itertools.groupby vs Run-length encoding

#pointers #hackerrank #itertools #run-length-encoding

LinkedIn: Md. Ziaul Karim

Find me here:

Compress the String

Medium

In this task, we would like for you to appreciate the usefulness of the *groupby()* function of *itertools* . To read more about this function, Check this out .

You are given a string S. Suppose a character 'c' occurs consecutively X times in the string. Replace these consecutive occurrences of the character with '(X, c)' in the string.

For a better understanding of the problem, check the explanation.

Input Format

A single line of input consisting of the string S.

Output Format

A single line of output consisting of the modified string.

Constraints

All the characters of S denote integers between 0 and 9 .
$1 \leq |S| \leq 10^4$

Sample Input

```
1222311
```

Sample Output

```
(1, 1) (3, 2) (1, 3) (2, 1)
```

Explanation

First, the character occurs only once. It is replaced by . Then the character occurs three times, and it is replaced by and so on.

Also, note the single space within each compression and between the compressions.

`itertools.groupby` Solution

```
from itertools import groupby as gb
def compress_the_string(string):
    result = ""
    for key, group in gb(string):
        result+=f"({len(list(group))}, {key}) "
    return result
if __name__ == "__main__":
    string = "1222311"
    result = compress_the_string(string)
    print(result)
```

Output

```
(1, 1) (3, 2) (1, 3) (2, 1)
```

`Run-length encoding` Solution

```
def compress_the_string(string):
    result = ""
    count = 1
    for i in range(1, len(string)):
        if string[i] == string[i - 1]:
            count += 1
        else:
            result+=f'({count}, {int(string[i - 1])}) '
            count = 1
    result+=f'({count}, {int(string[-1])})'
    return result

if __name__ == '__main__':
    string = "1222311"
    result = compress_the_string(string)
    print(result)
```

Output

```
(1, 1) (3, 2) (1, 3) (2, 1)
```

End