1.Write a function that takes a list of lists and returns the value of all of the symbols in it, where each symbol adds or takes something from the total score. Symbol values:

= 5

O = 3

X = 1

! = -1

!! = -3

!!! = -5

A list of lists containing 2 #s, a O, and a !!! would equal (0 + 5 + 5 + 3 - 5) 8.

If the final score is negative, return 0 (e.g. 3 #s, 3 !!s, 2 !!!s and a X would be (0 + 5 + 5 + 5 - 3 - 3 - 3 - 5 - 5 + 1) -3, so return 0).

Examples:

check_score([

["#", "!"],

["!!", "X"]

]) ➞ 2

check_score([

["!!!", "O", "!"],

["X", "#", "!!!"],

["!!", "X", "O"]

]) ➞ 0

2. Create a function that takes a variable number of arguments, each argument representing the number of items in a group, and returns the number of permutations (combinations) of items that you could get by taking one item from each group.

Examples:

combinations(2, 3) ➞ 6

combinations(3, 7, 4) ➞ 84

combinations(2, 3, 4, 5) ➞ 120

3. Create a function that takes a string as an argument and returns the Morse code equivalent.

encode_morse("EDABBIT CHALLENGE") ➞ ". -.. .- -... -... .. - -.-. .... .- .-.. .-.. . -. --. ."

encode_morse("HELP ME !") ➞ ".... . .-.. .--. -- . -.-.--"

This dictionary can be used for coding:

```
char_to_dots = {
  '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': '--..', ' ': ' ', '0': '-----', '1': '.----',
  '2': '..---', '3': '...--', '4': '....-', '5': '.....', '6': '-....',
  '7': '--...', '8': '---..', '9': '----.', '&': '.-...', "'": '.----.',
  '@': '.--.-.', ')': '-.--.-', '(': '-.--.', ':': '---...', ',': '--..--',
  '=': '-...-', '!': '-.-.--', '.': '.-.-.-', '-': '-....-', '+': '.-.-.',
  '"': '.-..-.', '?': '..--..', '/': '-..-.'
}
```

4. Write a function that takes a number and returns True if it's a prime; False otherwise. The number can be 2^64−1 (2 to the power of 63, not XOR). With the standard technique it would be

O(2 ^64−1), which is much too large for the 10 second time limit.

Examples:

prime(7) ➞ True

prime(56963) ➞ True

prime(5151512515524) ➞ False

5. Create a function that converts a word to a bitstring and then to a boolean list based on the following criteria:

Locate the position of the letter in the English alphabet (from 1 to 26).

Odd positions will be represented as 1 and 0 otherwise.

Convert the represented positions to boolean values, 1 for True and 0 for False.

Store the conversions into an array.

Examples:

to_boolean_list("deep") ⟶ [False, True, True, False]

deep converts to 0110

d is the 4th alphabet - 0

e is the 5th alphabet - 1

e is the 5th alphabet - 1

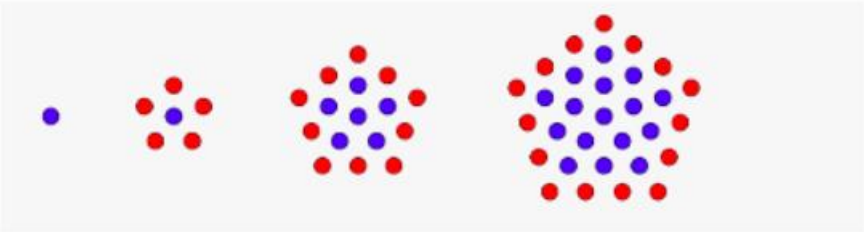
p is the 16th alphabet - 0

to_boolean_list("loves") ⟶ [False, True, False, True, True]

to_boolean_list("tesh") ⟶ [False, True, True, False]

6. Write a function that takes a positive integer num and calculates how many dots exist in a pentagonal shape around the center dot on the Nth iteration.

In the image below you can see the first iteration is only a single dot. On the second, there are 6 dots. On the third, there are 16 dots, and on the fourth there are 31 dots.



Return the number of dots that exist in the whole pentagon on the Nth iteration.

