Lab 6 String Manipulation

Purpose

To attain more experience using strings and indexing and to begin to understand how characters - and thus, strings - are represented by a machine.
To practice using a loop to iterate over characters in a string.

Description

ASCII Character Set

ASCII characters are represented in a machine as a single byte, which is a sequence of 8 bits. Each bit (or "binary digit") has a value of either 0 or 1. Thus, since each bit can hold one of two values and there are 8 such values in a byte, the number of possible values a byte can store is 2^8 or 256. As the English language comprises far fewer symbols than 256 (including numbers and punctuation), it is possible to represent each character in English in a byte. Thus, each character has a numerical representation, as can be referenced by entering `man ASCII` in a terminal if you are using a UNIX variant OS.

0	1	0	0	0	0	0	1
7	6	5	4	3	2	1	0

The bits displayed above represent the number 65, which, in ASCII, is the character "A". This number can be found by adding $2^0 + 2^6$ which equals 1 + 64 or 65.

Implementation

Create a file, **lab6.py**, and write the following functions in the file. Since we are emphasizing test-driven development, you should read the description and write a function docstring with at least three examples for each function. Use the doctest to test your functions as before.

Do not refer to an ASCII table to complete this assignment. No ASCII numbers should be used in your implementation - instead, use letters passed to the `ord()` function.

Character Manipulation

`is_lower(char:str)->bool`

Return `True` if the given character is lowercase (assuming only the English alphabet) and `False` otherwise. You may not use the `str.islower()` function. Hint: use ord('a') and ord('z') to determine if a character is a lower case alphabet letter. Also, in Python, 'a' <= 'c' <= 'z' evaluates to True, but 'a' <= 'C' <= 'z' evaluates to False.

`char_rot_13(char:str)->str`

Return the ROT-13 encoding of the given character, which is simple Caesar-cypher encryption that replaces each character of the English alphabet with the character 13 places forward or backward along with 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. This function may use the `str.isalpha()`, `str.islower()`, and `str.isupper()` functions. You may use the built-in `chr()` function to convert a character code to the character. For example, `chr(97)` returns 'a'.

String Manipulation

`str_rot_13(my_str:str)->str`

Return the ROT-13 encoding of the given multi-character string. This function must make calls to `char_rot_13()` rather than duplicating its implementation.

`str_translate(my_str:str, old:str, new:str)->str`

Return a new string where each occurrence of the string `old` is replaced with the string `new` and all other characters are left unchanged. `new` and `old` can be a multi-character string such as 'Fizz' and 'Buzz'. Do not use any built-in string functions with similar behavior, such as `str.replace()`. However, you may use str.find(). You might want to use while-loop for this function.

Example: `str_translate("abcdcba", "a", "x")` returns `"xbcdcbx"`
Example: `str_translate("abcdabc", "abc", "x")` returns `"xdx"`
Example: `str_translate("FizzBuzzFizz", "Fizz", "Buzz")` returns `"BuzzBuzzBuzz"`

reverse_substr(my_str:str, start:int, end:int)->str

Return a new string with its substring that starts at the position ‘start’ and end at the position ‘end’ reversed. Do not use any built-in string functions with similar behavior, such as `str.replace()`. You may use [ ].
Example: `reverse_substr("abcdcba", 1, 3)` returns `"adcbcba"`
Example: `reverse_substr("abcdcba", 4, 6)` returns `"abcdabc"`
Example: `reverse_substr("abcdcba", 2, 4)` returns `"abcdcba"`
Example: `reverse_substr("abcdcba", 2, 2)` returns `"abcdcba"`

Testing

Add the usual `if __name__ == '__main__':` line at the bottom of your file, import the `doctest` module, and test your functions using the `doctest.testmod()`.

```
if __name__ == '__main__':
    import doctest
    doctest.testmod()
```

Submission

Submit your lab6.py to Gradzilla to get it scored. Then, submit the lab6.py to Canvas.