**Soru 76**

Exercise-76  
There is an array with some numbers. All numbers are equal except for one. Try to find it!

find\_uniq([ 1, 1, 1, 2, 1, 1 ]) == 2

find\_uniq([ 0, 0, 0.55, 0, 0 ]) == 0.55

It’s guaranteed that array contains at least 3 numbers.  
The tests contain some very huge arrays, so think about performance.  
This is the first kata in series:

1. Find the unique number (this kata)
2. [Find the unique string](https://www.codewars.com/kata/585d8c8a28bc7403ea0000c3)
3. [Find The Unique](https://www.codewars.com/kata/5862e0db4f7ab47bed0000e5)

find\_uniq ([ 1, 1, 1, 2, 1, 1 ]) # uniq one == 2

find\_uniq2 ([ 0, 0, 0.55, 0, 0 ]) # uniq one == 0.55

def find\_uniq(liste):

    for i in liste:

        if liste.count(i) == 1:

            return i

print(find\_uniq([ 1, 1, 1, 2, 1, 1 ]))

2

**Exercise -75**You are given an m x n integer grid accounts where accounts[i][j] is the amount of money the i​​​​​​​​​​​th customer has in the j​​​​​​​​​​​th bank. Return*the****wealth****that the richest customer has.*  
A customer's **wealth** is the amount of money they have in all their bank accounts. The richest customer is the customer that has the maximum **wealth**.

**Example 1:**

Input: accounts = [[1,2,3],[3,2,1]]

Output: 6

Explanation:

1st customer has wealth = 1 + 2 + 3 = 6

2nd customer has wealth = 3 + 2 + 1 = 6

Both customers are considered the richest with a wealth of 6 each, so return 6.

**Example 2:**

Input: accounts = [[1,5],[7,3],[3,5]]

Output: 10

Explanation:

1st customer has wealth = 6

2nd customer has wealth = 10

3rd customer has wealth = 8

The 2nd customer is the richest with a wealth of 10.

Solution:

accounts = [[1,2,3],[3,2,1]]

richest = []

for i in accounts :

  richest += [sum(i)]

print((max(richest)))

**Exercise -74**

We define a harmonious array as an array where the difference between its maximum value and its minimum value is **exactly** 1.  
Given an integer array nums, return *the length of its longest harmonious subsequence among all its possible subsequences*.  
A **subsequence** of array is a sequence that can be derived from the array by deleting some or no elements without changing the order of the remaining elements.

**Example 1:**

Input: nums = [1,3,2,2,5,2,3,7]

Output: 5

Explanation: The longest harmonious subsequence is [3,2,2,2,3].

**Example 2:**

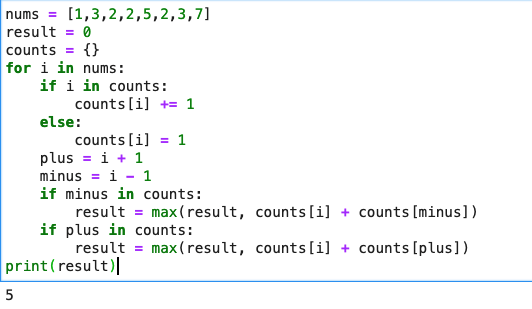
Input: nums = [1,2,3,4]

Output: 2

**Example 3:**

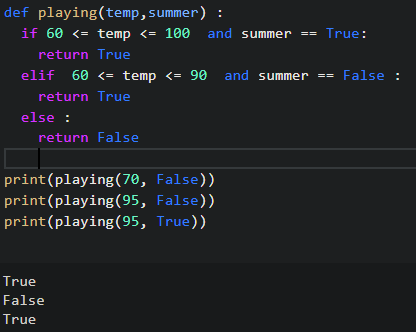
Input: nums = [1,1,1,1]

Output: 0



**Exercise-73**

The squirrels in Palo Alto spend most of the day playing. In particular, they play if the temperature is between 60 and 90 (inclusive). Unless it is summer, then the upper limit is 100 instead of 90. Given an int temperature and a boolean is\_summer, return True if the squirrels play and False otherwise.(70, False) → True  
(95, False) → False  
(95, True) → True