Examples

pentagonal(1) ⟶ 1

pentagonal(2) ⟶ 6

pentagonal(3) ⟶ 16

pentagonal(8) ⟶ 141

7. Make a function that encrypts a given input with these steps:

Input: "apple"

Step 1: Reverse the input: "elppa"

Step 2: Replace all vowels using the following chart:

a => 0

e => 1

i => 2

o => 2

u => 3

"1lpp0"

Step 3: Add "aca" to the end of the word: "1lpp0aca"

Output: "1lpp0aca"

Examples

encrypt("banana") ➞ "0n0n0baca"

encrypt("karaca") ➞ "0c0r0kaca"

encrypt("burak") ➞ "k0r3baca"

encrypt("alpaca") ➞ "0c0pl0aca"

8. Given the month and year as numbers, return whether that month contains a Friday 13th.(i.e You can check Python's datetime module)

Examples

has_friday_13(3, 2020) ➞ True

has_friday_13(10, 2017) ➞ True

has_friday_13(1, 1985) ➞ False

9. Write a regular expression that will help us count how many bad cookies are produced every day. You must use RegEx negative lookbehind.

Example

lst = ["bad cookie", "good cookie", "bad cookie", "good cookie", "good cookie"]

pattern = "yourregularexpressionhere"

len(re.findall(pattern, ", ".join(lst))) ➞ 2

10. Given a list of words in the singular form, return a set of those words in the plural form if they appear more than once in the list.

Examples

pluralize(["cow", "pig", "cow", "cow"]) ➞ { "cows", "pig" }

pluralize(["table", "table", "table"]) ➞ { "tables" }

pluralize(["chair", "pencil", "arm"]) ➞ { "chair", "pencil", "arm" }

11. Create a function to perform basic arithmetic operations that includes addition, subtraction, multiplication and division on a string number (e.g. "12 + 24" or "23 - 21" or "12 // 12" or "12 * 21").

Here, we have 1 followed by a space, operator followed by another space and 2. For the challenge, we are going to have only two numbers between 1 valid operator. The return value should be a number.

eval() is not allowed. In case of division, whenever the second number equals "0" return -1.

For example:

"15 // 0" ⟶ -1

Examples

arithmetic_operation("12 + 12") ⟶ 24 // 12 + 12 = 24

arithmetic_operation("12 - 12") ⟶ 24 // 12 - 12 = 0

arithmetic_operation("12 * 12") ⟶ 144 // 12 * 12 = 144

arithmetic_operation("12 // 0") ⟶ -1 // 12 / 0 = -1

12. Write a function that takes the coordinates of three points in the form of a 2d array and returns the perimeter of the triangle. The given points are the vertices of a triangle on a two-dimensional plane.

Examples

perimeter([[15, 7], [5, 22], [11, 1]]) ⟶ 47.08

perimeter([[0, 0], [0, 1], [1, 0]]) ⟶ 3.42

perimeter([[-10, -10], [10, 10], [-10, 10]]) ⟶ 68.28

13. A city skyline can be represented as a 2-D list with 1s representing buildings. In the example below, the height of the tallest building is 4 (second-most right column).

[[0, 0, 0, 0, 0, 0],

[0, 0, 0, 0, 1, 0],

[0, 0, 1, 0, 1, 0],

[0, 1, 1, 1, 1, 0],

[1, 1, 1, 1, 1, 1]]

Create a function that takes a skyline (2-D list of 0's and 1's) and returns the height of the tallest skyscraper.

Examples

```
tallest_skyscraper([
  [0, 0, 0, 0],
  [0, 1, 0, 0],
  [0, 1, 1, 0],
  [1, 1, 1, 1]
]) ⟶ 3
```

```
tallest_skyscraper([
  [0, 1, 0, 0],
  [0, 1, 0, 0],
  [0, 1, 1, 0],
  [1, 1, 1, 1]
]) ⟶ 4
```

```
tallest_skyscraper([
  [0, 0, 0, 0],
  [0, 0, 0, 0],
  [1, 1, 1, 0],
  [1, 1, 1, 1]
]) ⟶ 2
```

14. A financial institution provides professional services to banks and claims charges from the customers based on the number of man-days provided. Internally, it has set a scheme to motivate and reward staff to meet and exceed targeted billable utilization and revenues by paying a bonus for each day claimed from customers in excess of a threshold target.

