BEWD 10

LESSON 2

4 LEARNING GOALS

GIT TIME - BRANCH CREATION

SCALARS - EXPLORE STRINGS

COLLECTIONS - EXPLORE ARRAYS

GIT TIME

GIT TIME - PART 1

STEP 1 - CHANGE TO BEWD_SF_10 DIRECTORY

```
A - Change directories
    - `cd` to your `bewd_sf_10`

B - Make sure you in your `bewd_sf_10`
    - `pwd` your directory should end with `bewd_sf_10`
```

STEP 2 - COMMIT LESSON_ONE CHANGES

```
A - Check your branch
    - `git branch`

B - Check to see what's been staged on the current branch
    - `git status`

    Have notes that you'd like to keep from lesson_one?
    - Add it with: `git add . `
    - Commit it with: `git commit -m "Lesson One Notes"

C - Create a remote branch on github for lesson_one
      -git push origin +lesson_one
```

GIT TIME - PART 2

STEP 1: CHECKOUT YOUR MASTER BRANCH

```
git checkout master
 - checkout (or change to) your master branch
```

STEP 2: PULL THE LATEST VERSION OF `UPSTREAM`

```
git pull upstream master
 - pulls the latest version from the `mother_ship`
```

STEP 3: PUSH THE LATEST TO YOUR FORKED VERSION

```
git push origin +master
 - pushes the latest version from the upstream to your forked version
```

STEP 4: CREATE LESSON_TWO BRANCH

```
git branch lesson_two
 - creates a new branch called lesson_two

git checkout lesson_two
 - changes your current branch to the `lesson_one` branch
```

GIT TIME

1 - WHAT IS GIT BRANCH

2 - HOW TO CREATE A BRANCH LOCALLY

3 - HOW TO CREATE A BRANCH REMOTELY

SCALAR

<objects>

SCALAR

Numbers | **Strings**| Symbols

STRINGS: LEARNING GOALS

1 - WHAT IS A STRING

2 - HOW TO CONSTRUCT A STRING LITERAL

3 - HOW TO USE 5 STRING METHODS

4 - HOW TO DISCOVER MORE STRING METHODS USING RUBY DOCS

SCALAR OBJECTS

- *SCALAR* MEANS ONE DIMENSIONAL
- OBJECTS THAT REPRESENT A SINGLE VALUE
- THERE ARE 3 SCALAR TYPES
  - STRING . NUMERIC . SYMBOL

STRING	`"hello"`
NUMERIC	`12345`
SYMBOL	`:name`

WHAT'S A STRING?

A STRING IS AN OBJECT THAT REPRESENTS A BODY OF TEXT OF ARBITRARY CONTENT & LENGTH

```
- "This is a String"
```

STRINGS

5 COMMON STRING METHODS

```
1 - .length
2 - .prepend
3 - <<
4 - .strip
5 - .chars
```

RUBY DOCS FOR THE STRING CLASS

HTTP://RUBY-DOC.ORG/CORE-2.2.2/STRING.HTML

STRINGS

DISCOVERING NEW METHODS VIA RUBY DOCS

```
#Find these two methods within the Ruby Docs
#Let's try to use them

- start_with?
- include?
```

RUBY DOCS FOR THE STRING CLASS

HTTP://RUBY-DOC.ORG/CORE-2.2.2/STRING.HTML

CODE ALONG

<strings>

COLLECTION

<objects>

COLLECTIONS

- *CONTAINER FILLED WITH OBJECTS*
- THERE ARE TWO KINDS
  - ARRAYS & HASHES

ARRAY

```
cars = ["tesla", "ford", "bugatti"]
```

HASH

```
cars["tesla"] = {year: 2016, model: "Model X", price: "80000"}
```

ARRAYS: LEARNING GOALS

1 - WHAT IS AN ARRAY

2 - HOW TO CREATE AN ARRAY

3 - HOW TO USE 5 ARRAY METHODS

4 - DISCOVER METHODS VIA RUBY DOCS

WHAT'S AN ARRAY?

AN ARRAY IS AN ORDERED COLLECTION

```
rock_stars = ["Beyonce", "Beatles", "Jay-Z", "Kanye West", "Taylor Swift"]
```

zero-index

```
rock_stars[0]
  "Beyonce"
```

ARRAYS

3 WAYS TO CREATE AN ARRAY

```
1 - Via Instantiation
    Array.new

2 - Using the Literal Array Constructor (square brackets)
    rock_stars = []

3 - Special %w{...} notation
     %w{testing hello}
```

ARRAYS

COMMON METHODS

```
1 - .size
2 - .push or <<
3 - .pop
4 - .unshift
5 - .shift
6 - .uniq and uniq!
7 - .include?
```

RUBY DOCS FOR THE ARRAY CLASS

HTTP://RUBY-DOC.ORG/CORE-2.2.2/ARRAY.HTML

ARRAYS

LET'S DISCOVER NEW METHODS

RUBY DOCS FOR THE STRING CLASS

HTTP://RUBY-DOC.ORG/CORE-2.2.2/ARRAY.HTML

LET'S CODE!

CODE ALONG

CODE CHALLENGE!

REVERSE IT!!

REVERSE IT!

- WRITE OUR OWN 'REVERSE' METHOD
-USE IT TO DETERMINE IF A WORD IS A PALINDROME

RUBY DOCS FOR THE STRING CLASS

HTTP://RUBY-DOC.ORG/CORE-2.2.2/STRING.HTML

KEYS TO SUCCESS

- ONE BRICK AT TIME
- DEBUG WITH PRY EVERY TIME
- CODE PROLIFICALLY

REVERSE IT!

```ruby
def my_reverse(string)
  char = string.downcase.chars
  word = ""
  until char.length == 0
    word << char.pop
  end
  word.capitalize
end

def is_palindrome?(word)
  if word.downcase == my_reverse(word).downcase
    "Yay! A Palindrome!"
  else
    "Shucks, Not A Palindrome"
  end
end

#####
puts "Please provide a word \n"
word = gets.strip

puts my_reverse(word)
puts is_palindrome?(word)
```

LET'S BUILD IT!

CODE ALONG

HOMEWORK

1 - WELL GROUNDED RUBYIST - CHAPTER 6 THRU 9

2 - COMPLETE REVERSE IT ON YOUR OWN!