**1 - Two Strings**

"Write a function that takes in two strings, s1 and s2, containing only letters from 'a' to 'z'. The function should return a new string, containing distinct letters in alphabetical order, each taken only once from either s1 or s2. The returned string should be the longest possible.

Examples:

* The function called with the inputs 'xyaabbbccccdefww' and 'xxxxyyyyabklmopq' should return 'abcdefklmopqwxy'.
* The function called with the inputs 'abcdefghijklmnopqrstuvwxyz' and 'abcdefghijklmnopqrstuvwxyz' should return 'abcdefghijklmnopqrstuvwxyz'."

def onlyLettersFromaToz(string):

ENGLISHALPHABETATOZLOWERCASESTRING='abcdefghijklmnopqrstuvwxyz'

for x in list(string):

#find function is case sensitive

#index function is not case sensitive

index=ENGLISHALPHABETATOZLOWERCASESTRING.find(x)

if index==-1:

return False

return True

def run(s1, s2):

auxString=s1+s2

#containing only letters from 'a' to 'z'

if not onlyLettersFromaToz(auxString):

raise Exception("Only Letters from 'a' to 'z'")

#other away but is in this case dont validate case

#if not auxString.isalpha():

#raise Exception("Only Letters from 'a' to 'z'")

#containing distinct letters

lst = []

for letter in auxString:

if letter not in lst: lst.append(letter)

#alphabetical order

lst.sort()

result = ''.join([str(elem) for elem in lst])

#return a new string

return result

run('xyaabbbccccdefww', 'xxxxyyyyabklmopq')

run('abcdefghijklmnopqrstuvwxyz', 'abcdefghijklmnopqrstuvwxyz')

----------------------------------------

2 - Arrays of integers

Write a function that takes in an array of integers and returns an index N such that the sum of the integers to the left of N is equal to the sum of the integers to the right of N. If no such index exists, the function should return -1. The function should return the lowest index in case of multiple valid answers.

Examples:

* The function called with the input [1,2,3,4,3,2,1] should return 3, because the sum of the elements to the left of index 3 ([1, 2, 3]) is equal to the sum of the elements to the right of index 3 ([3, 2, 1]).
* The function called with the input [1,100,50,-51,1,1] should return 1, because the sum of the elements to the left of index 1 ([1]) is equal to the sum of the elements to the right of index 1 ([50, -51, 1, 1]).
* The function called with the input [20, 10, -80, 10, 10, 15, 35] should return 0, because the sum of the elements to the left of index 0 ([]) is equal to the sum of the elements to the right of index 0 ([10, -80, 10, 10, 15, 35]).

Note:

* The input array may have length 0 < arr < 1000 and may contain any integer values, positive or negative.

def validArraySize(array, min, max):

size = len(array)

if min < size and size < max:

return True

return False

#Write a function that takes in an array of integers

def run(integers):

#The input array may have length 0 < arr < 1000 and may contain any integer values, positive or negative.

if not validArraySize(integers, 0, 1000):

raise Exception("The input array may have length 0 < arr < 1000")

#If no such index exists, the function should return -1

n=-1

for pointer in range((len(integers)-1)):

auxinput = integers.copy()

del auxinput[pointer]

#left of N

left=auxinput[:pointer]

#right of N

right=auxinput[pointer:]

#sum of the integers

sumLeft = sum(left)

sumRight = sum(right)

#such that the sum of the integers to the left of N

#is equal to the sum of the integers to the right of N

#The function should return the lowest index in case of multiple valid answers (first find fisrt return)

if sumLeft == sumRight:

n = pointer

break

return n

run([1,2,3,4,3,2,1])

run([1,100,50,-51,1,1])

run([20, 10, -80, 10, 10, 15, 35])

----------------------------------------

3 - Binary code

Write a function that takes in an array of ones and zeroes, and converts the equivalent binary value to an integer.

