The first thing you want to do when creating an interactive Caesar Cipher program is to prompt the user to type in the message. So, the first line of code should be:  
puts “What message would you like to encrypt?”

Then, you want to set the user input as a variable. In this case, I chose “message”. So, the next line of code would be:

Message = gets.chomp

Here’s the tricky part now. What I did was turn the word into an array of its letters. This is where the “.chars” method comes in handy. The “.chars” method does exactly what I need, which is to turn a word into an array of its letters. Next, you want to turn the letters in the array into their respective ascii values, so that I can shift them. You need the ascii values in order to shift the letters, as you can’t perform operations (addition, subtraction, etc.) involving integers and strings. In order to do this, I would call upon the “.map” method, which essentially allows me to change the elements within an array. Now, bear with me here, as this gets a little tricky. The “.ord” method changes letters into their respective ascii values, however we can’t just call it upon our “message.chars.map”. The way to call the “.ord” is to do this:

{ |c| c.ord}

This means that all values in the array, represented by “c”, will have the “.ord” method called upon them. Now, we have to create an array to nest this data in, which I will call “asci”. The third line of our Caesar cipher will look like this:

ascii = message.chars.map { |c| c.ord }

Now, you want the program to prompt the user to type in the shift. The fourth line of the code is as follows:

puts “What would you like to shift left by?”

You want to set this shift to a variable, making the fifth line of code:

shift = gets.chomp

Now that you know the shift, what you need to do is create a “shifted” array, which imposes the shift upon the ascii array. Since you’re changing the array, you need to call the “.map” method upon the ascii array. In order to impose the shift, you need to do this:

{ |c| c – shift.to\_i }

Again, the “c” represents every value inside the array, and the shift, being turned into an integer, is subtracted from every value inside the array. The sixth line of code is:

shifted = ascii.map { |c| c – shift.to\_i }

Finally, you want the program to return the decrypted message. In order to do this, you need to “puts” the shifted array, however we need to turn the numbers back into letters. To do this, you need to do:

{ |c| c.chr }.join

Again, “c” represents every value inside the array, and the “.chr” method turns ascii numbers back into their respective letters. The “.join” method puts the array back into a word, making it easier for the user to read. The seventh and final line is as follows:  
puts shifted.map { |c| c.chr }.join

And the final program is:

puts "What message would you like to encrypt?"

message = gets.chomp

ascii = message.chars.map { |c| c.ord }

puts "What would you like to shift left by?"

shift = gets.chomp

shifted = ascii.map { |c| c - shift.to\_i }

puts shifted.map { |c| c.chr }.join

Sources:

<http://codereview.stackexchange.com/questions/35317/caesar-cipher-in-ruby>

<https://gist.github.com/matugm/db363c7131e6af27716c>

https://www.blackbytes.info/2015/03/caesar-cipher-in-ruby/