Lab 7, CSC 101

This lab provides exercises on characters and strings. In addition, the last part introduces command-line arguments.

Download lab7.zip, place it in your `cpe101` directory, and unzip the file.

Character Manipulation

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

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` type functions. Your solution should use character/string literals; avoid the direct use of the integer values for characters. Recall that you use can the `ord` function to determine the integer representation of a character.

char_rot_13

The `char_rot_13` function takes a single character and returns the rot-13 encoding of the character. 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 use can 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.

str_rot_13

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

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'`.

Command-line Arguments

Develop this part in the `cmdline` directory in the `cmdline.py` file.

This part introduces the notion of command-line arguments. These are arguments to the program itself when run from the command-line.

Command-line arguments are accessible, as a list, via the `argv` variable in the `sys` module.

Develop a program that prints each command-line argument on a separate line preceded by the argument index.