Word Search

CPE 101: Fundamentals of Computer Science
Winter 2019 - Cal Poly, San Luis Obispo

Purpose

To practice string operations (not list) and decomposing a problem into functional units.

Description

Download the project files from Polylearn.

In this project, you will implement a program which locates words in word search puzzles where all puzzles are 100 characters in size. A sample run of the program is shown below.

```
$ python3 wordsearch.py < test1.in

      WAQHGTTWEE
      CBMIVQQELS
      AZXWKWIIIL
      LDWLFXPIPV
      PONDTMVAMN
      OEDSOYQGOB
      LGQCKGMMCT
      YCSLOACUZM
      XVDMGSXCYZ
      UUIUNIXFNU

      UNIX: (FORWARD) row: 9 column: 3
      CALPOLY: (DOWN) row: 1 column: 0
      GCC: word not found
      SLO: (FORWARD) row: 7 column: 2
      COMPILE: (UP) row: 6 column: 8
      VIM: (BACKWARD) row: 1 column: 4
      TEST: word not found
```

Words can appear in the puzzle forward, backward, upward, and downward. You will not need to check diagonals. As will be discussed in a lecture, you will use the `python built-in str find` function on strings, which returns the index of the beginning of a given word located in a given string (or `-1` if the word is not present). For example, `"UUIUNIXFNU".find("UNIX")` returns `3` because the first character of `UNIX` starts at index `3` of the string. However, `"UUIUNIXFNU".find("SLO")` returns `-1` since `SLO` is not contained in the string.

Dimensionality Conversion

While the given puzzle is a 100-character string (a one-dimensional sequence), the process of finding the rows and columns of words requires operating in two dimensions. To do so, you must convert a given arbitrary index of the string into two values - one for the row and one for the column - which can be done using simple arithmetic operations.

0	1	2	3	4	5	6	7	8
A	B	C	G	I	T	X	Y	Z

One-Dimensional Character Sequence

	0	1	2
0	A	B	C
1	G	I	T
2	X	Y	Z

Two-Dimensional
Character Sequence

In the diagrams above, the word being searched (`GIT`) is highlighted in yellow, with the first letter of the word highlighted in green. Given this 9-character string, calling `"ABCGITXYZ".find("GIT")` evaluates to `3`, the index of the `G`. However, in order to report the row and column of this character, this 3 must be converted to two values: `1` for the row and `0` for the column.

The resulting output of this example would be:

```
GIT: (FORWARD) row: 1 column: 0
```

Implementation

You may not use lists, slicing operations, the `split` function or any language features and functions not yet discussed in lecture for this assignment.

Input

Your program must read a text file with the following properties:

- A 100-character string makes up the first line of the file
- The space-separated words to search for make up the second line of the file

Output

Your program must print the following text:

- The given puzzle as a 10x10 grid of characters
- The result of searching for each word, specifying its direction, row, and column if found or a message indicating that it was not found

Minimum Required Program Structure

`main()`

- Use the `input` function to read in the puzzle and words to find (without `split`)
  - Recall that each call to `input` reads one line from a file
  - Use `strip` to remove trailing newline characters
- Display the puzzle, one row per line (this step may be done in another function)
- Iterate through the words, calling function(s) to search for each one
  - Use the fact that each word to search for is delimited by a space
  - Ensure the order of the words printed matches the test files

`find_word(puzzle, word)`

- Searches the puzzle for the given word (in any direction)
  - This function may call other functions that search in a specific direction
- Returns a string containing the search result to be printed in `main`
  - In the example above, this function would return:
    `"GIT: (FORWARD) row: 1 column: 0"`

`reverse_string(string)`

- Returns the reverse of the input string

`transpose_string(string, row_len)`

- Returns a transposition of the input string, assuming `row_len` characters per row
  - Transposing a two-dimensional grid means converting its rows to columns and its columns to rows
  - Since strings are one-dimensional, the result will be a string with its characters shifted around
- For example, `"ABCGITXYZ"` transposes to `"AGXBIYCTZ"`

Testing

Each puzzle can be found in a separate file:

```
test1.in, test2.in, test3.in
```

Your program should be run using:

```
python3 wordsearch.py < test#.in
```

You should compare your output with the corresponding output files using `diff` (without the use of any flags):

```
test1.out, test2.out, test3.out
```

You are required to write at least 3 tests for each function that you create (except `main`). As done previously, tests are written using `assert` statements. Since we are emphasizing test-driven development, you should write tests for each function first. In doing so, you will have a better understanding as to what the functions take as input and produce as output, which makes writing the function definitions easier.

Submission

Zip `wordsearch.py` and `tests.py` into one zip file with name [your_calpoly_id]_project2.zip, where [your_calpoly_id] is a part of your calpoly email address excluding @calpoly.edu, and submit it to polylearn.