This quarterly scheme is calculated with a threshold target of 32 days per quarter, and the incentive payment for each billable day in excess of such threshold target is shown as follows:

Days	Bonus

0 to 32 days	Zero
33 to 40 days	SGD$325 per billable day
41 to 48 days	SGD$550 per billable day
Greater than 48 days	SGD$600 per billable day

Please note that incentive payment is calculated progressively. As an example, if an employee reached total billable days of 45 in a quarter, his/her incentive payment is computed as follows:

32*0 + 8*325 + 5*550 = 5350

Write a function to read the billable days of an employee and return the bonus he/she has obtained in that quarter.

Examples

bonus(15) ⟶ 0

bonus(37) ⟶ 1625

bonus(50) ⟶ 8200

15. A number is said to be Disarium if the sum of its digits raised to their respective positions is the number itself.

Create a function that determines whether a number is a Disarium or not.

Examples

is_disarium(75) ⟶ False

7^1 + 5^2 = 7 + 25 = 32

is_disarium(135) ⟶ True

1^1 + 3^2 + 5^3 = 1 + 9 + 125 = 135

is_disarium(544) ⟶ False

is_disarium(518) ⟶ True

is_disarium(466) ⟶ False

is_disarium(8) ⟶ True

16. In mathematics, the Fibonacci numbers, commonly denoted Fn, form a sequence, called the Fibonacci sequence, such that each number is the sum of the two preceding ones, starting from 0 and 1:

$$F_0 = 0, \quad F_1 = 1,$$

and

$$F_n = F_{n-1} + F_{n-2},$$

for n > 1

The beginning of the sequence is this: 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, ...

The function fastFib(num) returns the fibonacci number Fn, of the given num as an argument.

Examples

fib_fast(5) ⟶ 5

fib_fast(10) ⟶ 55

fib_fast(20) ⟶ 6765

fib_fast(50) ⟶ 12586269025

17. Create a function that takes a strings characters as ASCII and returns each characters hexadecimal value as a string.

Examples

convert_to_hex("hello world") ⟶ "68 65 6c 6c 6f 20 77 6f 72 6c 64"

convert_to_hex("Big Boi") ⟶ "42 69 67 20 42 6f 69"

convert_to_hex("Marty Poppinson") ⟶ "4d 61 72 74 79 20 50 6f 70 70 69 6e 73 6f 6e"

18. Someone has attempted to censor my strings by replacing every vowel with a *, l*k* th*s. Luckily, I've been able to find the vowels that were removed.

Given a censored string and a string of the censored vowels, return the original uncensored string.

Example

uncensor("Wh*r* d*d my v*w*ls g*?", "eeioeo") ⟶ "Where did my vowels go?"

uncensor("abcd", "") ⟶ "abcd"

uncensor("*PP*RC*S*", "UEAE") ⟶ "UPPERCASE"

19. Write a function that takes an IP address and returns the domain name using PTR DNS records.

Example

get_domain("8.8.8.8") ⟶ "dns.google"

get_domain("8.8.4.4") ⟶ "dns.google"

20. Create a function that takes an integer n and returns the factorial of factorials. See below examples for a better understanding:

Examples

fact_of_fact(4) ⟶ 288

4! * 3! * 2! * 1! = 288

fact_of_fact(5) ⟶ 34560

fact_of_fact(6) ⟶ 24883200

21. Create a function that takes a number n (integer greater than zero) as an argument, and returns 2 if n is odd and 8 if n is even.

You can only use the following arithmetic operators: addition of numbers +, subtraction of numbers -, multiplication of number *, division of number /, and exponentiation **.

You are not allowed to use any other methods in this challenge (i.e. no if statements, comparison operators, etc).

Examples

f(1) ⟶ 2

f(2) ⟶ 8

f(3) ➞ 2

22. Create a function that returns the majority vote in a list. A majority vote is an element that occurs > N/2 times in a list (where N is the length of the list).

Examples

majority_vote(["A", "A", "B"]) ➞ "A"

majority_vote(["A", "A", "A", "B", "C", "A"]) ➞ "A"

majority_vote(["A", "B", "B", "A", "C", "C"]) ➞ None

23. Create a function that takes a string txt and censors any word from a given list lst. The text removed must be replaced by the given character char.

Examples

censor_string("Today is a Wednesday!", ["Today", "a"], "-") ➞ "----- is - Wednesday!"

censor_string("The cow jumped over the moon.", ["cow", "over"], "*"), "The *** jumped **** the moon.")

censor_string("Why did the chicken cross the road?", ["Did", "chicken", "road"], "*") ➞ "Why *** the ******* cross the ****?"

