**Level 6 Descends. The End is Near**

**Objective:**

Well done, well done, well done, well done! Truly, truly, truly, truly, truuuuuuuuly!

Here's the last few minutes of your life in review:

* **Files.** -- Lots of methods exist for editing files and looking around in directories.
* **Arguments.** -- Arguments are a list of things sent into a method, separated by commas.
* **Block Changes.** -- You can use **do** and **end** as another way to make a code block.

Welp, you totally know how to use Ruby now. I mean, you've got down the essentials, right? You just need to keep learning more methods and try out more complex blocks on your way to mastery.

But, there's still one side of Ruby we haven't settled: building your own methods!

*Ahem!* Let's get it over with, then.

Start building our new method with: def load\_comics( path )

> def load\_­comics(pat­h)

..

## In Ruby, Def Leppard Means Define Leppard (a Method)!

### Objective:

Great, you've started it up. You're making your own method! First, you used def, followed by the name of the method. Next came a list of arguments which the method will need in order to work when you need it. Don't worry, this isn't too scary and dangerous!

All we have to do is fill it up with Ruby and finish it up with end. Feel free to quickly copy and paste.

Here's the code:

**def load\_comics( path )**

comics = {}

File.foreach(path) do |line|

name, url = line.split(': ')

comics[name] = url.strip

end

comics

end

Though the indentation isn't strictly necessary, it's a good habit to develop so that people can read your code easily.

> def load\_­comics(pat­h)

.. comics = {}

.. File.forea­ch(path) do |line­|

.... name, url = line.­split(': ')

.... comics[nam­e] = url.s­trip

.... end

.. comics

.. end

=> nil

>

## The Ripened Fruit of Your Own Creation

### Objective:

A new method is born. And now, let us use it! Type comics = load\_comics('/comics.txt').

Then, type next to start using your new comics.

> comics = load\_­comics('/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/"}

> next

>

## Hey, Cool, a Comics Thing

### Objective:

In your Ruby console, look at the earlier code you wrote for the load\_comics method.

What is happening here? You're passing in the path variable as an argument, and you're getting back the comics variable just before the end of the method. Ruby looks for something to return just before a method's end.

A number of methods were used to get the entire job done. See if you can spot them.

* **File.foreach** -- This method opens a file and hands each line of the file to the block. The line variable inside the do...end block took turns with each line in the file.
* **split** -- A method for strings which breaks the string up into an array, removing the piece you pass in. An axe is laid on the colon and the line is chopped in half, giving us the data to be stored in url and name for each comic.
* **strip** -- This quickie removes extra spaces around the url. Just in case.

Right on! Bravo. You've got the comics in a Ruby hash. But, uh, what now? What good is this work, really?

Welp, we can use it to make a page of links. Links are useful and cool, right? First, we'll need to load a little library I've made for you.

Here's the code for loading it: require 'popup'.

> require 'popu­p'

=> true

Success!

>

## Browser Puppetry

### Objective:

Excellent, you've loaded the popup library. It's saved in a file in the Libraries folder.

The popup library contains a bunch of methods I've written which will let you control a popup window here on the Try Ruby site.

For example, try this one: Popup.goto "http://bing.com"

When you've got that entered, checkout the new "Popup" tab that appeared near the top of your console! Click it to see the website you just loaded with the goto method. You can even try another one if you feel like it.

Once you're done investigating, type next to go to the next challenge.

> Popup.goto­ http­://bing.co­m

> next

## Making Links and Spinning Webs

### Objective:

Well now, we've got our own lovely, little popup to manipulate. You can also fill it with your own goodies. We'll start small:

Popup.make {

h1 "My Links"

link "Go to Bing", "http://bing.com"

}

The term h1 (h-one) means a level-one header. In HTML, this is the largest header size available. The term link is exactly what you'd think: a link to website.

When you've built the code (by entering it one line at a time), check the Popup tab to see your results.

And when the road finally calls again, type in next to move forward. The end is close.

> Popup.make­ {

.. h1 "My Links­"

.. link "Go to Bing"­, "http­://bing.co­m"

.. }

>

## Popups Are So Easy, It's Crazy

### Objective:

Looks great! You did it perfectly. So now, let's make a list.

Here's how you make a list with the popup library:

Popup.make do

h1 "Things To Do"

list do

p "Try out Ruby"

p "Ride a tiger"

p "(down River Euphrates)"

end

end

The p method is short for "paragraph". It will ensure each of your list items gets its own paragraph spot.

When you want to move on, next will bring you to the last moments of the course...

> Popup.make­ do

.. h1 "Thin­gs To Do"

.. list do

.... p "Try out Ruby"­

.... p "Ride­ a tiger­"

.... p "(dow­n River­ Euphr­ates)"

.... end

.. end

>

> next

## Spread the Comics on the Table

### Objective:

Your journey has progressed wonderfully. You may think that this may be simple stuff, but keep in mind that you didn't know any Ruby whatsoever just a little while ago!

It's time for the last step. Let's tie it all together, you know? Time to make it all sing together like a very nice set of glistening chimes on the beach in the maginificent sunlight! Yeah, imagery!

Remember: you have already loaded your comics with comics = load\_comics( '/comics.txt' ).

Now, let's make a list of the links to each comic:

Popup.make do

h1 "Comics on the Web"

list do

comics.each do |name, url|

link name, url

end

end

end

This ties each name of a comic to a url, creating a link! You can even click on the links and read the comics in the little window! Smashing! Dashing!

Congrats on completing Try Ruby! You can play through this course again by typing next.

If you're interested in Ruby on Rails, then check out Code School's [Rails for Zombies](https://www.codeschool.com/courses/rails-for-zombies-redux/) course! Or if you want to continue your quest towards Ruby mastery, you can play the first level of [Ruby Bits](https://www.codeschool.com/courses/ruby-bits) for free!

> Popup.make­ do

.. h1 "Comi­cs on the Web"

.. list do

.... comics.eac­h do |name­, url|

...... link name,­ url

...... end

.... end

.. end

>