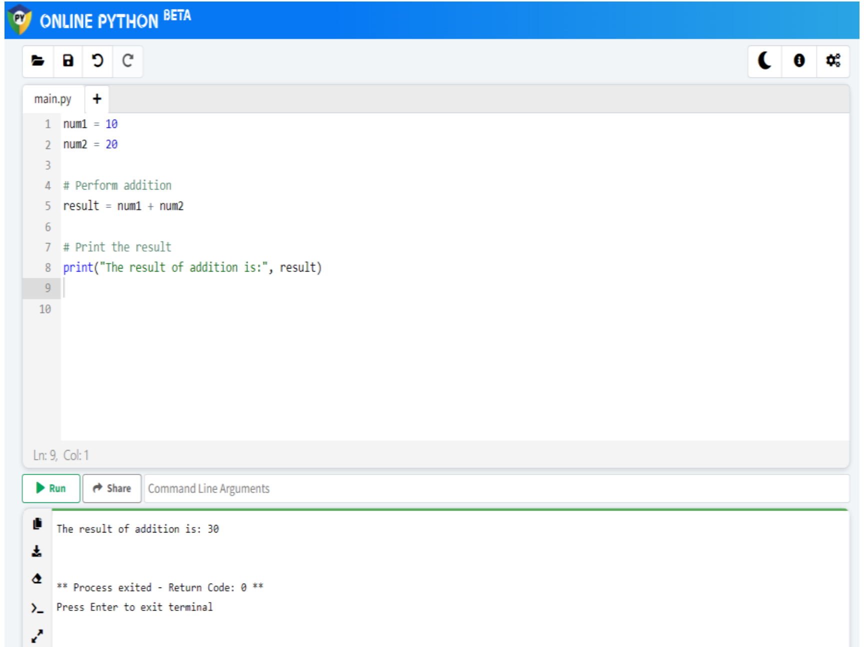
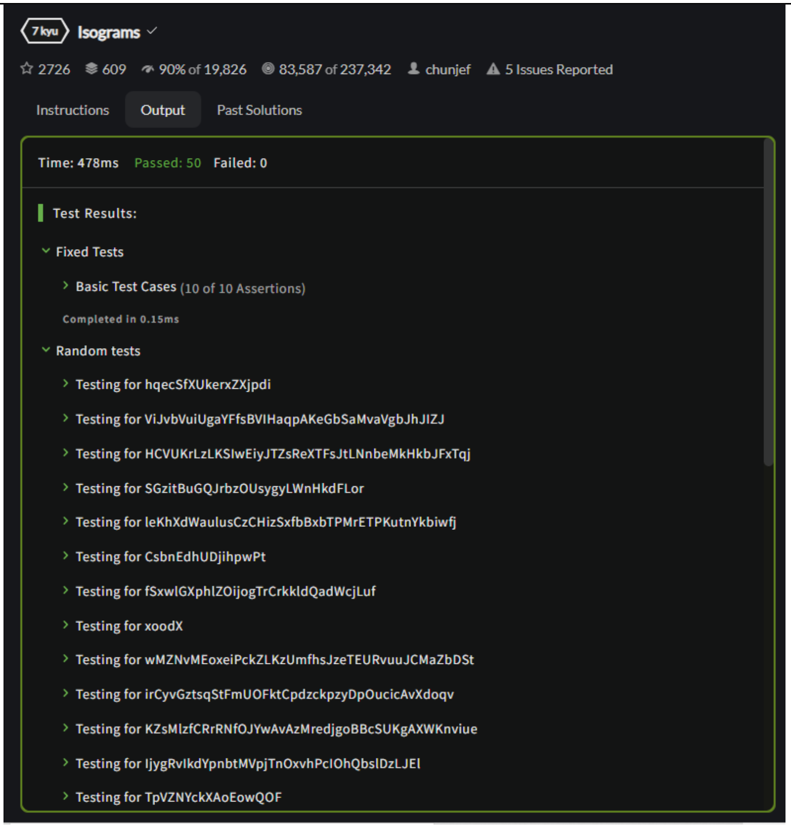
Koda ģenerācija – ChatGPT 3.5

Uzdevuma instrukcija (angļu valodā)	Iegūtais kods	Koda rezultāti

Please generate Python code that defines two integer variables and then does a simple addition operation.	num1 = 10
	num2 = 20
	
	# Perform addition
	result = num1 + num2
	
	# Print the result
	print("The result of addition is:", result)
		Šis uzdevums nebija ‘codewars’ vietnē, tāpēc šo kodu var manuāli pārbaudīt.
		
		 ONLINE PYTHON BETA
		main.py
		Ln: 9, Col: 1
		Run
		Share
		Command Line Arguments
		`num1 = 10`
		`num2 = 20`
		
		`# Perform addition`
		`result = num1 + num2`
		
		`# Print the result`
		`print("The result of addition is:", result)`
		
		`The result of addition is: 30`
		
		`** Process exited - Return Code: 0 **`
		`Press Enter to exit terminal`

An isogram is a word that has no repeating letters, consecutive or non-consecutive. Implement a function that determines whether a string that contains only letters is an isogram. Assume the empty string is an isogram. Ignore letter case.

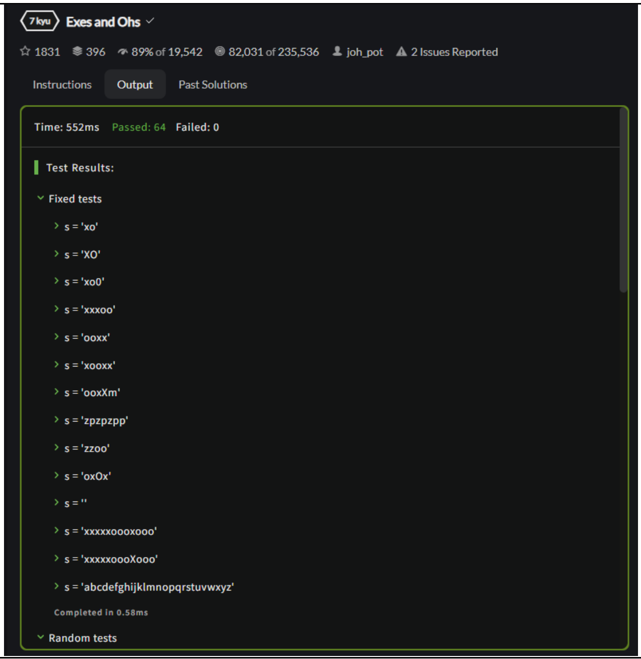
```python
def is_isogram(string):
    """
    Check if a string is an isogram.

    Args:
    string (str): The input string to check.

    Returns:
    bool: True if the string is an isogram, False otherwise.
    """
    # Convert the string to lowercase to ignore letter case
    string = string.lower()

    # Create a set to store unique letters
    seen = set()

    # Iterate through each character in the string
    for char in string:
        # Check if the character is already in the set
        if char in seen:
            return False
        # Add the character to the set
        seen.add(char)

    # If no repeating letters found, return True
    return True
```



```
Time: 478ms   Passed: 50   Failed: 0

Test Results:
  Fixed Tests
    > Basic Test Cases (10 of 10 Assertions)
    Completed in 0.15ms
  Random tests
    > Testing for hqecSfXUkerxZXjpdi
    > Testing for ViJvbVuiUgaYFfsBVIHaqpAKeGbSaMvaVgbJhJIZJ
    > Testing for HCVUKrLzLKSIwEiyJTZsReXTFsJtLNnbeMkHkbJFxTqj
    > Testing for SGzitBuGQJrbzOUsygyLWnHkdFLor
    > Testing for leKhXdWaulusCzCHizSxfbBxbTPMrETPKutnYkbiwfj
    > Testing for CsbnEdhUDjihpwPt
    > Testing for fSxwlGXphlZOijogTrCrkkldQadWcjLuf
    > Testing for xoodX
    > Testing for wMZNvMEoxeiPckZLKzUmfhsJzeTEURvuuJCMaZbDSt
    > Testing for irCyvGztsqStFmUOFktCpdzckpzyDpOucicAvXdoqv
    > Testing for KZsMlzfCRrRNfOJYwAvAzMredjgoBBcSUKgAXWKnviue
    > Testing for IjygRvlkdYpnbtMVpjTnOxvhPclOhQbslDzLJEl
    > Testing for TpVZNYckXAoEowQOF
```

Check to see if a string has the same amount of 'x's and 'o's. The method must return a boolean and be case insensitive. The string can contain any char.

