For each of the student submissions below, what feedback would you give to help them improve their submission, with a goal of eventually emulating the instructor solution (provided below)? Assume that the student will be able to resubmit their assignment after reviewing this feedback.

#### **Instructor solution:**

```
def combine anagrams(words)
    words.group_by { |word| word.downcase.chars.sort }.values
end
```

Please read all the submissions before beginning to give feedback. Please don't move on to the next page until you have completed all the questions on this page.

### Submission 1

```
1 def combine_anagrams(words)
    output_array = Array.new(0)
 2
 3
    words.each do |word_1|
 4
      temp array = []
 5
      words.each do |word 2|
 6
         if (word_2.downcase.split(//).sort == word_1.downcase.split(//).sort) then
 7
           temp_array.push(word_2)
 8
         end
 9
10
      output_array.push(temp_array)
11
12
    return output_array.uniq
13 end
```

(For our reference: 18.8 326)

The complexity of this algorithm is high.

There are special Enumerable methods that would help make this easier to understand. You do not need to explicitly return at the end in ruby.

## Submission 2

```
1 def combine_anagrams(words)
   words.group_by { |elt| elt.downcase.chars.sort }.values
3 end
```

(For our reference: 6.8 577)

You could be more descriptive with your variable names inside blocks

# Submission 3

```
1 def combine_anagrams(words)
    sort w = Hash.new
 3
    words.each do |w|
      w sort = w.downcase.split("").sort.join.gsub(/(.)\1{2,}/) do |s|
 5
         (s.length.to s + s[0, 1])
 6
      end
 7
      if (not sort_w.has_key?(w_sort)) then
        sort_w[w_sort] = [w]
 9
10
        sort_w[w_sort] = sort_w[w_sort].push(w)
11
      end
12
    end
13
    out array = Array.new
    sort_w.each_key { |ww| out_array.push(sort_w[ww]) }
14
    return out array
16 end
```

(For our reference: 27.2 312)

There are special Enumerable methods that would help make this easier to understand. You do not need to explicitly return at the end in ruby.

## Submission 4

```
1 def combine_anagrams(words)
h = Hash[]
   words.each do |w|
     h[w.downcase.chars.sort.join] = (h[w.downcase.chars.sort.join].to a << w)
5
   return h.to a.collect { |kv| kv[1] }
6
```

(For our reference: 23.3 296)

There are special Enumerable methods that would help make this easier to understand. You do not need to explicitly return at the end in ruby.

## Submission 5

```
1 def combine_anagrams(words)
   return words.group by { |word| word.downcase.chars.sort }.values
3 end
```

(For our reference: 6.8 652)

You do not need to explicitly return at the end in ruby

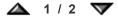
# Submission 6

```
1 def combine_anagrams(words)
   anagrams = {}
3
   words.each do |word|
4
      key = word.split("").sort.join
5
      anagrams[key] ||= []
6
      (anagrams[key] << word)</pre>
7
   return anagrams.values
9 end
```

(For our reference: 9.9 87)

There are special Enumerable methods that would help make this easier to understand. You do not need to explicitly return at the end in ruby.

\* Indicates Response Required



For each of the student submissions, rate the autogenerated hints on a scale of 1 to 5.

- 5 Highly relevant: covers a key concept that should be improved.
- 4 Somewhat relevant: helpful in a lesser way.
- 3 Neutral: not helpful.
- 2 Irrelevant: this hint is irrelevant
- 1 Harmful: this hint is completely wrong and harmful

Please also comment on the degree to which these hints capture the feedback you gave in the previous step.

### Submission 1

```
1 def combine_anagrams(words)
    output_array = Array.new(0)
 3
    words.each do |word 1|
 4
      temp array = []
 5
      words.each do |word 2|
 6
         if (word 2.downcase.split(//).sort == word 1.downcase.split(//).sort) then
 7
           temp array.push(word 2)
 8
 9
      end
10
      output array.push(temp array)
11
12
    return output array.uniq
13 end
```

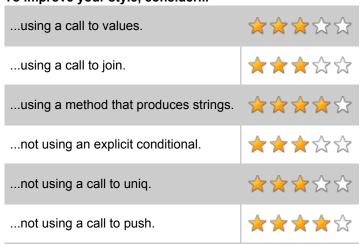
(For our reference: 18.8 326)

### Here's the feedback you gave for this submission:

The complexity of this algorithm is high. There are special Enumerable methods that would help make this easier to understand. You do not need to explicitly return at the end in ruby.

