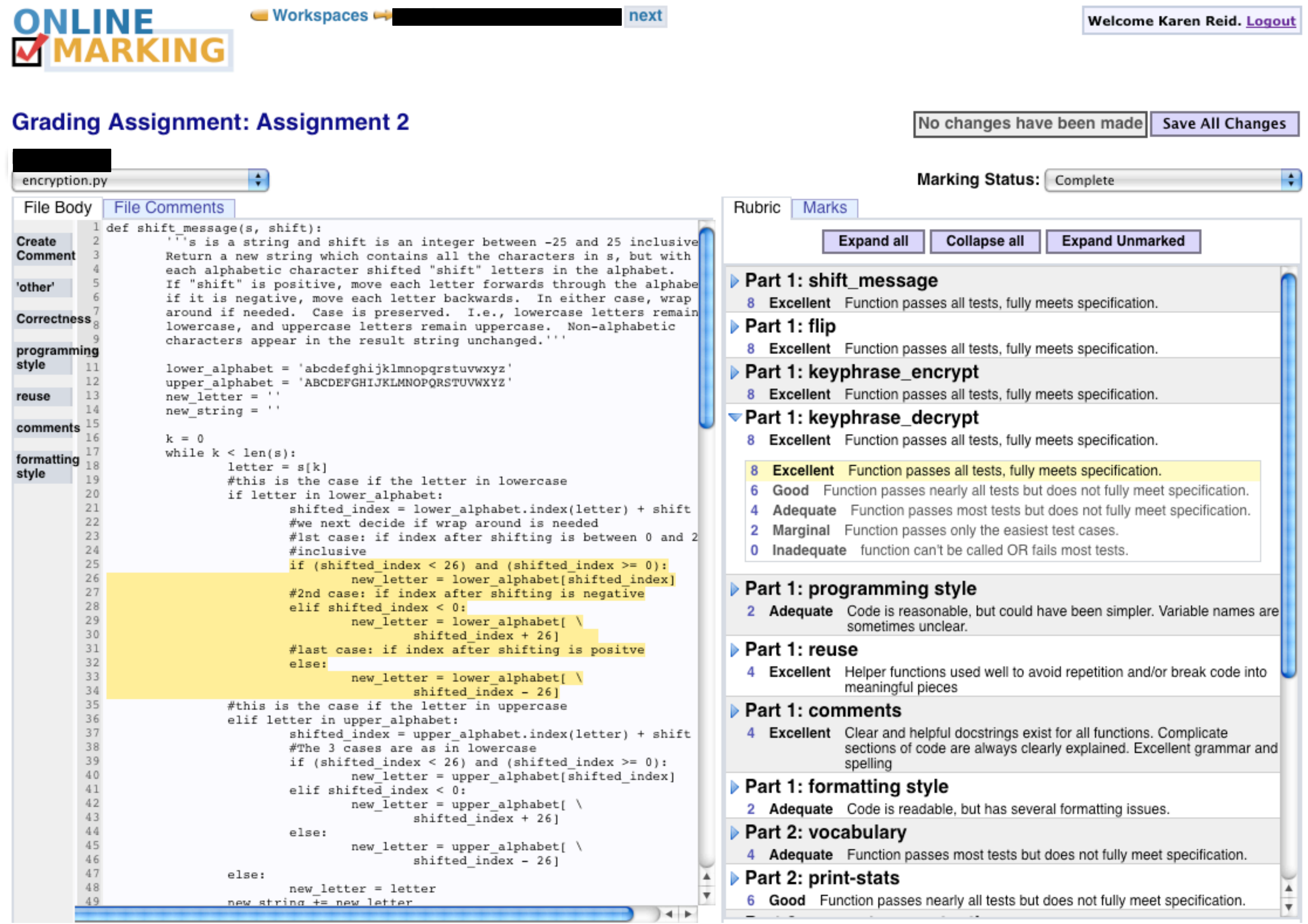

[redacted]

Grading Assignment: Assignment 2

[redacted]

encryption.py

Marking Status: Complete

```
def shift_message(s, shift):
        '''s is a string and shift is an integer between -25 and 25 inclusive
        Return a new string which contains all the characters in s, but with
        each alphabetic character shifted "shift" letters in the alphabet.
        If "shift" is positive, move each letter forwards through the alphabe
        if it is negative, move each letter backwards.  In either case, wrap
        around if needed.  Case is preserved.  I.e., lowercase letters remain
        lowercase, and uppercase letters remain uppercase.  Non-alphabetic
        characters appear in the result string unchanged.'''

        lower_alphabet = 'abcdefghijklmnopqrstuvwxyz'
        upper_alphabet = 'ABCDEFGHIJKLMNOPQRSTUVWXYZ'
        new_letter = ''
        new_string = ''

        k = 0
        while k < len(s):
                letter = s[k]
                #this is the case if the letter in lowercase
                if letter in lower_alphabet:
                        shifted_index = lower_alphabet.index(letter) + shift
                        #we next decide if wrap around is needed
                        #1st case: if index after shifting is between 0 and 2
                        #inclusive
                        if (shifted_index < 26) and (shifted_index >= 0):
                                new_letter = lower_alphabet[shifted_index]
                        #2nd case: if index after shifting is negative
                        elif shifted_index < 0:
                                new_letter = lower_alphabet[ \
                                        shifted_index + 26]
                        #last case: if index after shifting is positve
                        else:
                                new_letter = lower_alphabet[ \
                                        shifted_index - 26]
                #this is the case if the letter in uppercase
                elif letter in upper_alphabet:
                        shifted_index = upper_alphabet.index(letter) + shift
                        #The 3 cases are as in lowercase
                        if (shifted_index < 26) and (shifted_index >= 0):
                                new_letter = upper_alphabet[shifted_index]
                        elif shifted_index < 0:
                                new_letter = upper_alphabet[ \
                                        shifted_index + 26]
                        else:
                                new_letter = upper_alphabet[ \
                                        shifted_index - 26]
                else:
                        new_letter = letter
                new_string += new_letter
```

Part 1: shift_message
8 **Excellent** Function passes all tests, fully meets specification.

Part 1: flip
8 **Excellent** Function passes all tests, fully meets specification.

Part 1: keyphrase_encrypt
8 **Excellent** Function passes all tests, fully meets specification.

Part 1: keyphrase_decrypt
8 **Excellent** Function passes all tests, fully meets specification.

8	Excellent	Function passes all tests, fully meets specification.
6	Good	Function passes nearly all tests but does not fully meet specification.
4	Adequate	Function passes most tests but does not fully meet specification.
2	Marginal	Function passes only the easiest test cases.
0	Inadequate	function can't be called OR fails most tests.

Part 1: programming style
2 **Adequate** Code is reasonable, but could have been simpler. Variable names are sometimes unclear.

Part 1: reuse
4 **Excellent** Helper functions used well to avoid repetition and/or break code into meaningful pieces

Part 1: comments
4 **Excellent** Clear and helpful docstrings exist for all functions. Complicate sections of code are always clearly explained. Excellent grammar and spelling

Part 1: formatting style
2 **Adequate** Code is readable, but has several formatting issues.

Part 2: vocabulary
4 **Adequate** Function passes most tests but does not fully meet specification.

Part 2: print-stats
6 **Good** Function passes nearly all tests but does not fully meet specification.