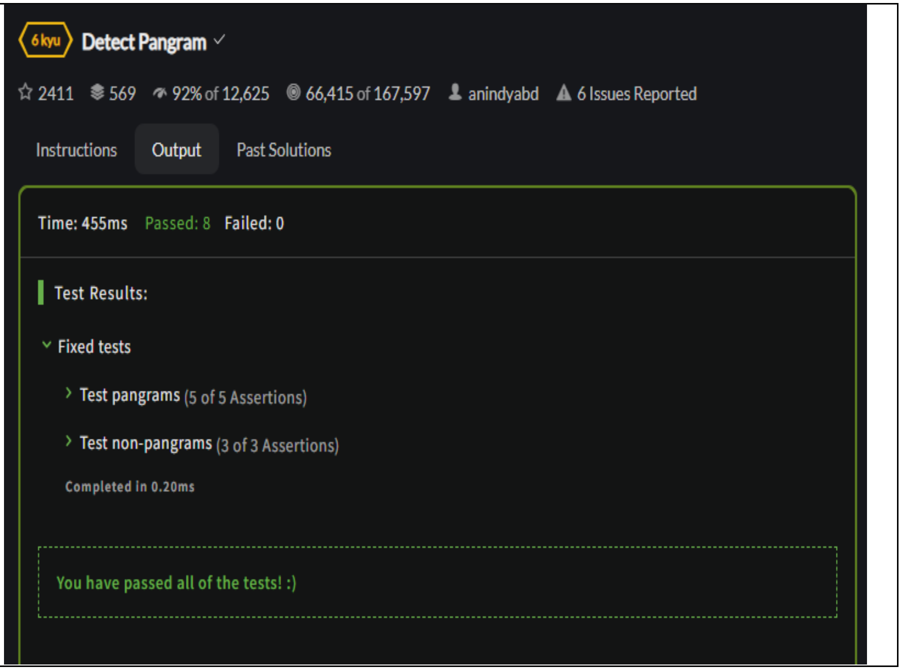
```python
def equal_x_and_o(string):
   # Count the occurrences of 'x' and 'o' in the string
   count_x = string.lower().count('x')
   count_o = string.lower().count('o')

   # Return True if the counts are equal, False otherwise
   return count_x == count_o

# Example usage:
print(equal_x_and_o("xooxxoX"))  # Output: True
print(equal_x_and_o("xooxX"))    # Output: False
```



```
Time: 552ms   Passed: 64   Failed: 0

Test Results:
  Fixed tests
    > s = 'xo'
    > s = 'XO'
    > s = 'xo0'
    > s = 'xxxoo'
    > s = 'ooxx'
    > s = 'xooxx'
    > s = 'ooxXm'
    > s = 'zpzpzpp'
    > s = 'zzoo'
    > s = 'oxOx'
    > s = ''
    > s = 'xxxxxoooxooo'
    > s = 'xxxxxoooXooo'
    > s = 'abcdefghijklmnopqrstuvwxyz'
    Completed in 0.58ms
  Random tests
```

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

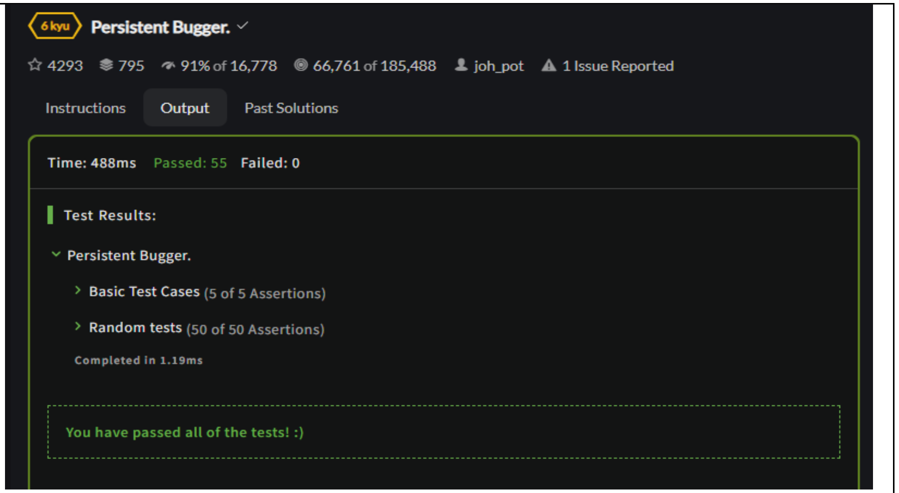
```
import string

def is_pangram(s):
    alphabet = set(string.ascii_lowercase) # Set of all lowercase letters

    # Iterate through each character in the string
    for char in s.lower():
        if char.isalpha():  # Check if the character is a letter
            alphabet.discard(char)  # Remove the letter from the set if found

    # If the set is empty, all letters have been found
    return not alphabet
```



```
Time: 455ms   Passed: 8   Failed: 0

Test Results:
  Fixed tests
    Test pangrams (5 of 5 Assertions)
    Test non-pangrams (3 of 3 Assertions)
    Completed in 0.20ms

You have passed all of the tests! :)
```

Write a function, persistence, that takes in a positive parameter num and returns its multiplicative persistence, which is the number of times you must multiply the digits in num until you reach a single digit.

```
def persistence(num):
    # Convert the number to a string to access its digits
    num_str = str(num)
    # Initialize persistence count
    persistence_count = 0

    # Keep looping until the number becomes a single digit
    while len(num_str) > 1:
        # Multiply the digits
        result = 1
        for digit in num_str:
            result *= int(digit)
        # Update the number and increment the persistence count
        num_str = str(result)
        persistence_count += 1

    return persistence_count
```



Time: 488ms Passed: 55 Failed: 0

Test Results:

Persistent Bugger.

Basic Test Cases (5 of 5 Assertions)

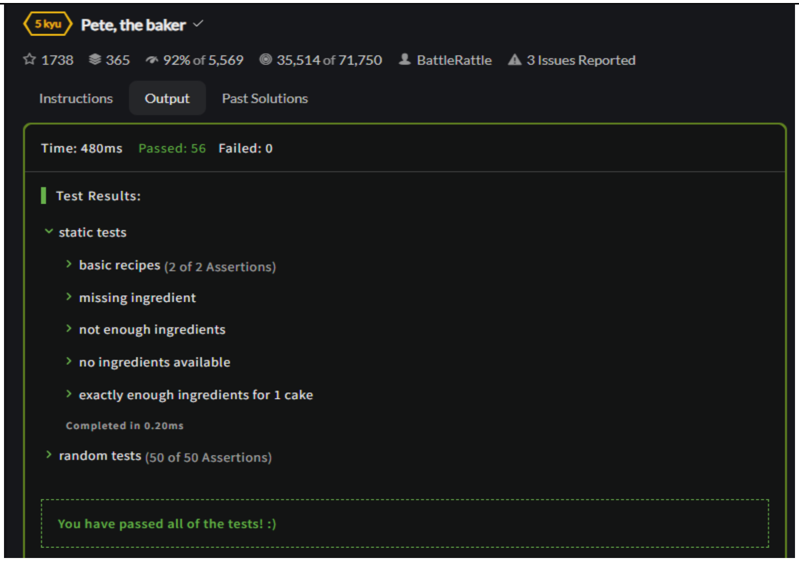
Random tests (50 of 50 Assertions)

