Programming Assignment 2

Getting started

Complete the reading and practice assignments posted on the course schedule. Review class handouts and examples. This assignment requires knowledge of String methods and conditional statements along with some math operations that we have already covered.

Programming Project: ShowKeyword **worth: 20 points**

Show keyword in text

In this week's assignment you will be processing text. In particular, the goal of the program is, given a chunk of text (through a file) and a keyword, locate the first occurrence of the keyword in the text, output the sentence in which the keyword appears and then display a schematic representation of the text showing the keyword formatted in a specified way.

I will supply the starting code, which takes care of reading and displaying the contents of a file, in *showKeyword.py.* I will also post the file with the input text used in this handout. Crate a new project and place *showKeyword.py* and the input text files in it. Run *showKeyword.py,* and review its output. Develop the rest of your code in that file.

To describe the program, let's start with a sample interaction.

As usual, boldface denotes the input provided by the user.

```
Please enter the name of the file containing the text: text2.txt
Contents of file text2.txt:
One two three. Four
five, six - seven! Eight, nine.
Ten eleven twelve. Thirteen fourteen fifteen,
Sixteen.

Enter the keyword:four
Enter the line length: 15
------
Outputting the sentence followed by the text schema with 15 characters per
line:
*****
FOUR five, six – seven!
...............
FOUR...........
...............
...............
...............
...............
...............
.
```

Let's review the **requirements**. The program should read the following input:

1. the name of a file, containing the text, located in the same folder as the program (note that your program will be tested on other files as well as the one shown here),
2. a keyword to be found in the text, and
3. an integer (greater than 10) defining number of characters to be displayed per line.

The program should output:

1. The **sentence** of the text containing the first occurrence of the specified keyword (appearing in possibly different combination of capital and lowercase letters). The sentence should appear exactly as in the text with the following exceptions:
   - all line breaks must be replaced with a space,
   - the keyword must be shown in all capitals.
2. The **schematic representation of the text**, in which
   - a dot (.) is displayed instead of every character except for the first occurrence of the keyword,
   - the first occurrence of the keyword is displayed in all uppercase letters,
   - the number of characters per line, not counting the '\n' at the end of each line, equals the line length provided via the third input parameter.

Explanation of interaction: The first interactions describes a basic case, when the word is found inside the text and not on the last line of the schema and does not wrap into the next line. It has user choosing file **`text2.txt`**, looking for keyword **`four`**, which was found in the second sentence. The overall length of the text is 106 characters (this includes the end-of-line chars as well), with the keyword occurring at position 15. The output schema includes a single full line of dots preceding the line with FOUR, because 15//15 = 1. The last line contains 106 % 15 = 1 dot; it is not a full line of dots. There are 106 // 15 = 7 full lines of dots, of which one is before the keyword and one contains the keyword, hence 7 - 1 – 1 = 5 full lines after the line with the keyword.

The following interactions demonstrate three other runs of the program. *All of them use the same text file, the content of which is presented only in the first interaction, for the sake of brevity of this handout.* The length of the text in these interactions is **106** characters (note that this count includes end-of-line characters as well).

The next interaction (note that the content of the file text2.txt is omitted from here and shown in the previous interaction) shows a word found in the end of the sentence:

```
Please enter the name of the file containing the text: text2.txt


Enter the keyword:seven
Enter the line length:20
--------------
Outputting the sentence followed by the text schema with 20 characters per
line:
*****
Four five, six - SEVEN!
....................
............SEVEN...
```

```
..................
..................
..................
......
```

The following interaction shows the keyword spanning multiple lines in the output text schema.

```
Please enter the name of the file containing the text: text2.txt

Contents of file text2.txt:
One two three. Four
five, six - seven! Eight, nine.
Ten eleven twelve. Thirteen fourteen fifteen,
Sixteen.

Enter the keyword:thirteen
Enter the line length:15
--------------
Outputting the sentence followed by the text schema with 15 characters per
line:
*****
THIRTEEN fourteen fifteen, Sixteen.
...............
...............
...............
...............
...........THIR
TEEN...........
...............
.
```

Finally, an interaction that shows what happens when the keyword does not appear in the text:

```
Please enter the name of the file containing the text: text2.txt

Enter the keyword:twenty
'twenty' does not appear in the text
```

Important Notes: regarding the input:

- The keyword will contain no spaces and will never be longer than 11 characters, thus, it will never extend over more than two lines.
- The only non-alphanumeric characters in the text will be the punctuation symbols, namely ,:;-.!?"'

Hints:

- To avoid having to enter the file name every time you test your program, at the beginning stages of developing your program, hardcode it, assigning it to a variable directly in the code. Later on, when you've tested the program on different keywords and line lengths, you can update it to read the file name from the user and test it again.

- There is no method that would search for a string ignoring lettercase, in other words, the call
  `"It was a very good breakfast.".find("GOOD")`
  will return -1. A common technique used in matching text written using different lettercase is to convert the text and the phrase you're looking for to the same lettercase and then do the searching. For example, if you're searching for word "GOOD" in "It was a very good breakfast.", then conversion to lowercase will produce "good" and "it was a very good breakfast.", so find() can be applied successfully to find the starting position of the word "good".

- Consider the following: a call `"this is a sentence i wrote.".find("i")` will return 2, finding "i" in the word 'this'. A trick that can be applied so you find the keyword as a *whole* word: search for the keyword surrounded by spaces (" i ", in the above case).

- The scenario above can be further complicated by the presence of punctuation symbols in the end of words, as when looking for 'cat' in 'There was no catastrophe; we found our cat. It was in the attic.' If we search for 'cat', we won't find it, unless we make sure that punctuation symbols in the text are replaced with spaces. String method *replace*() will do the job. For example, "a.b.c".replace(".", " ") will return "a b c".

Note that the hints and sample interactions here do not present all possible cases of how a keyword may appear inside a text. It is your job to think of them, develop an algorithm that works, and test it thoroughly. Your grade will depend on how many different cases your program handles correctly.

Grading

Your program should compile without syntax errors to receive any credit. If a part of your program is working, you will receive partial credit, but only if the program compiles without syntax errors. The grading schema for this project is, roughly, as follows:

- 2 points will be awarded for correctly handling the input
- 8 points will be awarded for displaying the sentence correctly. Of these, 5 points are allotted to the basic case (word not surrounded by punctuation and in the same lettercase), and 3 points for correctly handling more tricky cases (word in the beginning or end of a sentence, etc.)
- 8 points will be awarded for displaying the text schema correctly. Of these, 6 points are allotted to the basic case when the keyword appears on one internal line, and 2 points for correctly handling more tricky cases (e.g. keyword spanning more than one line, word on the last line of text).
- 2 points will be awarded for good programming style, as defined in handout 1.

Note that you do not need to use loops for this program and solutions that use loops will be penalized.

Created by Tamara Babaian on January 23, 2020