Programming Project C

Word Game

The file *words.txt* in the folder for this assignment contains about 81,000 words, one per line. You will use it for this assignment as a lexicon of English words.

Write a Python program that chooses a word of a specified length at random from the lexicon and displays it to the user in a random, scrambled order. Then let the user guess words of varying lengths that can be formed using the letters of the word and the same number of occurrences or less as they appear in the word. After every guess, respond to the user whether the guess was correct, and add it to the list of words that have been guessed so far. Print results grouped by length of words and alphabetized within each group.

Here is a sample execution:

```
$ python3 wordgame.py
Enter the range of word lengths (low,high):3,6
renbur:

['---', '---', '---', '---', '---', '---', '---', '---', '---']
['----', '----', '----', '----']
['-----']
['------']

Enter a guess: bur
Correct!

urnrbe:

['---', '---', 'bur', '---', '---', '---', '---', '---', '---']
['----', '----', '----', '----']
['-----']
['------']

Enter a guess: burr
Correct!

ubenrr:

['---', '---', 'bur', '---', '---', '---', '---', '---', '---']
['----', 'burr', '----', '----']
['-----']
['------']

Enter a guess: run
Correct!

rebnur:

['---', '---', 'bur', '---', '---', '---', '---', 'run', '---']
['----', 'burr', '----', '----']
```

```
['-----']
['------']

Enter a guess: **rune**
Correct!

brurne:

['---', '---', 'bur', '---', '---', '---', '---', 'run', '---']
['----', 'burr', '----', 'rune']
['-----']
['------']

Enter a guess: **foo**
Sorry. Try again

renrub:

['---', '---', 'bur', '---', '---', '---', '---', 'run', '---']
['----', 'burr', '----', 'rune']
['-----']
['------']

Enter a guess: **burner**
Correct!

nrrebu:

['---', '---', 'bur', '---', '---', '---', '---', 'run', '---']
['----', 'burr', '----', 'rune']
['-----']
['burner']

Enter a guess: **burn**
Correct!

enurrb:

['---', '---', 'bur', '---', '---', '---', '---', 'run', '---']
['burn', 'burr', '----', 'rune']
['-----']
['burner']

Enter a guess: **urn**
Correct!

nebrur:

['---', '---', 'bur', '---', '---', '---', '---', 'run', 'urn']
['burn', 'burr', '----', 'rune']
['-----']
['burner']

Enter a guess: **q**

['brr', 'bun', 'bur', 'err', 'nub', 'rub', 'rue', 'run', 'urn', 'burn',
'burr', 'rube', 'rune', 'rerun', 'burner']
```

Note that you first prompt the user for the range of word lengths to consider. The largest length requested (6 in this case) determines the length of the word that you will choose randomly from the lexicon ("burner" here). Note also that you shuffle the word on each iteration. Entering the string "q" quits the program. Display each list of same-length words guessed in alphabetical order.

Print all words at the end of the game, as shown above.

If there are no words of a particular length, just ignore them altogether (note no words of length 5 below in "seaway" with a length range of 4 through 6):

```
esaywa:

['away', '----', 'ayes', 'easy', 'sway', 'ways', 'yaws', 'yeas', 'yews']
['------']
```

You may find **itertools.groupby**, **random.choice**, and **random.shuffle** helpful.

Submit your code and a text file containing the transcript of a sample game session.

Hint: It is too time-consuming to check every permutation of subsets of letters for being a word for the game to be efficient. It is better to instead look at every possible word of the appropriate lengths in the dictionary and check to see if the letters are subsets of the sets of letters in the original, full-length word. Using **collections.Counter** makes easy work of this. (**Counter** is Python's version of a "bag" data structure. You can use <, <=, >, >= as needed for checking containment. This functionality is new in Python 3.10.)

If a command-line argument is passed to the program, that argument should be used as the base word. For example, with the command below, the base word would be "stormy":

```
$ python3 wordgame.py stormy
```

Name your Python file `wordgame.py`

Points

5	Gets the minimum and maximum word lengths from the user
10	Uses command-line argument for base word or randomly chooses a base word of the correct length from the dictionary if there are no command-line arguments
5	Scrambles the base word after each guess
20	Gives correct feedback for user's guesses
20	Shows the correct number of words of each length in alphabetical order
20	Keeps track of correct guesses and displays them
20	Prints all words at the end of the game
100	TOTAL