Completed in 1.19ms

You have passed all of the tests! :)

Write a function cakes(), which takes the recipe (object) and the available ingredients (also an object) and returns the maximum number of cakes Pete can bake (integer). For simplicity there are no units for the amounts (e.g. 1 lb of flour or 200 g of sugar are simply 1 or 200). Ingredients that are not present in the objects, can be considered as 0.

```
def cakes(recipe, available):
    # Initialize a list to store the ratios of available ingredients to required ingredients
    ratios = []

    # Iterate through each ingredient in the recipe
    for ingredient, amount in recipe.items():
        # If the ingredient is not available, return 0 cakes
        if ingredient not in available:
            return 0
        # Calculate the ratio of available ingredient to required ingredient
        ratio = available[ingredient] // amount
        # Add the ratio to the list
        ratios.append(ratio)

    # Return the minimum ratio, which represents the maximum number of cakes that can be made
    return min(ratios)
```



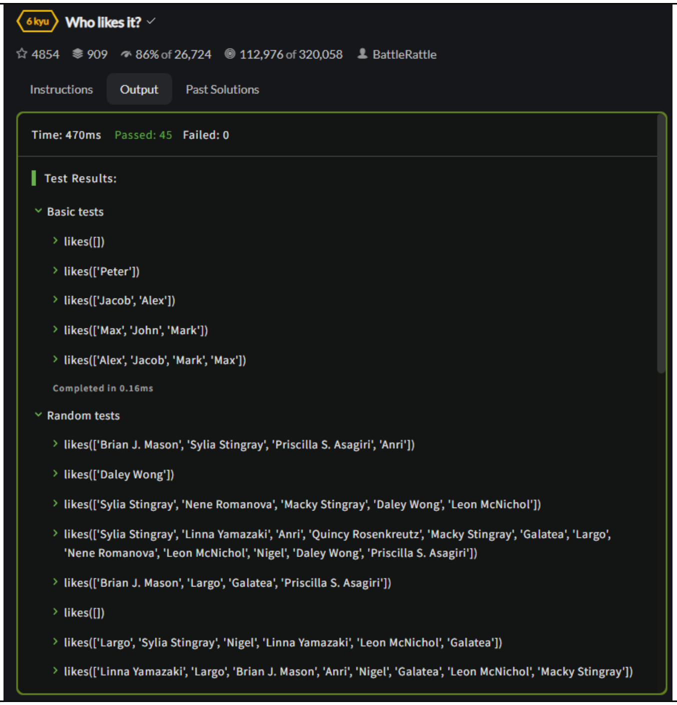
Pete, the baker ✓

☆ 1738 365 92% of 5,569 35,514 of 71,750 BattleRattle 3 Issues Reported

```
Time: 480ms   Passed: 56   Failed: 0

Test Results:
 static tests
   basic recipes (2 of 2 Assertions)
   missing ingredient
   not enough ingredients
   no ingredients available
   exactly enough ingredients for 1 cake
   Completed in 0.20ms
 random tests (50 of 50 Assertions)

You have passed all of the tests! :)
```

You probably know the "like" system from Facebook and other pages. People can "like" blog posts, pictures or other items. We want to create the text that should be displayed next to such an item.
Implement the function which takes an array containing the names of people that like an item. It must return the display text as shown in the examples

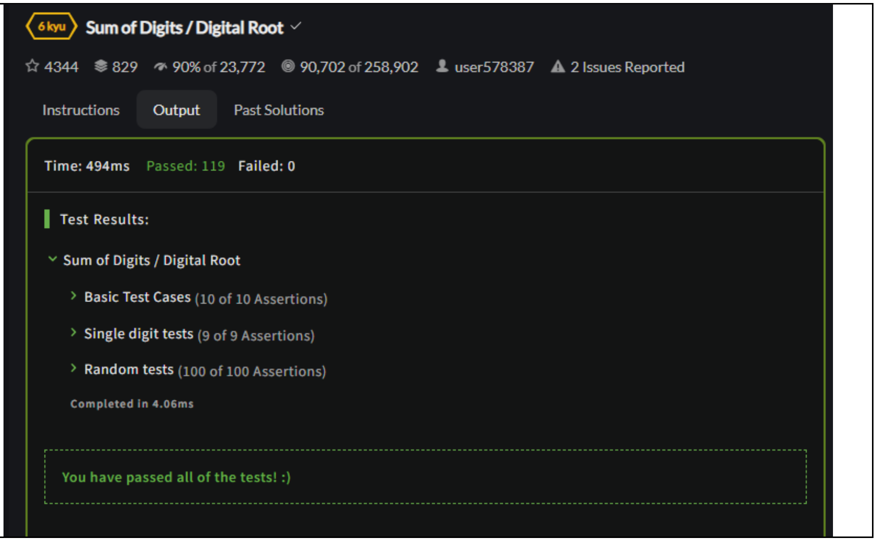
```
def likes(names):

    if len(names) == 0:
        return "no one likes this"
    elif len(names) == 1:
        return f"{names[0]} likes this"
    elif len(names) == 2:
        return f"{names[0]} and {names[1]} like this"
    elif len(names) == 3:
        return f"{names[0]}, {names[1]} and {names[2]} like this"
    else:
        return f"{names[0]}, {names[1]} and {len(names) - 2} others like this"
```



```
Time: 470ms   Passed: 45   Failed: 0

Test Results:
Basic tests
  likes([])
  likes(['Peter'])
  likes(['Jacob', 'Alex'])
  likes(['Max', 'John', 'Mark'])
  likes(['Alex', 'Jacob', 'Mark', 'Max'])
  Completed in 0.16ms
Random tests
  likes(['Brian J. Mason', 'Sylia Stingray', 'Priscilla S. Asagiri', 'Anri'])
  likes(['Daley Wong'])
  likes(['Sylia Stingray', 'Nene Romanova', 'Macky Stingray', 'Daley Wong', 'Leon McNichol'])
  likes(['Sylia Stingray', 'Linna Yamazaki', 'Anri', 'Quincy Rosenkreutz', 'Macky Stingray', 'Galatea', 'Largo',
  'Nene Romanova', 'Leon McNichol', 'Nigel', 'Daley Wong', 'Priscilla S. Asagiri'])
  likes(['Brian J. Mason', 'Largo', 'Galatea', 'Priscilla S. Asagiri'])
  likes([])
  likes(['Largo', 'Sylia Stingray', 'Nigel', 'Linna Yamazaki', 'Leon McNichol', 'Galatea'])
  likes(['Linna Yamazaki', 'Largo', 'Brian J. Mason', 'Anri', 'Nigel', 'Galatea', 'Leon McNichol', 'Macky Stingray'])
```

Digital root is the recursive sum of all the digits in a number.
Given n, take the sum of the digits of n. If that value has more than one digit, continue reducing in this way until a single-digit number is produced. The input will be a non-negative integer.

```python
def digital_root(n):
    # Keep looping until n is a single-digit number
    while n >= 10:
        # Convert n to a string to access its digits
        n_str = str(n)
        # Calculate the sum of digits
        digit_sum = sum(int(digit) for digit in n_str)
        # Update n with the sum of digits
        n = digit_sum
    return n
```



