**6 kyu**

**Number , number ... wait LETTER !**

27496% of 7294 of184[Cl3anKillx](https://www.codewars.com/users/Cl3anKillx)

Python

* [TRAIN AGAIN](https://www.codewars.com/kata/number-number-dot-dot-dot-wait-letter/train/python)
* [NEXT KATA](https://www.codewars.com/trainer/python)

Details

[Solutions](https://www.codewars.com/kata/number-number-dot-dot-dot-wait-letter/solutions/python)

[Forks (3)](https://www.codewars.com/kata/number-number-dot-dot-dot-wait-letter/forks/python)

[Discourse (35)](https://www.codewars.com/kata/number-number-dot-dot-dot-wait-letter/discuss/python)

* Add to Collection
* |
* Share this kata:

Your task is to write a function named do\_math that receives a single argument. This argument is a string that contains multiple whitespace delimited numbers. Each number has a single alphabet letter somewhere within it.

Example : "24z6 1x23 y369 89a 900b"

As shown above, this alphabet letter can appear anywhere within the number. You have to extract the letters and sort the numbers according to their corresponding letters.

Example : "24z6 1x23 y369 89a 900b" will become 89 900 123 369 246 (ordered according to the alphabet letter)

Here comes the difficult part, now you have to do a series of computations on the numbers you have extracted.

* The sequence of computations are + - \* /. Basic math rules do **NOT** apply, you have to do each computation in exactly this order.
* This has to work for any size of numbers sent in (after division, go back to addition, etc).
* In the case of duplicate alphabet letters, you have to arrange them according to the number that appeared first in the input string.
* Remember to also round the final answer to the nearest integer.

Examples :

"24z6 1x23 y369 89a 900b" = 89 + 900 - 123 \* 369 / 246 = 1299

"24z6 1z23 y369 89z 900b" = 900 + 369 - 246 \* 123 / 89 = 1414

"10a 90x 14b 78u 45a 7b 34y" = 10 + 45 - 14 \* 7 / 78 + 90 - 34 = 60

Good luck and may the CODE be with you!

<https://www.codewars.com/kata/number-number-dot-dot-dot-wait-letter/python>

**def** insertionSort(letras, numeros):

**for** i **in** range(1, len(letras)):

        indice = i

**while**(indice > 0 **and** letras[indice - 1] > letras[indice]):

            tempString = letras[indice - 1]

            letras[indice - 1 ] = letras [indice]

            letras[indice ] = tempString

            tempNum = numeros[indice - 1 ]

            numeros[indice -1] = numeros[indice]

            numeros[indice] = tempNum

            indice -= 1

**def** do\_math(s) :

*#Your code starts here ... may the FORCE be with you*

    s = s.strip()

    sp = s.split(' ')

    nums = [0.0]\* len(sp)

    letras = [""] \* len(sp)

**for** i **in** range(0, len(sp)) :

        numero = ""

        temp = sp[i]

**for** j **in** range(0, len(temp)):

**if** (temp[j].isalpha()):

*#letras[i] = str() temp[j]*

                letras[i] = (str((temp[j]).lower()))

**else**:

                numero += temp[j] + ""

        nums[i] = (int(numero))

        numero = ""

    insertionSort(letras, nums)

**for** item **in** nums:

**print**(item)

    op = "+-\*/"

    indOperador = 0

    suma = 0.0

    suma = float(nums[0])

**for** i **in** range(1, len(nums)):

**if**(op[indOperador] == '+'):

            suma = suma + nums[i]

**elif**(op[indOperador] == '-'):

            suma = suma - nums[i]

**elif** (op[indOperador] == '\*'):

            suma = suma \* nums[i]

**elif** (op[indOperador] == '/'):

*#aux = round(suma / nums[i])*

*#suma = int (round(aux));*

            suma = (float(suma) / float(nums[i]))

        indOperador += 1

**if**(indOperador >= len(op)): indOperador = 0

**return** int(round(suma))

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace ConsoleApp1

{

class Program

{

//static int find\_x(int n)

//{

// int x = 0;

// for (int i = 0; i < n; i++)

// {

// for (int j = 0; j < 2 \* n; j++)

