

Assignment for Module #2: Calculate Maximum Word Frequency

The overall goal of the assignment is to write a Ruby class and work with attributes, methods, hashes, and arrays.

The functional goal of the assignment is to read some text from a file and find the word or words that appear the most. However, the way we are instructed to measure “the words that appear the most” is by

1. finding the highest frequency word(s) per each line
2. finding the highest frequency line(s) that contain the "highest frequency words".

Functional Requirements

1. Write a class called LineAnalyzer that
 - records the location of a line of text in the file
 - analyzes a line of text
 - figures out the highest frequency word(s) on that line and their frequency count
2. Write a driver class called Solution that
 - reads provided ‘test.txt’ file
 - creates an array of LineAnalyzers
 - selects the ones with the highest frequency
 - prints out the results

Getting Started

1. Download and extract the starter set of files. The root directory of this starter set will be referred to as the root directory of your solution.

```
--- student-start
|-- .rspec (important hidden file)
|-- module2_assignment.rb
|-- solution.rb
|-- spec
|   |-- line_analyzer_spec.rb
|   |-- solution_spec.rb
|   '-- spec_helper.rb
'-- test.txt
```

- module2_assignment.rb - your solution must be placed within this file and spread across two classes: LineAnalyzer and Solution.

- spec - this directory contains tests to verify your solution. You should not modify anything in this directory
- solution.rb - a script that you can use to execute your Solution outside the scope of the unit tests
- test.txt - this file contains test data your solution will read and analyze.

2. Install the following gems. You may already have these installed.

```
$ gem install rspec
$ gem install rspec-its
```

3. Run the rspec command to execute the unit tests within the spec directory. This command should be run from the root directory of the project. This should result in several failures until you complete your solution in module2_assignment2.rb.

```
$ rspec
```

```
FFFFFFFFFFFFFFFFFFFF
```

```
Failures:
```

```
1) LineAnalyzer
   Failure/Error: subject { LineAnalyzer.new("test", 1) }
   ArgumentError:
     wrong number of arguments (2 for 0)
   # ./spec/line_analyzer_spec.rb:6:in 'initialize'
   # ./spec/line_analyzer_spec.rb:6:in 'new'
   ...
Finished in 0.04955 seconds (files took 0.10746 seconds to load)
17 examples, 17 failures
...
```

Technical Requirements

1. Implement all parts of this assignment within the module2_assignment.rb file in the root directory of your solution. The grader will load this specific file from this location.
2. Implement a class called LineAnalyzer. The grader will look for a class with this exact name.
3. Implement the following read-only attributes in the LineAnalyzer class. The grader will look for accessor methods with these exact names.

- `highest_wf_count` - a number with maximum number of occurrences for a single word (calculated)
 - `highest_wf_words` - an array of words with the maximum number of occurrences (calculated)
 - `content` - the string analyzed (provided)
 - `line_number` - the line number analyzed (provided)
4. Add the following methods in the `LineAnalyzer` class. The grader will look for methods with these exact names.
 - `initialize()` - taking a line of text (`content`) and a line number
 - `calculate_word_frequency()` - calculates result and stores in attributes
 5. Implement the `initialize()` method to:
 - take in a line of text and line number
 - initialize the `content` and `line_number` attributes
 - call the `calculate_word_frequency()` method.
 6. Implement the `calculate_word_frequency()` method to:
 - calculate the maximum number of times a single word appears within provided content and store that in the `highest_wf_count` attribute.
 - identify the words that were used the maximum number of times and store that in the `highest_wf_words` attribute.
 7. Implement a class called `Solution`. The grader will look for a class with this exact name.
 8. Implement the following read-only attributes in the `Solution` class. The grader will look for accessor methods with these exact names.
 - `highest_count_across_lines` - a number with the value of the highest frequency of a word
 - `highest_count_words_across_lines` - an array with the words with the highest frequency
 9. Implement the following methods in the `Solution` class. The grader will look for methods with these exact names.
 - `analyze_file()` - processes 'test.txt' into an array of `LineAnalyzers`
 - `calculate_line_with_highest_frequency()` - determines which line(s) of text has the highest number of occurrence of a single word
 - `print_highest_word_frequency_across_lines()` - prints the word(s) with the highest number of occurrences and its corresponding line number
 10. Implement the `analyze_file()` method() to:

- Read the 'test.txt' file in lines
 - Create an array of LineAnalyzers for each line in the file
11. Implement the `calculate_line_with_highest_frequency()` method to:
- calculate the highest number of occurrences of a word across all lines and stores this result in the `highest_count_across_lines` attribute.
 - identifies the word(s) that were used with the highest number of occurrences and stores them in `print_highest_word_frequency_across_lines`.
12. Implement the `print_highest_word_frequency_across_lines()` method to
- print the result in the following format

```
The following words have the highest word frequency per line:
["word1"] (appears in line #)
["word2", "word3"] (appears in line #)
```

Self Grading/Feedback

You can self-grade yourself by calculating the result from the simple text file provided and by using the `rspec` unit tests also provided. When the solution has been completed successfully, the `rspec` test result should look as follows.

```
$ rspec
```

```
Finished in 0.02748 seconds (files took 0.16322 seconds to load)
19 examples, 0 failures
```

Submission

Submit an `.zip` archive (other archive forms not currently supported) created with your solution root directory as the top-level. The grader will replace the spec files with fresh copies and will perform a test with a different `test.txt`.

```
--- student-start
|-- .rspec (important hidden file)
|-- module2_assignment.rb
|-- solution.rb
|-- spec
|   |-- line_analyzer_spec.rb
|   |-- solution_spec.rb
|   '-- spec_helper.rb
'-- test.txt
```

Updated: 2015-09-13