Examples:

* The function called with the input [0, 0, 0, 1] should return 1, because 0001 is the binary representation of 1.
* The function called with the input [0, 0, 1, 0] should return 2, because 0010 is the binary representation of 2.
* The function called with the input [0, 1, 0, 1] should return 5, because 0101 is the binary representation of 5.
* The function called with the input [1, 0, 0, 1] should return 9, because 1001 is the binary representation of 9.
* The function called with the input [0, 0, 1, 0] should return 2, because 0010 is the binary representation of 2.
* The function called with the input [0, 1, 1, 0] should return 6, because 0110 is the binary representation of 6.
* The function called with the input [1, 1, 1, 1] should return 15, because 1111 is the binary representation of 15.
* The function called with the input [1, 0, 1, 1] should return 11, because 1011 is the binary representation of 11.

Note:

* The input array can have varying lengths, not just limited to 4

def validBinaryArray(array):

VALID\_NUMBERS=[0,1]

for x in array:

try:

index=VALID\_NUMBERS.index(x)

except:

return False

return True

def run(binaryString):

#takes in an array of ones and zeroes

if not validBinaryArray(binaryString):

raise Exception("Just ones and zeroes")

#The input array can have varying lengths, not just limited to 4

#and converts the equivalent binary value to an integer

binaryString.reverse()

aux = []

for index, item in enumerate(binaryString):

aux.append(item\*pow(2, index))

result = sum(aux)

return result

run([0, 0, 0, 1])

run([0, 0, 1, 0])

run([0, 1, 0, 1])

run([1, 0, 0, 1])

run([0, 0, 1, 0])

run([0, 1, 1, 0])

run([1, 1, 1, 1])

run([1, 0, 1, 1])

----------------------------------------

4 - Find the missing letter

Write a method that takes an array of consecutive (increasing) letters as input and returns the missing letter in the array. You will always receive a valid array and it will always be missing exactly one letter. The length of the array will always be at least 2. The array will always contain letters in only one case.

Examples:

['a', 'b', 'c', 'd', 'f'] => 'e'

['O', 'Q', 'R', 'S'] => 'P'

["a", "b", "c", "d", "f"] => "e"

["O", "Q", "R", "S"] => "P"

Note: Use the English alphabet with 26 letters.

import sys

#Write a method that takes an array of consecutive (increasing) letters as input

def run(input):

if not validArraySize(input, 2, sys.maxsize):

raise Exception("The length of the array will always be at least 2")

englishAlphabet = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j', 'k', 'l', 'm', 'n', 'o', 'p', 'q', 'r', 's', 't', 'u', 'v', 'w', 'x', 'y', 'z']

standardformat = input.copy()

#[standardformat.lower() for x in standardformat]

standardformat = list(map(lambda x: x.lower(), standardformat))

startIndex = englishAlphabet.index(standardformat[0])

aux = -1

#You will always receive a valid array and it will always be missing exactly one letter.

#The length of the array will always be at least 2

for index, itEnglishAlphabet in enumerate(englishAlphabet[startIndex:]):

if itEnglishAlphabet!=standardformat[index]:

aux = itEnglishAlphabet

break

#The array will always contain letters in only one case.

aux = aux if input[0].islower() else aux.upper()

result=aux

#returns the missing letter in the array

return result

run(['a', 'b', 'c', 'd', 'f'])

run(['O', 'Q', 'R', 'S'])

run(["a", "b", "c", "d", "f"])

run(["O", "Q", "R", "S"])

----------------------------------------

5 - Welcome

In this task, you are required to replace every letter in a given string with its position in the alphabet. If the string contains any non-letter characters, ignore them and do not include them in the output. For example, the letter "a" should be replaced with "1", "b" should be replaced with "2", and so on.

Example:

alphabet\_position("The sunset sets at twelve o'clock.")

