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:

```ruby
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

```ruby
def combine_anagrams(words)
  occurance = Hash.new
  words.each do |word|
    normalWord = word.downcase.chars.sort.join
    if occurance[normalWord].nil? then
      occurance[normalWord] = [word]
    else
      occurance[normalWord].push(word)
    end
  end
  retVal = []
  occurance.each { |key, value| retVal.push(value) }
  return retVal
end
```

(For our reference: 16.5 200)

*

Look into the many methods available to you through on ruby Arrays and Hashes. A lot of the heavy lifting done here can be handled in simple method calls.

Submission 2

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

(For our reference: 6.8 471)

*

Almost perfect, but maybe use a more descriptive variable than 'elt'

Submission 3

```
1 def combine_anagrams(words)
2   tmp_hash = Hash.new([])
3   words.each { |word| tmp_hash[word.downcase.split(//).sort] += [word] }
4   tmp_hash.values
5 end
```

(For our reference: 7.0 363)

*

Look into the Array.group_by method to avoid using a temporary hash variable.

Submission 4

```
 1 def combine_anagrams(words)
 2   split_words = []
 3   anagrams = []
 4   result = []
 5   words.each { |w| (split_words << w.downcase.split(//).sort) }
 6   for i in (0..(split_words.length - 1)) do
 7     if anagrams.include?(split_words[i]) then
 8       (result[anagrams.index(split_words[i])] << words[i])
 9     else
10       (anagrams << split_words[i])
11       (result << [words[i]])
12     end
13   end
14   return result
15 end
```

(For our reference: 23.8 179)

*

Consider using a hash to make grouping anagrams together more easy. If you use each 'split_word' as a hash key, grouping becomes much easier (you don't need to manually make comparisons).

Submission 5

```ruby
def combine_anagrams(words)
  hash = {}
  anagrams = []
  words.each do |w|
    sorted = w.downcase.each_char.sort.join
    if hash.has_key?(sorted) then
      hash[sorted].push(w)
    else
      hash[sorted] = Array.new.push(w)
    end
  end
  hash.each_value { |v| anagrams.push(v) }
  return anagrams
end
```

(For our reference: 16.8 467)

*

if you use Hash.new(arg), then arg becomes the default value of undefined keys in the hash. Use this to make grouping easier to read (don't have to check if the key exists, just add values directly).

Submission 6

```ruby
def combine_anagrams(words)
  h = Hash.new
  words.length.times do |i|
    v = 0
    w = words[i].split("").sort!.join("")
    w.length.times { |j| v = (v + w.clone.slice!(j).downcase.ord) }
    if h.key?(v) then
      (h[v] << words[i])
    else
      h[v] = Array.new
      (h[v] << words[i])
    end
  end
  return h.values
end
```

(For our reference: 33.5 594)

*

- Use Hash.new([]) to have a hash's default value be an empty array
- You should downcase words to make sure there are no casing issues
- all of line 6 is very confusing and I honestly don't know what it does.

* 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)
 2   occurance = Hash.new
 3   words.each do |word|
 4     normalWord = word.downcase.chars.sort.join
 5     if occurance[normalWord].nil? then
 6       occurance[normalWord] = [word]
 7     else
 8       occurance[normalWord].push(word)
 9     end
10   end
11   retVal = []
12   occurance.each { |key, value| retVal.push(value) }
13   return retVal
14 end
```

(For our reference: 16.5 200)

Here's the feedback you gave for this submission:

Look into the many methods available to you through on ruby Arrays and Hashes. A lot of the heavy lifting done here can be handled in simple method calls.

*** Here's the feedback the autograder gave:**

To improve your style, consider...

...using a method that produces hashes.	★★★★★
...using a call to has_key?.	★★★☆☆

...not using a call to new.	★★★☆☆
...not using a call to nil?.	★★★☆☆

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

The main issue here is that the student isn't using hashes, which make the problem way easier, so the first hint is the best one. The

Submission 2

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

(For our reference: 6.8 471)

Here's the feedback you gave for this submission:

Almost perfect, but maybe use a more descriptive variable than 'elt'

*** Here's the feedback the autograder gave:**

To improve your style, consider...