24. In mathematics a Polydivisible Number (or magic number) is a number in a given number base with digits abcde... that has the following properties:

- Its first digit a is not 0.

- The number formed by its first two digits ab is a multiple of 2.

- The number formed by its first three digits abc is a multiple of 3.

- The number formed by its first four digits abcd is a multiple of 4.

Create a function which takes an integer n and returns True if the given number is a Polydivisible Number and False otherwise.

Examples

is_polydivisible(1232) ➞ True

1 / 1 = 1

12 / 2 = 6

123 / 3 = 41

1232 / 4 = 308

is_polydivisible(123220) ➞ False

1   / 1 = 1

12   / 2 = 6

123   / 3 = 41

1232   / 4 = 308

12322   / 5 = 2464.4   # Not a Whole Number

123220   /6 = 220536.333... # Not a Whole Number

25. Create a function that takes a list of numbers and returns the sum of all prime numbers in the list.

Examples

sum_primes([1, 2, 3, 4, 5, 6, 7, 8, 9, 10]) ➞ 17

sum_primes([2, 3, 4, 11, 20, 50, 71]) ➞ 87

sum_primes([]) ➞ None

26. You are given two strings s and t. String t is generated by randomly shuffling string s and then adding one more letter at a random position. Return the letter that was added to t.

Examples

find_the_difference("abcd", "abcde") ➞ "e"

find_the_difference("", "y") ➞ "y"

find_the_difference("ae", "aea") ➞ "a"

27. Given a function that accepts unlimited arguments, check and count how many data types are in those arguments. Finally return the total in a list.

List order is:

[int, str, bool, list, tuple, dictionary]

Examples

count_datatypes(1, 45, "Hi", False) ➞ [2, 1, 1, 0, 0, 0]

count_datatypes([10, 20], ("t", "Ok"), 2, 3, 1) ➞ [3, 0, 0, 1, 1, 0]

count_datatypes("Hello", "Bye", True, True, False, {"1": "One", "2": "Two"}, [1, 3], {"Brayan": 18}, 25, 23) ➞ [2, 2, 3, 1, 0, 2]

count_datatypes(4, 21, ("ES", "EN"), ("a", "b"), False, [1, 2, 3], [4, 5, 6]) ➞ [2, 0, 1, 2, 2, 0]

28. A Fibonacci string is a precedence of the Fibonacci series. It works with any two characters of the English alphabet (as opposed to the numbers 0 and 1 in the Fibonacci series) as the initial items and concatenates them together as it progresses in a similar fashion as the Fibonacci series.

Examples

fib_str(3, ["j", "h"]) ➞ "j, h, hj"

fib_str(5, ["e", "a"]) ➞ "e, a, ae, aea, aeaae"

fib_str(6, ["n", "k"]) ➞ "n, k, kn, knk, knkkn, knkknknk"

29. Given an integer between 0 and 26, make a variable (self.answer). That variable would be assigned to a string in this format:

"nines:your answer, threes:your answer, ones:your answer"

You need to find out how many ones, threes, and nines it would at least take for the number of each

to add up to the given integer when multiplied by one, three or nine (depends).

Examples

ones_threes_nines(10) ➞ "nines:1, threes:0, ones:1"

ones_threes_nines(15) ➞ "nines:1, threes:2, ones:0"

ones_threes_nines(22) ➞ "nines:2, threes:1, ones:1"

30. The Fibonacci sequence is a classic use case for recursive functions since the value of the sequence at a given index is dependent on the sum of the last two values. However, the recursion tree created by solving the Fibonacci sequence recursively can grow quite fast. Therefore it can be important to think about the implications of running a function recursively. Depending on the size of n needed and the capabilities of the system in question you might want to take a different approach.

Write a non-recursive function that takes an integer n and returns the Fibonacci sequence's value at index n.

Examples

fib(6) ⟶ 8

0 + 1 = 1, 1 + 1 = 2, 1 + 2 = 3, 2 + 3 = 5, 3 + 5 = 8

fib(1) ⟶ 1

fib(2) ⟶ 1

31. Write a function that counts how many concentric layers a rug.

Examples

```
count_layers([
  "AAAA",
  "ABBA",
  "AAAA"
]) ⟶ 2

count_layers([
  "AAAAAAAAA",
  "ABBBBBBBA",
  "ABBAAABBA",
  "ABBBBBBBA",
  "AAAAAAAAA"
]) ⟶ 3

count_layers([
  "AAAAAAAAAAA",
  "AABBBBBBBAA",
  "AABCCCCCBAA",
  "AABCAAACBAA",
  "AABCADACBAA",
  "AABCAAACBAA",
  "AABCCCCCBAA",
```

"AABBBBBBBAA",

"AAAAAAAAAAA"

]) ➞ 5

32. There are many different styles of music and many albums exhibit multiple styles. Create a function that takes a list of musical styles from albums and returns how many styles are unique.

