

[My Path](#)[Catalog](#)[Codecademy Resources](#) > ruby

Ruby: Reading and Writing Files

Learn to read and write ruby files from the command line



Background

In the Ruby programs you've written thus far with Codecademy, there has been no data persistence. In other words, none of the data your programs generated was saved.

For example, in [Banking On Ruby](#), you created new bank accounts and performed various methods on them. But when you signed out of Codecademy or shutdown your computer for the day, your account objects were lost forever.

In this article, we'll walk through a few basic ways to read and write files with Ruby. Reading and writing files keeps track of the data our programs generate. It's an important part of any program that requires data persistence.

Read a File with Ruby

Ruby is pretty smart. It can look inside a file and read it back to you. For instance, if you have a text file with a list of todos typed out inside of it, you could ask Ruby to open the file for you, and then Ruby can print out everything in the list. It's like Ruby is reading to you. Isn't that cool?

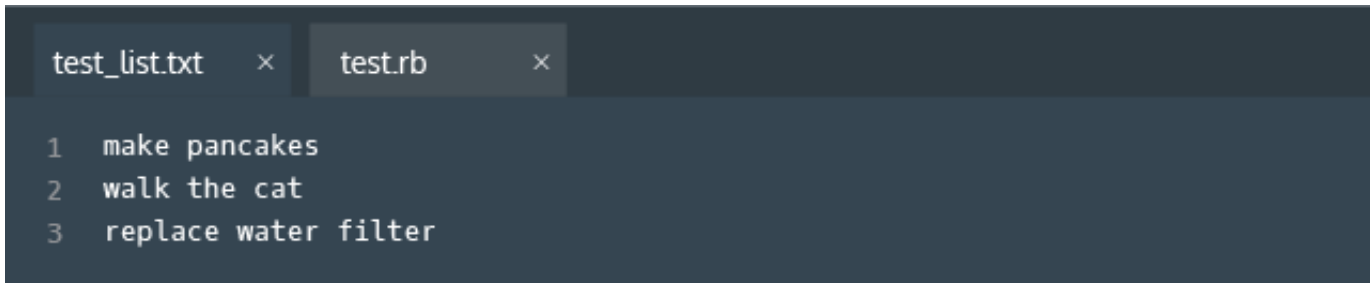
To demonstrate how this works, we are going to create a demo todo list, and a ruby file to read the list to us.

Throughout this article, we are going to use a text editor to write our todo list and our Ruby file. The one we prefer at Codecademy is Sublime Text. If you don't have a text editor like Sublime Text, read this article first to get it set up.

Okay, let's dive in:

1. Create a folder to hold our demo todo list and our Ruby program. You can create a folder using the Command Line, or by right clicking on your desktop, and selecting "New Folder."
2. Open the folder in Sublime Text, by going to File > Open File ...

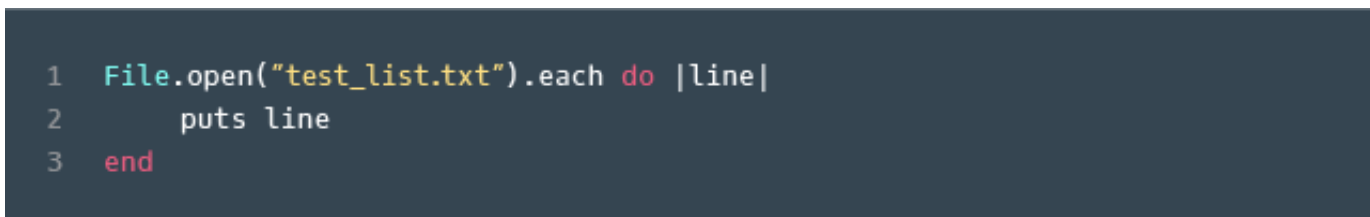
3. Now create a text file called `test_list.txt` and another file named `test.rb`. You can do this by going to `File > New File` in the Sublime Text menu, or by using the shortcut command `+ N`, (`control + N` for Windows).
4. Once you've created these files, make sure to save them in the folder you made in the first step.
5. Open up `test_list.txt` in Sublime Text, and type in two or three sample todo items there, one item per line. This will be the file we will read with Ruby.

A screenshot of a text editor window with two tabs: 'test_list.txt' and 'test.rb'. The 'test_list.txt' tab is active, showing three lines of text: '1 make pancakes', '2 walk the cat', and '3 replace water filter'. The lines are numbered on the left side of the editor.

```
test_list.txt  ×  test.rb  ×  
1 make pancakes  
2 walk the cat  
3 replace water filter
```

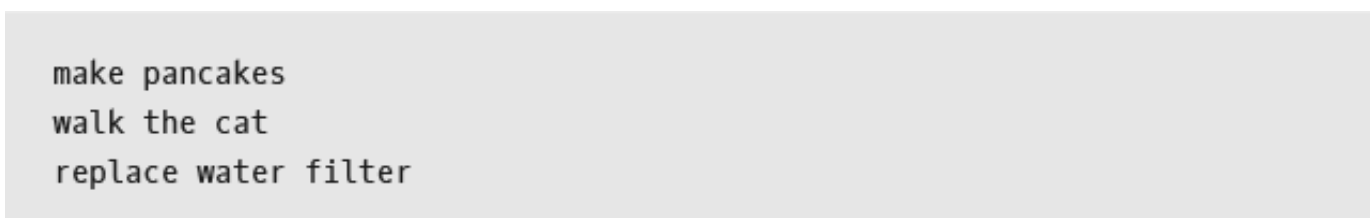
Now comes the magic part: we're going to enlist a Ruby Class called `File`, which will read your `test_list.txt` file line by line, and print each of those lines to your command line console. For many of you, this will be a completely new way to use Ruby. Excited? Good!