No feedback generated	★☆☆☆☆

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

This is almost perfect, so I'm not surprised that there is no feedback generated.

Submission 3

```
1 def combine_anagrams(words)
2   tmp_hash = Hash.new([])
3   words.each { |word| tmp_hash[word.downcase.split(//).sort] += [word] }
4   tmp_hash.values
5 end
```

(For our reference: 7.0 363)

Here's the feedback you gave for this submission:

Look into the Array.group_by method to avoid using a temporary hash variable.

*** Here's the feedback the autograder gave:**

To improve your style, consider...

...using a call to group_by.	★★★★★

...not using a call to each.	★★★★☆
...not using a call to new.	★★★☆☆

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

The first piece of feedback will sort of implicate the second and third pieces of feedback. They are all closely interrelated.

Submission 4

```
 1 def combine_anagrams(words)
 2   split_words = []
 3   anagrams = []
 4   result = []
 5   words.each { |w| (split_words << w.downcase.split(//).sort) }
 6   for i in (0..(split_words.length - 1)) do
 7     if anagrams.include?(split_words[i]) then
 8       (result[anagrams.index(split_words[i])] << words[i])
 9     else
10       (anagrams << split_words[i])
11       (result << [words[i]])
12     end
13   end
14   return result
15 end
```

(For our reference: 23.8 179)

Here's the feedback you gave for this submission:

Consider using a hash to make grouping anagrams together more easy. If you use each 'split_word' as a hash key, grouping becomes much easier (you don't need to manually make comparisons).

*** Here's the feedback the autograder gave:**

To improve your style, consider...

...using a method that produces hashes.	★★★★★
...using a call to has_key?.	★★★★☆
...using a call to each_char.	★☆☆☆☆
...not using a call to split.	★★★★★
...not using a call to index.	★★★★★
...not using a call to length.	★★★★★

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

I think the main issue here is that this person got really tripped up on manually iterating over strings and lists. So any

Submission 5

```
1 def combine_anagrams(words)
2   hash = {}
3   anagrams = []
4   words.each do |w|
5     sorted = w.downcase.each_char.sort.join
6     if hash.has_key?(sorted) then
7       hash[sorted].push(w)
8     else
9       hash[sorted] = Array.new.push(w)
10    end
11  end
12  hash.each_value { |v| anagrams.push(v) }
13  return anagrams
14 end
```

(For our reference: 16.8 467)

Here's the feedback you gave for this submission:

if you use Hash.new(arg), then arg becomes the default value of undefined keys in the hash. Use this to make grouping easier to read (don't have to check if the key exists, just add values directly).

*** Here's the feedback the autograder gave:**

To improve your style, consider...

...using a call to chars.	★★★☆☆
...not using a call to new.	★☆☆☆☆
...not using a call to each_char.	★☆☆☆☆
...not using a call to each_value.	★☆☆☆☆

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

I think this is a submission where the person should be told to 'try to use a method that produces a hash'

Submission 6

```
1 def combine_anagrams(words)
2   h = Hash.new
3   words.length.times do |i|
4     v = 0
```

```
 5     w = words[i].split("").sort!.join("")
 6     w.length.times { |j| v = (v + w.clone.slice!(j).downcase.ord) }
 7     if h.key?(v) then
 8       (h[v] << words[i])
 9     else
10       h[v] = Array.new
11       (h[v] << words[i])
12     end
13   end
14   return h.values
15 end
```

(For our reference: 33.5 594)

Here's the feedback you gave for this submission:

- Use Hash.new([]) to have a hash's default value be an empty array - You should downcase words to make sure there are no casing issues - all of line 6 is very confusing and I honestly don't know what it does.

*** Here's the feedback the autograder gave:**

To improve your style, consider...

...using a call to sort.	★★★★★
...using a call to each.	★★★★★
...using a call to has_key?.	★★★★★
...not using a call to split.	★★★★☆
...not using a call to ord.	★★★★★
...not using a call to sort!.	★★★★★

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

Honestly, this submission is so wonky that any advice is really needed. Their solution goes beyond bad style and actually has a lot

* Indicates Response Required

Close