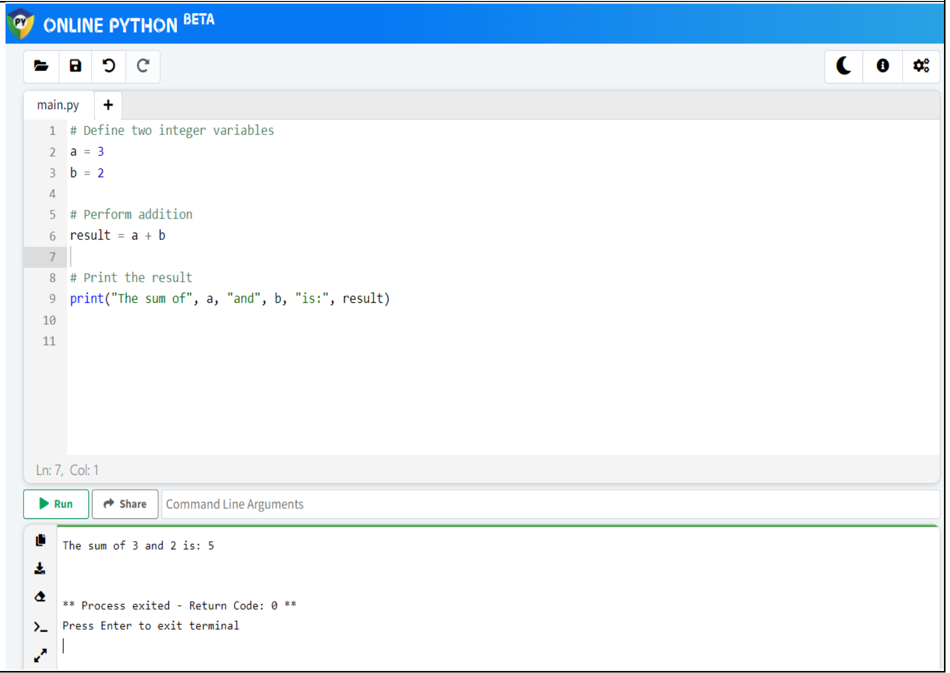
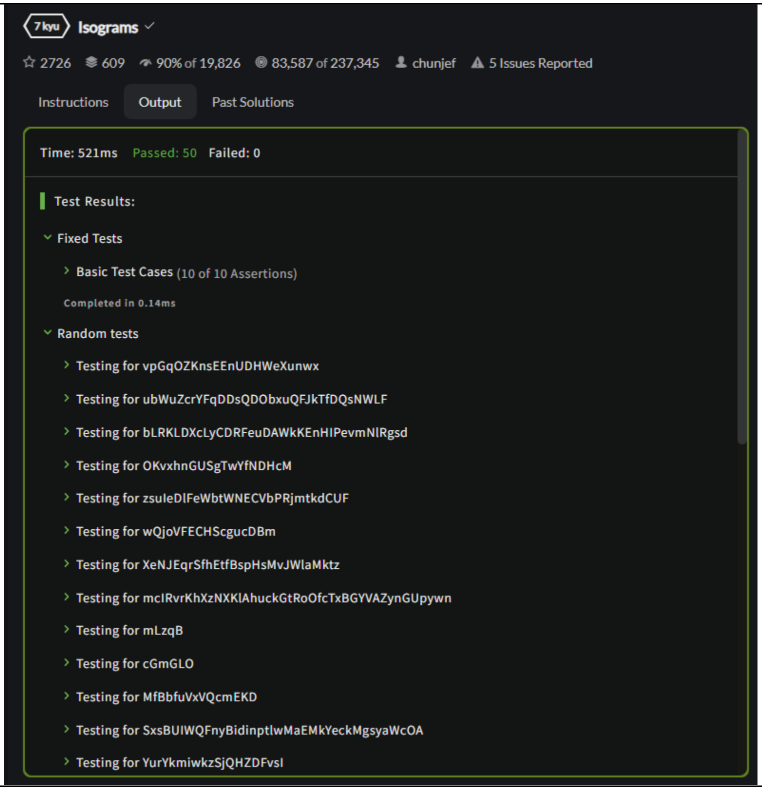
Koda ģenerēšana – ChatGPT 4

Instruction	Output	Koda rezultāti

Please generate Python code that defines two integer variables and then does a simple addition operation.	# Define two integer variables
	a = 3
	b = 2
	
	# Perform addition
	result = a + b
	
	# Print the result
	print("The sum of", a, "and", b, "is:", result)
		
		 ONLINE PYTHON BETA
		main.py
		Ln: 7, Col: 1
		Run
		Share
		Command Line Arguments
		`# Define two integer variables`
		`a = 3`
		`b = 2`
		
		`# Perform addition`
		`result = a + b`
		
		`# Print the result`
		`print("The sum of", a, "and", b, "is:", result)`
		
		`The sum of 3 and 2 is: 5`
		
		`** 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.

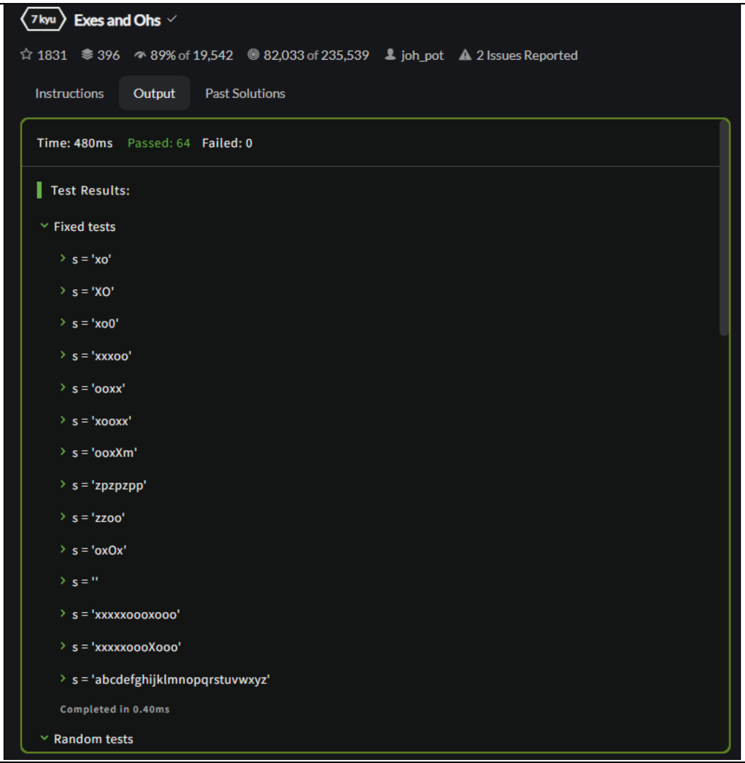
```python
def is_isogram(word):
    """
    Determine if the provided word is an isogram.

    Args:
    word (str): The word to check.

    Returns:
    bool: True if the word is an isogram, False otherwise.
    """
    seen = set()
    for char in word.lower():
        if char in seen:
            return False
        seen.add(char)
    return True
```



```
Time: 521ms   Passed: 50   Failed: 0

Test Results:
 Fixed Tests
   Basic Test Cases (10 of 10 Assertions)
   Completed in 0.14ms
 Random tests
   Testing for vpGqOZKnsEEnUDHWeXunwx
   Testing for ubWuZcrYFqDDsQDObxuQFJkTfDQsNWLF
   Testing for bLRKLDXcLyCDRFeuDAWkKEnHIPevmNlRgsd
   Testing for OKvxhnGUSgTwYfNDHcM
   Testing for zsuleDlFeWbtWNECVbPRjmtkdCUF
   Testing for wQjoVFECHScgucDBm
   Testing for XeNJEqrSfhEtfBspHsMvJWlaMktz
   Testing for mcIRvrKhXzNXKlAhuckGtRoOfcTxBGYVAZynGUpywn
   Testing for mLzqB
   Testing for cGmGLO
   Testing for MfBbfuVxVQcmEKD
   Testing for SxsBUIWQFnyBidinptlwMaEMkYeckMgsyaWcOA
   Testing for YurYkmiwkzSjQHZDFvsI
```

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.