Sum of Digits / Digital Root ✓

☆ 4344 829 90% of 23,772 90,702 of 258,902 user578387 ⚠ 2 Issues Reported

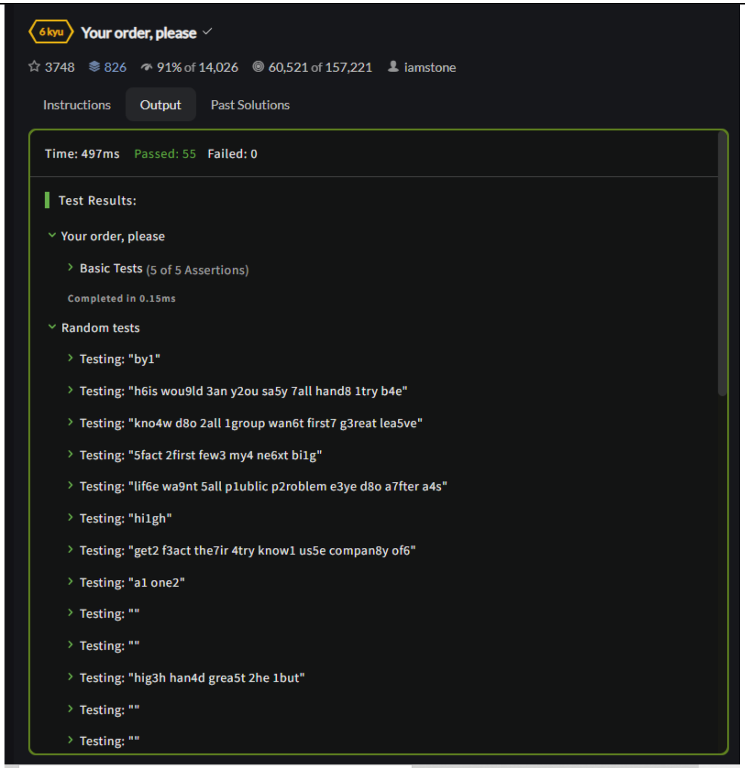
```
Time: 494ms   Passed: 119   Failed: 0

Test Results:
 Sum of Digits / Digital Root
   Basic Test Cases (10 of 10 Assertions)
   Single digit tests (9 of 9 Assertions)
   Random tests (100 of 100 Assertions)
   Completed in 4.06ms

You have passed all of the tests! :)
```

Your task is to sort a given string. Each word in the string will contain a single number. This number is the position the word should have in the result.

Note: Numbers can be from 1 to 9. So 1 will be the first word (not 0).

If the input string is empty, return an empty string. The words in the input String will only contain valid consecutive numbers.

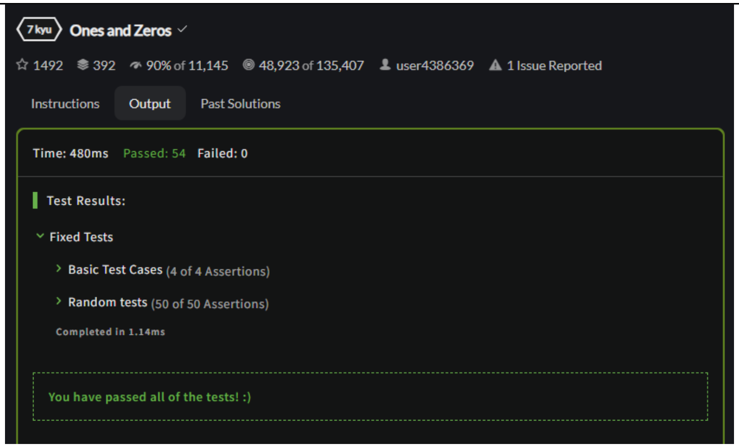
```python
def order(sentence):
    if not sentence:
        return ""

    # Split the sentence into words
    words = sentence.split()

    # Sort the words based on the number within each word
    sorted_words = sorted(words, key=lambda x: int(''.join(filter(str.isdigit, x))))

    # Join the sorted words into a single string
    sorted_sentence = ' '.join(sorted_words)

    return sorted_sentence
```



```
Time: 497ms   Passed: 55   Failed: 0

Test Results:
  Your order, please
    Basic Tests (5 of 5 Assertions)
    Completed in 0.15ms
  Random tests
    Testing: "by1"
    Testing: "h6is wou9ld 3an y2ou sa5y 7all hand8 1try b4e"
    Testing: "kno4w d8o 2all 1group wan6t first7 g3reat lea5ve"
    Testing: "5fact 2first few3 my4 ne6xt bi1g"
    Testing: "lif6e wa9nt 5all p1ublic p2roblem e3ye d8o a7fter a4s"
    Testing: "hi1gh"
    Testing: "get2 f3act the7ir 4try know1 us5e compan8y of6"
    Testing: "a1 one2"
    Testing: ""
    Testing: ""
    Testing: "hig3h han4d grea5t 2he 1but"
    Testing: ""
    Testing: ""
```

Given an array of ones and zeroes, convert the equivalent binary value to an integer. Eg: [0, 0, 0, 1] is treated as 0001 which is the binary representation of 1.

```
def binary_array_to_number(arr):
    # Initialize the result as 0
    result = 0

    # Iterate through the array in reverse order
    for i in range(len(arr)):
        # Multiply the digit by its corresponding power of 2 and add it to the result
        result += arr[len(arr) - i - 1] * (2 ** i)

    return result
```

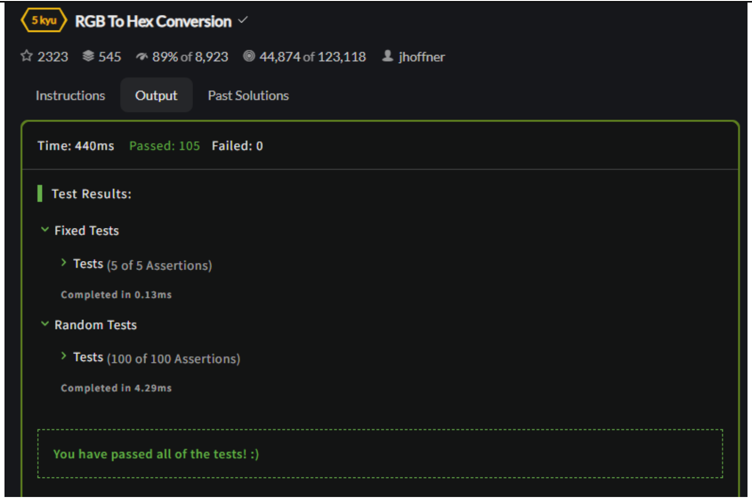


```
Time: 480ms   Passed: 54   Failed: 0

Test Results:
  Fixed Tests
    Basic Test Cases (4 of 4 Assertions)
    Random tests (50 of 50 Assertions)
    Completed in 1.14ms

You have passed all of the tests! :)
```

The rgb function is incomplete. Complete it so that passing in RGB decimal values will result in a hexadecimal representation being returned. Valid decimal values for RGB are 0 - 255. Any values that fall out of that range must be rounded to the closest valid value.

Note: Your answer should always be 6 characters long, the shorthand with 3 will not work here.

