String Manipulation with Ruby

String Information

```
awesomeString = "Hello World"
```

Is the string empty?

```
awesomeString.empty?
>false
```

How many characters is the string?

```
awesomeString.length
```

String Information

```
awesomeString = "Hello World"
```

Concatenation: bringing two strings together

```
awesomeString = "Hello"+ "World"
```

No + necessary, as long as it's just two strings (no variables):

```
awesomeString = "Hello" "World"
```

String Concatenation II

Chains

```
awesomeString = "Hello" << "World"
```

Concat method

```
awesomeString = "Hello".concat("World")
```

String Interpolation

An easy way to put a variable inside a string:

```
awe = "Hello"
run = "World"
u = "Well #{awe}, #{run}!"
```

Any Ruby code can be put in between the curly braces, but typically variables are

Accessing String Elements

Loading specific parts of the string:

```
awesomeString[0]
>"H"
awesomeString[0,3]
>"Hel"
awesomeString[0..5]
>"Hello"
awesomeString[2..-2]
>"llo Worl"
```

Replacing Words

```
yourString = "Hello World!"
yourString["World"] = "Universe"
yourString
>"Hello Universe"
```

Substitution, Repeating

Substitution with gsub

```
yourString = "Hello World!"
yourString.gsub "Universe", "World"
```

gsub can also be used with a Regular Expression, a built-in Ruby pattern matcher:

```
yourString.gsub /./, "World"
```

Check out http://rubular.com/ for a great guide and RegEx tester.

Repeating Strings

```
yourString * 3
>Hello World!Hello World!Hello World!
```

Inserting Text

```
yourString = "Hello World!"
yourString.insert 5, " to the"
>"Hello to the world!"
```

Chomp and Chop, Reverse

```
yourString = "Hello World! H"

yourString.chop
> "Hello World! "
myString = "Hello World\n"
myString.chomp
> "Hello World"

yourString.reverse
> "!dlroW olleH"
```

Capitalization

```
.upcase
.downcase
.swapcase
.capitalize
```

Here Documents

Heredocs are free-format strings.

They allow you to specify long strings easily:

```
yourText = <<DOC
"Hello Sir,
I know you enjoy learning about programming."
DOC
```

Here Documents

You can use whatever word you'd like after the <<:

```
yourText = <<SOMEWORD
"Hello Sir,
I know you enjoy learning about programming."
SOMEWORD
```

Exercise

Create a program to analyze a block of text supplied in a heredoc. The program should be encapsulated in a function that returns a hash of results like so:

```
{words: 323, spaces: 100, vowels: 1003, consonants: 2232, most_used: "the"}
```

Feel free to add more dimensions of analysis to your results.