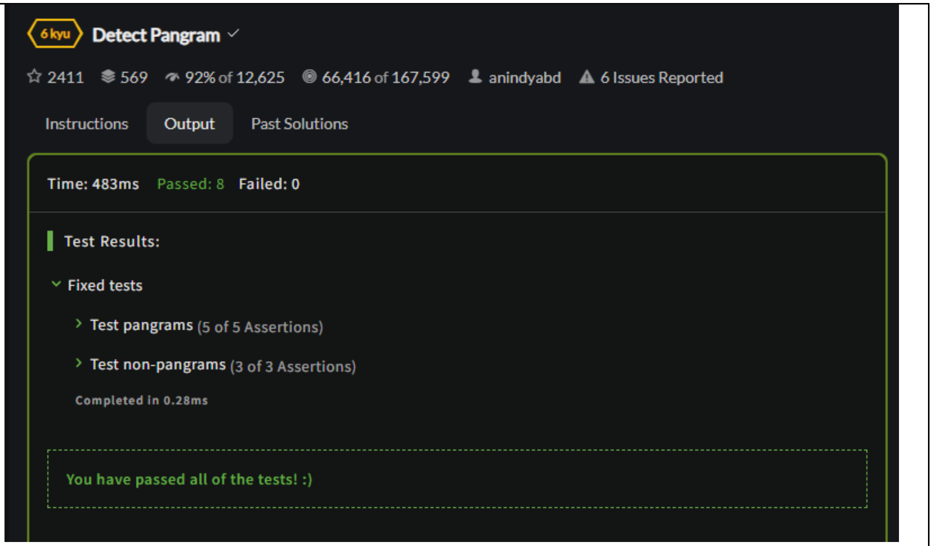
```python
def equal_xo_count(s):
    """
    Check if the string has equal numbers of 'x's and 'o's.

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

    Returns:
    bool: True if the number of 'x's and 'o's are equal, False otherwise.
    """
    s = s.lower()  # Convert the string to lowercase to make the check case-insensitive
    count_x = s.count('x')
    count_o = s.count('o')

    return count_x == count_o
```



```
Time: 480ms   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 0.40ms
  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

```
def is_pangram(sentence):
    """
    Determine if the provided sentence is a pangram.

    Args:
    sentence (str): The sentence to check.

    Returns:
    bool: True if the sentence is a pangram, False otherwise.
    """
    # Convert to lowercase and filter out non-alphabetic characters
    filtered_sentence = [char for char in sentence.lower() if char.isalpha()]

    # Use a set to find all unique letters
    unique_letters = set(filtered_sentence)

    # Check if we have all 26 letters of the alphabet
    return len(unique_letters) == 26
```



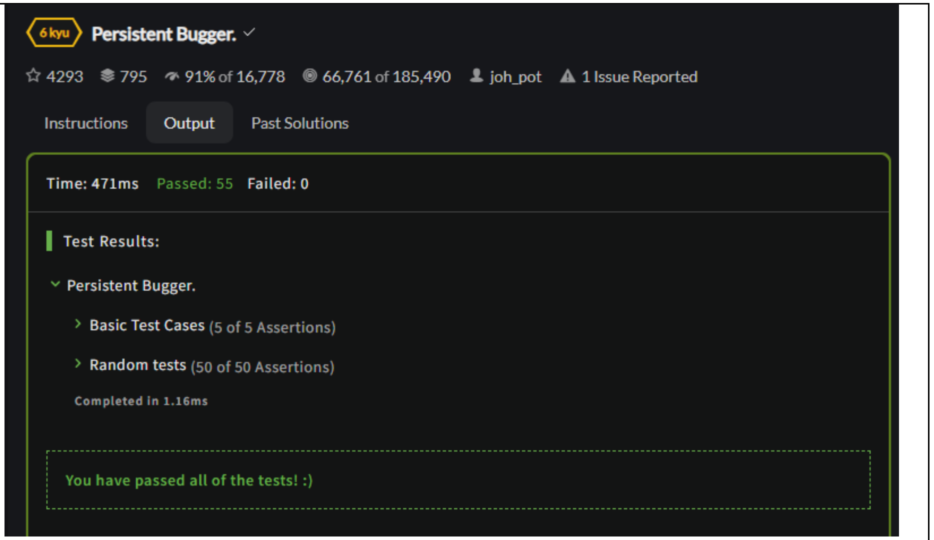
6 kyu **Detect Pangram** ✓

☆ 2411 569 92% of 12,625 66,416 of 167,599 anindyabd ⚠ 6 Issues Reported

```
Time: 483ms Passed: 8 Failed: 0

Test Results:
Fixed tests
  Test pangrams (5 of 5 Assertions)
  Test non-pangrams (3 of 3 Assertions)
  Completed in 0.28ms

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.

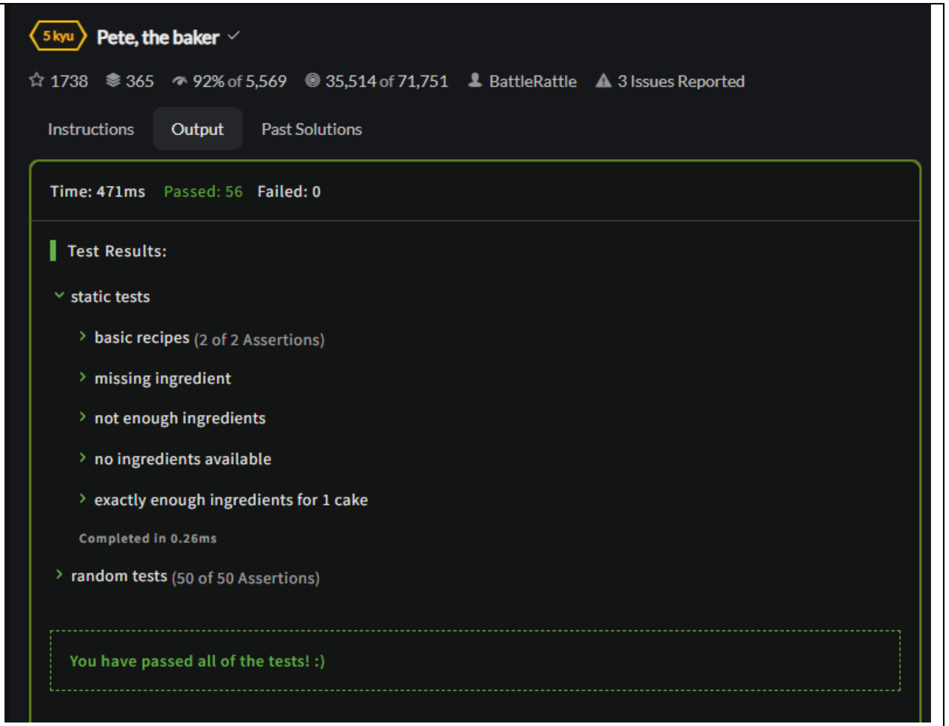
```python
def persistence(num):
    """
    Calculate the multiplicative persistence of a number.

    Args:
    num (int): The number to calculate the persistence for.

    Returns:
    int: The number of steps required to reduce num to a single digit by repeatedly multiplying its digits.
    """
    if num < 10:
        return 0  # If the number is already a single digit, no steps are needed

    steps = 0
    while num >= 10:  # Continue until num is a single digit
        # Multiply all digits of the current num
        product = 1
        while num > 0:
            product *= num % 10
            num //= 10
        num = product
        steps += 1

    return steps
```



```
Time: 471ms   Passed: 55   Failed: 0

Test Results:
  Persistent Bugger.
    > Basic Test Cases (5 of 5 Assertions)
    > Random tests (50 of 50 Assertions)
    Completed in 1.16ms

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):
    """
    Calculate the maximum number of cakes that can be baked based on the available ingredients.

    Args:
    recipe (dict): A dictionary containing the ingredients and their quantities needed for one cake.
    available (dict): A dictionary containing the ingredients and their quantities currently available.

    Returns:
    int: The maximum number of cakes that can be baked.
    """
    max_cakes = float('inf')  # Start with a theoretically infinite maximum

    for ingredient, amount_needed in recipe.items():
        if ingredient in available:
            # Calculate the maximum number of cakes for this ingredient
            max_cakes_for_ingredient = available[ingredient] // amount_needed
        else:
            # Ingredient is missing, so no cakes can be made
            return 0
```