Should return "20 8 5 19 21 14 19 5 20 19 5 20 19 1 20 20 23 5 12 22 5 15 3 12 15 3 11" (as a string)

Note: The output should be a string of space-separated integers

def run(input):

englishAlphabet = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j', 'k', 'l', 'm', 'n', 'o', 'p', 'q', 'r', 's', 't', 'u', 'v', 'w', 'x', 'y', 'z']

standardformat = input

standardformat = list(map(lambda x: x.lower(), standardformat))

auxStandardformat = list(standardformat)

auxx = ''

for itAuxStandardformat in auxStandardformat:

try:

#replace every letter in a given string with its position in the alphabet

#For example, the letter "a" should be replaced with "1"

a = englishAlphabet.index(itAuxStandardformat[0])+1

auxx+= str(a) + ' '

except:

#If the string contains any non-letter characters, ignore them and do not include them in the output

continue

auxx = auxx[:-1]

result = auxx

return result

run("The sunset sets at twelve o'clock.")

----------------------------------------

6- The (((()

The goal of this exercise is to convert a string to a new string where each character in the new string is represented by "(" if it appears only once in the original string, or ")" if it appears more than once. The case of the characters should be ignored.

Examples:

"din" => "((("

"recede" => "()()()"

"Success" => ")())())"

"(( @" => "))(("

Note:

The output string should contain only "(" and ")" characters.

def run(input):

standardformat = input

#The case of the characters should be ignored.

standardformat = list(map(lambda x: x.lower(), standardformat))

aux = {}

for itInput in standardformat:

if itInput not in aux:

aux[itInput]=1

else:

aux[itInput] = aux[itInput]+1

result = ''

for itInput in list(standardformat):

#represented by "(" if it appears only once

if aux[itInput] == 1:

result+='('

#")" if it appears more than once

else:

result+=')'

#The output string should contain only "(" and ")" characters.

return result

run("din")

run("recede")

run("Success")

run("(( @")

----------------------------------------

7- Numbers have funny properties

Write a function that determines if there exists a positive integer k such that the sum of the digits of a given positive integer n, taken to successive powers of p, is equal to k times n.

In other words, the function should return k if the following equation holds:

(a^p + b^(p+1) + c^(p+2) + d^(p+3) + ...) = n \* k

where a, b, c, d are the digits of n.

If no such positive integer k exists, the function should return -1.

Examples:

dig\_pow(89, 1) returns 1, since 8^1 + 9^2 = 89 = 89 \* 1

dig\_pow(92, 1) returns -1, since there is no k such that 9^1 + 2^2 = 92 \* k

dig\_pow(695, 2) returns 2, since 6^2 + 9^3 + 5^4 = 1390 = 695 \* 2

dig\_pow(46288, 3) returns 51, since 4^3 + 6^4 + 2^5 + 8^6 + 8^7 = 2360688 = 46288 \* 51

Notes:

The input integers n and p will always be strictly positive.

def intPositive(value):

if type(value) == int and value > 0:

return True

return False

def run(n,p):

#default value (error)

result = -1

#The input integers n and p will always be strictly positive.

if not intPositive(n) or not intPositive(p):

raise Exception("The input integers n and p will always be strictly positive.")

auxN=str(n)

auxN = list(map(lambda x: int(x), auxN))

#the sum of the digits of a given positive integer n, taken to successive powers of p

aux = []

for it in auxN:

aux.append(pow(it, p))

p+=1

sumPow = sum(aux)

#is equal to k times n.

k = 0

x = n\*k

while x <= sumPow:

k += 1

x = n\*k

if sumPow == x:

result = k

return result

run(89,1)

run(92,1)

run(695,2)

run(46288,3)

----------------------------------------

8- Square

Write a function that determines whether a given integer is a perfect square. A perfect square is an integer that is the product of some integer with itself.

