# Задание по регулярным выражениям

**Описание задания**

В этом задании нужно написать программу, которая найдет в тексте ряд простых арифметических выражений и подсчитает их.

Любое выражение начинается с имени переменной **a**, **b** или **c**, затем может идти **+** или **-**, затем идет **=**, затем может идти имя переменной **a**, **b** или **c**, а затем может идти **+** или **-** и целое число. Если в правой части выражения нет переменной, то число может быть без знака **+** или **-**

Не бывает названий переменных, кроме a, b или c, и действий, кроме описанных тут. Не бывает пробелов вокруг знаков. В тексте не встречаются некорректные выражения, в которых справа от = нет ни переменной, ни числа. Таким образом, список типов выражений, которые могут встречаться, выглядит примерно так:

a=1, a=+1, a=-1, a=b, a=b+100, a=b-100, b+=10, b+=+10, b+=-10, b+=b, b+=b+100, b+=b-100, c-=101, c-=+101, c-=-101, c-=b, c-=b+101, c-=b-101

Выражения могут встречаться внутри текста, например lorem**c-=a+10**ipsum**a-=a**d**b+=10**olorsitamet.

В вашу функцию calculate(data, findall) будет передан словарь с начальными значениями переменных a, b и c: data = {"a":1, "b":2, "c": 3} и ссылка на функцию findall, а вы должны вернуть такой же словарь с новыми значениями для a, b и c.

Работает findall() аналогично re.findall(), только у нее всего один параметр – регулярное выражение. А текст, в котором она будет искать выражения, она знает сама (см. приложенный архив, чтобы понять, о чем речь). С помощью findall() нужно как минимум найти все выражения в тексте, а как максимум найти их и разбить на группы так, чтобы было удобно их обработать. *Если findall() будет вызвано больше одного раза или если размер списка, который она вернет, будет отличаться от количества выражений в тексте, тест провалится.*

Эталонное решение занимает 11 строк, не содержит импортов, не использует eval, а регулярное выражение находит выражения и бьет каждое на четыре группы (некоторые группы для некоторых выражений оказываются пустыми):

1. Имя переменной слева.
2. Знак перед = (если есть).
3. Имя переменной справа (если есть).
4. Число (если есть) со знаком (если есть).

Это позволяет легко (буквально в одну строку) посчитать правую часть, а потом, в зависимости от наличия знака перед =, произвести действие с левой частью. Однако ваш алгоритм может быть другим, требуется только выполнить ограничения на вызов findall() и оставить сигнатуру calculate() неизменной.

Вы должны скачать архив к этому уроку, изучить оба файла и переписать функцию calculate() внутри regexp.py. Для проверки вашего решения локально запустите *python test.py*. На сервере задание будет проверяться похожим образом, но реализация findall() будет иной (с проверкой всех ограничений), и тестов будет несколько. Решением будет файл regexp.py, который надо загрузить на сервер для проверки.