ZED SHAW'S HARD WAY SERIES

Learn RUBY the HARD WAY

THIRD EDITION

A Simple and Idiomatic Introduction to the Imaginative World of Computational Thinking with Code

ZED A. SHAW

LEARN RUBY THE HARD WAY

Third Edition

EXERCISE 24

More Practice

You are getting to the end of this section. You should have enough Ruby "under your fingers" to move on to learning about how programming really works, but you should do some more practice. This exercise is longer and all about building up stamina. The next exercise will be similar. Do them, get them exactly right, and do your checks.

ex24.rb

```ruby
puts "Let's practice everything."
puts 'You\'d need to know \'bout escapes with \\ that do \n newlines and \t tabs.'

poem = <<END
\tThe lovely world
with logic so firmly planted
cannot discern \n the needs of love
nor comprehend passion from intuition
and requires an explanation
\n\t\twhere there is none.
END

puts "--------------"
puts poem
puts "--------------"


five = 10 - 2 + 3 - 6
puts "This should be five: #{five}"

def secret_formula(started)
  jelly_beans = started * 500
  jars = jelly_beans / 1000
  crates = jars / 100
  return jelly_beans, jars, crates
end


start_point = 10000
beans, jars, crates = secret_formula(start_point)

puts "With a starting point of: #{start_point}"
puts "We'd have #{beans} beans, #{jars} jars, and #{crates} crates."

start_point = start_point / 10
```

What You Should See

Exercise 24 Session

```
$ ruby ex24.rb
Let's practice everything.
You'd need to know 'bout escapes with \ that do \n newlines and \t tabs.
--------------
  The lovely world
with logic so firmly planted
cannot discern
 the needs of love
nor comprehend passion from intuition
and requires an explanation

          where there is none.
--------------
This should be five: 5
With a starting point of: 10000
We'd have 5000000 beans, 5000 jars, and 50 crates.
```

Study Drills

1. Make sure to do your checks: read it backward, read it out loud, and put comments above confusing parts.
2. Break the file on purpose, then run it to see what kinds of errors you get. Make sure you can fix it.

Common Student Questions

Why do you call the variable `jelly_beans` but the name beans later?
That's part of how a function works. Remember that inside the function the variable is temporary. When you return it, then it can be assigned to a variable for later. I'm just making a new variable named beans to hold the return value.

What do you mean by reading the code backward?
Start at the last line. Compare that line in your file to the same line in mine. Once it's exactly the same, move up to the next line. Do this until you get to the first line of the file.

Who wrote that poem?
I did. Not all of my poems suck.

Even More Practice

We're going to do some more practice involving functions and variables to make sure you know them well. This exercise should be straightforward for you to type in, break down, and understand.

However, this exercise is a little different. You won't be running it. Instead, *you* will import it into Ruby and run the functions yourself.

ex25.rb

```ruby
module Ex25

  # This function will break up words for us.
  def Ex25.break_words(stuff)
    words = stuff.split(' ')
    return words
  end

  # Sorts the words.
  def Ex25.sort_words(words)
    return words.sort
  end

  # Prints the first word after shifting it off.
  def Ex25.print_first_word(words)
    word = words.shift
    puts word
  end

  # Prints the last word after popping it off.
  def Ex25.print_last_word(words)
    word = words.pop
    puts word
  end

  # Takes in a full sentence and returns the sorted words.
  def Ex25.sort_sentence(sentence)
    words = Ex25.break_words(sentence)
    return Ex25.sort_words(words)
  end

  # Prints the first and last words of the sentence.
  def Ex25.print_first_and_last(sentence)
    words = Ex25.break_words(sentence)
    Ex25.print_first_word(words)
    Ex25.print_last_word(words)
  end

  # Sorts the words then prints the first and last one.
  def Ex25.print_first_and_last_sorted(sentence)
    words = Ex25.sort_sentence(sentence)
    Ex25.print_first_word(words)
    Ex25.print_last_word(words)
  end

end
```

First, run this with `ruby ex25.rb` to find any errors you have made. Once you have found all of the errors you can and fixed them, you will then want to follow the *What You Should See* section to complete the exercise.

What You Should See

In this exercise we're going to interact with your `ex25.rb` file inside the `irb` interpreter you used periodically to do calculations. You run `irb` from the terminal like this:

```
>>>
$ irb
irb(main):001:0>
```

Your output should look like mine, and after the > character (called the prompt) you can type Ruby code in and it will run immediately. Using this I want you to type each of these lines of Ruby code into `irb` and see what it does:

Exercise 25 Ruby Session

```
require "./ex25.rb"

sentence = "All good things come to those who wait."
words = Ex25.break_words(sentence)
words
sorted_words = Ex25.sort_words(words)
sorted_words
Ex25.print_first_word(words)
Ex25.print_last_word(words)
words
Ex25.print_first_word(sorted_words)
Ex25.print_last_word(sorted_words)
sorted_words
sorted_words = Ex25.sort_sentence(sentence)
sorted_words
Ex25.print_first_and_last(sentence)
Ex25.print_first_and_last_sorted(sentence)
```

Here's what it looks like when I work with the ex25.rb module in irb:

Exercise 25 Ruby Session

```
>> require "./ex25.rb"
=> true
>>
?> sentence = "All good things come to those who wait."
=> "All good things come to those who wait."
>> words = Ex25.break_words(sentence)
=> ["All", "good", "things", "come", "to", "those", "who", "wait."]
>> words
=> ["All", "good", "things", "come", "to", "those", "who", "wait."]
>> sorted_words = Ex25.sort_words(words)
=> ["All", "come", "good", "things", "those", "to", "wait.", "who"]
>> sorted_words
=> ["All", "come", "good", "things", "those", "to", "wait.", "who"]
>> Ex25.print_first_word(words)
=> nil
>> Ex25.print_last_word(words)
wait.
=> nil
>> words
=> ["All", "good", "things", "come", "to", "those", "who"]
>> Ex25.print_first_word(sorted_words)
=> nil
>> Ex25.print_last_word(sorted_words)
who
=> nil
>> sorted_words
=> ["All", "come", "good", "things", "those", "to", "wait."]
>> sorted_words = Ex25.sort_sentence(sentence)
=> ["All", "come", "good", "things", "those", "to", "wait.", "who"]
>> sorted_words
=> ["All", "come", "good", "things", "those", "to", "wait.", "who"]
>> Ex25.print_first_and_last(sentence)
wait.
=> nil
>> Ex25.print_first_and_last_sorted(sentence)
who
=> nil
```

As you go through each of these lines, make sure you can find the function that's being run in ex25.rb and you understand how each one works. If you get different results or errors, you'll have to fix your code, exit irb, and start over.