


【华为机试真题 Python实现】火星文计算【2022 Q1 Q2 | 100分】

原创 不太灵光的程序员 于 2022-06-29 00:51:19 发布 700 收藏 3 版权

分类专栏: 华为机试真题详解 华为机试真题 文章标签: 华为 华为OD 机试 Python 算法

 华为机试真题 同时被 2 个专栏收录
该专栏为热销专栏榜 第3名

¥59.90
¥99.00

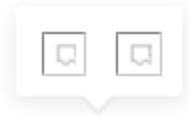
761 订阅 213 篇文章

已订阅

超级会员免费看

文章目录

- 前言
- 题目描述
- 示例 1
- 参考代码



前言

如果您在准备华为的面试，期间有想了解的可以私信我，我会尽可能帮您解答，也可以给您一些建议！

本文解法非最优解（即非性能最优）。

特别提醒！！！！

注意1：机试为ACM 模式

你的代码需要处理输入输出， `input` 接收输入、 `print` 格式化输出

注意2：机试按通过率记分

复杂题目可以考虑暴力破解，再逐步优化，不是运行超时就无法得分，如下，提交结果运行超时，但用例通过率>92.31%，如果是100分的题目，可以得92.3分。

代码提交记录



提交结果：运行超时 运行时间：2001ms 占用内存：4532KB 使用语言：Python 3
用例通过率：92.31%

```
1 import re
2 while True:
3     try:
4         s1 = input().lower()
5         s2 = input().lower()
6         s1 = s1.replace('.', '\\.').replace('?', '[0-9a-z]').replace('*', '[0-9a-z]*')
7         # s1 = re.sub('#+', '[0-9a-z]*', s1)
8         if bool(re.fullmatch(s1, s2)):
9             print('true')
10        else:
11            print('false')
12    except:
13        break
```

CSDN @不太灵光的程序员

题目描述

已知火星使用的运算符为#、\$，

其与地球人的等价公式如下：

$$x\#y = 2x+3y+4$$

$$x\$y = 3*x+y+2$$

其中x、y是无符号整数

地球人公式按C语言规则计算，火星公式中，\$的 优先级 高于#，相同的运算符，按从左到右的顺序计算 现有一段火星人的字符串报文，请你来翻译并计算结果。

输入描述：

火星 字符串 表达式（结尾不带回车换行）

输入的字符串说明：

字符串为仅由无符号整数和操作符（#、\$）

组成的计算 表达式。例如：123#45#6778

用例保证字符串中，操作数与操作符之间没有任何分隔符。

用例保证操作数取值范围为32位无符号整数。

保证输入以及计算结果不会出现整型溢出。

保证输入的字符串为合法的求值报文，例如：123#45#6778

保证不会出现非法的求值报文，例如类似这样字符串：

#4\$5 //缺少操作数

4\$5# //缺少操作数

4#\$5 //缺少操作数

4 \$5 //有空格

3+4-5*6/7 //有其它操作符

12345678987654321\$54321 //32位整数计算溢出

输出描述：

根据输入的火星字符串输出计算结果（结尾不带回车换行）

示例 1

输入：

7#6\$5#12

输出：

226

说明：

7#6\$5#12

=7#(36+5+2)#12

=7#25#12

=(27+325+4)#12

=93#12

=293+3*12+4

=226

参考代码

```
1 while 1:
2     try:
3         nums = input()
4         # 无符号数
5         index_list = [i for i, c in enumerate(nums) if c in "#$"]
6
7         # 按操作符 分隔字符串
8         stack = []
9         s = 0
10        for e in index_list:
11            stack.append(int(nums[s:e]))
12            stack.append(nums[e])
13            s = e + 1
14        else:
15            stack.append(int(nums[s:]))
16
17        # 计算 优先级高的 f"3*{x}+{y}+2"
18        r_stack = []
19        while stack:
20            item = stack.pop()
21            if item == "$":
22                x = stack.pop()
23                y = r_stack.pop()
24                stack.append(eval(f"3*{x}+{y}+2"))
25            else:
26                r_stack.append(item)
27        stack += r_stack[::-1]
28
29        # 计算 2*{x}+3*{y}+4"
30        x = stack[0]
31        for i in range(2, len(stack), 2):
32            y = stack[i]
33            x = eval(f"2*{x}+3*{y}+4")
34        print(x)
35    except Exception as e:
36        break
```