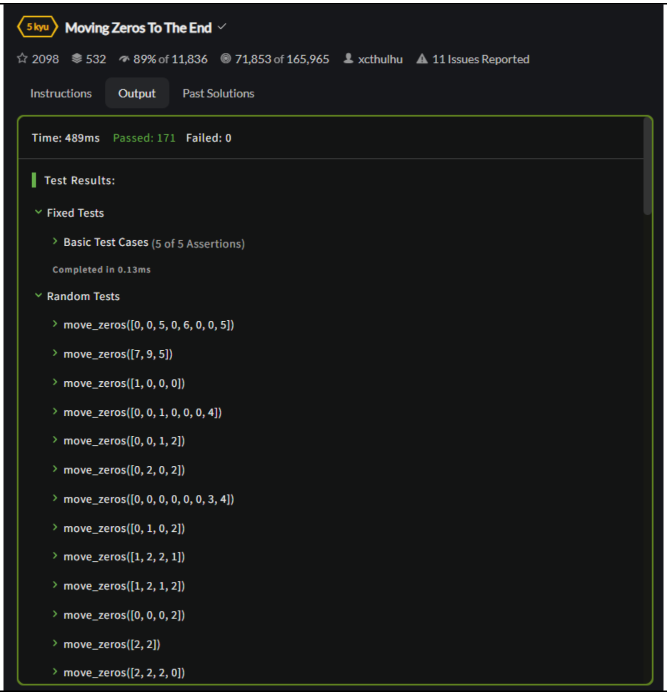
```
def rgb(r, g, b):
    # Function to clamp a value within the range [0, 255]
    def clamp(x):
        return max(0, min(x, 255))

    # Convert decimal values to hexadecimal representation and ensure each component is 2 characters long
    hex_r = format(clamp(r), '02X')
    hex_g = format(clamp(g), '02X')
    hex_b = format(clamp(b), '02X')

    # Concatenate the hexadecimal components
    return hex_r + hex_g + hex_b
```



```
Time: 440ms   Passed: 105   Failed: 0

Test Results:
  Fixed Tests
    Tests (5 of 5 Assertions)
    Completed in 0.13ms
  Random Tests
    Tests (100 of 100 Assertions)
    Completed in 4.29ms

You have passed all of the tests! :)
```

Write an algorithm that takes an array and moves all of the zeros to the end, preserving the order of the other elements.

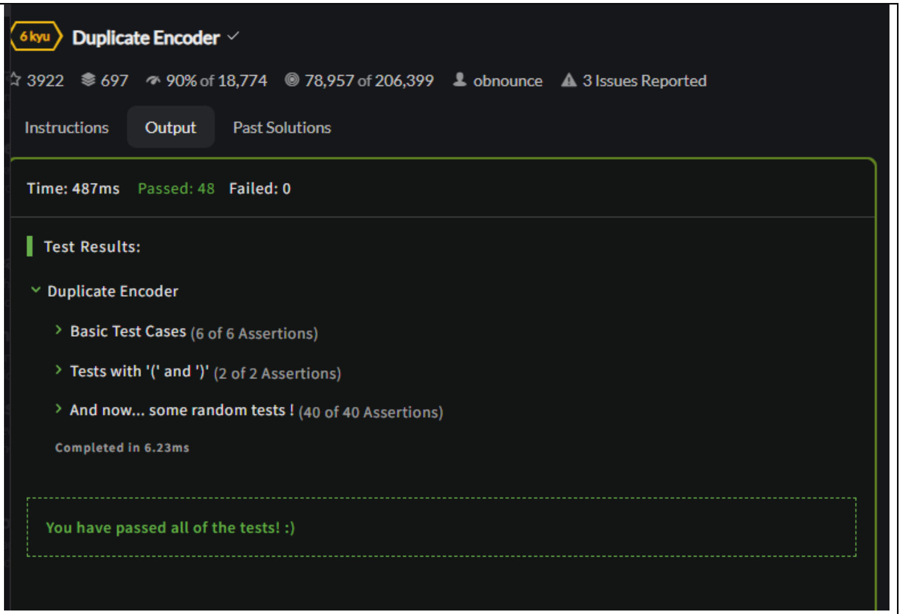
```python
def move_zeros(arr):
    # Initialize a variable to keep track of the count of non-zero elements
    non_zero_count = 0

    # Iterate through the array and move non-zero elements to the front
    for num in arr:
        if num != 0:
            arr[non_zero_count] = num
            non_zero_count += 1

    # Fill the remaining positions with zeros
    for i in range(non_zero_count, len(arr)):
        arr[i] = 0

    return arr
```



```
Time: 489ms   Passed: 171   Failed: 0

Test Results:
  Fixed Tests
    Basic Test Cases (5 of 5 Assertions)
    Completed in 0.13ms
  Random Tests
    move_zeros([0, 0, 5, 0, 6, 0, 0, 5])
    move_zeros([7, 9, 5])
    move_zeros([1, 0, 0, 0])
    move_zeros([0, 0, 1, 0, 0, 0, 4])
    move_zeros([0, 0, 1, 2])
    move_zeros([0, 2, 0, 2])
    move_zeros([0, 0, 0, 0, 0, 0, 3, 4])
    move_zeros([0, 1, 0, 2])
    move_zeros([1, 2, 2, 1])
    move_zeros([1, 2, 1, 2])
    move_zeros([0, 0, 0, 2])
    move_zeros([2, 2])
    move_zeros([2, 2, 2, 0])
```

The goal of this exercise is to convert a string to a new string where each character in the new string is "(" if that character appears only once in the original string, or ")" if that character appears more than once in the original string. Ignore capitalization when determining if a character is a duplicate.

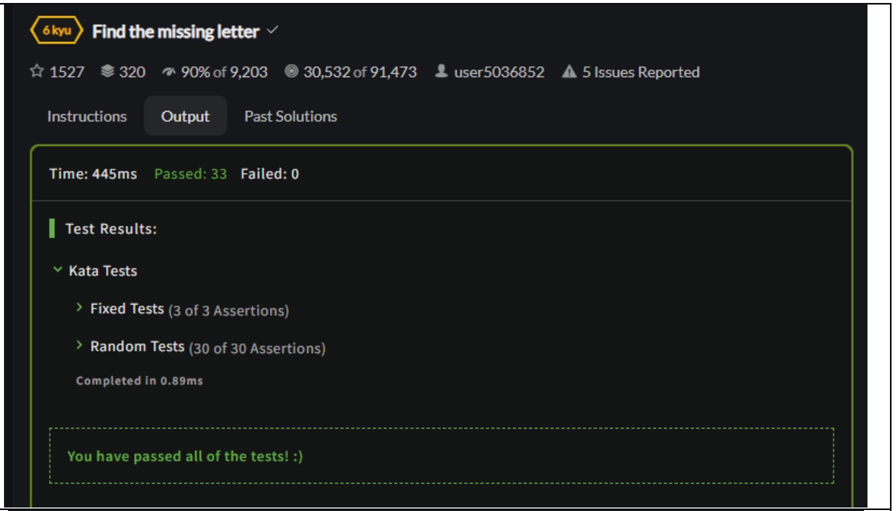
```
def duplicate_encode(word):
    # Convert the word to lowercase to ignore capitalization
    word = word.lower()

    # Initialize a dictionary to store the count of each character
    char_count = {}

    # Count the occurrences of each character in the word
    for char in word:
        char_count[char] = char_count.get(char, 0) + 1

    # Construct the new string based on the counts
    new_word = ''
    for char in word:
        if char_count[char] == 1:
            new_word += '('
        else:
            new_word += ')'

    return new_word
```



```
Time: 487ms   Passed: 48   Failed: 0

Test Results:
  Duplicate Encoder
    Basic Test Cases (6 of 6 Assertions)
    Tests with '(' and ')' (2 of 2 Assertions)
    And now... some random tests ! (40 of 40 Assertions)
    Completed in 6.23ms

You have passed all of the tests! :)
```

