

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)
2   new = {}
3   words.each do |x|
4     y = x.downcase.split('').sort.join
5     new.has_key?(y) ? (new[y].concat([x])) : (new[y] = [x])
6   end
7   return new.values
8 end
```

(For our reference: 12.7 421)

*

Look up group_by under the enumerable module. Is there a way we can use this to simplify our code?

Submission 2

```
1 def combine_anagrams(words)
2   h = words.group_by { |word| word.chars.sort }
3   print(h.values)
4 end
```

(For our reference: 6.3 596)

*

Do you need to use a variable just to store a result then return it, rather than just returning directly?

Submission 3

```
1 def combine_anagrams(words)
2   anagrams = words.group_by { |word| word.chars.sort }.values
3   p(anagrams)
4 end
```

(For our reference: 6.7 568)

*

In ruby, what is the value returned if there is no explicit return value?

Submission 4

```
1 def combine_anagrams(words)
2   h = {}
3   r = []
4   words.each { |x| h[x.downcase.chars.sort.join] = [] }
5   words.each { |x| (h[x.downcase.chars.sort.join] << x) }
6   h.each { |k, v| (r << h[k]) }
7   return r
8 end
```

(For our reference: 21.0 315)

*

Is there a way we can combine the lines of code that looks like we just copied and modified it slightly?

Submission 5

```

1 def combine_anagrams(words)
2   groups = Hash.new
3   words.each do |word|
4     key = word.downcase.chars.sort.join
5     groups[key] = (groups[key].to_a + [word])
6   end
7   result = []
8   groups.each { |key, value| result = (result + [value]) }
9   result
10 end

```

(For our reference: 15.5 208)

*

Can you think of a better way to order based on key?

Submission 6

```

1 def combine_anagrams(words)
2   sortWords = Array.new
3   agHash = Hash.new
4   agWords = Array.new
5   words.each do |w|
6     x = w.downcase
7     sortWords.push(x.chars.sort { |a, b| a.casecmp(b) }.join)
8   end
9   sortWords.each_index do |index|
10    if agHash.has_key?(sortWords[index]) then
11      agHash[sortWords[index]].push(words[index])
12    else
13      agHash[sortWords[index]] = [words[index]]
14    end
15  end
16  agHash.each_value { |v| agWords.push(v) }
17  return agWords
18 end

```

(For our reference: 27.5 180)

*

Logically, you have 3 distinct steps where you iterate over your three collections. Can you think of a way to combine them; pass the output of one as the input to the next in a cleaner method?

* 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   new = {}
3   words.each do |x|
4     y = x.downcase.split('').sort.join
5     new.has_key?(y) ? (new[y].concat([x])) : (new[y] = [x])
6   end
7   return new.values
8 end
```

(For our reference: 12.7 421)

Here's the feedback you gave for this submission:

Look up group_by under the enumerable module. Is there a way we can use this to simplify our code?

* Here's the feedback the autograder gave:

To improve your style, consider...

...using a call to chars. ★★★★★

...not using a call to split. ★★★★★

...not using a call to concat. ★★★★★

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

Useful hints, not really what I had in my comments though.

Submission 2

```
1 def combine_anagrams(words)
2   h = words.group_by { |word| word.chars.sort }
3   print(h.values)
4 end
```

(For our reference: 6.3 596)

Here's the feedback you gave for this submission:

Do you need to use a variable just to store a result then return it, rather than just returning directly?

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

To improve your style, consider...

...not using a call to print. ★★★★★

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

Pretty much would solve the issues my hint brings up.

Submission 3

```
1 def combine_anagrams(words)
2   anagrams = words.group_by { |word| word.chars.sort }.values
3   p(anagrams)
4 end
```

(For our reference: 6.7 568)

Here's the feedback you gave for this submission:

In ruby, what is the value returned if there is no explicit return value?

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

To improve your style, consider...

...not using a call to p. ★★★★★

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

Looking at how to return things would make me not use a call to p.

Submission 4

```
1 def combine_anagrams(words)
2   h = {}
3   r = []
4   words.each { |x| h[x.downcase.chars.sort.join] = [] }
5   words.each { |x| (h[x.downcase.chars.sort.join] << x) }
6   h.each { |k, v| (r << h[k]) }
7   return r
8 end
```

(For our reference: 21.0 315)

Here's the feedback you gave for this submission:

Is there a way we can combine the lines of code that looks like we just copied and modified it slightly?

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

To improve your style, consider...

...using a call to `each_value`.



...not using a call to `join`.



...not using a method that produces strings.



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

My comment really only address the `each_value` part, but other ideas all good.

Submission 5

```
1 def combine_anagrams(words)
2   groups = Hash.new
3   words.each do |word|
4     key = word.downcase.chars.sort.join
5     groups[key] = (groups[key].to_a + [word])
6   end
7   result = []
8   groups.each { |key, value| result = (result + [value]) }
9   result
10 end
```

(For our reference: 15.5 208)

Here's the feedback you gave for this submission:

Can you think of a better way to order based on key?

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

To improve your style, consider...

...using a call to `split`.



...using a method that produces hashes.



...using a call to `map`.



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



...not using a call to `chars`.



...not using a call to `to_a`.



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

All the hints help; again, not really related to my comments.

Submission 6

```

1 def combine_anagrams(words)
2   sortWords = Array.new
3   agHash = Hash.new
4   agWords = Array.new
5   words.each do |w|
6     x = w.downcase
7     sortWords.push(x.chars.sort { |a, b| a.casecmp(b) }.join)
8   end
9   sortWords.each_index do |index|
10    if agHash.has_key?(sortWords[index]) then
11      agHash[sortWords[index]].push(words[index])
12    else
13      agHash[sortWords[index]] = [words[index]]
14    end
15  end
16  agHash.each_value { |v| agWords.push(v) }
17  return agWords
18 end

```

(For our reference: 27.5 180)

Here's the feedback you gave for this submission:

Logically, you have 3 distinct steps where you iterate over your three collections. Can you think of a way to combine them; pass the output of one as the input to the next in a cleaner method?

* Here's the feedback the autograder gave:

To improve your style, consider...

...using a call to each_char.



...not using a method that produces hashes.



...not using a call to push.



...not using a call to chars.



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

Big one is not using hashes, other ones seem ok.

* Indicates Response Required

