CS 120 (Spring 22): Introduction to Computer Programming II

Long Project #2 - Word Search

due at 5pm, Tue 1 Feb 2022

REMEMBER: The `itertools` and `copy` libraries in Python are **banned.**

1 Overview

Write a program named `word_search.py` .

In this project, we will be implementing a word search. You will read your input from a file: it will contain a grid of letters, a blank line, and then a set of words to find.

You will search the grid for each word. If you find it, you will print out the grid again, but with all letters (except for the word you've found) turned into periods. If you don't find it, you will print a message to that effect.

I will not tell you how large the grid is; you will have to figure that out from the file. I **will** guarantee that all of the lines in the grid will have the same number of characters, but I will **not** guarantee that the grid is a square; there it might be a tall or wide rectangle, sometimes.

2 How to Search

Doing a word search simply requires that you do a more general version of the grid-scan that you did in the Short problem. In fact, I hope that you can re-use several functions from that project (perhaps with small modifications).

To structure your search, there are two key issues you must consider:

- The word you are searching for might exist **starting** at any (x, y) location in the grid.
- The word you are searching for might exist in any one of 8 directions, starting from that (x, y) location.

3 Input File Format

The input file will have a series of lines, with no blank lines or comments. Each one will be made up **only** of lowercase alphabetical characters - there won't be any whitspace, numerals, or other strange characters to worry about.

After the grid, there will be a single blank line.

After the blank line, you should read each additional word from the file; there will only be one word per line. (You can assume that, you don't need to error-check it.) You can also assume that the words are all made up entirely of lowercase alpha characters.

While you're in the "words to search for" part of the file, you should ignore any line that is blank, or that is made up entirely of whitespace. Don't print anything out.

4 Your Output

At the start of your program, print out a prompt. (To make it easier to autograde on GradeScope, please print this on its own line with `print()`. Then call `input()`, with no parameter, on the next line.)

```
Please give the puzzle filename:
```

If the file doesn't exist, print out the following message:

```
Sorry, the file doesn't exist or cannot be opened.
```

and terminate the program. Otherwise, open the file, and read its contents. (How to see if a file doesn't exist? See the bottom of this spec.)

While we will test you on checking to see if the file exists, we will **not** test you on invalid file contents. If the file exists, you may assume that it is valid.

For each word that you search for, either print the grid out (with everything except the word that you've found replaced by periods), or print the message

```
Word 'word_here' not found.
```

In either case, print a blank line after you print out the search result.

5 Directions

You will be required to find words going in all 8 directions: right, left, up, down, and all four diagonal directions. However, you can earn most of the points without having to search in the "northwest" and "southwest" directions. We will test those - but only in a few of the testcases.

6 Example Code?

Take a look at the .zip file I've provided. In addition to testcases, I've also provided a program that will **generate** new testcases. Not only is it a cool tool, it also might be useful to you, when you're trying to figure out how to write your own code! (Remember, it's never a violation of the Code if you are re-using or adapting something that I've provided you.)

Unfortunately, since I wrote this code for myself - not for you - it's probably not commented as well as you would like. Guess you'll have to dig into it, to figure out how it works...

7 Turning in Your Solution

You must turn in your code using GradeScope.

---

8 Appendix: Does the File Exist?

If you try to open up a file, and the file doesn't exist, Python will throw an **exception.** You've seen this before, probably - if not, open up your favority Python environment and try this out:

```
handle = open("no_such_file.txt")
```

If we don't take any action to prevent it, an exception will kill your program. But don't worry - we have a solution! To "catch" an exception, we add a `try/except` block around it. Then, any exception that happens inside the block will be caught; instead of killing your program, it drops into the "exception handler," like this:

```
try:
    handle = open(some_filename)
except FileNotFoundError:
    print("If we get here, the file didn't exist.")
    print("Terminate the program with a useful error message.")
    return    # or something, you get to choose...

print("If we get here, then open() ran just fine, and we've opened the file.")
```