



- ▶ Getting Started Overview
- ▶ W0M: Overview and Introduction to Software Engineering (Week 0, Monday Aug. 31)
- ▶ W0W: Beautifully Engineered Software, Plan & Document vs Agile (Week 0, Wednesday Sept. 2)
- ▶ W1W: Introduction to Ruby (Week 1, Wednesday Sept. 9)
- ▶ W2W: More Ruby and Intro to BDD & TDD (Week 2, Wednesday Sept. 16)
- ▼ W3M: SaaS

HW 1-1: FUN WITH STRINGS (100/100 points)

Specs: `spec/fun_with_strings_spec.rb`

In this problem, you'll implement three functions that perform basic string processing. You can start from the template

`fun_with_strings.rb`

Part A — Palindromes:

A palindrome is a word or phrase that reads the same forwards as backwards, ignoring case, punctuation, and nonword characters. (A "nonword character" is defined for our purposes as "a character that Ruby regular expressions would treat as a nonword character".)

You will write a method `palindrome?` that returns true if and only if its receiver is a palindrome.

As you can see in the template `fun_with_strings.rb`, we arrange to mix your method into the `String` class so it can be called like this:

```
"redivider".palindrome?    # => should return true
"adam".palindrome?         # => should return false or nil
```

Your solution shouldn't use loops or iteration of any kind. Instead, you will find regular-expression syntax very useful; it's reviewed briefly in the book, and the website rubular.com lets you try out Ruby regular expressions "live". Some methods that you might find useful (which you'll have to look up in Ruby documentation, ruby-doc.org) include:

`String#downcase`, `String#gsub`, `String#reverse`.

The spec file contains a number of test cases. At a minimum, all should pass before you submit your code. We may run additional cases as well.

Part B — Word Count: Define a function `count_words` that, given an input string, return a hash whose keys are words in the string and whose values are the number of times each word appears:

Architecture and REST (Week 3, Monday Sept. 21)

ESaaS Ch. 2.1-2:
The Web as a
Client-Server
System; TCP/IP
intro (13:25)

ESaaS Ch. 2.3:
HTML+CSS (9:33)

ESaaS Ch. 2.4:
3-tier shared-
nothing
architecture &
scaling (11:53)

ESaaS Ch. 2.5:
Model-
View-Controller
(8:06)

Homework 1: More Ruby (Due Tues. 9/29 at Midnight)

Homework 1 due Oct
06, 2015 at 05:00 UTC

W3.0M - Goals
and Activities for
Week 3, Monday

W3.2M -
Background:
Introduction to
Git and HTML

W3.3M - Activities

W3.4M -
Preparation for
Monday, Sept. 28

```
"To be or not to be" # => {"to"=>2, "be"=>2, "or"=>1, "not"=>1}
```

Your solution shouldn't use for-loops, but iterators like `each` are permitted. As before, nonwords and case should be ignored. A word is defined as a string of characters between word boundaries.

Part C — Anagrams:

An anagram group is a group of words such that any one can be converted into any other just by rearranging the letters. For example, "rats", "tars" and "star" are an anagram group.

Given a space separated list of words in a single string, write a method that groups them into anagram groups and returns the array of groups. Case doesn't matter in classifying string as anagrams (but case should be preserved in the output), and the order of the anagrams in the groups doesn't matter.

No files selected.

On Time

palindrome detection

- should work for simple strings [10 points]

- should be case-insensitive [10 points]

- should ignore nonword characters [10 points]

word count

- should return a hash [5 points]

- works on simple strings [10 points]

- ignores punctuation [5 points]

- works on the empty string [10 points]

- ignores leading whitespace [10 points]

- ignores embedded whitespace [10 points]

anagram grouping

- for "scream cars for four scar creams" [10 points]

- sanity checks

 - should work on the empty string [5 points]

 - should return an array of arrays for nonempty string [5 points]

Finished in 0.01011 seconds

12 examples, 0 failures

SUBMIT URL TO PAIRING VIDEO (SCREENCAST)

► W4M: SaaS

Architecture
and REST
(Week 4,
Monday Sept.
28)

(10 points possible)

Please submit the URL to an unlisted youtube video recording
(screencast) of your pairing session on this assignment below.

?

If you are unable to access YouTube and/or G+, feel free to submit a link
to a video hosted on some other service (such as Zoom).

- ▶ W4W: Rails
Intro (Week 4,
Wednesday
Sept. 30)

- ▶ W5M: Rails
cont. (Week 5,
Monday Oct.
5)

Note: we are hoping to see screencasts with screen sharing plus text
chat, or even better, audio chat, but video from webcams are not
required.

- ▶ W5W:
Enhancing
SaaS with
JavaScript
(Week 5,
Wednesday
Oct. 7)

- ▶ W6M: Agile
Methodology:
Working with
the Customer
(Week 6,
Monday Oct.
12)

- ▶ W6W: BDD
with
Cucumber and
Capybara
(Week 6,
Wednesday
Oct. 14)

- ▶ W7M: TDD
with RSpec
(Week 7,
Monday Oct.
19)
- ▶ W7W: TDD
with RSpec
cont. and
Review So Far
(Week 7,
Wednesday
Oct. 21)
- ▶ W8M: Wrap Up
and
Assessment of
Part 1 (Week 8,
Monday, Oct.
26)
- ▶ W8W: Project
Poster Session
- ▶ W9M:
Introduction to
Part 2 and
Advanced Rails
(Week 9,
Monday Nov.
2)
- ▶ W9W:
Advanced Rails
(Week7,
Wednesday,
Nov. 4)
- ▶ W10M:

Refactoring &
Legacy (Week
10, Monday
Nov. 9)

- ▶ W10W:
Refactoring &
Legacy (Week
10, Nov. 11)
- ▶ W11M: Project
Management
(Week 11,
Monday Nov.
16)
- ▶ W11W: Project
Management
(Week 11,
Wednesday,
Nov. 18)
- ▶ W12M: More
Enhancing
SaaS with
Javascript
(Week 12,
Monday Nov.
23)
- ▶ W13M: Design
Patterns for
SaaS (Week 13,
Monday Nov.
30)
- ▶ W13W: Design
Patterns for
SaaS (Week 13,

Wednesday,
Dec. 2)

- ▶ W14M:
Practical
DevOps:
Deployment,
Upgrades,
Performance,
Security (Week
14, Monday
Dec. 7)
- ▶ W14W:
Practical
DevOps:
Deployment,
Upgrades,
Performance,
Security (Week
14,
Wednesday
Dec. 9)
- ▶ W15M:
EarlyBird
Project Demos
(Monday, Dec.
14)
- ▶ W15W-W16T:
Project Demos
and Final
Exam
(Wednesday-
Friday and
Monday and
Tuesday Dec.
16-18 and Dec
21-22)

- ▶ Bonus Videos

© All Rights Reserved