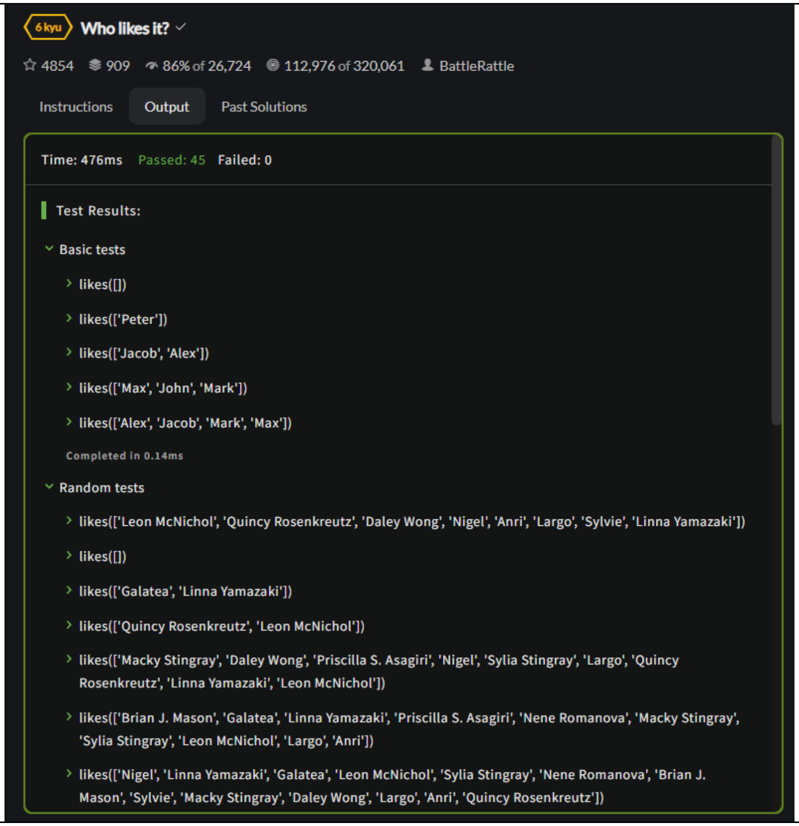


```
Time: 471ms  Passed: 56  Failed: 0

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 0.26ms
  random tests (50 of 50 Assertions)

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

```
        # Update the overall maximum cakes if this ingredient is more limiting
        if max_cakes_for_ingredient < max_cakes:
            max_cakes = max_cakes_for_ingredient

    return max_cakes
```

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):
    """
    Generates a string based on the number of likes from different people.

    Args:
    names (list): A list of strings containing the names of people who liked an item.

    Returns:
    str: A descriptive string about who likes the item.
    """
    n = len(names)
    if n == 0:
        return "no one likes this"
    elif n == 1:
        return f"{names[0]} likes this"
    elif n == 2:
        return f"{names[0]} and {names[1]} like this"
    elif n == 3:
        return f"{names[0]}, {names[1]} and {names[2]} like this"
    else:
```

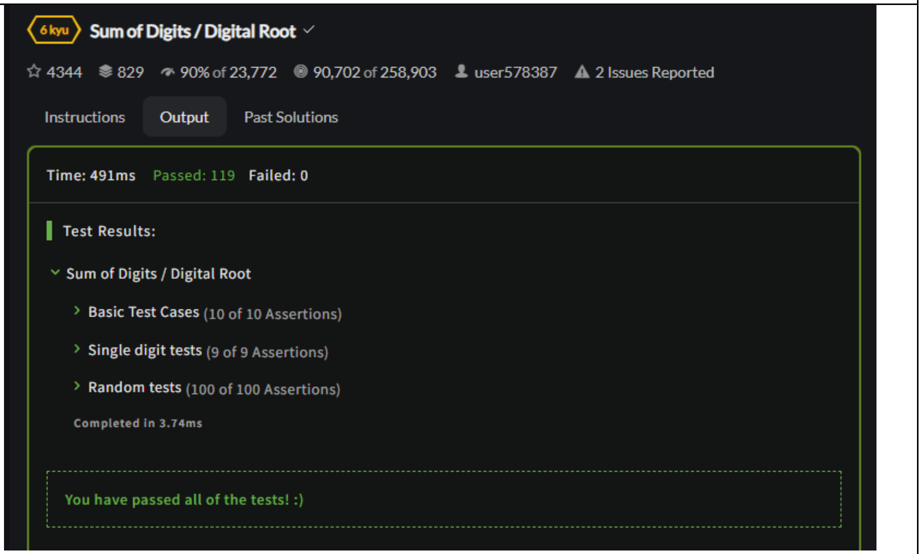


```
Time: 476ms   Passed: 45   Failed: 0

Test Results:
  Basic tests
    > likes([])
    > likes(['Peter'])
    > likes(['Jacob', 'Alex'])
    > likes(['Max', 'John', 'Mark'])
    > likes(['Alex', 'Jacob', 'Mark', 'Max'])
    Completed in 0.14ms
  Random tests
    > likes(['Leon McNichol', 'Quincy Rosenkreutz', 'Daley Wong', 'Nigel', 'Anri', 'Largo', 'Sylvie', 'Linna Yamazaki'])
    > likes([])
    > likes(['Galatea', 'Linna Yamazaki'])
    > likes(['Quincy Rosenkreutz', 'Leon McNichol'])
    > likes(['Macky Stingray', 'Daley Wong', 'Priscilla S. Asagiri', 'Nigel', 'Sylia Stingray', 'Largo', 'Quincy
      Rosenkreutz', 'Linna Yamazaki', 'Leon McNichol'])
    > likes(['Brian J. Mason', 'Galatea', 'Linna Yamazaki', 'Priscilla S. Asagiri', 'Nene Romanova', 'Macky Stingray',
      'Sylia Stingray', 'Leon McNichol', 'Largo', 'Anri'])
    > likes(['Nigel', 'Linna Yamazaki', 'Galatea', 'Leon McNichol', 'Sylia Stingray', 'Nene Romanova', 'Brian J.
      Mason', 'Sylvie', 'Macky Stingray', 'Daley Wong', 'Largo', 'Anri', 'Quincy Rosenkreutz'])
```

```
        return f"{names[0]}, {names[1]} and {n - 2} others like this"
```

