LinuxAcademy.com

Ruby 101

Intermediate Ruby:Symbols, Strings, Comments

Intermediate Ruby:

Strings -> data between "..."
    Each new literal -> A new object_id
        (unless frozen)

Symbols -> names or labels for things, with a :...
    Each reference to the same symbol -> The same object_id

Symbols can:
- Make code easier to read
- Trim excess overhead vs strings
- Return a string with .to_s

Method names, object names, class names

class names -> typically CamelCase MyClassName
method names -> underscore .my_method_name
object names -> underscore my_object

Comments:

Begin with '#' for single line comments

```
# This is a one line comment

##################################
# This is commented out too. But it
# is also very eye-catching
##################################

=begin
    Multi line comments go here,
    but these aren't used very often.
=end
```

Self commenting?

Code should always be readable, without recourse to comments.

- descriptive class, method, object/variable names
- verbs in method names
- no single letter variable names ... too hard to read

Comments mandatory?

All code should be throroughly commented with separate comments.

- Code can still be hard to read

My preference for comments:

```
    ###########################################
    # Multi-line "visual" comments for methods
    ###########################################

    #single line comments to hold places for future code or explain
             single lines of code

=begin
    For commenting out sections of code
=end
```

Our model.rb file:

```
################################
# Setup the object ...
# make sure @log_entries is an
# array.
################################
def initialize
        cd "./"
        @log_entries = Array.new
end

################################
# change directory of LogFile
# object, to path if possible.
# If not, return false.
################################
```

Try it yourself:

Comment on each of the methods we have created.

Download the sample files for some hints.