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

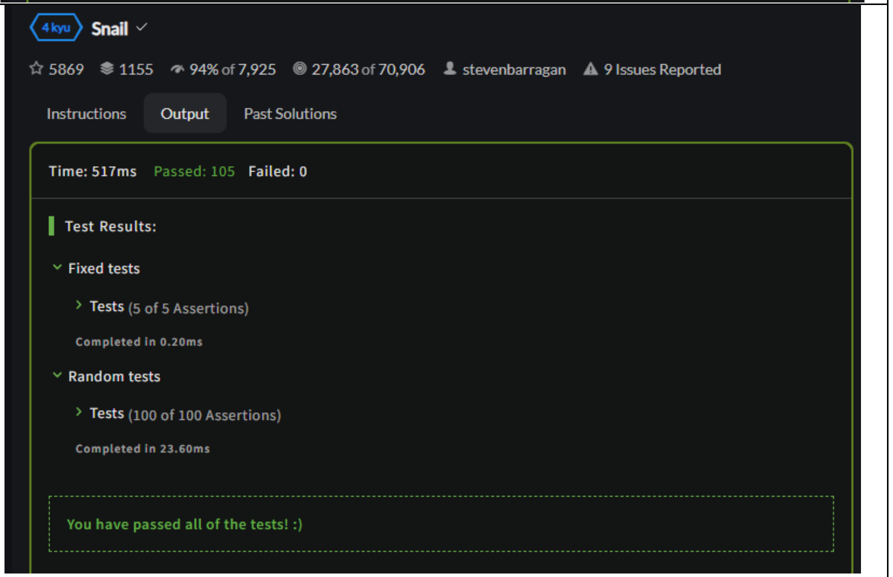
```
def find_missing_letter(chars):
    # Iterate through the array
    for i in range(len(chars) - 1):
        # Check if the ASCII value of the next character is not consecutive
        if ord(chars[i + 1]) - ord(chars[i]) != 1:
            # Return the missing letter
            return chr(ord(chars[i]) + 1)

# Example:
print(find_missing_letter(['a', 'b', 'c', 'd', 'f']))  # Output: 'e'
```



```
Time: 445ms   Passed: 33   Failed: 0

Test Results:
  Kata Tests
    Fixed Tests (3 of 3 Assertions)
    Random Tests (30 of 30 Assertions)
    Completed in 0.89ms

You have passed all of the tests! :)
```

Given an n x n array, return the array elements arranged from outermost elements to the middle element, traveling clockwise.

```
def spiral_order(matrix):
    result = []
    if not matrix:
        return result

    # Define initial boundaries
    top, bottom, left, right = 0, len(matrix) - 1, 0, len(matrix[0]) - 1

    while top <= bottom and left <= right:
        # Traverse top row
        for j in range(left, right + 1):
            result.append(matrix[top][j])
        top += 1

        # Traverse rightmost column
        for i in range(top, bottom + 1):
            result.append(matrix[i][right])
        right -= 1
```

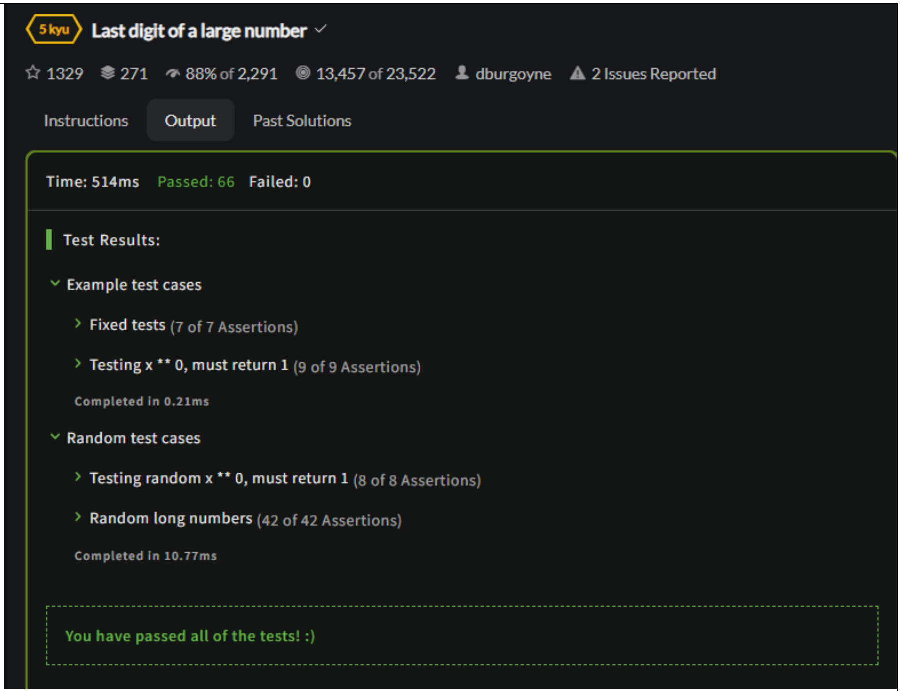


```
Time: 517ms   Passed: 105   Failed: 0

Test Results:
  Fixed tests
    Tests (5 of 5 Assertions)
    Completed in 0.20ms
  Random tests
    Tests (100 of 100 Assertions)
    Completed in 23.60ms

You have passed all of the tests! :)
```

```
        # Check if there is a bottom row
        if top <= bottom:
            # Traverse bottom row
            for j in range(right, left - 1, -1):

result.append(matrix[bottom][j])
            bottom -= 1

        # Check if there is a leftmost
column
        if left <= right:
            # Traverse leftmost column
            for i in range(bottom, top - 1, -1):
                result.append(matrix[i][left])
            left += 1

    return result
```

Define a function that takes in two non-negative integers �a and �b and returns the last decimal digit of ��ab. Note that �a and �b may be very large!
For example, the last decimal digit of 9797 is 99,
since 97=478296997=4782969. The last decimal digit of (2200)2300(2200)2300, which has over 10921092 decimal digits, is 66. Also, please take 0000 to be 11.
You may assume that the input will always be valid.

```
def last_digit(a, b):
    if b == 0:
        return 1

    # Map the last digit of a to the
possible last digits of a^b
    digit_to_power = {
        0: [0],
        1: [1],
        2: [2, 4, 8, 6],
        3: [3, 9, 7, 1],
        4: [4, 6],
        5: [5],
        6: [6],
        7: [7, 9, 3, 1],
        8: [8, 4, 2, 6],
        9: [9, 1]
    }

    # Find the last digit of a
    last_digit_a = a % 10

    # Find the possible last digits of a^b
    possible_last_digits =
digit_to_power[last_digit_a]

    # Find the position of b in the cycle of
possible last digits
    position_in_cycle = (b - 1) %
len(possible_last_digits)

    # Return the last digit of a^b
    return
possible_last_digits[position_in_cycle]

# Example usage:
print(last_digit(3, 3))   # Output: 7
```

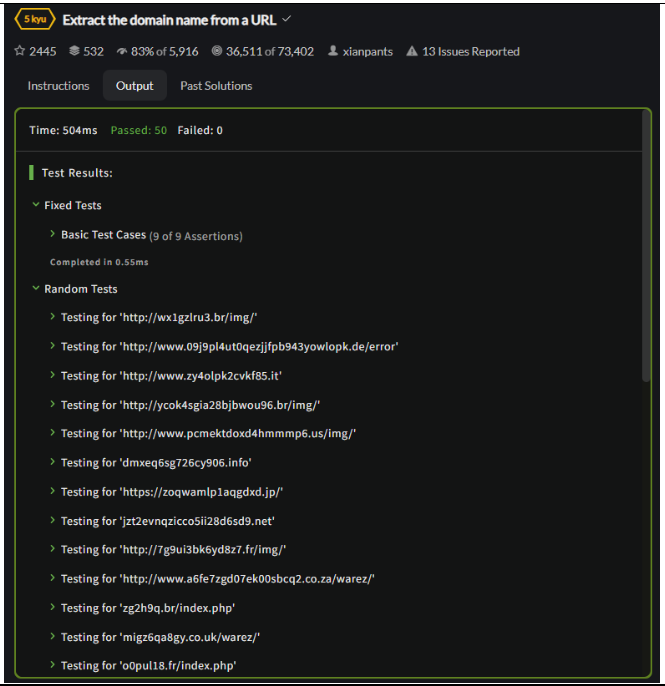


```
Time: 514ms   Passed: 66   Failed: 0

Test Results:
  Example test cases
    Fixed tests (7 of 7 Assertions)
    Testing x ** 0, must return 1 (9 of 9 Assertions)
    Completed in 0.21ms
  Random test cases
    Testing random x ** 0, must return 1 (8 of 8 Assertions)
    Random long numbers (42 of 42 Assertions)
    Completed in 10.77ms

You have passed all of the tests! :)
```

	print(last_digit(4, 2)) # Output: 6 print(last_digit(7, 8)) # Output: 1 print(last_digit(0, 0)) # Output: 1	

