Ruby: Command Line Arguments with ARGV

Feed your Programs Ruby Arguments via the Command Line

Background

In the same way that we pass arguments to methods in Ruby, we can also pass arguments to whole programs. That's exactly what command line arguments do. Each time we run our Ruby program we can feed in different command line arguments, and get different results.

Why Learn It?

In the Ruby lesson titled Putting the Form in Formatter, you used `gets.chomp` to bring user input into your Ruby program. Using command line arguments adds a new tool to your Ruby toolbelt, which you can use to get user input into your program if you choose.

Using Command Line Arguments

Ruby captures command line arguments with a special array named `ARGV`. When written inside your Ruby program, `ARGV` will take take a command line command that looks like this:

```
ruby testing_argv.rb these are elements in the argv array
```

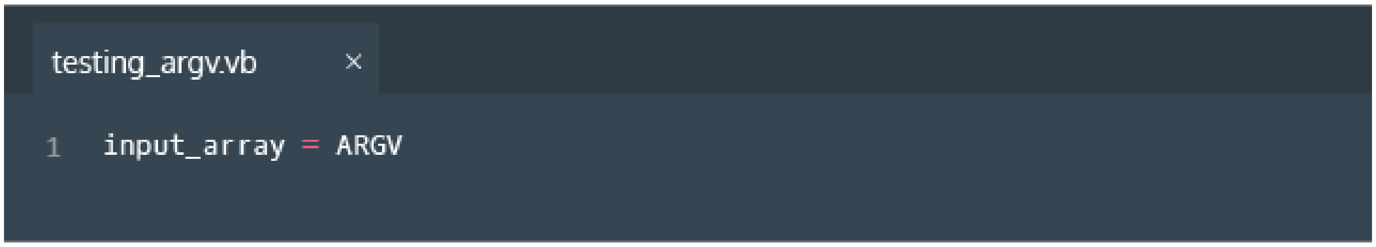
and create an array that looks like this:

```
["these", "are", "elements", "in", "the", "argv", "array"]
```

Now you try!

Trying out ARGV

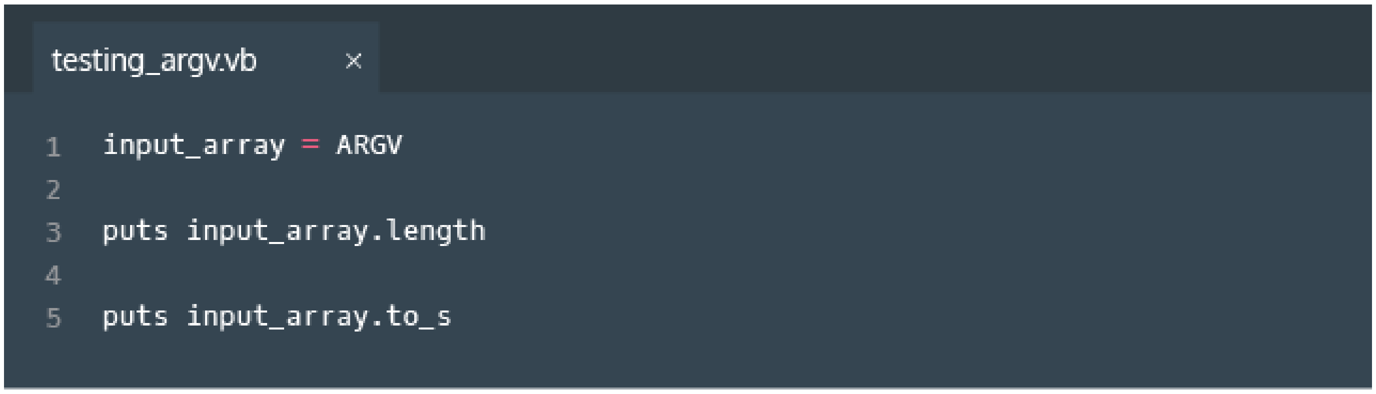
1. In Sublime Text, create a file named `testing_argv.rb` by going to File > New File ...
2. Once you've created your test file, make sure to save it in an easy-to-find location, and let's start writing our code!
3. Set up `ARGV` to capture command line arguments Inside `testing_argv.rb`, assign `ARGV` to an array called `input_array`, which will bind the contents of `ARGV` to a variable.



```
testing_argv.vb
input_array = ARGV
```

Since `ARGV` creates an array, you can call any method on `input_array` that you can call on a normal array. For instance, `puts input_array.length` will return with the number of arguments a user passed.

You can also print the results of the `ARGV` array using puts `input_array.to_s`. The full `testing_argv.rb` file might look something like this:



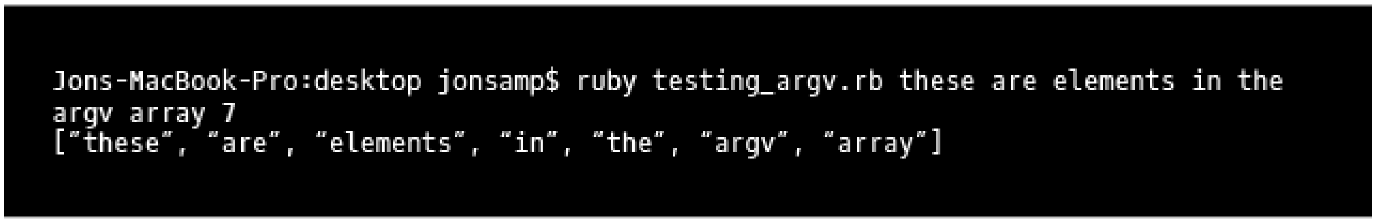
```
testing_argv.vb
input_array = ARGV

puts input_array.length

puts input_array.to_s
```

To test your file, run the program on the command line with multiple arguments following the filename, like this: `ruby testing_argv.rb these are elements in the argv array`

Your command line should tell you there are 7 items in the array, and print them out to your console, like this:
`["these", "are", "elements", "in", "the", "argv", "array"]`



```
Jons-MacBook-Pro:desktop jonsamp$ ruby testing_argv.rb these are elements in the
argv array 7
["these", "are", "elements", "in", "the", "argv", "array"]
```

Using Splat to separate the first argument from the rest

For your program, you may want to use the first command line argument as a command, and then use the rest of the arguments as a string of text that you can do things with. Here's a new Ruby expression that can help you accomplish this simply:
`first_arg, *the_rest = ARGV`

Here, you're assigning two variables at the same time! The comma separates them, and the `*` is called a splat. You've used the splat in a similar way in the Codecademy lesson on Ruby Methods, Blocks and Sorting. Because of the `*`, Ruby knows to assign `ARGV[0]` to `first_arg`, and the rest of the arguments to the_rest. Let's prove it:

In your program, write:

```
puts first_arg
puts the_rest
```

Then run `ruby testing_argv.rb first and all the rest!` inside your command line.

##How can you use this Technique?##
The command line can be a powerful interface for a program you've written. These methods will enable you to capture user input, and then put it to good use.

For instance, if you built and ATM program with Ruby, you could use command line arguments to call methods and modify your bank account. You could write an ARGV array that takes your first command line argument to perform a method, like "withdraw", and then use the arguments after it, perhaps "$20", to specify how much money to withdraw.

There are infinite ways to use ARGV and splats to get data from the command line and make it do something for you. If you want to get to the bottom of how ARGV works, check out its Ruby-doc here.

codecademy

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

English ▼