String Compression

Problem

Given a string in the form 'AAAABBBBCCCCCDDEEEE' compress it to become 'A4B4C5D2E4'. For this problem, you can falsely "compress" strings of single or double letters. For instance, it is okay for 'AAB' to return 'A2B1' even though this technically takes more space.

The function should also be case sensitive, so that a string 'AAAaaa' returns 'A3a3'.

Solution

Since Python strings are immutable, we'll need to work off of a list of characters, and at the end convert that list back into a string with a **join** statement.

The solution below should yield us with a Time and Space complexity of O(n). Let's take a look with careful attention to the explanatory comments:

In [1]:
```
def compress(s):
    """
    This solution compresses without checking. Known as the RunLength Compression algorithm.
    """

    # Begin Run as empty string
    r = ""
    l = len(s)

    # Check for length 0
    if l == 0:
        return ""

    # Check for length 1
    if l == 1:
        return s + "1"

    #Intialize Values
    last = s[0]
    cnt = 1
    i = 1

    while i < l:

        # Check to see if it is the same letter
        if s[i] == s[i - 1]:
            # Add a count if same as previous
            cnt += 1
        else:
            # Otherwise store the previous data
            r = r + s[i - 1] + str(cnt)
            cnt = 1

        # Add to index count to terminate while loop
        i += 1
```

```
    # Put everything back into run
    r = r + s[i - 1] + str(cnt)

    return r
```

In [2]:
```
compress('AAAAABBBBCCCC')
```

Out[2]:
```
'A5B4C4'
```

Test Your Solution

In [3]:
```
"""
RUN THIS CELL TO TEST YOUR SOLUTION
"""
from nose.tools import assert_equal

class TestCompress(object):

    def test(self, sol):
        assert_equal(sol(''), '')
        assert_equal(sol('AABBCC'), 'A2B2C2')
        assert_equal(sol('AAABCCDDDDD'), 'A3B1C2D5')
        print('ALL TEST CASES PASSED')

# Run Tests
t = TestCompress()
t.test(compress)
```

```
ALL TEST CASES PASSED
```

Good Job!