Our Solution(s) Run Code

Your Solutions Run Code

```
Solution 1
             Solution 2
 1 # Copyright © 2020 AlgoExpert, LLC. All rights reserved.
   # O(w * n * log(n) + n * w * log(w)) time | O(wn) space - where
   \# n is the length of the longest word
   def groupAnagrams(words):
        if len(words) == 0:
           return []
8
        sortedWords = ["".join(sorted(w)) for w in words]
9
10
        indices = [i for i in range(len(words))]
        \verb|indices.sort(key=lambda x: sortedWords[x])|\\
11
12
13
        result = []
        currentAnagramGroup = []
14
15
        currentAnagram = sortedWords[indices[0]]
16
        for index in indices:
17
            word = words[index]
            sortedWord = sortedWords[index]
18
19
20
            if sortedWord == currentAnagram:
21
                currentAnagramGroup.append(word)
                continue
23
            result.append(currentAnagramGroup)
24
25
            currentAnagramGroup = [word]
            currentAnagram = sortedWord
26
27
28
        result.append(currentAnagramGroup)
29
30
        return result
```

```
Solution 1 Solution 2 Solution 3

1 def groupAnagrams(words):
2  # Write your code here.
3  pass
4
```

31

the latest part of the latest pa --and a local last residence of the local last residence of ___ THE RESIDENCE OF THE PERSON NAMED IN COLUMN 2 IN COLUM Run or submit code when you're ready. ___ manufacturer, but, but, but, but, but,

_