**Exercise-72**

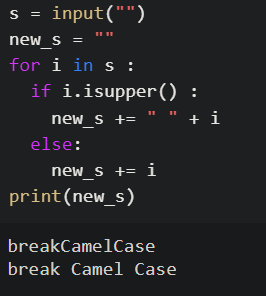
Complete the solution so that the function will break up camel casing, using a space between words.  
Example

"camelCasing" => "camel Casing"

"identifier" => "identifier"

"" => ""

"breakCamelCase" => "break Camel Case"



Exercise - 71  
Alice has n candies, where the ith candy is of type candyType[i]. Alice noticed that she started to gain weight, so she visited a doctor.  
The doctor advised Alice to only eat n / 2 of the candies she has (n is always even). Alice likes her candies very much, and she wants to eat the maximum number of different types of candies while still following the doctor's advice.  
Given the integer array candyType of length n, return *the****maximum****number of different types of candies she can eat if she only eats*n / 2*of them*.**Example 1:**

Input: candyType = [1,1,2,2,3,3]

Output: 3

Explanation: Alice can only eat 6 / 2 = 3 candies. Since there are only 3 types, she can eat one of each type.

**Example 2:**

Input: candyType = [1,1,2,3]

Output: 2

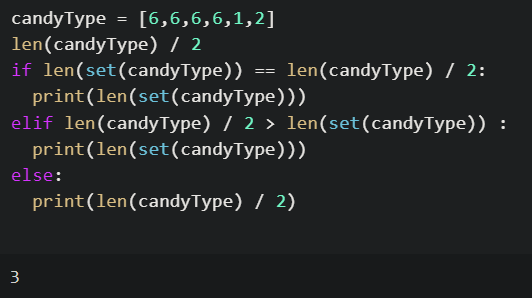
Explanation: Alice can only eat 4 / 2 = 2 candies. Whether she eats types [1,2], [1,3], or [2,3], she still can only eat 2 different types.

**Example 3:**

Input: candyType = [6,6,6,6]

Output: 1

Explanation: Alice can only eat 4 / 2 = 2 candies. Even though she can eat 2 candies, she only has 1 type.



**Exercise -70**

Given a string s, *find the first non-repeating character in it and return its index*. If it does not exist, return -1.**Example 1:**

Input: s = "leetcode"

Output: 0

**Example 2:**

Input: s = "loveleetcode"

Output: 2

**Example 3:**

Input: s = "aabb"

Output: -1

Sol:

s = input("")

for i in s :

  if s.count(i) == 1 :

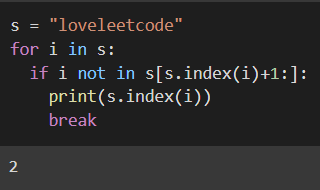
    print(s.index(i))

    break

else :

  print(-1)

veya



**Exercise – 69**

You are keeping score for a baseball game with strange rules. The game consists of several rounds, where the scores of past rounds may affect future rounds' scores.  
At the beginning of the game, you start with an empty record. You are given a list of strings ops, where ops[i] is the i th operation you must apply to the record and is one of the following:

1. An integer x - Record a new score of x.
2. "+" - Record a new score that is the sum of the previous two scores. It is guaranteed there will always be two previous scores.
3. "D" - Record a new score that is double the previous score. It is guaranteed there will always be a previous score.
4. "C" - Invalidate the previous score, removing it from the record. It is guaranteed there will always be a previous score.

Return *the sum of all the scores on the record*.  
**Example :**

Input: ops = ["5","-2","4","C","D","9","+","+"]

Output: 27

Explanation:

"5" - Add 5 to the record, record is now [5].

"-2" - Add -2 to the record, record is now [5, -2].

"4" - Add 4 to the record, record is now [5, -2, 4].

"C" - Invalidate and remove the previous score, record is now [5, -2].