Examples:

isSquare(-1) returns false

isSquare(0) returns true

isSquare(3) returns false

isSquare(4) returns true

isSquare(25) returns true

isSquare(26) returns false

Notes:

The input will always be an integral number.

import math

def run(integer):

n = integer

if n < 0:

return False

else:

sqrtN = int(math.sqrt(n))

return ((sqrtN\*sqrtN)==n)

run(-1)

run(0)

run(3)

run(4)

run(25)

run(26)

----------------------------------------

9- Fibonacci

Write a function that generates the first n elements of the Tribonacci sequence, given a starting signature of 3 integers. The Tribonacci sequence is similar to the Fibonacci sequence, but the next element is obtained by summing the last 3 elements of the sequence instead of the last 2.

Examples:

If the starting signature is [1, 1, 1] and n is 5, the function should return [1, 1, 1, 3, 5]

If the starting signature is [0, 0, 1] and n is 5, the function should return [0, 0, 1, 1, 2]

Notes:

The input signature will always contain 3 integers.

The input n will always be a non-negative integer.

If n is 0, the function should return an empty array (except in C, where it should return NULL)

def run(signature, n):

#The input signature will always contain 3 integers. (no validation needed)

#The input n will always be a non-negative integer. (no validation needed)

#If n is 0, the function should return an empty array

if n==0:

return[]

aux = signature

#control var because signature will always contain 3 integers

i=3

while i<n:

value = aux[-1]+ aux[-2] + aux[-3]

aux.append(value)

i+=1

result = aux

return result

run([1, 1, 1], 5)

run([0, 0, 1], 5)

----------------------------------------

10- Accumulate Characters

Write a function that takes a string as input and returns a new string where each character is repeated and separated by a hyphen (-). The number of repetitions of a character should be equal to its position in the input string, with the first character being repeated once. The characters in the output string should be capitalized.

Examples:

accum("abcd") returns "A-Bb-Ccc-Dddd"

accum("RqaEzty") returns "R-Qq-Aaa-Eeee-Zzzzz-Tttttt-Yyyyyyy"

accum("cwAt") returns "C-Ww-Aaa-Tttt"

Notes:

The input string will always contain only lowercase letters.

#takes a string as input

def run(input):

standardformat=input.lower()

#each character is repeated

aux = {}

for itInput in list(standardformat):

if itInput not in aux:

aux[itInput]=1

else:

aux[itInput] = aux[itInput]+1

#The number of repetitions of a character should be equal to its position in the input string

result = ''

index = 0

for itInput in list(standardformat):

#The characters in the output string should be capitalized.

result+=itInput.upper()

i=1

while i <= index and index > 0:

result+=itInput

i+=1

#separated by a hyphen (-).

result+='-'

index+=1

result=result[:-1]

return result

run('abcd')

run('RqaEzty')

run('cwAt')

----------------------------------------

11- A Pangram

Write a function that takes a string as input and returns a boolean indicating whether or not the string is a pangram. A pangram is a sentence that contains every single letter of the alphabet at least once. The function should ignore numbers, punctuation, and the case of the letters.

Example:

isPangram("The quick brown fox jumps over the lazy dog") returns true

isPangram("The quick brown fox jumps over the lazy cat") returns false

Notes:

The input string will only contain letters from a-z and A-Z.

#that takes a string as input

#The input string will only contain letters from a-z and A-Z.

def run(input):

englishAlphabet = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j', 'k', 'l', 'm', 'n', 'o', 'p', 'q', 'r', 's', 't', 'u', 'v', 'w', 'x', 'y', 'z']

standardformat=input.lower()

count=0

aux=list(input)

#returns a boolean (i return string bool to encapsulate on test structure)

result=True

for x in englishAlphabet:

try:

z = aux.index(x)

#The function should ignore numbers, punctuation, and the case of the letters.

except:

#returns a boolean (i return string bool to encapsulate on test structure)

return False

return result

run('The quick brown fox jumps over the lazy dog')

run('The quick brown fox jumps over the lazy cat')