SelfAssessment73

October 18, 2015

1 Exercise 7.3

Write a function to count occurrences of a particular character in a string. Both the string and the character are parameters of the function. Test the function.

Answer appears after one blank page (so you don't peek).

Are you sure you're ready to peek?

2 Possible Solutions

```
In [12]: def count_chars(string, letter):
             return string.count(letter)

         count_chars('banana', 'a')
```

```
Out[12]: 3
```

This is the most straightforward way. We take advantage of the fact that there's a method called `count` that does exactly what we're looking to do. Reember that "methods" are slightly different in their construction than functions. They are appended to the end of a variable and may (or may not) take arguments.

```
In [11]: def count_chars(string, letter):
             counter = 0
             if isinstance(string, str) & isinstance(letter, str):
                 for i in string:
                         if letter in i:
                             counter += 1
                 return counter
             else:
                 raise Exception('Both arguments must be strings.')

         print '\'count_chars(\'banana\', \'a\')\' - Expecting a sum of 3:', count_chars('banana', 'a')
         print ' \'count_chars(\'banana\', \'a\')\' - Expecting a sum of 1:', count_chars('banana', 'b')
         print ' \'count_chars(\'banana\', 4)\'   - Expecting an error:', count_chars('banana', 4)
```

```
'count_chars('banana', 'a')' - Expecting a sum of 3: 3
 'count_chars('banana', 'a')' - Expecting a sum of 1: 1
 'count_chars('banana', 4)'   - Expecting an error:


        ---------------------------------------------------------------------------

        Exception                                 Traceback (most recent call last)

        <ipython-input-11-87250874d45c> in <module>()
         11 print '\'count_chars(\'banana\', \'a\')\' - Expecting a sum of 3:', count_chars('banana', 'a
         12 print ' \'count_chars(\'banana\', \'a\')\' - Expecting a sum of 1:', count_chars('banana', '
    ---> 13 print ' \'count_chars(\'banana\', 4)\'   - Expecting an error:', count_chars('banana', 4)


        <ipython-input-11-87250874d45c> in count_chars(string, letter)
           7         return counter
           8     else:
        ----> 9         raise Exception('Both arguments must be strings.')
          10
          11 print '\'count_chars(\'banana\', \'a\')\' - Expecting a sum of 3:', count_chars('banana', 'a


        Exception: Both arguments must be strings.
```

Here we can make use of the `in` operator to sum across the sequence of letters `string`. We've used this method of looping before. I've tried a few different tests. One thing yo haven't seen before are the functions `isinstance` and the `raise` statement. `isinstance` lets us test the variable type, making sure we're doing this for strings only. `raide` allows us to throw user defined errors. This lets me control how the function behaves a bit better, and provide a more specific error message for the user. Python has a very well defined set of error types and you can use any one of these. The general rule is to use the most specific error type.