**User Input**

So far in the course we have only written code that executes without any input from a user. In order to make our programs interactive, we'll need to learn how to allow the user to interact through their keyboard.

**Getting Input**

The built in method we'll use to allow a user to give input is gets. The gets method is unique in that when it is called, it will halt execution of the code and allow the user to type characters until they press enter on their keyboard. Once enter is hit, the gets method will return a string containing the user's keystrokes. Let's take a look at gets in action:

p "Enter your name:"

name = gets

p "hello " + name

The program above will ask a use to enter their name and then print "hello *name*". Try putting this code in a file and running it to see for yourself!

**Dealing with New Lines**

When using gets, the string returned represents the keys typed by the user. However, since the user presses enter to end their input, this will add a newline character at the end of the string. \n is how we represent the newline character in programming. You'll notice that every string returned from gets will end in \n as a result of this. We should be very aware of this extra character when using gets. Here's a common mistake:

puts "Enter 'yes' or 'no'"

response = gets

if response == "yes"

puts "yup!"

elsif response == "no"

puts "nah!"

else

puts "I'm sorry, my responses are limited."

end

As this codes stands, if the user enters a valid response of *yes* the conditional would not be able to catch this. This is because gets will really return "yes\n". "yes" is not equal to "yes\n", bummer.

**Chomping New Lines**

To fix the issue in the previous code, we can use a string method, chomp to remove all newline chars (\n) at the end of a string by returning a new string. Note that chomp is just a plain string method:

my\_string = "yes\n"

p my\_string # "yes\n"

p my\_string.chomp # "yes"

Since gets returns a string, let's chomp it in our old example. Here is the correct code:

puts "Enter 'yes' or 'no'"

response = gets.chomp

if response == "yes"

puts "yup!"

elsif response == "no"

puts "nah!"

else

puts "I'm sorry, my responses are limited."

end

When the user responds with *yes* or *no*, the code above will run accordingly.

**Getting Numbers**

Another common mistake happens when we try to get number input from the user. Take a look at this faulty code:

puts "Enter a number: "

num = gets.chomp

puts 2 \* num #TypeError: String can't be coerced into Integer

When the user enters a "number", the code will get an error because gets will always return a string of characters. So if the user intended to enter the number 42, num would really be the string "42".

**Converting Strings to Numbers**

To fix the previous error we'll use the to\_i on strings. This method will convert a string **to** an **i**nteger:

numeric\_string = "42"

p numeric\_string # "42"

p numeric\_string.to\_i # 42

Let's apply this to the last example:

puts "Enter a number: "

num = gets.chomp.to\_i

puts 2 \* num

If the user enters 42, the program will correctly print 84.