Lab Exercise #3

Assignment Overview

This lab exercise provides practice with strings and functions in Python.

You will work with a partner on this exercise during your lab session. Two people should work at one computer. Occasionally switch the person who is typing. Talk to each other about what you are doing and why so that both of you understand each step.

PART A: DEBUGGING EXERCISE
The art of debugging is an integral part of programming. The following exercise is designed to help you understand the string manipulation using a debugger. The debugger is described in Appendix C of the text.
Set breakpoints on lines by double clicking the left pane of the Spyder IDE next to the line numbers. Try it and you will see a red circle; double-click again to make it disappear. Run the program in *debug* mode by pressing the Debug file (`CTRL + F5`) button (leftmost). Execution "freezes" when a breakpoint is encountered. You can then progress the program one statement at a time by pressing the Run Current Line (`CTRL + F10`) button (second from left). A pointer (`--->`) in the console (in the lower right corner of Spyder) shows you the next statement to be executed. This console output will change every time you progress your program further (using `CTRL + F10`). You can use this pointer along with `CTRL + F10` to understand which lines of your code are executed during conditional statements. Notice the values stored in variables in the *Variable Explorer* window (above the console).
Open up the program `debug3.py` in Spyder. In the upper-right window click on Variable Explorer to open that window. Click 'Remove all variables' button () in Variable Explorer to clear all variables. *Set a breakpoint at the first executable line of the program* (`string1`), and start the debugger by pressing the "start debugger" button (leftmost blue button). Answer the following questions:

1) When the pointer (`--->`) shows that the next instruction to be executed is 14, which variables are held in memory and what are their values?
   ANSWER: Variables → values

   string1 → noitalupinam gnirtS
   string2 → NOITALUPINAM GNIRTS
   string3 → RTS
   string4 → <empty>
2) List all values of `i` and ch observed until the completion of the `for` loop.
   ANSWER:

`i`	`ch`
0	R
1	T
2	S

Note: Please *do not* use the `print()` statements during the debug exercise.
Additional Resources
https://docs.spyder-ide.org/debugging.html

PART B: PROGRAMMING WITH STRINGS

Develop a Python program which will convert English words into their Pig Latin form, as described below.

The program will repeatedly prompt the user to enter a word. First convert the word to lower case. The word will be converted to Pig Latin using the following rules:

a) If the word begins with a vowel, append “way” to the end of the word.
b) If the word begins with a consonant, remove all consonants from the beginning of the word and append them to the end of the word. Then, append “ay” to the end of the word.

For example:

```
"dog" becomes "ogday"
"scratch" becomes "atchscray"
"is" becomes "isway"
"apple" becomes "appleway"
"Hello" becomes "ellohay"
"a" becomes "away"
```

The program will halt when the user enters “quit” (any combination of lower and upper case letters, such as “QUIT”, “Quit” or “qUIt”).

Suggestions:

a) Use `.lower()` to change the word to lower case.
b) How do you find the position of the first vowel? I like using `enumerate(word)` as in `for i,ch in enumerate(word)` where `ch` is each character in the word and `i` is the character’s index (position).
c) Use *slicing* to isolate the first letter of each word.
d) Use *slicing* and *concatenation* to form the equivalent Pig Latin words.
e) Use the **`in`** operator and the string `"aeiou"` to test for vowels. Good practice: define a constant `VOWELS = "aeiou"`

★ **Test your program on Mimir and demonstrate it to your TA. On-line students should submit the completed program (named “lab03.py”) for grading via the Mrmir system.**