Write a function that when given a URL as a string, parses out just the domain name and returns it as a string.

```
import re

def domain_name(url):
   # Regular expression pattern to match the domain name
   pattern = r'(?:http[s]?://)?(?:www\.)?([a-zA-Z0-9-]+)\.'

   # Use regular expression to find the domain name in the URL
   match = re.search(pattern, url)

   # Return the domain name if found, otherwise return an empty string
   return match.group(1) if match else ''
```



Time: 504ms   Passed: 50   Failed: 0

Test Results:

Fixed Tests

Basic Test Cases (9 of 9 Assertions)

Completed in 0.55ms

Random Tests

Testing for 'http://wx1gzlru3.br/img/'

Testing for 'http://www.09j9pl4ut0qezjjfpb943yowlopk.de/error'

Testing for 'http://www.zy4olpk2cvkf85.it'

Testing for 'http://ycok4sgia28bjbwou96.br/img/'

Testing for 'http://www.pcmektdoxd4hmmmp6.us/img/'

Testing for 'dmxeq6sg726cy906.info'

Testing for 'https://zoqwamlp1aqgdxd.jp/'

Testing for 'jzt2evnqzicco5ii28d6sd9.net'

Testing for 'http://7g9ui3bk6yd8z7.fr/img/'

Testing for 'http://www.a6fe7zgd07ek00sbcq2.co.za/warez/'

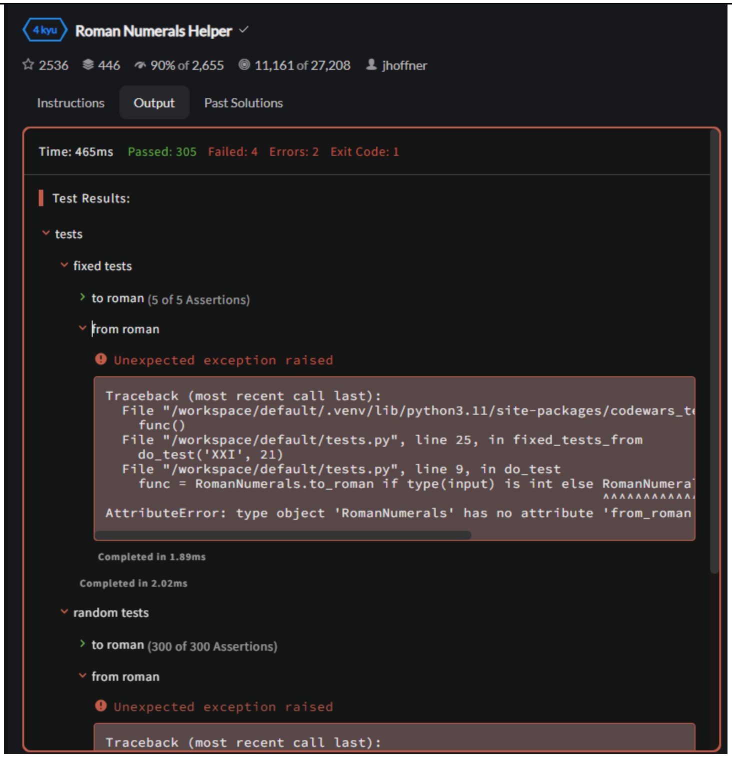
Testing for 'zg2h9q.br/index.php'

Testing for 'migz6qa8gy.co.uk/warez/'

Testing for 'o0pul18.fr/index.php'

Write two functions that convert a roman numeral to and from an integer value. Multiple roman numeral values will be tested for each function.
Modern Roman numerals are written by expressing each digit separately starting with the left most digit and skipping any digit with a value of zero. In Roman numerals:
1990 is
rendered: 1000=M, 900=CM, 90=XC; resulting in MCMXC
2008 is written as 2000=MM, 8=VIII; or MMVIII
1666 uses each Roman symbol in descending order: MDCLXVI.
Input range : 1 <= n < 4000
In this kata 4 should be represented as IV, NOT as IIII (the "watchmaker's four").

```
class RomanNumerals:
  @staticmethod
  def to_roman(val: int) -> str:
    # Define the mapping of integer
values to Roman numerals
    int_to_roman_map = {
      1000: 'M',
      900: 'CM',
      500: 'D',
      400: 'CD',
      100: 'C',
      90: 'XC',
      50: 'L',
      40: 'XL',
      10: 'X',
      9: 'IX',
      5: 'V',
      4: 'IV',
      1: 'I'
    }

    # Initialize the result string
    result = ''

    # Iterate through the integer values
in descending order
    for integer in
sorted(int_to_roman_map.keys(),
reverse=True):
      # Append the corresponding
Roman numeral to the result while
subtracting the integer value
      while val >= integer:
        result +=
int_to_roman_map[integer]
        val -= integer
```



```
Time: 465ms   Passed: 305   Failed: 4   Errors: 2   Exit Code: 1

Test Results:
  tests
    fixed tests
      to roman (5 of 5 Assertions)
      from roman
        Unexpected exception raised
        Traceback (most recent call last):
          File "/workspace/default/.venv/lib/python3.11/site-packages/codewars_te
            func()
          File "/workspace/default/tests.py", line 25, in fixed_tests_from
            do_test('XXI', 21)
          File "/workspace/default/tests.py", line 9, in do_test
            func = RomanNumerals.to_roman if type(input) is int else RomanNumera
                                                                     ^^^^^^^^^^^
        AttributeError: type object 'RomanNumerals' has no attribute 'from_roman
        Completed in 1.89ms
      Completed in 2.02ms
    random tests
      to roman (300 of 300 Assertions)
      from roman
        Unexpected exception raised
        Traceback (most recent call last):
```

```
    return result
```

In this kata we want to convert a string into an integer. The strings simply represent the numbers in words.

Examples:

"one" => 1

"twenty" => 20

"two hundred forty-six" => 246

"seven hundred eighty-three thousand nine hundred and nineteen" => 783919

Additional Notes:

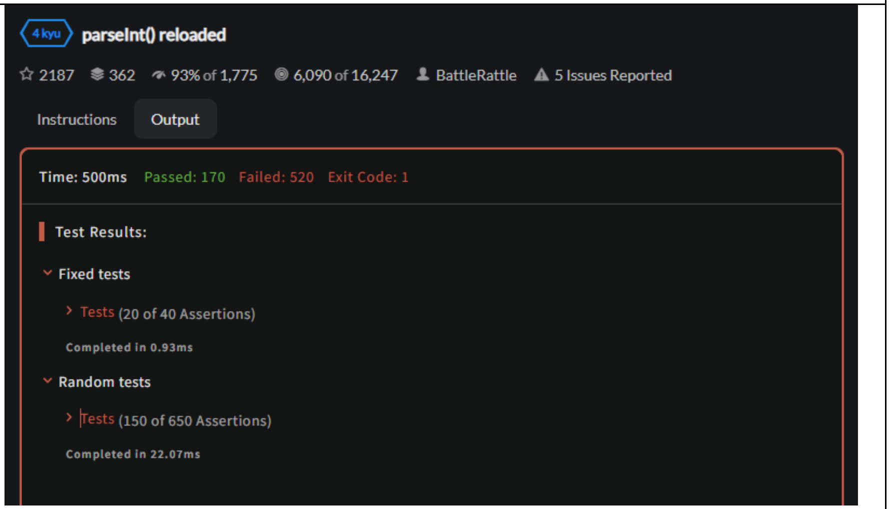
The minimum number is "zero" (inclusively)

The maximum number, which must be supported is 1 million (inclusively)

The "and" in e.g. "one hundred and twenty-four" is optional, in some cases it's present and in others it's not

All tested numbers are valid, you don't need to validate them

```
def parse_int(string):
    # Define a mapping between words and their numerical values
    word_to_number = {
        'zero': 0,
        'one': 1,
        'two': 2,
        'three': 3,
        'four': 4,
        'five': 5,
        'six': 6,
        'seven': 7,
        'eight': 8,
        'nine': 9,
        'ten': 10,
        'eleven': 11,
        'twelve': 12,
        'thirteen': 13,
        'fourteen': 14,
        'fifteen': 15,
        'sixteen': 16,
        'seventeen': 17,
        'eighteen': 18,
        'nineteen': 19,
        'twenty': 20,
        'thirty': 30,
        'forty': 40,
        'fifty': 50,
        'sixty': 60,
        'seventy': 70,
        'eighty': 80,
        'ninety': 90,
        'hundred': 100,
        'thousand': 1000,
        'million': 1000000
    }
```



```
Time: 500ms   Passed: 170   Failed: 520   Exit Code: 1

Test Results:
  Fixed tests
    > Tests (20 of 40 Assertions)
    Completed in 0.93ms
  Random tests
    > Tests (150 of 650 Assertions)
    Completed in 22.07ms
    # Split the input string into individual words
    words = string.replace('-', ' ').split()

    # Initialize variables to track the total value and the current value
    total = 0
    current = 0

    # Iterate through the words
    for word in words:
        # If the word represents a numerical value, add it to the current value
        if word in word_to_number:
            current += word_to_number[word]
        # If the word represents a multiple of 100, multiply the current value by it
        elif word == 'hundred':
            current *= 100
        # If the word represents a multiple of 1000 or 1000000, multiply the current value by it and add it to the total
        elif word in ['thousand', 'million']:
            total += current * word_to_number[word]
            current = 0

    # Add the remaining current value to the total
    total += current

    return total
```

Your task in order to complete this Kata is to write a function which formats a duration, given as a number of seconds, in a human-friendly way.

The function must accept a non-negative integer. If it is zero, it just returns "now". Otherwise, the duration is expressed as a combination of years, days, hours, minutes and seconds.

It is much easier to understand with an example:

* For seconds = 62, your function should return
  "1 minute and 2 seconds"
* For seconds = 3662, your function should return
  "1 hour, 1 minute and 2 seconds"

For the purpose of this Kata, a year is 365 days and a day is 24 hours.

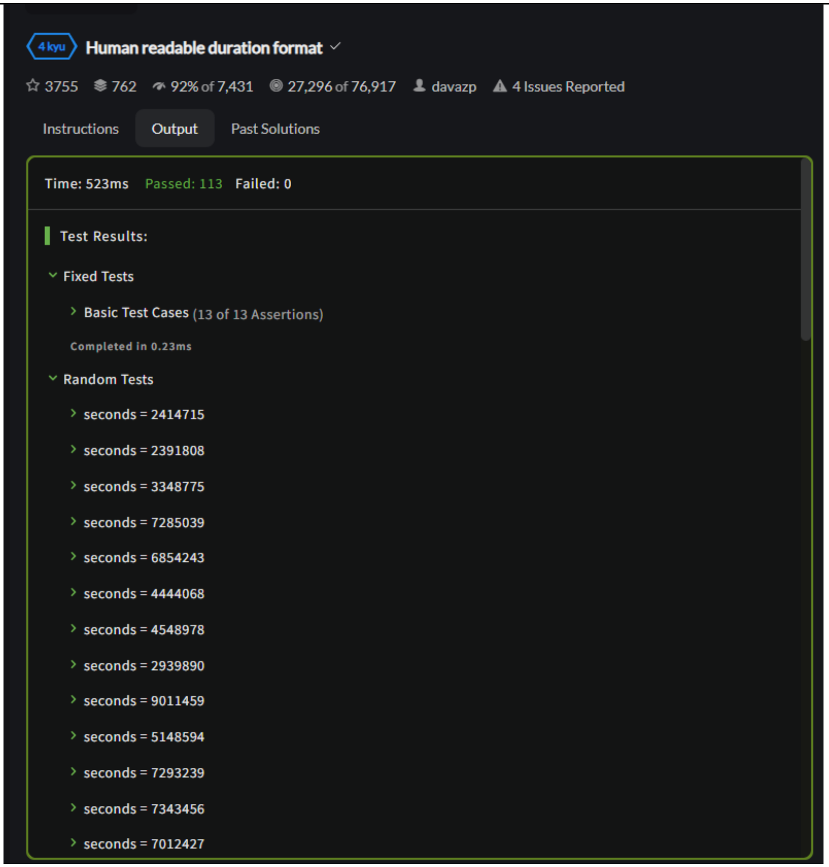
Note that spaces are important.

Detailed rules

The resulting expression is made of components like 4 seconds, 1 year, etc. In general, a positive integer and one of the valid units of time, separated by a space. The unit of time is used in plural if the integer is greater than 1.

The components are separated by a comma and a space (", "). Except the last

```
def format_duration(seconds):
    if seconds == 0:
        return "now"

    # Define the units of time and their
corresponding values in seconds
    units = [
        ('year', 365 * 24 * 60 * 60),
        ('day', 24 * 60 * 60),
        ('hour', 60 * 60),
        ('minute', 60),
        ('second', 1)
    ]

    # Initialize an empty list to store the
formatted components
    formatted_components = []

    # Iterate through the units of time
    for unit, value in units:
        # Calculate the number of
occurrences of the current unit in the
remaining seconds
        count = seconds // value
        # If the count is greater than zero,
add the formatted component to the list
        if count > 0:

formatted_components.append(f"{coun
t} {unit}" + ('' if count == 1 else 's'))
            # Subtract the corresponding
amount from the remaining seconds
            seconds -= count * value

    # Join the formatted components with
commas and 'and' where necessary
```



Human readable duration format

```
Time: 523ms   Passed: 113   Failed: 0

Test Results:
 ˅ Fixed Tests
   › Basic Test Cases (13 of 13 Assertions)
   Completed in 0.23ms
 ˅ Random Tests
   › seconds = 2414715
   › seconds = 2391808
   › seconds = 3348775
   › seconds = 7285039
   › seconds = 6854243
   › seconds = 4444068
   › seconds = 4548978
   › seconds = 2939890
   › seconds = 9011459
   › seconds = 5148594
   › seconds = 7293239
   › seconds = 7343456
   › seconds = 7012427
```

component, which is separated by " and ", just like it would be written in English.

A more significant units of time will occur before than a least significant one. Therefore, 1 second and 1 year is not correct, but 1 year and 1 second is.

Different components have different unit of times. So there is not repeated units like in 5 seconds and 1 second.

A component will not appear at all if its value happens to be zero. Hence, 1 minute and 0 seconds is not valid, but it should be just 1 minute.

A unit of time must be used "as much as possible". It means that the function should not return 61 seconds, but 1 minute and 1 second instead. Formally, the duration specified by of a component must not be greater than any valid more significant unit of time.

```
    formatted_duration = ', '.join(formatted_components[:-1])
    if formatted_duration:
        formatted_duration += ' and ' + formatted_components[-1]
    else:
        formatted_duration = formatted_components[-1]

    return formatted_duration
```