Examples

unique_styles([

"Dub,Dancehall",

"Industrial,Heavy Metal",

"Techno,Dubstep",

"Synth-pop,Euro-Disco",

"Industrial,Techno,Minimal"

]) ➞ 9

unique_styles([

"Soul",

"House,Folk",

"Trance,Downtempo,Big Beat,House",

"Deep House",

"Soul"

]) ➞ 7

33. Create a function that finds a target number in a list of prime numbers. Implement a binary search algorithm in your function. The target number will be from 2 through 97. If the target is prime then return "yes" else return "no".

Examples

primes = [2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 67, 71, 73, 79, 83, 89, 97]

is_prime(primes, 3) ➞ "yes"

is_prime(primes, 4) ➞ "no"

is_prime(primes, 67) ➞ "yes"

is_prime(primes, 36) ➞ "no"

34. Create a function that takes in n, a, b and returns the number of positive values raised to the nth power that lie in the range [a, b], inclusive.

Examples

power_ranger(2, 49, 65) ➞ 2

2 squares (n^2) lie between 49 and 65, 49 (7^2) and 64 (8^2)

power_ranger(3, 1, 27) ➞ 3

3 cubes (n^3) lie between 1 and 27, 1 (1^3), 8 (2^3) and 27 (3^3)

power_ranger(10, 1, 5) ➞ 1

1 value raised to the 10th power lies between 1 and 5, 1 (1^10)

power_ranger(5, 31, 33) ➞ 1

power_ranger(4, 250, 1300) ➞ 3

35. Given a number, return the difference between the maximum and minimum numbers that can be formed when the digits are rearranged.

Examples

rearranged_difference(972882) ➞ 760833

988722 - 227889 = 760833

rearranged_difference(3320707) ➞ 7709823

7733200 - 23377 = 7709823

rearranged_difference(90010) ➞ 90981

91000 - 19 = 90981

36. Given a sentence as txt, return True if any two adjacent words have this property: One word ends with a vowel, while the word immediately after begins with a vowel (a e i o u).

Examples

vowel_links("a very large appliance") ➞ True

vowel_links("go to edabit") ➞ True

vowel_links("an open fire") ➞ False

vowel_links("a sudden applause") ➞ False

37. You are given three inputs: a string, one letter, and a second letter.

Write a function that returns True if every instance of the first letter occurs before every instance of the second letter.

Examples

first_before_second("a rabbit jumps joyfully", "a", "j") ➞ True

Every instance of "a" occurs before every instance of "j".

first_before_second("knaves knew about waterfalls", "k", "w") ➞ True

first_before_second("happy birthday", "a", "y") ➞ False

The "a" in "birthday" occurs after the "y" in "happy".

first_before_second("precarious kangaroos", "k", "a") ➞ False

38. Create a function that returns the characters from a list or string r on odd or even positions, depending on the specifier s. The specifier will be "odd" for items on odd positions (1, 3, 5, ...) and "even" for items on even positions (2, 4, 6, ...).

Examples

char_at_pos([2, 4, 6, 8, 10], "even") ➞ [4, 8]

4 & 8 occupy the 2nd & 4th positions

char_at_pos("EDABIT", "odd") ➞ "EAI"

"E", "A" and "I" occupy the 1st, 3rd and 5th positions

char_at_pos(["A", "R", "B", "I", "T", "R", "A", "R", "I", "L", "Y"], "odd") ➞ ["A", "B", "T", "A", "I", "Y"]

39. Write a function that returns the greatest common divisor of all list elements. If the greatest common divisor is 1, return 1.

