**6 kyu**

**String Evaluation**

13194% of 4792 of131[sataman](https://www.codewars.com/users/sataman)

Python

* [TRAIN AGAIN](https://www.codewars.com/kata/string-evaluation/train/python)
* [NEXT KATA](https://www.codewars.com/trainer/python)

Details

[Solutions](https://www.codewars.com/kata/string-evaluation/solutions/python)

[Forks (6)](https://www.codewars.com/kata/string-evaluation/forks/python)

[Discourse (10)](https://www.codewars.com/kata/string-evaluation/discuss/python)

* Add to Collection
* |
* Share this kata:

Evaluate the given string with the given conditons.

The conditions will be passed in an array and will be formatted like this:

{symbol or digit}{comparison operator}{symbol or digit}

Return the results in an array.

The characters in the conditions will always be in the string. Characters in the string are chosen from ascii letters + @#$%^&\*()\_{}[]

Example

input string: "aab#HcCcc##l#"

conditions: ["a<b", "#==4", "c>=C", "H!=a"]

The conditions in this example array can be interpreted as:

* "a<b": The number of times "a" occurs in the string should be less than the number of times "b" occurs in the string
* "#==4": "#" should occur exactly 4 times in the string
* "c>=C": "c" should occur greater than or equal to the number of times "C" occurs
* "H!=a": The number of times "H" occurs should not equal the number of times "a" occurs

In this example condition 1 is false and 2, 3, 4 are true. So the return value will be an array as such:

[False, True, True, True]

<https://www.codewars.com/kata/string-evaluation/python>

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace ConsoleApp1

{

class Program

{

static List<bool> string\_evaluation(string strng, string[] conditions)

{

Dictionary<string , int> diccio = new Dictionary<string , int>();

for (int i = 0; i < strng.Length; i++)

{

if (diccio.ContainsKey(strng[i].ToString())) diccio[strng[i].ToString()]++;

else diccio[strng[i].ToString()] = 1;

}

List<bool> ans = new List<bool>();

string[] c = { "==", "!=", ">=", "<=", ">", "<" };

for (int i = 0; i < conditions.Length; i++)

{

string[] variables = conditions[i].Split(c, StringSplitOptions.RemoveEmptyEntries);

string op = "";

foreach (string operador in c)

{

if (conditions[i].Contains(operador))

{

op = operador;

break;

}

}

string a = variables[0];

string b = variables[1];

int cantA = 0;

int cantB = 0;

if (diccio.ContainsKey(a))

{

cantA = diccio[a];

}

int ia = 0;

if (char.IsNumber(a[0]))

{

ia = int.Parse(a);

cantA = ia;

}

if (diccio.ContainsKey(b))

{

cantB = diccio[b];

}

int ib = 0;

if (char.IsNumber(b[0]))

{

ib = int.Parse(b);

cantB = ib;

}

if (op == c[0]) ans.Add(cantA == cantB);

else if (op == c[1]) ans.Add(cantA != cantB);

else if (op == c[2]) ans.Add(cantA >= cantB);

else if (op == c[3]) ans.Add(cantA <= cantB);

else if (op == c[4]) ans.Add(cantA > cantB);

else if (op == c[5]) ans.Add(cantA < cantB);

}

return ans;

}

static void Main(string[] args)

{

string s = "aab#HcCcc##l#";

string[] cond = { "a<b", "#==4", "c>=C", "H!=a" };

//string s = "abc#$%KDAyyaa@@@";

//string[] cond2 = { "#>@", "A==2", "a>A", "$!=2" };

// EvaluarCondiciones(cond);

List<bool> ans = string\_evaluation(s, cond);

foreach(bool item in ans)

{

Console.Write(item + " ");

}

Console.ReadLine();

}

}

}

---------------------------PYTHON ---------------

**def** miSplit(cad):

    c = [ "==", "!=", ">=", "<=", ">", "<" ]

    sp = []

**for** i **in** range(0, len(c)):

        sp = cad.split(c[i])

**if**(len(sp) > 1) : **return** sp

**return** []

**def** string\_evaluation(strng, conditions):

    diccio = {}

**for** i **in** range(0, len(strng)):

**if**(str(strng[i]) **in** diccio):

            diccio[str(strng[i])] += 1

**else**:

            diccio[str(strng[i])] = 1

    ans = []

    c = [ "==", "!=", ">=", "<=", ">", "<" ]

**for** i **in** range(0, len(conditions)):

        variables = miSplit(conditions[i])

        op = ""

**for** operador **in** c:

**if**(operador **in** conditions[i]):

                op = operador

**break**

        a = variables[0]

        b = variables[1]

        cantA = 0

        cantB = 0

**if**(a **in** diccio):

            cantA = diccio[a]

**if**(a.isnumeric()):

            cantA = int(a)

**if**(b **in** diccio):

            cantB = diccio[b]

**if**(b.isnumeric()):

            cantB = int(b)

**if** (op == c[0]): ans.append(cantA == cantB)

**elif** (op == c[1]): ans.append(cantA != cantB)

**elif** (op == c[2]): ans.append(cantA >= cantB)

**elif** (op == c[3]): ans.append(cantA <= cantB)

**elif** (op == c[4]): ans.append(cantA > cantB)

**elif** (op == c[5]): ans.append(cantA < cantB)

**return** ans