Jane Street

Jane Street Coding Problem

04m : 35s
to test end

1

☆ Programmer String

We consider a string to be a *programmer string* if some subset of its letters can be rearranged to form the word "programmer". For example, the strings "programmer", "grammproer", and "xproxmerqgram" are all programmer strings.

Consider a string, $s = s_0, s_1, s_2, \ldots s_{n-1}$ of n lowercase English letters. We denote a substring of s starting at index i and ending at index j as $s_{i,\, j}$. We want to find the number of indices i such that the substrings $s_{0,\, i-1}$ and $s_{i+1,\, n-1}$ are programmer strings. In other words, for each index i satisfying this property, the substring to the left of index i and the substring to the right of index i must both be programmer strings.

Complete the *programmerStrings* function in the editor below. It has one parameter: a string, s, of lowercase English letters. The function must return an integer denoting the number of indices i such that the substring of s from s_0 through s_{i-1} and the substring of s from s_{i+1} through s_{n-1} are both programmer strings.

Input Format

Locked stub code in the editor reads string s from stdin and passes it to the function.

Constraints

- String s consists of lowercase English alphabetic letters only.
- $1 \leq$ length of $s \leq 10^5$

Output Format

The function must return a single integer denoting the number of indices i such that the strings $s_{0,\, i-1}$ and $s_{i+1,\, n-1}$ are both programmer strings.

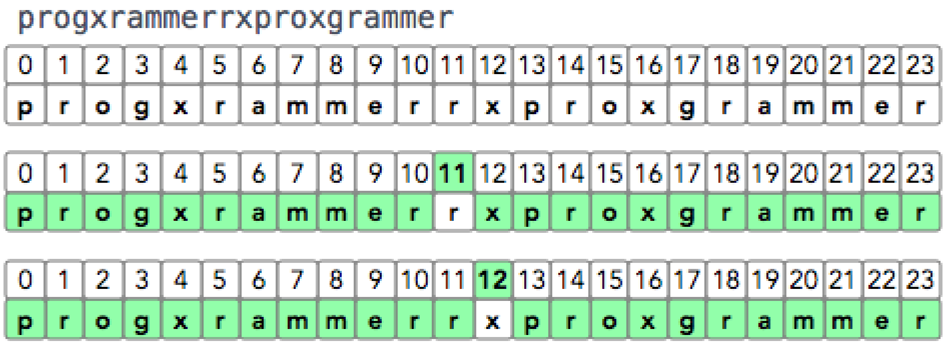
Sample Input 0

```
progxrammerrxproxgrammer
```

Sample Output 0

Explanation 0

There are two indices, *i = 11* and *i = 12*, that satisfy the property that substrings $s_{0,\ i-1}$ and $s_{i+1,\ n-1}$ are both programmer strings:



```
progxrammerrxproxgrammer
```

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
p	r	o	g	x	r	a	m	m	e	r	r	x	p	r	o	x	g	r	a	m	m	e	r

0	1	2	3	4	5	6	7	8	9	10	**11**	12	13	14	15	16	17	18	19	20	21	22	23
p	r	o	g	x	r	a	m	m	e	r	r	x	p	r	o	x	g	r	a	m	m	e	r

0	1	2	3	4	5	6	7	8	9	10	11	**12**	13	14	15	16	17	18	19	20	21	22	23
p	r	o	g	x	r	a	m	m	e	r	r	x	p	r	o	x	g	r	a	m	m	e	r

Thus, the function returns *2* as the answer.

Sample Input 1

```
xprogxrmaxemrppprmmograeiruu
```

Sample Output 1

```
2
```

Explanation 1

There are two indices, *i = 13* and *i = 14*, that satisfy the property that substrings $s_{0,\ i-1}$ and $s_{i+1,\ n-1}$ are both programmer strings:

0	1	2	3	4	5	6	7	8	9	10	11	12	**13**	14	15	16	17	18	19	20	21	22	23	24	25	26	27
x	p	r	o	g	x	r	m	a	x	e	m	r	p	p	p	r	m	m	o	g	r	a	e	i	r	u	u

0	1	2	3	4	5	6	7	8	9	10	11	12	13	**14**	15	16	17	18	19	20	21	22	23	24	25	26	27
x	p	r	o	g	x	r	m	a	x	e	m	r	p	p	p	r	m	m	o	g	r	a	e	i	r	u	u

Thus, the function returns *2* as the answer.

