CS 1110, LAB 03: STRINGS; TESTING

http://www.cs.cornell.edu/courses/cs1110/2018sp/labs/lab03/lab03.pdf

First Name: _____ **Last Name**: _____ **NetID**: _____

Correction on pg 2 made Tue Feb 13, 3:15pm

Getting Credit: **As always, strive to finish during the lab session** — it's the best way to stay on track in this course.[1] (You are getting two weeks for this lab because of February break.)

1. Practice with String Operations and String Methods

Start up Python interactive mode[2], which is what we use for quick experiments, and enter the *second* line below, the one with a mixed-case word, at the >>> prompt.

```
#    0123456789      These numbers show you the indices of the first 10 characters
s = 'HeLLo WorLd!'
```

Now fill in the tables below, as usual.

Expression	Expected value	Actual value, if different
`s[1]`		
`s[15]`		
`s[1:7]`		
`s[:7]`		
`s[4:]`		
`'e' in s`		
`'x' in s`		

Expression	Expected Value	Calculated Value
`s.index('L')`		
`s.index('x')`		
`s.count('o')`		
`s.index('L',5)` `# this means, "look starting from index 5"`		

Lab authors: D. Gries, L. Lee, S. Marschner, W. White

[1]But if you don't manage finish during lab, here are the alternate checkoff opportunities: (a) at ACCEL Green room consulting hours, listed at http://www.cs.cornell.edu/courses/cs1110/2018sp/about/staff.php , **from today until Tue Feb 27 inclusive**, (b) at non-professorial TA office hours **from today to Wed Feb 28 3:45pm inclusive**, although at TA office hours, questions about course material or assignments take precedence over lab check-offs; or (c) during the **first 10 minutes of your next scheduled lab (Tue Feb 27 or Wed Feb 28)**. Beyond that time, the staff have been instructed not to give you credit.

Labs are graded on effort, not correctness. We just want to see that you tried all the exercises, and to clarify any misunderstandings or questions you have.

[2]Enter `python` at the command shell prompt.

2. The Function and Implementation of replace_first in lab03.py

In this lab, you will test and debug an implementation of the following function, which could be useful for fixing typos:

> `replace_first(word,target,rep)` returns a copy of string `word` with the *first* instance of string `target` in `word` replaced by string `rep`. Precondition: `target` has length ≥ 1 , and occurs at least once in `word`.

From this specification, we expect that `replace_first('THanks', 'H', 'h')` returns `'Thanks'`.

2.1. **Develop further understanding through test cases.** For each of the following potential calls to `replace_first`, state (a) whether it has valid inputs according to the specification, (b) if so, what the output should be, (c) whether and why it would be a good additional test case given the set of valid test cases already given.

To help you, we've done the first few for you.

`replace_first('methos', 's', 'd')`

valid, `'method'`, good case — tests `target` at the very end of the string

`replace_first('Misissippi', 's', 'ss')`

valid, `'Mississippi'`, good case — more than one occurrence of `target`.

`replace_first('decrepif', 'f', 't')` (THIS IS A FIX OF THE ORIGINAL, which had "decrepid")

valid, `'decrepit'`, bad case — already tested target at end and single-letter targets.

`replace_first('aggreived', 'ei', 'ie')`

`replace_first('em', 'em', 'umm')`

```
replace_first('judgement', 'e', '')
```

```
replace_first('judgement', '', '!')
```

2.2. Use the good test cases to test an implementation of `replace_first`.

2.2.1. *The implementation and the testing files.* Create a new directory on your hard drive for this lab's files. Then, download into that new directory the files you need for lab 03; get them from the Labs section of the course web page, http://www.cs.cornell.edu/courses/cs1110/2018sp/labs .

In file `lab03.py`, there's a slightly incorrect implementation of `replace_first`, which you are going to debug. But don't look at it yet! Instead, open the separate testing file `lab03_test.py` in Komodo Edit. In it is an incomplete test function, `test_replace_first()`, for checking `lab03.replace_first`. This function is called near the end of the script.

2.2.2. *Understanding flow of execution in the testing file.* Open a command shell and navigate[3] to your new directory with the lab 03 files in it. Then, run Python on `lab03_test.py`.[4]

You should get a message

```
Testing lab03.replace_first
Module lab03: all tests passed
```

If you get an error message instead, ask for help now.

