CSC/CPE 101: Fundamental of Computer Science I

Spring 2018 (LAB-6)

Due: 4/18/18

This lab provides additional exercises on iteration over lists and exercises on characters and strings.

Download Lab6.zip, from PolyLearn (place it in your `CPE101` directory, and unzip the file) or from:

```
unzip  lab6.zip
```

NOTE: For each function you need to write signature and purpose statement before you implement the functions.

Fold Pattern

Develop these functions in the `fold` directory in files `fold.py` and `fold_tests.py`. Don't use the **library functions such as sum, index,** etc. The List can be list of list!

Each of the functions for this part of the lab must be implemented using the fold or reduce pattern. In this pattern, the values in the input list are combined in some manner (e.g., an arithmetic or relational operation) to compute a single result value. Each of these functions will use a local variable (or local variables) to hold the computed value as the list is traversed. This variable is often updated at each step of the iteration.

1) sum

The `sum` function returns the sum of all values in the input list.

2) index_of_smallest

The `index_of_smallest` function returns the index of the smallest value in the input list. If the list is empty (i.e., the length is ≤ 0), then the function returns -1. If there is more than one smallest value, return the index of the first occurrence of it.

Character Manipulation

Develop these functions in the `char` directory in files `char.py` and `char_tests.py`.

1) is_lower_101

The `is_lower_101` function takes a single character and returns True if the character is lowercase (assuming only the English alphabet) and False otherwise. You are required to write this function without using the predefined `lower` functions. Your solution should use character/string literals; avoid the direct use of the integer values for characters. Recall that you can use the `ord` function to determine the integer representation of a character.

2) char_rot13

The char_rot13 function takes a single character and returns the rot13 encoding of the character (**rot13:** The rot13 algorithm obscures text. It does not encrypt it. The algorithm shifts each character back, or forward, 13 places. It is a cipher algorithm.) Your solution should avoid the direct use of integer values for characters (use character/string literals). You may use the str.isalpha, str.islower,and str.isupper functions, if desired. Recall that you can use the ord function to determine the integer representation of a character and the chr function to determine the character represented by an integer (in the valid range).

Rot-13

rot-13 (rotate by 13 places) is a simple Caesar-cypher encryption that replaces each character of the English alphabet with the character 13 places forward or backward along the alphabet (e.g, 'a' becomes 'n', 'b' becomes 'o', 'N' becomes 'A', etc.). This only works on the characters in the alphabet. All other characters are left unchanged. Moreover, the rotation is only within the characters of the same case (i.e., a lowercase letter always rotates to a lower case letter and the same for uppercase letters). An encoded character can be decoded by applying the rotation a second time. More information, for those curious, can be found at the Wikipedia entry for rot13.

String Manipulation

Develop these functions in the `string` directory in files `string_101.py` and `string_101_tests.py`.

Note: There are a few different approaches to the following functions. For the sake of learning, you should implement one of them using an explicit loop and one without using an explicit loop.

1) str_rot_13

The `str_rot_13` function takes an input string argument and returns the rot-13 encoding of the input string.

2) str_translate_101

The `str_translate_101` function is a simplification of the `translate` function. Write `str_translate_101` to take a string and two characters (*old* and *new*) as arguments. The function will return a string where each occurrence of *old* is replaced with *new* (and all other characters are left unchanged). You are required to write this function without using any predefined string functions with similar behavior (i.e., think of this as an exercise in implementing a provided function).

As an example, `str_translate_101('abcdcba', 'a', 'x')` should result in `'xbcdcbx'`.

Demonstration

Demonstrate the test cases from each part of the lab to your instructor to have this lab recorded as completed. In addition, be prepared to show the source code to your instructor.

Submission

Demo your work and Submit your files in the PolyLearn.

1. `fold.py`,
2. `fold_tests.py`,
3. `char.py`,
4. `char_tests.py`,
5. `string_101.py`, and
6. `string_101_tests.py`.