"D" - Add 2 \* -2 = -4 to the record, record is now [5, -2, -4].

"9" - Add 9 to the record, record is now [5, -2, -4, 9].

"+" - Add -4 + 9 = 5 to the record, record is now [5, -2, -4, 9, 5].

"+" - Add 9 + 5 = 14 to the record, record is now [5, -2, -4, 9, 5, 14].

The total sum is 5 + -2 + -4 + 9 + 5 + 14 = 27.

Sol:

ops = ["5","-2","4","C","D","9","+","+"]

total = []

for i in range(len(ops)):

  if ops[i]=='C':

    total.pop()

  elif ops[i]=='D':

    total.append(2\*total[-1])

  elif ops[i]=='+':

    total.append(total[-1]+total[-2])

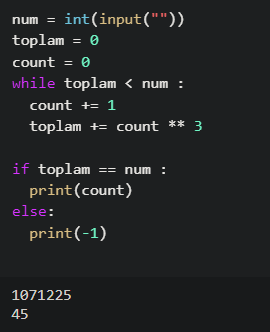
  else:

    total.append(int(ops[i]))

print(sum(total))

**Exercise-68**

Your task is to construct a building which will be a pile of n cubes. The cube at the bottom will have a volume of n^3, the cube above will have volume of (n-1)^3 and so on until the top which will have a volume of 1^3.  
You are given the total volume m of the building. Being given m can you find the number n of cubes you will have to build?  
The parameter of the function findNb (find\_nb, find-nb, findNb, ...) will be an integer m and you have to return the integer n such as n^3 + (n-1)^3 + ... + 1^3 = m if such a n exists or -1 if there is no such n.  
Examples:  
findNb(1071225) --> 45  
findNb(91716553919377) --> -1



**Exercise – 67**

:exclamation:Don't use any module or library for question solution  
Given an integer n, return T*rue if it is a power of two. Otherwise, return False*.  
An integer n is a power of two, if there exists an integer x such that n == 2x.**Example 1:**

Input: n = 1

Output: true

Explanation: 2\*\*0 = 1

**Example 2:**

Input: n = 16

Output: True

Explanation: 2\*\*4 = 16

**Example 3:**

Input: n = 3

Output: False

**Example 4:**

Input: n = 4

Output: True

**Example 5:**

Input: n = 5

Output: False

Sol:

n = int(input(""))

n\_list = []

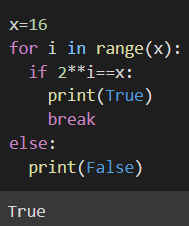
for i in range(2,n+1) :

  if n % i == 0 :

    n\_list += [i]

all([i % 2 == 0 for i in n\_list])

veya:



**Exercise-66**

A pangram is a sentence that contains every single letter of the alphabet at least once. For example, the sentence "The quick brown fox jumps over the lazy dog" is a pangram, because it uses the letters A-Z at least once (case is irrelevant).Given a string, detect whether or not it is a pangram. Return True if it is, False if not. Ignore numbers and punctuation.

Sol:

cumle = input("").lower()

yeni\_cumle = ""

alpha = "abcdefghijklmnopqrstuvwxyz"

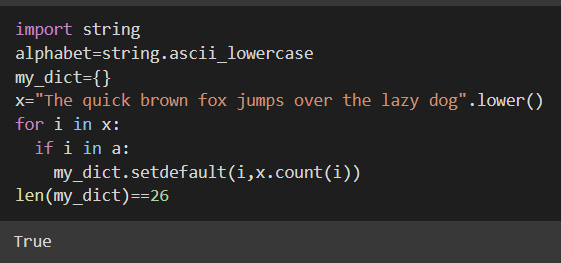
for i in cumle :

  if i.isalpha() :

    yeni\_cumle += i

print(set(alpha) - set(yeni\_cumle) == set())

veya:



**Exercise-65**

Write a function that takes in a string of one or more words, and returns the same string, but with all five or more letter words reversed .

* Strings passed in will consist of only letters and spaces.
* Spaces will be included only when more than one word is present.