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.

```
def digital_root(n):
    """
    Calculate the digital root of a non-negative integer.

    Args:
    n (int): The non-negative integer to calculate the digital root of.

    Returns:
    int: The digital root of the integer.
    """
    while n >= 10:  # Continue while the number has more than one digit
        sum_digits = 0
        while n > 0:
            sum_digits += n % 10
            n //= 10
        n = sum_digits  # Update n to the sum of its digits
    return n
```



6 kyu Sum of Digits / Digital Root

☆ 4344 829 90% of 23,772 90,702 of 258,903 user578387 2 Issues Reported

Time: 491ms 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 3.74ms

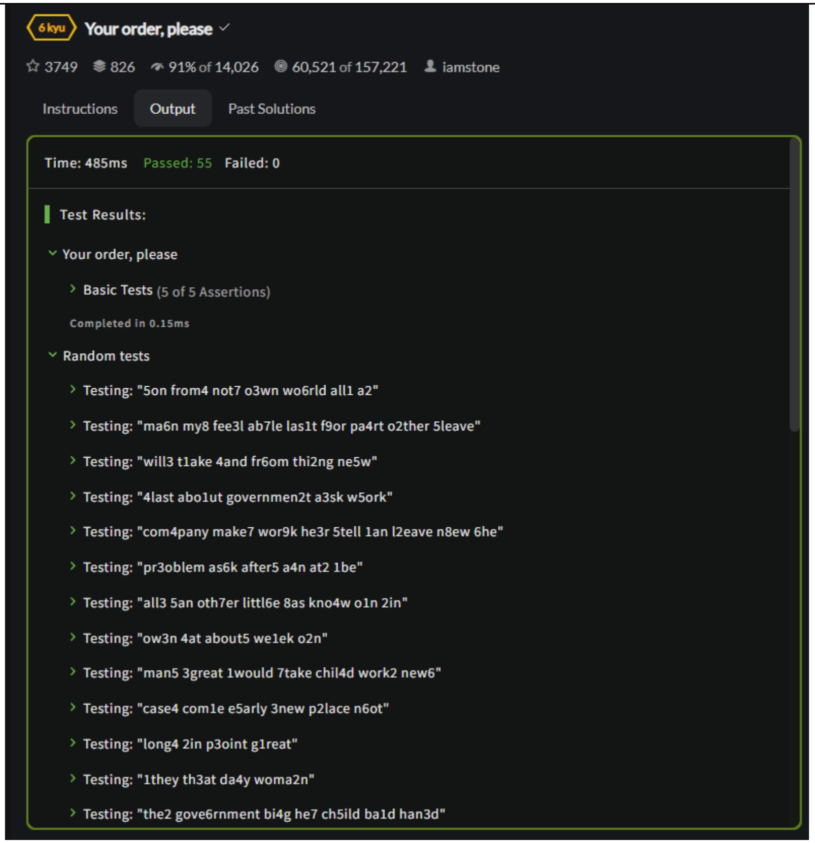
You have passed all of the tests! :)

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.

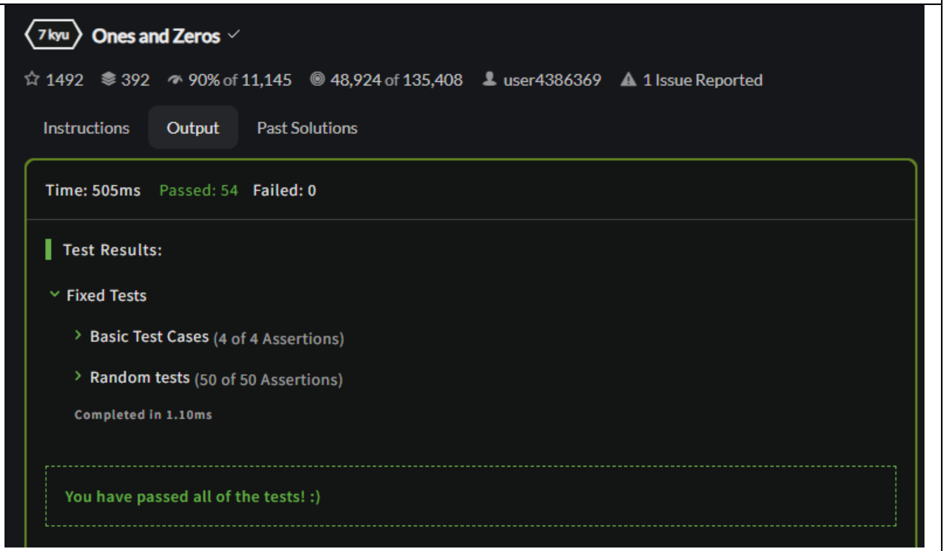
```
def order(sentence):
    """
    Sorts words in a sentence based on the numeric digit found in each word.

    Args:
    sentence (str): The string containing words with embedded numbers.

    Returns:
    str: A string with the words sorted according to the embedded numbers.
    """
    words = sentence.split()
    sorted_words = [''] * len(words)  # Create a list with the same number of empty elements as words

    for word in words:
        for char in word:
            if char.isdigit():  # Check if the character is a digit
                # Place the word in the correct position (subtract 1 since lists are zero-indexed)
                sorted_words[int(char) - 1] = word
                break  # No need to check further characters once the digit is found

    # Join the sorted words back into a single string
```



```
Time: 485ms   Passed: 55   Failed: 0

Test Results:
  Your order, please
    Basic Tests (5 of 5 Assertions)
    Completed in 0.15ms
  Random tests
    Testing: "5on from4 not7 o3wn wo6rld all1 a2"
    Testing: "ma6n my8 fee3l ab7le las1t f9or pa4rt o2ther 5leave"
    Testing: "will3 t1ake 4and fr6om thi2ng ne5w"
    Testing: "4last abo1ut governmen2t a3sk w5ork"
    Testing: "com4pany make7 wor9k he3r 5tell 1an l2eave n8ew 6he"
    Testing: "pr3oblem as6k after5 a4n at2 1be"
    Testing: "all3 5an oth7er littl6e 8as kno4w o1n 2in"
    Testing: "ow3n 4at about5 we1ek o2n"
    Testing: "man5 3great 1would 7take chil4d work2 new6"
    Testing: "case4 com1e e5arly 3new p2lace n6ot"
    Testing: "long4 2in p3oint g1reat"
    Testing: "1they th3at da4y woma2n"
    Testing: "the2 gove6rnment bi4g he7 ch5ild ba1d han3d"
    return ' '.join(sorted_words)
```

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):
    """
    Converts a list of binary digits (0s and 1s) to the corresponding decimal integer.

    Args:
    arr (list): List of integers (0s and 1s) representing a binary number.