Type in the method below -- Read-File Example 1 -- into `test.rb`. After you run this program from the command line, we'll explain what's happening in the code:

A screenshot of a text editor showing the contents of test.rb. The code consists of three lines: '1 File.open("test_list.txt").each do |line|', '2 puts line', and '3 end'. The code is syntax-highlighted with colors: 'File.open' is blue, 'puts' is green, and 'end' is red.

```
1 File.open("test_list.txt").each do |line|  
2 puts line  
3 end
```

1. Save `test.rb`
2. Then, go to your command line window and navigate to your folder's location using `cd`.
3. Once you're inside the folder, type this into the command line to run your Ruby code: `ruby test.rb`
4. You should see each line of `test_list.txt` print to the console:

A screenshot of a command line window showing the output of the Ruby program. The output consists of three lines: 'make pancakes', 'walk the cat', and 'replace water filter'.

```
make pancakes  
walk the cat  
replace water filter
```

How cool is that?

To explain very generally, Ruby – like all other programming languages – has the power to read data from one file and spit that data out somewhere else. Which data Ruby grabs and where it delivers it is up to you as the programmer. That's where the `File.open("test_list.txt")` method comes in. Let's revisit [line 5](#) of the code above:

```
1 File.open("test_list.txt").each do |line|
```

Ruby has a Class named `File` that can be used to perform a variety of methods on a file. One of those methods is `.open`, which looks inside a file. We can use `File.open` to look inside a file, by feeding it `test_list.txt` inside the `.open` method's parentheses.

Pro Tip: Reading Ruby Documentation will take your Ruby skills to the next level. Take a look at all the things you can do with the `File` Class [here](http://ruby-doc.org/). Ruby's documentation is your friend! When in doubt, search <http://ruby-doc.org/>.

Then, we can use the `.each` method to iterate over each line of the file. As each line is iterated over, we assign each line to a variable named `line`, as you see between the pipes: `| line |`

If [line 1](#) above has "opened" the file, then, what do we do once we've got access to each line? We simply puts it to the console. That's what [line 2](#) does:

```
2 puts line
```

Once `.each` has looped through each line and printed it to the console, the method ends.

Here's a hint for how you can use `File.open("text_list.txt").each` later on. What if, instead of simply printing each line of `test_list.txt` to the console, we did a wildly different Ruby operation with it? With `File.open("text_list.txt").each`, you've got each line of your text file in the palm of your hand.

Write to a File with Ruby

Now we want to write more lines inside `test_list.txt`. We can achieve this by writing a method that opens up a file, then appends text to the bottom of the file. Let's add some items to the bottom of the todo list that we just read in the last step.

Inside `test.rb`, add the method in Write Example 1 to your code, save it, and run it using `ruby test.rb`. Then, check `test_list.txt` and notice that there is a new line of text appended to the bottom.

```
1 File.open("test_list.txt","a") do |line|
2   line.puts "\r" + "air up bike tires"
3 end
```

Let's talk about how this method works, line by line.

```
1 File.open("test_list.txt","a") do |line|
```

Just like in the last step “Read a File with Ruby,” we can use the `Class File` with the method `.open` to look inside `test_list.txt`.

Next, inside the parentheses, after we named the file `test_list.txt`, we added “a”, which stands for append. This tells Ruby to add anything that comes next after the end of the `test_list.txt` file.

We added “a” for append, but there are other modes you can add. If you do the Ruby Final project, you'll use the “w” mode, to completely write over the existing file. Check out the complete list of modes [here](#).

Next, there's a block that passes in `| line |` as a parameter, which is the variable we can work with to append something to the bottom of the file we just opened.

```
2 line.puts "\r" + "air up bike tires"
```

Inside the block, we use `line` and call `.puts` to add the string “air up bike tires”, which will be appended to the bottom of `test_list.txt`.

We also added “\r”, which is an escape character in Ruby. It stands for return, and will make sure the string we add appears on a new line inside `test_list.txt`.

Once the method has appended the string, it ends.

Check your `test_list.txt` file to see the newly appended line.

```
make pancakes  
walk the cat  
replace water filter  
air up bike tires
```

Here's a hint for how you can use `File.open("test_list.txt", "a")`. With this method, you have the ability to change any file. What if, instead of hardcoding “air up bike tires” inside our method, we passed in a variable from somewhere else? We could add a whole list of todo tasks to a text file. Can you feel your Ruby-powers leveling up? Great work!

Wrapping up

Reading files and writing to files is an essential programming skill, as it allows you to take information from a file, edit it, and add to it. If you want to get a better idea of what is going on under the hood, check out the Ruby-docs. You can find the docs for reading a file [here](#), and for writing to a file [here](#).



Teaching the world how to code.

Company

- [About](#)
- [Stories](#)
- [We're hiring](#)
- [Blog](#)

Resources

- [Articles](#)
- [Schools](#)

Learn To Code

- [Make a Website](#)
- [Make an Interactive Website](#)
- [Learn Rails](#)
- [Ruby on Rails Authentication](#)
- [Learn AngularJS](#)
- [Learn the Command Line](#)
- [Learn SQL](#)
- [SQL: Analyzing Business Metrics](#)
- [Learn Java](#)
- [Learn Git](#)

- [HTML & CSS](#)
- [JavaScript](#)
- [jQuery](#)
- [PHP](#)
- [Python](#)
- [Ruby](#)
- [Learn APIs](#)

[Privacy Policy](#) [Terms](#)

Made in NYC © 2016 Codecademy

English ▼