THITH MEANTH WAR!

What You'll Be Building

Now that we can direct our program using `if` / `else` statements, we can produce different results based on different user input.

In this project, we'll combine control flow with a few new Ruby string methods to Daffy Duckify a user's string, replacing each `"s"` with `"th"`.

Instructions

1.
Click Run to see the Daffy Duckifier in action and to start building your own!

script.rb

```
print "Thtring, pleathe!: "
user_input = gets.chomp
user_input.downcase!

if user_input.include? "s"
  user_input.gsub!(/s/, "th")
else
  puts "Nothing to do here!"
end

puts "Your string is: #{user_input}"
```

Getting User Input

First, we should `print` a statement to prompt the user for input, then set that input to a variable using `gets.chomp`.

Instructions

1.
Use `print` to ask the user for input.

Declare a variable called `user_input` and set it equal to the user's input using `gets.chomp`.

Hint
Note: Be sure to give input for the terminal. When it expects input but never receives it an error will be displayed after some time. This prevents it from running indefinitely.

script.rb

```
print "Thtring, pleathe!: "
user_input = gets.chomp
```

Downcase!

We want to make sure we capture both `"S"` and `"s"` in the user's input. We could write separate `if` / `else` statements to handle this, but we can also use `.downcase!` to convert the user's input to all lower case. That way, we only have to search for `"s"`.

Instructions

1.
Call the `.downcase!` method on `user_input`. Make sure to include the `!` so that the user's string is modified in-place; otherwise, Ruby will create a copy of `user_input` and modify that instead.

After running the code, make sure to enter a string in the terminal.

script.rb

```
print "Thtring, pleathe!: "
user_input = gets.chomp
user_input.downcase!
```

Setting Up the 'If' Branch, Part 1

All right! Time to add in a little control flow.

For the `if` half of our branch, we want to check whether the user's input contains an `"s"`.

```
if string_to_check.include? "substring"
```

We can do that using Ruby's `.include?` method, which evaluates to `true` if it finds what it's looking for and `false` otherwise.

(As a general rule, Ruby methods that end with `?` evaluate to the boolean values `true` or `false`.)

Instructions

1.
We want to check `user_input` for the substring `"s"`.

Write an `if` statement in the editor. It should check to see if `user_input` includes `"s"`.

For now, `print` a string of your choice to the console if it finds `"s"`.

After running the code, make sure to enter a string in the terminal.

Hint
Feel free to peek back at the first exercise if you need help.

script.rb

```
print "Thtring, pleathe!: "
user_input = gets.chomp
user_input.downcase!

if user_input.include? "s"
  print "A s was found"
end
```

Setting Up the 'If' Branch, Part 2

Good! Now let's complete our `if` statement.

When we find `"s"`, we want Ruby to replace every instance of `"s"` it finds with `"th"`. We can do this with the `.gsub!` method, which stands for **g**lobal **sub**stitution.

The syntax looks like this:

```
string_to_change.gsub!(/s/, "th")
```

When we get to later lessons, we'll explain why the `/s/` has to be between slashes instead of between quotes. Note: you *cannot* put a space between `gsub!` and the bit in parentheses.

Remember, you want the `!` at the end of the method name so that Ruby will change the string in-place.

Instructions

1.
Remove the print statement you added to your `if` statement and replace it with a call to `.gsub!` on `user_input`. Have it replace `/s/` with `"th"`.

After pressing **Run**, enter a message in the console and hit enter to check your code!

script.rb

```
print "Thtring, pleathe!: "
user_input = gets.chomp
user_input.downcase!

if user_input.include? "s"
  user_input.gsub!(/s/, "th")
end
```

Setting Up the 'Else' Branch

The hard part's over! Now we just need to let the user know if we don't find any instances of the letter "s".

Instructions

1.
Add an `else` statement that displays a string to the user to let them know if there are no "s"s in their string.

script.rb

```
print "Thtring, pleathe!: "
user_input = gets.chomp
user_input.downcase!
```

```
if user_input.include? "s"
  user_input.gsub!(/s/, "th")
else
  print ("No s was found!")
end
```

Returning the Final String—Er, "Thtring"

Home stretch—now we want to display the Daffy Duckified string to the user. You can do that using the string interpolation we learned earlier:

```
my_string = "muchachos"
print "Adios, #{my_string}!"
# ==> "Adios, muchachos!"
```

Instructions

1.
Add a `puts` statement that uses string interpolation to show the user their transformed string.

script.rb

```
print "Thtring, pleathe!: "
user_input = gets.chomp
user_input.downcase!

if user_input.include? "s"
  user_input.gsub!(/s/, "th")
else
  print ("No s was found!")
end

puts "Your Dickified string is #{user_input}"
```

Congratulationth!

Great work!

How might you improve this project? You could:

- Add an additional `if` statement to re-prompt the user for input if they don't enter anything
- Think about how you might account for words in which the letter "c" sounds like an "s"
- Think about how you might preserve the user's original capitalization

Instructions

1.

Enough pondering for now. When you're ready, click Run to complete this project.

script.rb

```
print "Thtring, pleathe!: "
user_input = gets.chomp
user_input.downcase!

if user_input.include? "s"
  user_input.gsub!(/s/, "th")
else
  print ("No s was found!")
end

puts "Your Dickified string is #{user_input.capitalize!}"
```