Sample Input 2

```
programmerprogrammer
```

Sample Output 2

```
0
```

Explanation 2

There are no indices satisfying the property that substrings $s_{0,\ i-1}$ and $s_{i+1,\ n-1}$ are *both* programmer strings:

`programmerprogrammer`

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
p	r	o	g	r	a	m	m	e	r	p	r	o	g	r	a	m	m	e	r

Thus, the function returns *0* as the answer.

YOUR ANSWER

We recommend you take a quick tour of our editor before you proceed. The timer will pause up to 90 seconds for the tour. Start tour ✖

Draft saved 02:51 pm Original code Python 3

```
import os


# Complete the function below.

from collections import Counter

# Check if a string is a programmer string
def is_programmer_string(s):
    s = Counter(s)
    programmer = Counter('programmer')
    return all(programmer[letter] <= s[letter] for letter in programmer)

# Use 2 pointers coming from left and right of the minimal possible case: programmerprogrammer
# Then move the left pointer to the right and right pointer to the left
# If at some point both substrings on left and right are programmer string then every indices
# between left and right pointers are valid
# Else if left pointer has gone past right pointer and still nothing found then there is no such indice.
def  programmerStrings(s):
    if len(s) < 2*len('programmer')+1:
        return 0
    start = len('programmer')
    end = len(s)-1-len('programmer')
    while (start<=end):
        has_string_left = is_programmer_string(s[0:start])
        has_string_right = is_programmer_string(s[end+1:len(s)])
        condition = has_string_left and has_string_right
        # print(start,end,condition,has_string_left,has_string_right,s[end+1:len(s)])
        if condition:
            return end-start+1
        if not has_string_left:
            start += 1
        if not has_string_right:
            end -= 1
```

Jane Street

Jane Street Coding Problem

04m : 35s
to test end

```
f = open(os.environ['OUTPUT_PATH'], 'w')


try:
    _s = str(input())
except:
    _s = None

res = programmerStrings(_s)
f.write(str(res) + "\n")

f.close()

```

Line: 13 Col: 19

□ Test against custom input

Run Code

Submit code & Continue

(You can submit any number of times)

Download sample test cases *The input/output files have Unix line endings. Do not use Notepad to edit them on windows.*

Compiled successfully. All available test cases passed!

Tip: Debug your code against custom input

Test Case #1: ✓
Test Case #2: ✓
Test Case #3: ✓
Test Case #4: ✓
Test Case #5: ✓
Test Case #6: ✓
Test Case #7: ✓
Test Case #8: ✓
Test Case #9: ✓
Test Case #10: ✓
Test Case #11: ✓
Test Case #12: ✓
Test Case #13: ✓

Testcase 1: *Success*

Input [Download]

```
progxrammerrxproxgrammer
```

Your Output

```
2
```

Expected Output [Download]

```
2
```

Testcase 2: *Success*

Input [Download]

```
xprogxrmaxemrppprmmograeiruu
```

Your Output

```
2
```

Expected Output [Download]

```
2
```

Testcase 3: *Success*

Input [Download]

```
programmerprogrammer
```

Your Output

```
0
```

Expected Output [Download]

```
0
```

Testcase 4: *Success*

Testcase 5: *Success*

Your Output

```
Output hidden
```

Testcase 6: *Success*

Your Output

```
Output hidden
```

Testcase 7: *Success*

Your Output

```
Output hidden
```

Testcase 8: *Success*

Your Output

```
Output hidden
```

Testcase 9: *Success*

Your Output

```
Output hidden
```

Testcase 10: *Success*

Your Output

```
Output hidden
```

Testcase 11: *Success*

Testcase 12: *Success*

Your Output

```
Output hidden
```

Testcase 13: *Success*

Your Output

```
Output hidden
```

About Privacy Policy Terms of Service