题目描述

输入单行英文句子，里面包含英文字母，空格以及,.?三种标点符号，请将句子内每个单词进行倒序，并输出倒序后的语句。

输入描述

输入字符串S，S的长度 1 ≤ N ≤ 100

输出描述

输出倒序后的字符串

备注

标点符号左右的空格 ≥ 0，单词间空格 > 0

用例

输入	yM eman si boB.
输出	My name is Bob.
说明	无

输入	woh era uoy ? I ma enif.
输出	how are you ? I am fine.
说明	无

题目解析

从用例可以看出，单词的倒序并不难，将字符串单词转为字符数组后，reverse一下就行了。但是单词中如果有标点符号的话，则标点符号的位置不能改变，比如enif. 倒序后为 fine. 其中 . 的位置在倒序前后是一样的。

我的解题思路如下，从左到右遍历每一个字符，如果字符是 , . ? 或者空格，则看成一个分界符，将分界符之间的单词片段进行倒序。

JavaScript算法源码

```
/* JavaScript Node ACM模式 控制台输入获取 */
const readline = require("readline");

const rl = readline.createInterface({
  input: process.stdin,
  output: process.stdout,
});

rl.on("line", (line) => {
  console.log(getResult(line));
});

function getResult(str) {
  const reg = /[\,\.\?\s]/;
  const idxs = [-1];

  for (let i = 0; i < str.length; i++) {
    if (reg.test(str[i])) {
      idxs.push(i);
    }
  }

  idxs.push(str.length);

  const arr = [...str];

  idxs.reduce((p, c) => {
    let l = p + 1;
    let r = c - 1;

    while (l < r) {
      let tmp = arr[l];
```

JavaScript算法源码

```javascript
/* JavaScript Node ACM模式 控制台输入获取 */
const readline = require("readline");

const rl = readline.createInterface({
  input: process.stdin,
  output: process.stdout,
});

rl.on("line", (line) => {
  console.log(getResult(line));
});

function getResult(str) {
  const reg = /[\,\.\?\s]/;
  const idxs = [-1];

  for (let i = 0; i < str.length; i++) {
    if (reg.test(str[i])) {
      idxs.push(i);
    }
  }

  idxs.push(str.length);

  const arr = [...str];

  idxs.reduce((p, c) => {
    let l = p + 1;
    let r = c - 1;

    while (l < r) {
      let tmp = arr[l];
      arr[l] = arr[r];
      arr[r] = tmp;
      l++;
      r--;
    }

    return c;
  });

  return arr.join("");
}
```

更精简的解法，可以利用String.prototype.repalce的正则匹配出输入字符串中各个英文子串，将这些英文子串替换为倒序子串，关于repalce的正则匹配用法请看：

String.prototype.replace() - JavaScript | MDN (mozilla.org)

```javascript
/* JavaScript Node ACM模式 控制台输入获取 */
const readline = require("readline");

const rl = readline.createInterface({
  input: process.stdin,
  output: process.stdout,
});

rl.on("line", (line) => {
  console.log(getResult(line));
});

function getResult(str) {
  return str.replace(/[a-zA-Z]+/g, (s) => [...s].reverse().join(""));
}
```

Java算法源码

```java
import java.util.ArrayList;
import java.util.Scanner;

public class Main {
  public static void main(String[] args) {
    Scanner sc = new Scanner(System.in);

    String str = sc.nextLine();
    System.out.println(getResult(str));
  }

  public static String getResult(String str) {
    ArrayList<Integer> idxs = new ArrayList<>();

    idxs.add(-1);
    for (int i = 0; i < str.length(); i++) {
      if (",.? ".indexOf(str.charAt(i)) != -1) {
        idxs.add(i);
      }
    }
    idxs.add(str.length());
```

Java算法源码

```java
import java.util.ArrayList;
import java.util.Scanner;

public class Main {
  public static void main(String[] args) {
    Scanner sc = new Scanner(System.in);

    String str = sc.nextLine();
    System.out.println(getResult(str));
  }

  public static String getResult(String str) {
    ArrayList<Integer> idxs = new ArrayList<>();

    idxs.add(-1);
    for (int i = 0; i < str.length(); i++) {
      if (",.? ".indexOf(str.charAt(i)) != -1) {
        idxs.add(i);
      }
    }
    idxs.add(str.length());

    char[] chars = str.toCharArray();

    for (int i = 0; i < idxs.size() - 1; i++) {
      int l = idxs.get(i) + 1;
      int r = idxs.get(i + 1) - 1;

      while (l < r) {
        char tmp = chars[l];
        chars[l] = chars[r];
        chars[r] = tmp;
        l++;
        r--;
      }
    }

    StringBuilder sb = new StringBuilder();
    for (char c : chars) {
      sb.append(c);
    }
    return sb.toString();
  }
}
```

Python算法源码

```python
import re

s = input()

p = re.compile(r"[\\,\\.\\?\s]")

idxs = [-1]

for i in range(len(s)):
    if p.match(s[i]):
        idxs.append(i)

idxs.append(len(s))

arr = [c for c in s]

for i in range(len(idxs) - 1):
    l = idxs[i] + 1
    r = idxs[i + 1] - 1

    while l < r:
        arr[l], arr[r] = arr[r], arr[l]
        l += 1
        r -= 1

print("".join(arr))
```

更精简的，可以利用re.sub(pattern, repl, string, count=0, flags=0)

其中repl可以是一个函数，该函数接收被pattern正则匹配结果re.Match类型的值作为入参matched，我们可以通过matched.group()获取出匹配的子串，并对它做反转处理，然后返回给re.sub进行替换

```python
import re

s = input()


def rep(matched):
    tmp = list(matched.group())
    tmp.reverse()
    return "".join(tmp)
```

Python算法源码

```python
import re

s = input()

p = re.compile(r"[\\,\\.\\?\s]")

idxs = [-1]

for i in range(len(s)):
    if p.match(s[i]):
        idxs.append(i)

idxs.append(len(s))

arr = [c for c in s]

for i in range(len(idxs) - 1):
    l = idxs[i] + 1
    r = idxs[i + 1] - 1

    while l < r:
        arr[l], arr[r] = arr[r], arr[l]
        l += 1
        r -= 1

print("".join(arr))
```

更精简的，可以利用re.sub(pattern, repl, string, count=0, flags=0)

其中repl可以是一个函数，该函数接收被pattern正则匹配结果re.Match类型的值作为入参matched，我们可以通过matched.group()获取出匹配的子串，并对它做反转处理，然后返回给re.sub进行替换

```python
import re

s = input()


def rep(matched):
    tmp = list(matched.group())
    tmp.reverse()
    return "".join(tmp)


print(re.sub(r"[a-zA-Z]+", rep, s))
```