#! /home/douglasallen/.rbenv/versions/2.2.0/bin/ruby

```
##########################################################
#
# Ruby interactive input/eval loop
# Written by matz           (matz@netlab.co.jp)
# Modified by Mark Slagell  (slagell@ruby-lang.org)
#    with suggestions for improvement from Dave Thomas
#                           (Dave@Thomases.com)
#
##########################################################
#
# NOTE - this file has been renamed with a .txt extension to
# allow you to view or download it without the rubyist.net
# web server trying to run it as a CGI script.  You will
# probably want to rename it back to eval.rb.
#
##########################################################

module EvalWrapper
  # Constants for ANSI screen interaction.  Adjust to your liking.
  Norm = "\033[0m".freeze
  PCol = Norm # Prompt color
  Code = "\033[1;32m".freeze   # yellow
  Eval = "\033[0;36m".freeze   # cyan
  Prompt = PCol + 'ruby> ' + Norm
  PrMore = PCol + '    | ' + Norm
  Ispace = '   '.freeze        # Adjust length of this for indentation.
  Wipe = "\033[A\033[K".freeze # Move cursor up and erase line

  # Return a pair of indentation deltas. The first applies before
  # the current line is printed, the second after.
  def self.indentation(code)
    case code
    when /^\s*(class|module|def|if|case|while|for|begin)\b[^_]/
      [0, 1]      # increase indentation because of keyword
    when /^\s*end\b[^_]/
      [-1, 0]     # decrease because of end
    when /\{\s*(\|.*\|)?\s*$/
      [0, 1]      # increase because of '{'
    when /^\s*\}/
      [-1, 0]     # decrease because of '}'
    when /^\s*(rescue|ensure|elsif|else)\b[^_]/
      [-1, 1]     # decrease for this line, then come back
```

```
  else
    [0, 0]      # we see no reason to change anything
  end
end

# On exit, restore normal screen colors.
END { print Norm, "\n" }

######################################################################
# Execution starts here.
######################################################################

indent = 0
while TRUE # Top of main loop.

  # Print prompt, move cursor to tentative indentation level, and get
  # a line of input from the user.
  if indent == 0
    expr = ''; print Prompt  # (expecting a fresh expression)
  else
    print PrMore             # (appending to previous lines)
  end
  print Ispace * indent, Code
  line = gets
  print Norm

  if !line
    # end of input (^D) - if there is no expression, exit, else
    # reset cursor to the beginning of this line.
    expr == '' ? break : (print "\r")
  else

    # Append the input to whatever we had.
    expr << line

    # Determine changes in indentation, reposition this line if
    # necessary, and adjust indentation for the next prompt.
    begin
      ind1, ind2 = indentation(line)
      if ind1 != 0
        indent += ind1
        print Wipe, PrMore, (Ispace * indent), Code, line, Norm
      end
      indent += ind2
    rescue # On error, restart the main loop.
```

```
        print Eval, "ERR: Nesting violation\n", Norm
        indent = 0
        redo
      end

      # Okay, do we have something worth evaulating?
      if (indent == 0) && (expr.chop =~ /[^; \t\n\r\f]+/)
        begin
          result = eval(expr, TOPLEVEL_BINDING).inspect
          if $ERROR_INFO # no exception, but $! non-nil, means a warning
            print Eval, $ERROR_INFO, Norm, "\n"
            $! = nil
          end
          print Eval, '   ', result, Norm, "\n"
        rescue ScriptError, StandardError
          $! = 'exception raised' unless $ERROR_INFO
          print Eval, 'ERR: ', $ERROR_INFO, Norm, "\n"
        end
        break unless line
      end
    end
  end # Bottom of main loop
  print "\n"
end
```