here) depend