Non-Textbook Lab 7: String Functions (50 points)

Objectives

Practice defining functions with parameters. Practice manipulating strings.

Implement your own version of Python string methods

The provided file *string_functions*.py defines several string functions, but does not include the code for them. Add code to implement functions that are equivalent in behavior to the corresponding built-in Python string function. Every function MUST return a value.

Use loops to implement your functions. You **cannot** use built-in string methods, string functions, or string-operators to implement your functions. Each function is worth a maximum of 10 points.

1. `cs110_upper:` Takes a string as a parameter and returns it as a string in all uppercase. *Hint:* Use the `ord()` function to get the ASCII value of a character. For example, `ord('a')` returns the integer 97. You will also need to use the `chr(value)` function that returns a string of one character whose ASCII code is the integer *value*. For example, `chr(97)` returns the string 'a'. You **cannot** use the built-in string function `upper()` for this.

2. `cs110_lstrip:` Takes a string as a parameter and returns the same string with the whitespace stripped out from the left side of the string.

3. `cs110_replace:` Takes a string and two characters (char1 and char2) as parameters and returns a string with char1 replaced by char2 in the string.

4. `cs110_in:` Takes a long string and a short string as parameters and returns True if the short string is contained in the long string, and False if it is not.

5. `cs110_title:` Takes a string and returns a string with the first character capitalized for every word. The rest of the characters are in lower case. For example, if the input to the function is "I like Python a lot", the function should return a string that looks like "I Like Python A Lot".

Note: The point of this lab is for you to use **loops** to reimplement the string functions. You may **not** use Python's built-in string methods or operators to implement yours – for example, **`in`**, **`.upper(), .isUpper()`**, **`.strip(), .split()`**. You may use these Python functions: **`len()`**, **`ord()`**, **`str()`**. If you are not sure whether you can use a specific function or method, ask in Piazza.

Do **not** add a **main()** function or any non-function code to **string_functions.py**. Use the provided program **test_string_functions.py** to test your string functions. Put **test_string_functions.py** in the same directory with **string_functions.py** and run

test_string_functions.py. It imports string_functions.py and tests each of the functions you implemented. For each function that works correctly, **test_string_functions.py** will award 10 points.

6. Extra credit - Delete repetitive characters (5 points)

The extra credit can be attempted **only if all the other functions are working** and functional. For this part, write a function **`cs110_remove_repeats(string)`**, that takes in a string parameter and returns a string with **all** repeated characters removed. For example, if **`string`** is "bookkeeper", the function returns "bokeper", and if `string` is “hello”, the function returns “helo”.

Submit

Upload **string_functions.py** and a **readme.txt** file to NT Lab 7 in Canvas.