    Returns:
    int: Decimal integer representation of the binary number.
    """
    result = 0
    for digit in arr:
        result = (result << 1) | digit  # Shift result left by 1 and bitwise OR with the digit
    return result
```

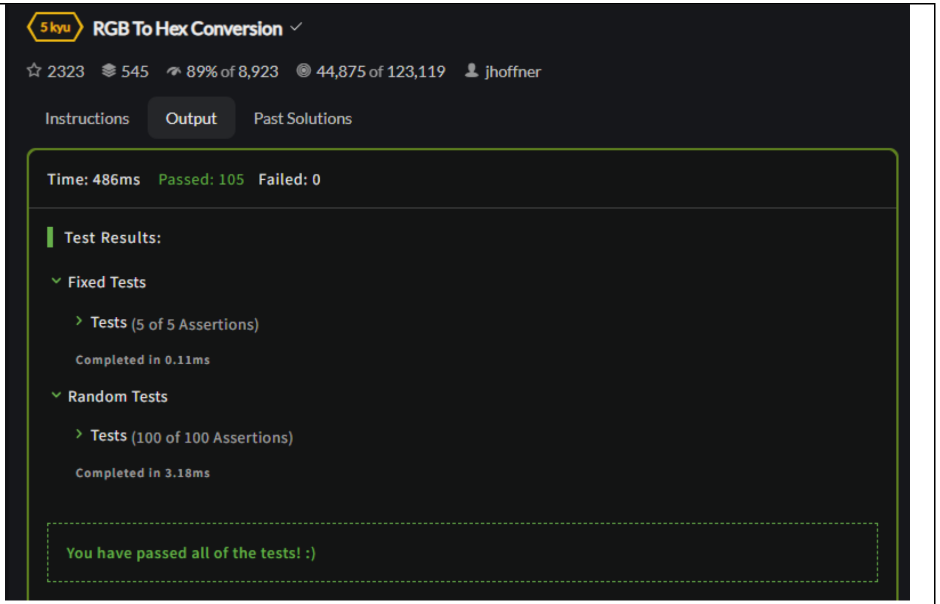


```
Time: 505ms   Passed: 54   Failed: 0

Test Results:
  Fixed Tests
    Basic Test Cases (4 of 4 Assertions)
    Random tests (50 of 50 Assertions)
    Completed in 1.10ms

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):
    """
    Convert RGB decimal values to a hexadecimal string.

    Args:
    r (int): Red component (0-255)
    g (int): Green component (0-255)
    b (int): Blue component (0-255)

    Returns:
    str: Hexadecimal string representation of the RGB color.
    """
    # Helper function to convert a single color component
    def clamp_and_convert(x):
        # Clamp x between 0 and 255
        x = max(0, min(255, x))
        # Convert to hexadecimal and return two character format
        return f"{x:02X}"

    # Convert each component and concatenate
    return clamp_and_convert(r) + clamp_and_convert(g) + clamp_and_convert(b)
```



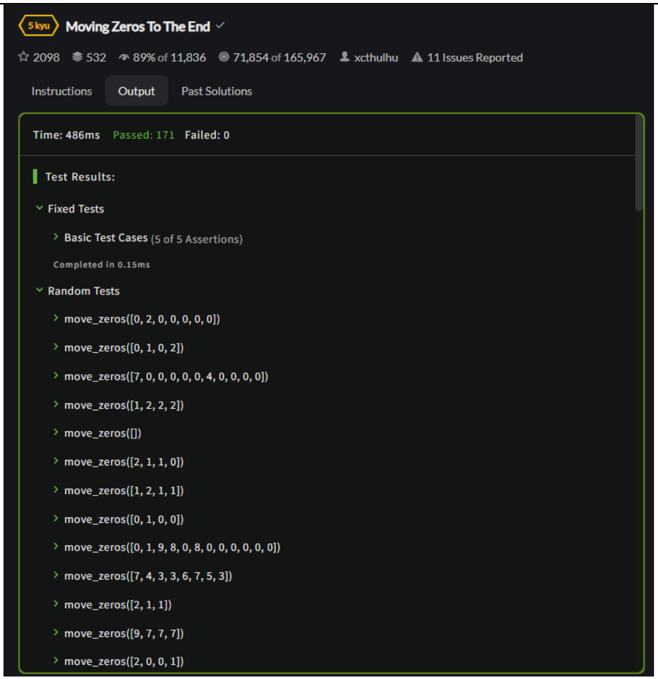
RGB To Hex Conversion ✓

☆ 2323 545 89% of 8,923 44,875 of 123,119 jhoffner

```
Time: 486ms   Passed: 105   Failed: 0

Test Results:
 Fixed Tests
   Tests (5 of 5 Assertions)
   Completed in 0.11ms
 Random Tests
   Tests (100 of 100 Assertions)
   Completed in 3.18ms

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.

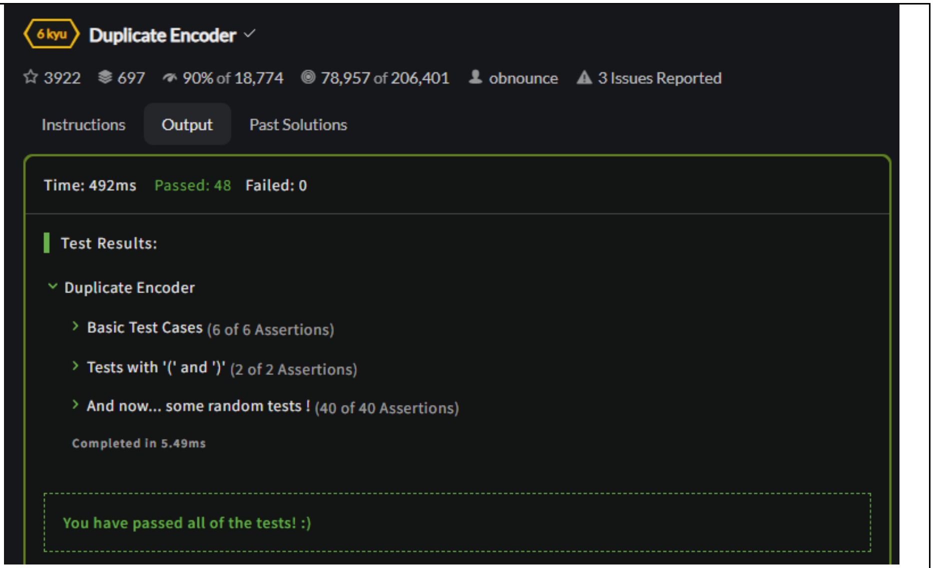
```
def move_zeros(arr):
    """
    Moves all zeros in the list to the end while preserving the order of non-zero elements.

    Args:
    arr (list): List of integers which may contain zeros.

    Returns:
    list: The modified list with all zeros moved to the end.
    """
    zero_count = arr.count(0)  # Count how many zeros are in the list
    # Create a new list without zeros
    new_arr = [x for x in arr if x != 0]
    # Append the necessary amount of zeros at the end
    new_arr.extend([0] * zero_count)
    return new_arr
```



```
Time: 486ms   Passed: 171   Failed: 0

Test Results:
  Fixed Tests
    Basic Test Cases (5 of 5 Assertions)
    Completed in 0.15ms
  Random Tests
    move_zeros([0, 2, 0, 0, 0, 0, 0])
    move_zeros([0, 1, 0, 2])
    move_zeros([7, 0, 0, 0, 0, 0, 4, 0, 0, 0, 0])
    move_zeros([1, 2, 2, 2])
    move_zeros([])
    move_zeros([2, 1, 1, 0])
    move_zeros([1, 2, 1, 1])
    move_zeros([0, 1, 0, 0])
    move_zeros([0, 1, 9, 8, 0, 8, 0, 0, 0, 0, 0, 0])
    move_zeros([7, 4, 3, 3, 6, 7, 5, 3])
    move_zeros([2, 1, 1])
    move_zeros([9, 7, 7, 7])
    move_zeros([2, 0, 0, 1])
```

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):
    """
    Encodes a string by replacing each character with '(' if it appears only once
    in the original string, and ')' if it appears more than once. The comparison
    is case-insensitive.

    Args:
    word (str): The input string to encode.