We claim that if the second-to-last non-blank line, `test_replace_first`, had been commented out, then the output would have been only this single line instead:

```
Module lab03: all tests passed
```

Why would the output line `Testing lab03.replace_first` no longer get printed out?

[3] Use the `cd` commands you practiced in Lab 02; see http://www.cs.cornell.edu/courses/cs1110/2018sp/materials/command.php for our documentation.

[4] That is, in the command shell, enter `python lab03_test.py`

2.2.3. *Adding the test cases to the testing file.* Lines 18-20 have, commented out, an instantiation or implementation of the first test case from Section 2.1. This implementation uses `assert_equals` from module `cornellasserts`, which we introduced in lecture.

Finish `test_replace_first()` by uncommenting that first test case[5], and then adding all your good test cases from Section 2.1. Use lines 18-20 as a guide.

2.2.4. *Finally, run the testing file to test the correctness of `lab03.replace_first`.* Save `lab03_test.py`, and, in the command shell, run Python on the file again.

Because we've planted one or more errors in `lab03.replace_first`, you'll get an error message. What is it?

Ugh!

2.3. **Use `print` statements to check the values of variables.** Now we know there's a problem with the given implementation. But how will we find all the problems?

`print` statements are perhaps the least elegant tool to use for isolating errors, but they work for any language and environment. These statements allow us to *inspect* a variable immediately after it is assigned a value.

Open up file `lab03.py` in Komodo Edit and look at the comments explaining what the variables `pos`, `before`, `after`, and `result` are supposed to mean. **According to those comments**, *not* the code itself, for the test case `word: 'methos'`, `target: 's'`, `rep: 'd'`, what *should* the values of these four variables be?

pos: before: after: result:

Let's add a `print` statement to inspect the variable `pos`. Inside of `replace_first`, *right after* the assignment to `pos`, add, properly indented, the informative statement

```
print("DEBUG: pos is:  " + str(pos))
```

While the point of the above command is to print out the value of `pos`, the "tag" text "DEBUG: pos is: " serves as a label in your output, making it more readable.

Do the analogous thing for the other three variables, `before`, `after`, and `result`.

Interlude: get a copy of your work off the lab machines! If you are working on a lab machine, know that **your files will be automatically deleted at some point soon after you log out or are auto-logged out**. It is therefore vitally important that, as you get near the end of the lab, **GET A COPY OF YOUR FILES TO YOURSELF — MAIL THEM TO YOURSELF, SAVE THEM TO A USB FlASH DRIVE**, or whatever works for you.

[5]Shortcut: select the three commented-out lines, and then in the Komodo Edit Menu go to item Code and use "Un-comment region".

Save `lab03.py` and run the *test* program again.[6]. Before you see the error message, you should see four DEBUG lines printed to the screen. These are the results of your print statements. The output helps you "visualize" what is going on in `replace_first()`. What does the output tell you, for the test case `word: 'methdo', target: 'do', rep: 'od'`, that the four variables `pos`, `before`, `after`, and `result` are *actually* set to by the code?

pos:	before:	after:	result:

By comparing your "should" answers two boxes above with your "actually" answers in the box above, you should see that there is a problem with the variable `pos`. Look where there's an assignment to `pos` in `replace_first`. What is the error in that assignment statement?

Fix the error (ask a staff member if you don't know how to fix it), save your files, and test the procedure again by running the test script.

Alas, you should get *another error*, for a different test case! Using the methodology above — looking at the printed-out values of your variables and comparing them to the values they "should" have — how should another line in `replace_first` be corrected?

Fix the error (ask a staff member if you don't know how to fix it), test your program, and repeat until all your test cases pass! What other errors did you find, if any?

2.4. **Show your code to a staff member.**

2.5. **Afterwards, remove or comment out your debugging print statements.** While your print statements proved very useful for debugging, you do not want those print statements showing information on the screen every time you run the procedure.[7]

So once you are sure the program is running correctly, you should remove all of your debugging print statements. You can either comment them out (fine in small doses, as long as it does not make your code unreadable), or you can delete them entirely.

[6] Enter `python lab03_test.py` at the command shell.

[7] In fact, their presence technically violates the function specification, since no mention of printing is made there.

However, once you remove these, it is important that you test the procedure one last time. You want to be sure that you did not delete the wrong line of code by accident. Run the test script one last time to make sure no errors were introduced by your deletions.