**Now Arriving at Level #5**

**Objective:**

Blocks are always attached to methods. You saw this with the times method, which took the block and ran its code over and over. (In this case: **five** times.)

This last lesson was a bit longer. You've probably used up three minutes learning about:

* **Hashes.** The little 'dictionary' with the curly braces: {}.
* **Symbols.** Tiny, efficiently reusable code words with a colon: :splendid.
* **Blocks.** Chunks of code which can be tacked on to many of Ruby's methods. Here's the code you used to build a scorecard: books.values.each { |rate| ratings[rate] += 1 }.

On to the next thing, okay? On your computer, you probably have a lot of different files. Some files have pictures in them, some have programs in them. And files are often organized into folders, also called: **directories**.

I've prepared a few directories for you. Take a look, using the following command: Dir.entries "/"

> Dir.entrie­s "/"

=> [".", "..", "Home", "Libraries", "MouseHole", "Programs", "Tutorials", "comics.txt"]

Success!

>

**The Private Collection of Dr. Dir**

**Objective:**

You've just listed out everything in the top directory, which is called the *root*. It's indicated by the single slash in your string parameter. It contains some programs, as well as other tutorials and such.

So, what exactly is that Dir.entries method? Well, it's just a method, like the others you've seen. Dir has a collection of methods for checking out file directories, and entries is being called *on* the Dir variable. The entries method just lists everything in the directory you've indicated!

One other little thing we haven't really talked about quite yet: method arguments. A few are highlighted below.

* Dir.entries **"/"** -- Anything listed after a method is considered an 'attachment'.
* print **poem** -- See, print is just an ordinary method, while the poem is what got attached for printing.
* print **"pre", "event", "ual", "ism"** -- This bit has several arguments! Ruby makes us use commas to distinguish between them.

Next up, we'll list just the text files in our root directory using a bracket notation. Remember how it searches?

Try: Dir["/\*.txt"]

> Dir["/\*.tx­t"]

=> ["/comics.txt"]

Success!

## Come, Read Comics With Me

### Objective:

The Dir[] syntax is kind of like entries, but instead searches for files with wildcard characters.

Here, we see those square brackets again! Notice how they still mean, "I am looking for \_\_\_\_\_?"

Dir["/\*.txt"] says to Ruby: "I am looking for any files which end with .txt." The asterisk indicates the "any file" part. Ruby then hands us every file that matches our request.

Alright, let's crack open this comics file, then! We'll use a new method to do it.

Here's the way: print File.read("/comics.txt")

> print File.­read("/com­ics.txt")

=> "Achewood: http://achewood.com/  
Dinosaur Comics: http://qwantz.com/  
Perry Bible Fellowship: http://cheston.com/pbf/archive.html  
Get Your War On: http://mnftiu.cc/  
"

Success!

> print File.­read("/com­ics.txt");­

=> nil

can't find '/Home/comics.txt', where did you put it?

> FileUtils.­cp('/comic­s.txt', '/Hom­e/comics.t­xt')

=> nil

Success!

>

## Mi Comicas, Tu Comicas

### Objective:

Okay, you've got a copy, and it's located in the right directory. Check it out!

Use Dir["/Home/\*.txt"]

Type next to move to the next lesson when you're finished.

> Dir["/Home­/\*.txt"]

=> ["/Home/comics.txt"]

> next

## Your Own Turf

### Objective:

To add your own comic to the list, let's open the file in **append** mode, which we indicate with the "a" parameter. This will allow us to put new stuff at the end of the file.

Start by entering this code: File.open("/Home/comics.txt", "a") do |f|

> File.open(­"/Home/com­ics.txt", "a") do |f|

..

.. f << "Cat and Girl:­ http:­//catandgi­rl.com" end

=> #<File:/Home/comics.txt (closed)>

>

## Ruby Sits Still

### Objective:

The last line will add the Cat and Girl comic to the list, but Ruby's going to wait until you're totally finished to take action.

This means we'll also need to wrap up the code block you've started. Turns out, you actually opened a new block when you typed that do keyword.

So far the blocks we've seen have used curly braces, but this time we'll be using do and end instead. A lot of Rubyists will use a do...end setup when the block goes on for many lines.

Let's get that block finished now, with your very own end.

**File.open("/Home/comics.txt", "a") do |f|**

**f << "Cat and Girl: http://catandgirl.com/"**

end

> File.open(­"/Home/com­ics.txt", "a") do |f| f << "Cat and Girl:­ http:­//catandgi­rl.com/" end

=> #<File:/Home/comics.txt (closed)>

>

## And Now For the Startling Conclusion

### Objective:

So your prompt has changed, see that? Your prompt is a **double** dot now.

In this tutorial, this prompt means that Ruby is expecting you to type a little bit more. As you write further lines of Ruby code, the double-dots will continue until the tutorial sees you are completely finished.

Alright, so here's more code. You've already typed the first line, so just enter the second line.

**File.open("/Home/comics.txt", "a") do |f|**

f << "Cat and Girl: http://catandgirl.com/"

**end**

> File.open(­"/Home/com­ics.txt", "a") do |f|

.. f << "Cat and Girl:­ http:­//catandgi­rl.com/"

.. end

=> #<File:/Home/comics.txt (closed)>

>

## Ruby Sits Still

### Objective:

Sweet! You've added that brand new comic to the end of the file. You can see for yourself, using the read method you saw earlier: print File.read("/Home/comics.txt").

When you want to move on to the next lesson, type next.

> next

## The Clock Nailed To the File

### Objective:

I wonder, what time was it when you changed your file? We can check that out.

Type: File.mtime("/Home/comics.txt").

> File.mtime­("/Home/co­mics.txt")­

=> 2015-05-13 18:26:01 UTC

Success!

>

## Just the Hour Hand

### Objective:

Excellent, there's the exact time, precisely when you added to the file. The mtime method gives you a nice Ruby Time object.

If you want to check just what hour it was, hit the up arrow key to put your previous entry in the console. Then modify the line to say: File.mtime("/Home/comics.txt").hour.

> File.mtime­("/Home/co­mics.txt")­.hour

=> 18

Success!

>