    Returns:
    str: The encoded string.
    """
    # Normalize the string to lowercase
    normalized_word = word.lower()
    # Create a dictionary to count occurrences of each character
    char_count = {}
    for char in normalized_word:
        if char in char_count:
            char_count[char] += 1
        else:
            char_count[char] = 1

    # Construct the encoded string
    encoded = ""
    for char in normalized_word:
        if char_count[char] > 1:
            encoded += ")"
        else:
            encoded += "("
```

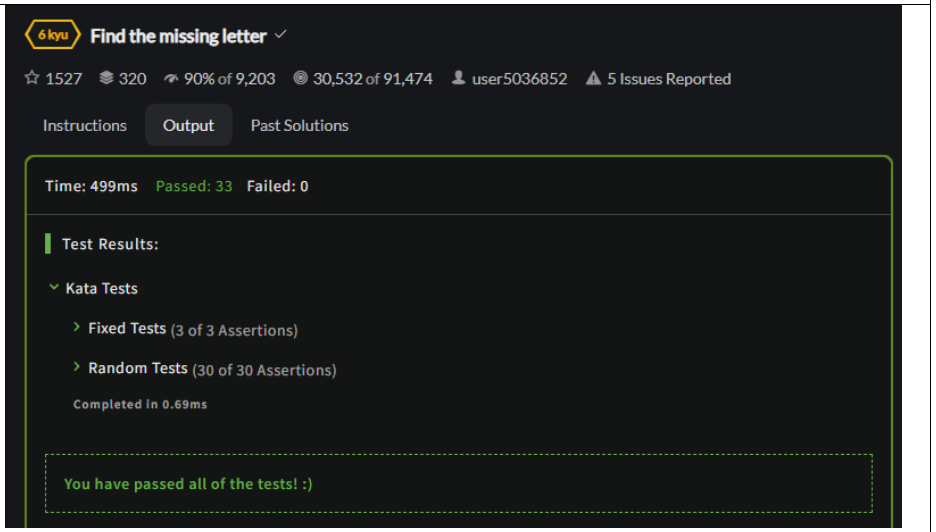


```
Time: 492ms   Passed: 48   Failed: 0

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 5.49ms

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

```
    return encoded
```

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):
    """
    Finds the missing letter in an array of consecutive letters.

    Args:
    chars (list): A list of consecutive (increasing) letters with exactly one missing.

    Returns:
    str: The missing letter.
    """
    # Convert characters to their ASCII values and find the missing one
    previous = ord(chars[0])
    for char in chars[1:]:
        current = ord(char)
        if current != previous + 1:
            # Return the missing character
            return chr(previous + 1)
        previous = current

    # Return None if no missing character is found (should not happen with valid input)
    return None
```



Time: 499ms Passed: 33 Failed: 0

Test Results:

Kata Tests

Fixed Tests (3 of 3 Assertions)

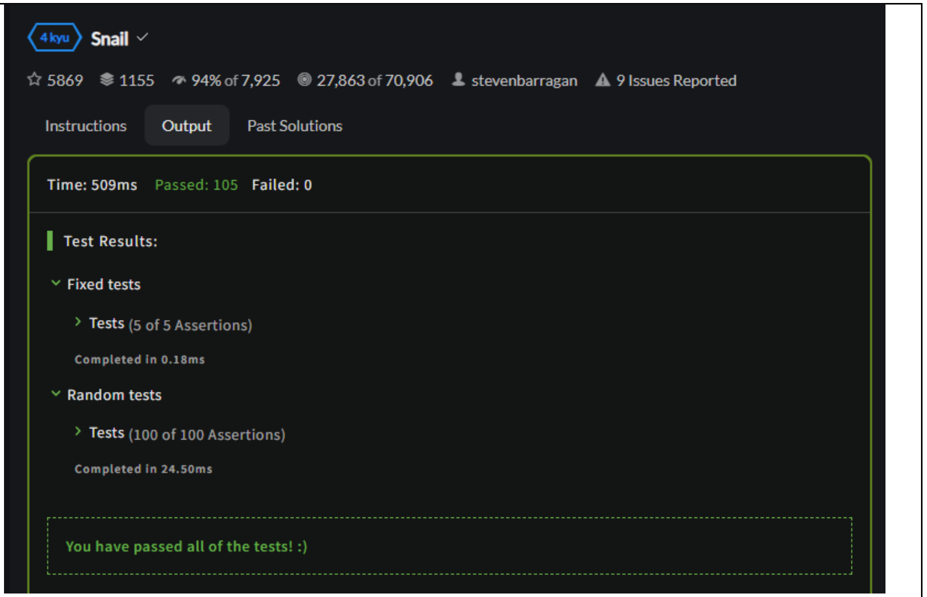
Random Tests (30 of 30 Assertions)

Completed in 0.69ms

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):
    """
    Returns the elements of an n x n
array arranged from outermost
elements to the middle element,
traveling clockwise.

    Args:
    matrix (list of list of int): An n x n
2D array.

    Returns:
    list: List of integers representing
the spiral order of elements.
    """
    result = []

    while matrix:
        # Remove the first row
        result.extend(matrix.pop(0))

        if matrix and matrix[0]:
            # Take the last element of
each remaining row
            for row in matrix:
                result.append(row.pop())

        if matrix:
            # Remove the last row and
add its elements in reverse order
            result.extend(matrix.pop()[::-
1])

        if matrix and matrix[0]:
            # Take the first element of
each remaining row (going upward)
            for row in reversed(matrix):
```



```
Time: 509ms   Passed: 105   Failed: 0

Test Results:
  Fixed tests
    Tests (5 of 5 Assertions)
    Completed in 0.18ms
  Random tests
    Tests (100 of 100 Assertions)
    Completed in 24.50ms

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

```
            result.append(row.pop(0))

    return result
```

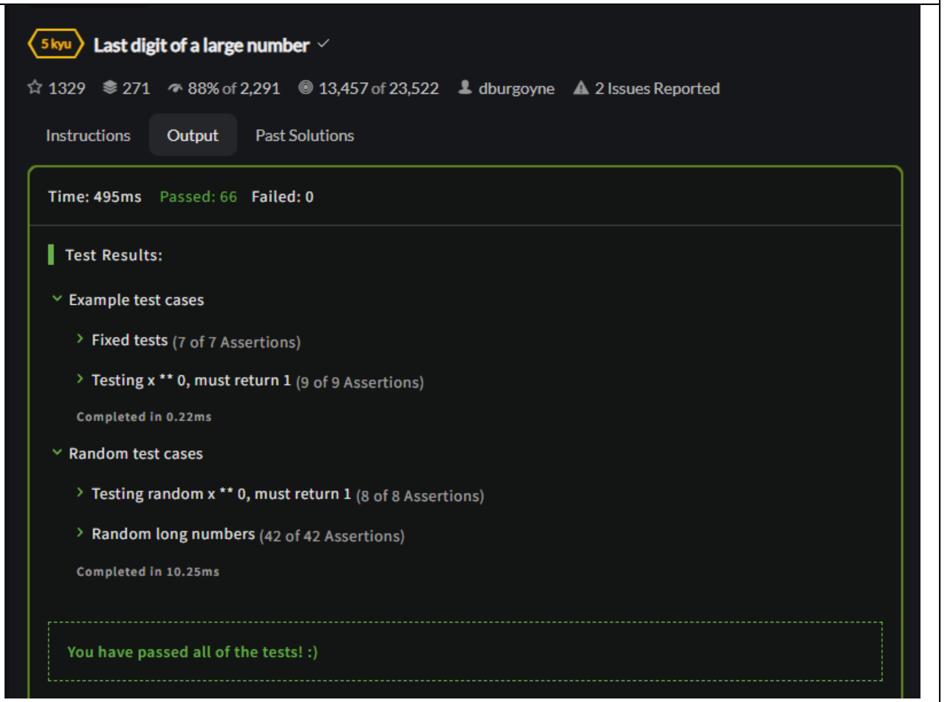
Define a function that takes in two non-negative integers �a and �b and returns the last decimal digit of ��ab. Note that �a and �b may be very large!

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):
    """
    Returns the last digit of a^b.

    Args:
    a (int): Base of the exponentiation.
    b (int): Exponent in the exponentiation.

    Returns:
    int: Last digit of a^b.
    """
    if b == 0:  # handling the special case where any number to the power of 0 is 1
        return 1

    # Cycles of last digits for bases 0-9
    last_digit_cycles = {
        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]
    }

    # Get last digit of a
```