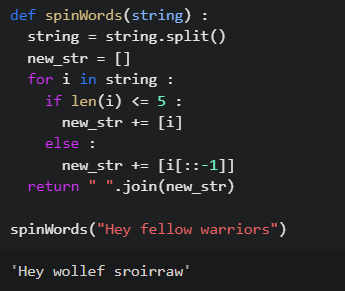
Examples:

spinWords("Hey fellow warriors") => "Hey wollef sroirraw"

spinWords("This is a test") => "This is a test"

spinWords("This is another test") => "This is rehtona test"

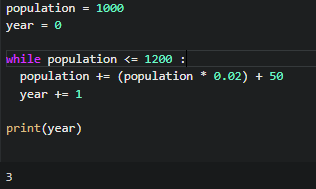
sol:



**Exercise-64**

 In a small town the population is p0 = 1000 at the beginning of a year. The population regularly increases by 2 percent per year and moreover 50 new inhabitants per year come to live in the town. How many years does the town need to see its population greater or equal to p = 1200 inhabitants?

Sol:



**Exercise – 63**

Lets make a username and password to login an account.  
Firstly define the username and password then take inputs for this two argument from user.  
Design python program 3 possibilities that first one is True user name and password,   
second is wrong username input and third is wrong password input. Then finish with try again output

Sol:

name = input("Lütfen isminizi giriniz :")

password = input("Lütfen şifrenizi giriniz :")

while True :

  if name == "Melek" and password == "1234" :

    print("Başarılı bir şekilde giriş yaptınız!")

    break

  elif name != "Melek" and password == "1234" :

    print("Lütfen kullanıcı isminizi kontrol ediniz!")

    name = input("Lütfen isminizi giriniz :")

    password = input("Lütfen şifrenizi giriniz :")

  elif name != "Melek" and password != "1234" :

    print("Lütfen kullanıcı isminizi ve şifrenizi kontrol ediniz!")

    name = input("Lütfen isminizi giriniz :")

    password = input("Lütfen şifrenizi giriniz :")

**Exercise – 62**

A **square triple** (a,b,c) is a triple where a, b, and c are **integers** and a2 + b2 = c2.  
Given an integer n, return *the number of****square triples****such that*1 <= a, b, c <= n.**Example 1:**

Input: n = 5

Output: 2

Explanation: The square triples are (3,4,5) and (4,3,5).

**Example 2:**

Input: n = 10

Output: 4

Explanation: The square triples are (3,4,5), (4,3,5), (6,8,10), and (8,6,10).

Sol:

n = int(input(""))

counter = 0

for i in range(1,n+1) :

  for j in range(1,n+1) :

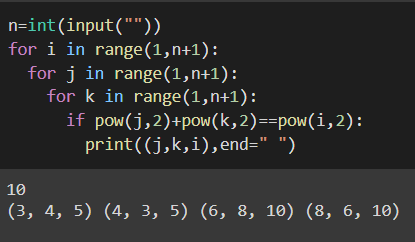
    for k in range(1,n+1) :

      if i \*\* 2 + j \*\* 2 == k \*\* 2 :

        counter += 1

print(counter)

veya:



**Exercise -61**

Given a string s, return True*if*s*is a****good****string, or F*alse*otherwise*.  
A string s is **good** if **all** the characters that appear in s have the **same** number of occurrences (i.e., the same frequency).**Example 1:**

Input: s = "abacbc"

Output: True

Explanation: The characters that appear in s are 'a', 'b', and 'c'. All characters occur 2 times in s.

**Example 2:**

Input: s = "aaabb"

Output: False

Explanation: The characters that appear in s are 'a' and 'b'.

'a' occurs 3 times while 'b' occurs 2 times, which is not the same number of times.

Sol:

s = input("")

s\_dict = {}

for i in s :

  if i in s\_dict :

    s\_dict[i] +=1

  else :

    s\_dict[i] = 1

print(len(set(s\_dict.values())) == 1)

veya:

