CSC 110
Introduction to Computer Science

Homework Assignment 5
Work with Basic loops

Due: midnight after Lecture 20

- Note: Homework Assignment 5 is to be completed individually.

Objective

In this homework we are going to use a long string, as well as a few functions with loops to do some "text processing"

The theme of this homework is a **word count program**.

This word search program should be able to:

1. ask a user for a word, and
2. search a list of words and count how many times the user-provided word is in that list (*even within other words*!).

Before you code: A PLAN

I will ask you to **plan before you code.** If you need any help or want to debug, the FIRST THING I will ask you to show me will be your plan.

- Your plan can be a diagram of steps, a set of steps written in bullet-points, even using semi-python notes.
- You should make this before you start editing the program.
- You should have also considered potential errors in your logic before you start editing the program.

Your Tasks:

For this homework you need to complete two tasks:

1) Make the `get_user_word` function with input checking for the requested option.
2) Make the `count_in_text` function.

Before we explain the tasks, pay careful attention to the following notes and tips.

Also, read ALL of these notes and instructions before you start your work.

The Template we provide

We provide a main that has a default text, extracts the list of words from the text, and calls the two functions. Your task is simply to make the functions.

Feel free to check out the provided text in the main as well as the way the words are extracted. This is not something we'll see until we talk about strings and string methods.

Expected printout:

So you understand what we are shooting for, an example run looks like this:

```
Provide a word with 3 or more characters: the
The user selected the word: 'the'
The word 'the' is found 5 times in the text
The End
```

another example is:

```
Provide a word with 3 or more characters: cat
The user selected the word: 'cat'
The word 'cat' is found 0 times in the text
The End
```

another example is:

```
Provide a word with 3 or more characters: no way
Error: You provided more than one word
Provide a word with 3 or more characters: one
The user selected the word: 'one'
The word 'one' is found 2 times in the text
The End
```

Note, in the example above, that the word `one` is only present, in the text, inside the word `gone`, but it still counts.

another example is:

```
Provide a word with 3 or more characters: is
Error: The word is too short
Provide a word with 3 or more characters: ear
The user selected the word: 'ear'
The word 'ear' is found 5 times in the text
The End
```

Note, in the example above, that the word `ear` is only present, in the text, inside the words `Fear` or `fear`, but it still counts.

The DocStrings below every function explain what they do. Do not remove them.

A Note on passing the tests

Remember to run the test file every time you have completed a code block. Any tests are very strict with respect to the format of requested prompts and printouts so pay attention to exactly what is requested and replicate it as exactly as possible.

Using strings and loops with lists

- In this homework, we'll be using a loop to compare a word to a list of other words. remember that the comparison operator is `==`
- to obtain the length of a list (to help you with loop limits or to check word length) you can use the `len` function. For example : `len(word)` returns the length of the word, and `len(word_list)` would return the length of the list.
- to place quotations inside a string, do it like in this example: `"My friend said 'hello' to me"`. You can also do it by **escaping** the quotation symbols, as in: `"My friend said \'hello\' to me"`.
- Check your notes for the different ways to use a for loop. In one way we use range to vary a iteration variable through a set of integers; In another way, we simply have the iteration variable take each of the values contained inside a list (this is the one you'll need). Check chapters 4.4.to 4.6 in our book: How to Think like a Computer Scientist: Interactive Edition

Task 1: get_user_word

The `get_user_word` function asks for a single word from the user that has at least 3 characters

define the function

For this problem, you need to define the function by yourself from scratch. It needs no input parameters but it returns one string. Inside the function, you should have a while loop that asks for a word, checks if it is good, and either a_ returns it if it is ok, or b) runs the iteration again.

The actions inside the while loop should be:

1. ask for a string with the prompt: `"Provide a word with 3 or more characters: "`
2. If the word has a space (`" "`) `in` the word, you should print the message `"Error: You provided more than one word."` and make sure the loop repeats by using `continue` to restart the loop. If it does not have a loop, go to the next check.
3. If the word has fewer than 3 characters, you should print the message `"Error: The word is too short"` and make sure the loop repeats by using `continue` to restart the loop. If the word is not too short, return the word.

Testing

The main has a print statement right after the call to `get_user_word`, so if you see the word printed after the call, it passed the two checks.

Task 2: count_in_text

The `count_in_text` function receives the user word and the word list to be examined and counts how many times the user word is found **within** each of the searched words!

define the function

For this problem, you need to define the function by yourself from scratch. It needs to accept two input parameters, called `user_word` and `word_list` Inside the function, you should have a for loop that runs once per word in the `word_list` and that checks to see if the word that the user provided, or `user_word` is `in` each of the words of the `word_list`.

Note that instead of using equality of strings (`==`) we ask about containment (`in`) because a word like `not` is contained inside the word `nothing` even if they are not equal. We want to count appearances, even if inside another word.

- Be careful to define a `counter` variable that starts at zero and that goes up by 1 every time the `user_word` is found.

The actions inside the for loop should be:

1. Check to see if `user_word` is `in` the current word from the `word_list`
2. if it is, increment the counter.

- After the loop, make sure to print the number of times the word was found in the text using this statement: `print(f"The word '{user_word}' is found {counter} times in the text")`

Your usual issues:

- Remember that indentation matters everywhere!
- Remember that a local variable cannot be seen inside a different function unless you return it
- remember global variables can be seen (read from) inside all functions in the program
- READ the instructionns carefully and ask me over slack if something is unclear

STEPS TO SOLVE THE HOMEWORK

1. PLAN your solution in paper
2. test your paper solution with example values
3. transcribe your paper solution to python in small increments… running the program as you go to make sure you have not introduced syntax errors
4. once a full function is finished, you should run the tests to see if the function makes sense logically.

Grading

- **IMPORTANT**: If your code does not compile, you get a large poitnt reduction, so make sure you run your code often (to avoid syntax or runtime errors) and you always have it in a "running" state, even if it does not

get the desired results.
- [no grade] If you require assistance with debugging, I will ask for a paper version of your functions with the plans for how you wanted to solve each task.
- 100% of your your grade will be the percentage of tests you pass.

Grading criteria:

General

The submission:

- includes a header with the name of any peers and any references (or -10%)
- runs without syntax errors (or -50%)
- uses appropriate, informative variable names (or -10%)
- adds a few small but informative comments (or -10%)

Operations

The program:

- Passes all 8 tests (or lose 12% per failed test)

To pass all tests:

- get_user_word prints the error message for more than one word
- after a word more than one word error, get_user_word should repeat the request for a valid word (because of the loop)
- get_user_word prints the error message for words that are too short
- after a word too short error, get_user_word should repeat the request for a valid word (because of the loop)
- get_user_word should return the user input word if it satisfies the restrictions
- count_in_text prints out the count message with the correct count for the test word 'not'
- count_in_text prints out the count message with the correct count for the test word 'fear'
- count_in_text prints out the count message with the correct count for the test word 'cat'