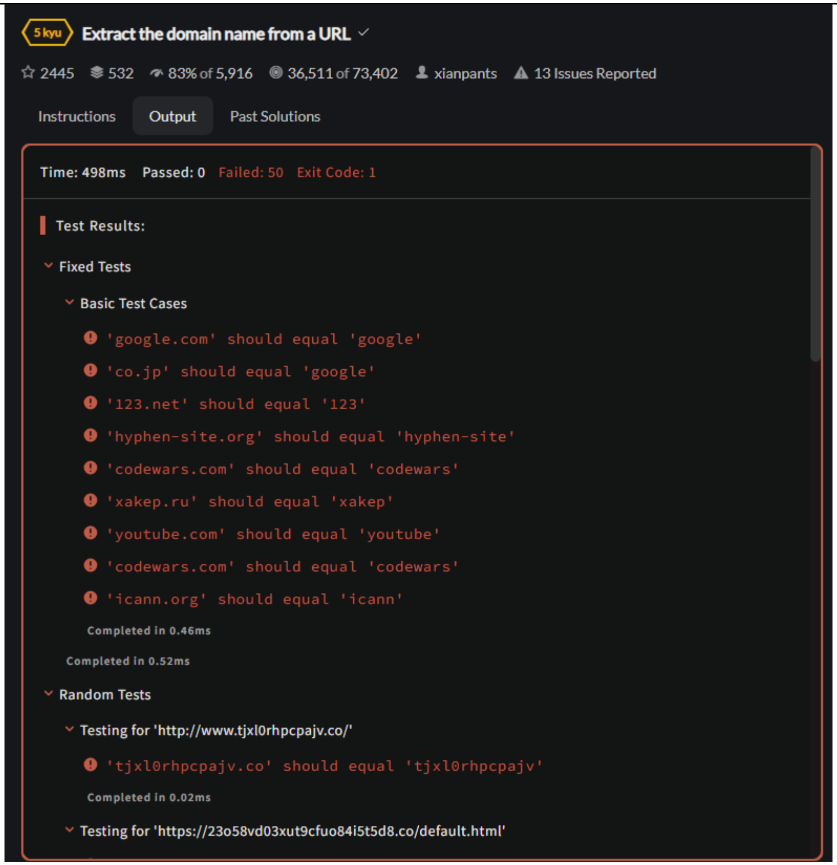


```
Time: 495ms   Passed: 66   Failed: 0

Test Results:
  Example test cases
    > Fixed tests (7 of 7 Assertions)
    > Testing x ** 0, must return 1 (9 of 9 Assertions)
    Completed in 0.22ms
  Random test cases
    > Testing random x ** 0, must return 1 (8 of 8 Assertions)
    > Random long numbers (42 of 42 Assertions)
    Completed in 10.25ms

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

```
    last_digit_of_a = a % 10
    # Get the cycle based on the last digit of a
    cycle = last_digit_cycles[last_digit_of_a]
    cycle_length = len(cycle)

    # Find the index in the cycle
    # b % cycle_length gives index in the cycle, subtract 1 because index is 0-based
    index = (b % cycle_length) - 1

    # Handle the case where index is -1 (meaning b is exactly divisible by cycle_length)
    return cycle[index]
```

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 get_domain_name(url):
    """
    Extracts the domain name from a given URL.

    Args:
    url (str): The URL from which to extract the domain name.

    Returns:
    str: The extracted domain name.
    """
    # Regex pattern to find the domain
    pattern = r'(?:http[s]?://)?(?:www\.)?([^/:\?]+)'

    # Search the URL with the regex pattern
    match = re.search(pattern, url)
    if match:
        domain = match.group(1)
        # Split by '.' and remove potential www or other subdomains to get the main domain
        domain_parts = domain.split('.')
        if len(domain_parts) > 2:
            # Usually the main domain parts are the last two parts, except known SLDs
            return '.'.join(domain_parts[-2:])
```



Extract the domain name from a URL

☆ 2445 532 83% of 5,916 36,511 of 73,402 xianpants 13 Issues Reported

```
Time: 498ms   Passed: 0   Failed: 50   Exit Code: 1

Test Results:
  Fixed Tests
    Basic Test Cases
      'google.com' should equal 'google'
      'co.jp' should equal 'google'
      '123.net' should equal '123'
      'hyphen-site.org' should equal 'hyphen-site'
      'codewars.com' should equal 'codewars'
      'xakep.ru' should equal 'xakep'
      'youtube.com' should equal 'youtube'
      'codewars.com' should equal 'codewars'
      'icann.org' should equal 'icann'
      Completed in 0.46ms
    Completed in 0.52ms
  Random Tests
    Testing for 'http://www.tjxl0rhpcpajv.co/'
      'tjxl0rhpcpajv.co' should equal 'tjxl0rhpcpajv'
      Completed in 0.02ms
    Testing for 'https://23o58vd03xut9cfuo84i5t5d8.co/default.html'
```

```
        return domain
    return ""
```

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

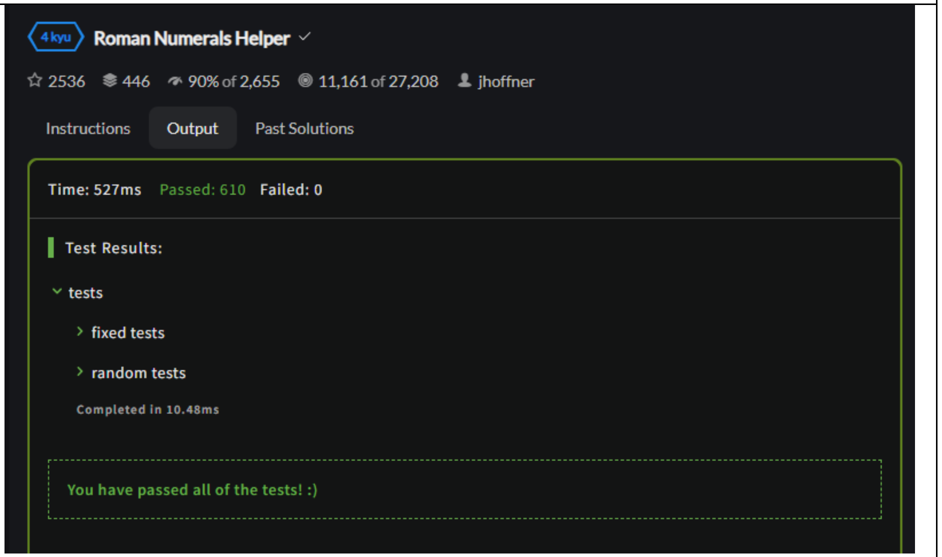
2008 is written as 2000=MM, 8=VIII; or MMVIII

