# Ruby Practice Exercise #3

*Note that these practice exercises are NOT optional. They are not add-ons to this module – they are part of the **core content** of your course. Do not skip these practice exercises!*

## *Exercise 3.1 (review Learn Ruby 4 and 5)*

The `/etc/passwd` file on Linux stores information about each of the system's users. This file is colon-delimited (i.e., `each:entry:is:separated:from:the:next:by:a:colon`). Write a Ruby program that processes this file on your computer and provides a count of the number of system users who do not have a login-shell enabled (hint: their shell will be set to `/bin/false`).

Note: Do not use regular expressions in your code.

## *Exercise 3.2 (review Learn Ruby 4 and 5)*

Write a Ruby program which process a text file and removes any blank lines (i.e., those lines which only contain a newline character). Such a program is commonly referred to as a "filter".

Note: Do not use regular expressions in your code.

## *Exercise 3.3 (review Learn Ruby 6 and the regex handout)*

Rewrite the programs you developed in 3.1 and 3.2, above, to take advantage of regular expressions. Ask yourself which solution(s) do you prefer? Ask yourself why?

## *Exercise 3.4 (review Learn Ruby 1-6)*

Write a Ruby program that accepts the name of an existing text file from the command-line, opens the file and transfers its contents in reversed order to a new file which takes as its name the existing text file's name in reverse (while preserving the extension). For example, if `mydata.txt` is the existing text file with these contents:

```
And if you walk you're gonna get there
although it takes just a little longer
and when you see 'em in the distance
you will wring your hands and moan ...
```

your program should create a new file called `atadym.txt` with the following contents:

```
you will wring your hands and moan ...
and when you see 'em in the distance
although it takes just a little longer
And if you walk you're gonna get there
```