CSCE A101, Python
Assignment 3
Due on Blackboard
10 points

Complete the following programming problem. Be sure to use good style and documentation (worth about 10% of the point total; see the ***DocumetationAndStyle.pdf*** file on Blackboard). When complete, please upload a single .py file to Blackboard.

Wordle Help

Write a program to provide help with the initial guesses for the Wordle puzzle, https://www.nytimes.com/games/wordle/index.html.

For this assignment, you will need to read a list of words from a text file and find the top 5 words that maximize the number of high-frequency letters. I will help you by laying out an incremental develop strategy here.

Step 1. Read in the file that I provide to you, words.txt.

To save the file to disk, right-click the file in Blackboard and select 'Save link as…' and specify a location. Then, read the contents of the file using the following code (you may also want to read zyBooks *Section 12.1, Reading files*):

```
f = open('words.txt', 'r') # open the file for reading
contents = f.read() # read the entire file contents as one big string
f.close() # close the file
```

This code relies on a *relative* path, not an *absolute*, or hardcoded directory in the code (which makes it easier for us to grade). Note that for this code to work, the file, `words.txt`, must be in the current working directory (i.e., same directory as your .py file is located).

Step 2. Extract just the 5 letter words.

After you have read in the file contents, you'll need to get each word (think `.split`) and then subset the list of words to those that have 5 letters (think conditional list comprehension).

Step 3. Calculate the frequency distribution of the letters.

Count up the number of times each letter of the alphabet occurs in all of the 5 letter words. A good way to do this is to use a dictionary where the key is the letter, and the value is the associated count. This can be done using nested loops, where the outer loop iterates over the words, and the inner loop iterates over the characters in each word. The `dict.get` function provides a nice way to update the counts, but you don't have to use it.

To check your work, **print** out the count values for ‘a’, ‘s’, and ‘q’. Hopefully they match mine:

```
a = 1986
s = 2606
q = 47
```

Include these print outs as part of your solution.

Step 4. Calculate a score for each word that is the sum of its letter frequencies.

To do this, loop through all of the 5-letter words and for each word, add up the value associated with each character in the word using the dictionary you created in Step 3. I recommend using another dictionary to store these word-value pairs.

Step 5. Print 5 words that maximize the number of high-frequency letters.

In order to do this, you have to sort your dictionary based on the values, not the keys. One way to do this is to use the `dict.items` function to get all of the (key, value) pairs, and then reverse the order to be (value, key) using a list comprehension. Then, you can use sorted() to sort this list of tuples. Hint: the highest scoring 5-letter word should make you think of many donkeys, and nothing else!

Bonus (3 pts). Print the 5 words with *unique* letters that maximize the number of high-frequency letters.

To do this, you need to print the top 5 words with no duplicate characters. There are many ways to test a word to see if all the characters are unique, but it can be a bit tricky. Feel free to use the code below as part of your Bonus solution.

```
def unique_chars(word):
    '''Return True if all characters in word are unique, else
False'''
    chr_exists = [False] * 128
    for ch in word:
        unicode_val = ord(ch)
        if chr_exists[unicode_val]:
            return False
        chr_exists[unicode_val] = True
    return True
```

There is much more that could be done with this game, like writing a function that displays a subset of words that meet certain letter constraints, e.g. all words with an ‘s’ in the 2nd position. Or, you could suggest a good 2nd guess that has unique letters from the 1st guess. But I think we’ve had enough fun, so I’ll stop here. Feel free to explore further on your own and/or read about other Wordle strategies… https://www.wired.com/story/best-wordle-tips/.

Sample output:

Here is the program sample output.

```
a = 1986
s = 2606
q = 47
Top 5 guesses:
[(12343, 'asses'), (12276, 'eases'), (11854, 'seers'), (11354, 'sense'), (11301, 'sears')]
(10209, 'arose')
(10070, 'laser')
(10070, 'earls')
(10035, 'raise')
(10035, 'arise')
```

Note: This is just a reference output. Though optional, you are encouraged to produce a better output in any aspect such as improved formatting, attractive graphical user interface (GUI) etc.