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").

```
def roman_to_int(s):
    """
    Converts a Roman numeral string to an integer.
    """
    roman_map = {'I': 1, 'V': 5, 'X': 10, 'L': 50, 'C': 100, 'D': 500, 'M': 1000}
    integer_value = 0
    prev_value = 0

    for char in reversed(s):
        int_val = roman_map[char]
        if int_val >= prev_value:
            integer_value += int_val
        else:
            integer_value -= int_val
        prev_value = int_val

    return integer_value

def int_to_roman(num):
    """
    Converts an integer to a Roman numeral string.
    """
    roman_pairs = [
        (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")
    ]
    roman_result = []
```



Roman Numerals Helper

```
Time: 527ms   Passed: 610   Failed: 0

Test Results:
  tests
    fixed tests
    random tests
  Completed in 10.48ms

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

```
  for value, numeral in
roman_pairs:
      while num >= value:

roman_result.append(numeral)
         num -= value

  return ''.join(roman_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:

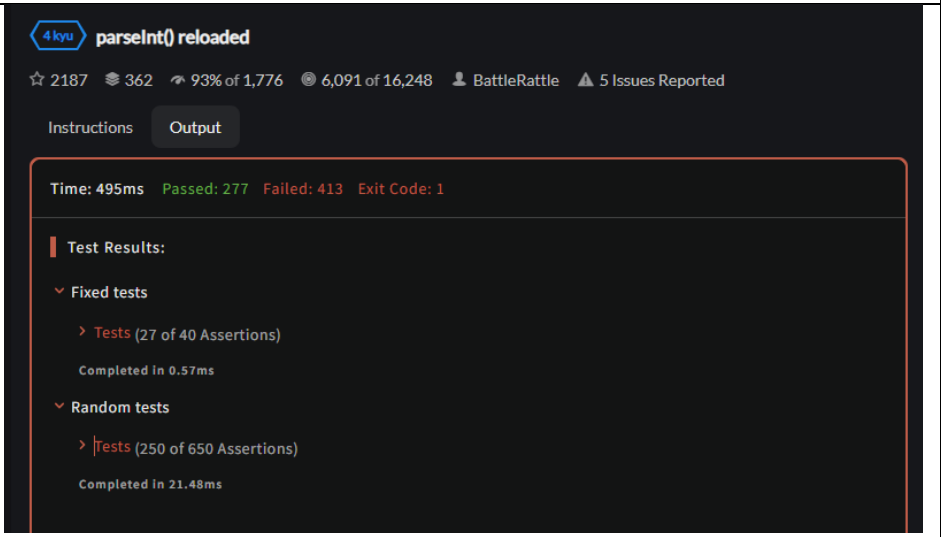
The minimum number is "zero" (inclusively)

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):
  num_dict = {
     '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
  }
  multipliers = {
     'hundred': 100,
     'thousand': 1000,
     'million': 1000000
  }

  words = string.replace('-', '
').replace('and', '').split()
  total = 0
  current = 0

  for word in words:
```



parseInt() reloaded

☆ 2187 362 93% of 1,776 6,091 of 16,248 BattleRattle 5 Issues Reported

```
Time: 495ms  Passed: 277  Failed: 413  Exit Code: 1

Test Results:
  Fixed tests
    > Tests (27 of 40 Assertions)
    Completed in 0.57ms
  Random tests
    > Tests (250 of 650 Assertions)
    Completed in 21.48ms
```

```
        if word in num_dict:
            current += num_dict[word]
        elif word in multipliers:
            current *= multipliers[word]
            if word in ['thousand', 'million']:
                total += current
                current = 0

    return total + current
```

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 second s.

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.

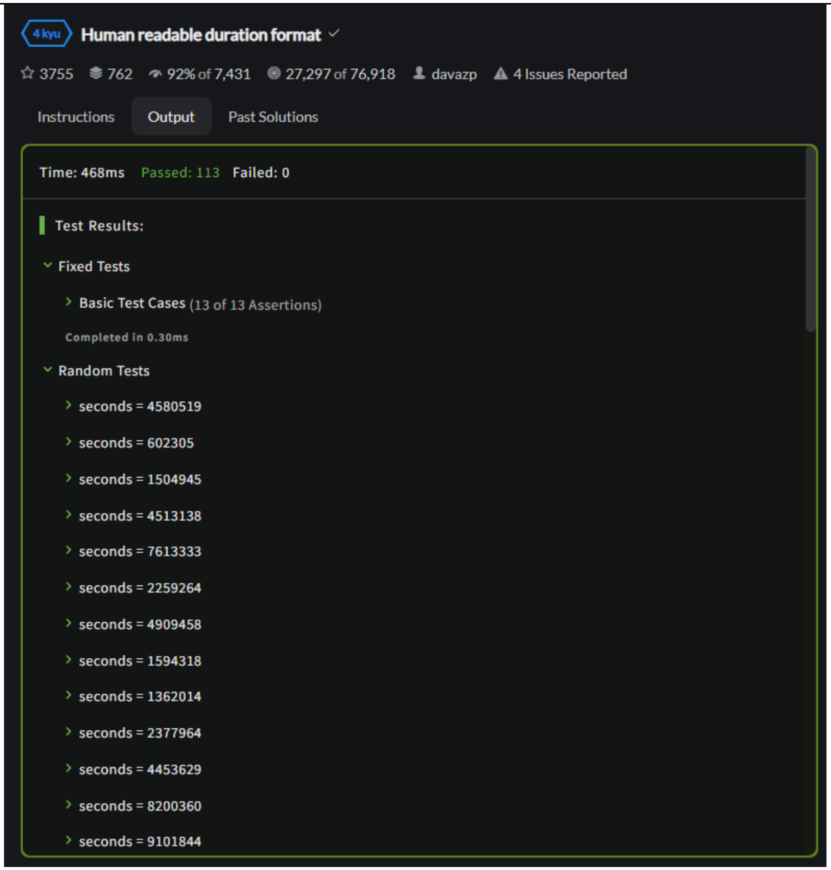
Note that spaces are important.

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"

    # Conversion factors
    minute = 60
    hour = 60 * minute
    day = 24 * hour
    year = 365 * day

    # Calculate each component
    years = seconds // year
    seconds %= year
    days = seconds // day
    seconds %= day
    hours = seconds // hour
    seconds %= hour
    minutes = seconds // minute
    seconds %= minute

    # Build components list
    components = []
    if years:
components.append(f"{years}
year{'s' if years > 1 else ''}")
    if days:
components.append(f"{days}
day{'s' if days > 1 else ''}")
    if hours:
components.append(f"{hours}
hour{'s' if hours > 1 else ''}")
    if minutes:
components.append(f"{minutes}
minute{'s' if minutes > 1 else ''}")
    if seconds:
components.append(f"{seconds}
second{'s' if seconds > 1 else ''}")
```



```
Time: 468ms   Passed: 113   Failed: 0

Test Results:
  Fixed Tests
    Basic Test Cases (13 of 13 Assertions)
    Completed in 0.30ms
  Random Tests
    seconds = 4580519
    seconds = 602305
    seconds = 1504945
    seconds = 4513138
    seconds = 7613333
    seconds = 2259264
    seconds = 4909458
    seconds = 1594318
    seconds = 1362014
    seconds = 2377964
    seconds = 4453629
    seconds = 8200360
    seconds = 9101844
```

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.

```
    # Format components into a
string with commas and "and"
    if len(components) == 1:
        return components[0]
    elif len(components) == 2:
        return ' and '.join(components)
    else:
        return ', '.join(components[:-
1]) + ' and ' + components[-1]
```