\* Here's the feedback the autograder gave:

To improve your style, consider...



\* Comment here on the degree to which degree these hints capture the feedback you gave this submission.

This code differs from the staff solution significantly. Saying to use/not to use is somewhat irrelevant because it might just

### Submission 2

```
1 def combine_anagrams(words)
   words.group_by { |elt| elt.downcase.chars.sort }.values
3 end
```

(For our reference: 6.8 577)

#### Here's the feedback you gave for this submission:

You could be more descriptive with your variable names inside blocks

\* Here's the feedback the autograder gave:

\* Comment here on the degree to which degree these hints capture the feedback you gave this submission.

Yeah this code is right.

# Submission 3

```
1 def combine_anagrams(words)
 2
    sort_w = Hash.new
 3
    words.each do |w|
 4
      w_sort = w.downcase.split("").sort.join.gsub(/(.)\1{2,}/) do |s|
         (s.length.to_s + s[0, 1])
 6
      end
 7
      if (not sort_w.has_key?(w_sort)) then
 8
         sort_w[w_sort] = [w]
      else
10
         sort_w[w_sort] = sort_w[w_sort].push(w)
11
      end
12
    end
13
    out_array = Array.new
    sort w.each key { |ww| out array.push(sort w[ww]) }
15
    return out_array
16 end
```

(For our reference: 27.2 312)

#### Here's the feedback you gave for this submission:

There are special Enumerable methods that would help make this easier to understand. You do not need to explicitly return at the end in ruby.

\* Here's the feedback the autograder gave:

To improve your style, consider...

...using a call to chars.



...not using a method that produces hashes.





\* Comment here on the degree to which degree these hints capture the feedback you gave this submission.

I'm not sure I understand what 'not using a method that produces hashes' refers to. Everything else makes sense.

### Submission 4

```
1 def combine anagrams(words)
   h = Hash[]
3
   words.each do |w|
     h[w.downcase.chars.sort.join] = (h[w.downcase.chars.sort.join].to a << w)
4
   return h.to_a.collect { |kv| kv[1] }
7 end
```

(For our reference: 23.3 296)

#### Here's the feedback you gave for this submission:

There are special Enumerable methods that would help make this easier to understand. You do not need to explicitly return at the end in ruby.

\* Here's the feedback the autograder gave:

To improve your style, consider...

```
...using a call to new.
...not using a call to collect.
...not using a call to to a.
```

\* Comment here on the degree to which degree these hints capture the feedback you gave this submission.

He's using to\_a to make sure he doesn't get nil in the 4th line. In the 6th line it makes sense.

### Submission 5

```
1 def combine anagrams(words)
   return words.group_by { |word| word.downcase.chars.sort }.values
3 end
```

(For our reference: 6.8 652)

#### Here's the feedback you gave for this submission:

You do not need to explicitly return at the end in ruby

\* Here's the feedback the autograder gave:

No feedback generated	***	

\* Comment here on the degree to which degree these hints capture the feedback you gave this submission.

```
Yeah this code is right
```

### Submission 6

```
1 def combine_anagrams(words)
   anagrams = {}
3
  words.each do |word|
    key = word.split("").sort.join
5
     anagrams[key] ||= []
     (anagrams[key] << word)</pre>
7
   end
   return anagrams.values
9 end
```

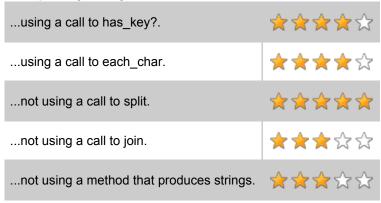
(For our reference: 9.9 87)

#### Here's the feedback you gave for this submission:

There are special Enumerable methods that would help make this easier to understand. You do not need to explicitly return at the end in ruby.

\* Here's the feedback the autograder gave:

To improve your style, consider...



\* Comment here on the degree to which degree these hints capture the feedback you gave this submission.

It captures		

\* Indicates Response Required



Close