// {

// x += j + i;

// }

// }

// return x;

//}

static int do\_math(string s)

{

//# Your code starts here ... may the FORCE be with you

// string s = "a21 5a6 6666a221 89a87";

string[] sp = s.Split(' ');

int[] nums = new int[sp.Length];

string[] letras = new string[sp.Length];

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

{

string numero = "";

string temp = sp[i];

for (int j = 0; j < temp.Length; j++)

{

if (char.IsLetter(temp[j]))

{

//letras.Add(temp[j].ToString());

letras[i] = temp[j].ToString();

}

else

{

numero += temp[j] + "";

}

}

//nums.Add(int.Parse(numero));

nums[i] = int.Parse(numero);

numero = "";

}

insertionSort(letras, nums);

foreach (int item in nums)

{

Console.Write(item + " ");

}

Console.WriteLine();

string op = "+-\*/";

int indOperador = 0;

double sum = nums.First();

for (int i = 1; i < nums.Length; i++)

{

if (op[indOperador] == '+')

{

sum = sum + nums[i];

}

else if (op[indOperador] == '-')

{

sum = sum - nums[i];

}

else if (op[indOperador] == '\*')

{

sum = sum \* nums[i];

}

else if (op[indOperador] == '/')

{

double aux = (sum / nums[i]);

sum = aux;

}

indOperador++;

if (indOperador >= op.Length)

{

indOperador = 0;

}

}

Console.WriteLine(sum);

return (int)Math.Round( sum);

}

static void insertionSort(string[] letras, int[] numeros)

{

for (int i = 1; i < letras.Length; i++)

{

int indice = i;

while (indice > 0 && letras[indice - 1].CompareTo(letras[indice]) > 0) //ar[indice - 1] > ar[indice])

{

string tempString = letras[indice - 1];

letras[indice - 1] = letras[indice];

letras[indice] = tempString;

int tempNum = numeros[indice - 1];

numeros[indice - 1] = numeros[indice];

numeros[indice] = tempNum;

indice--;

}

}

}

static void Main()

{

//string s = "a21 5a6 6666a221 89a87";

//List<string> sp = s.Split(' ').ToList();

//List<string> letras = new List<string>();

//letras.Add("a");

//letras.Add("a");

//letras.Add("a");

//letras.Add("a");

//List<int> nums = new List<int>(new int[] { 41, 56, 44, 32 });

//string[] letras = { "c", "a", "b", "a" };

//int[] nums = { 41, 56, 44, 32 };

//OrdenarParalelos(letras, nums);

//foreach (int item in nums)

//{

// Console.Write(item + " ");

//}

// string s = "24z6 1z23 y369 89z 900b";

string s = "1566e 461p 1670e 501b 746t 799d 525x 1420n"; //-313950 should equal -313861

Console.WriteLine(do\_math(s));

Console.ReadLine();

}

//static void Main(string[] args)

//{

// //for (int i = 1; i < 20; i++)

// //{

// // Console.WriteLine(find\_x(i));

// //}

// /\*

// test.assert\_equals(do\_math("24z6 1z23 y369 89z 900b"), 1414)

// test.assert\_equals(do\_math("24z6 1x23 y369 89a 900b"), 1299)

// test.assert\_equals(do\_math("10a 90x 14b 78u 45a 7b 34y"), 60)

// test.assert\_equals(do\_math("111a 222c 444y 777u 999a 888p"), 1459)

// test.assert\_equals(do\_math("1z 2t 3q 5x 6u 8a 7b"), 8)

// \*/

// string s = "24z6 1z23 y369 89z 900b";

// string[] sp = s.Split(' ');

// List<int> nums = new List<int>();

// List<string> letras = new List<string>();

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

// {

// string numero = "";

// string temp = sp[i];

// for (int j = 0; j < temp.Length; j++)

// {

// if (char.IsLetter(temp[j]))

// {

// letras.Add(temp[j].ToString());

// }

// else

// {

// numero += temp[j] + "";

// }

// }

// nums.Add(int.Parse(numero));

// numero = "";

// }

// foreach (int num in nums)

// {

// Console.Write(num + " ");

// }

// Console.ReadLine();

//}

}

}