Examples

GCD([10, 20, 40]) ⟶ 10

GCD([1, 2, 3, 100]) ⟶ 1

GCD([1024, 192, 2048, 512]) ⟶ 64

40. A number/string is a palindrome if the digits/characters are the same when read both forward and backward. Examples include "racecar" and 12321. Given a positive number n, check if n or the binary representation of n is palindromic. Return the following:

- "Decimal only." if only n is a palindrome.
- "Binary only." if only the binary representation of n is a palindrome.
- "Decimal and binary." if both are palindromes.
- "Neither!" if neither are palindromes.

Examples

palindrome_type(1306031) ⟶ "Decimal only."

decimal = 1306031

binary = "10011111010110101111"

palindrome_type(427787) ⟶ "Binary only."

decimal = 427787

binary = "1101000011100001011"

palindrome_type(313) ⟶ "Decimal and binary."

decimal = 313

binary = 100111001

palindrome_type(934) ⟶ "Neither!"

decimal = 934

binary = "1110100110"

41. YouTube offers different playback speed options for users. This allows users to increase or decrease the speed of the video content. Given the actual duration and playback speed of the video, calculate the playback duration of the video.

Examples

playback_duration("00:30:00", 2) ⟶ "00:15:00"

playback_duration("01:20:00", 1.5) ⟶ "00:53:20"

playback_duration("51:20:09", 0.5) ⟶ "102:40:18"

42. We needs your help 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.

Given the total volume m of the building, can you find the number of cubes n required for the building?

In other words, you have to return an integer n such that:

n^3 + (n-1)^3 + ... + 1^3 == m

Return None if there is no such number.

Examples

pile_of_cubes(1071225) ⟶ 45

pile_of_cubes(4183059834009) ⟶ 2022

pile_of_cubes(16) ⟶ None

43. A fulcrum of a list is an integer such that all elements to the left of it and all elements to the right of it sum to the same value. Write a function that finds the fulcrum of a list.

To illustrate:

find_fulcrum([3, 1, 5, 2, 4, 6, -1]) ⟶ 2

// Since [3, 1, 5] and [4, 6, -1] both sum to 9

Examples

find_fulcrum([1, 2, 4, 9, 10, -10, -9, 3]) ➞ 4

find_fulcrum([9, 1, 9]) ➞ 1

find_fulcrum([7, -1, 0, -1, 1, 1, 2, 3]) ➞ 0

find_fulcrum([8, 8, 8, 8]) ➞ -1

44. Given a list of integers representing the color of each sock, determine how many pairs of socks with matching colors there are. For example, there are 7 socks with colors [1, 2, 1, 2, 1, 3, 2]. There is one pair of color 1 and one of color 2. There are three odd socks left, one of each color. The number of pairs is 2.

Create a function that returns an integer representing the number of matching pairs of socks that are available.

Examples

sock_merchant([10, 20, 20, 10, 10, 30, 50, 10, 20]) ➞ 3

sock_merchant([50, 20, 30, 90, 30, 20, 50, 20, 90]) ➞ 4

sock_merchant([]) ➞ 0

45. Create a function that takes a string containing integers as well as other characters and return the sum of the negative integers only.

Examples

negative_sum("-12 13%14&-11") ➞ -23

-12 + -11 = -23

negative_sum("22 13%14&-11-22 13 12") ➞ -33

-11 + -22 = -33

46. Create a function that takes the width, height and character and returns a picture frame as a 2D list.

Examples

get_frame(4, 5, "#") ➞ [

["####"],

```
 ["#  #"],
 ["#  #"],
 ["#  #"],
 ["####"]
]
# Frame is 4 characters wide and 5 characters tall.


get_frame(10, 3, "*") ⟶ [
 ["**********"],
 ["*       *"],
 ["**********"]
]
# Frame is 10 characters and wide and 3 characters tall.


get_frame(2, 5, "0") ⟶ "invalid"
# Frame's width is not more than 2.
```

47. Write three functions:

1. boolean_and
2. boolean_or
3. boolean_xor

These functions should evaluate a list of True and False values, starting from the leftmost element and evaluating pairwise.

Examples

```
boolean_and([True, True, False, True]) ⟶ False
# [True, True, False, True] => [True, False, True] => [False, True] => False
boolean_or([True, True, False, False]) ⟶ True
# [True, True, False, True] => [True, False, False] => [True, False] => True
```

boolean_xor([True, True, False, False]) ⟶ False

[True, True, False, False] => [False, False, False] => [False, False] => False

48. Create a function that creates a box based on dimension n.

Examples

make_box(5) ⟶ [

"#####",

"#   #",

"#   #",

"#   #",

"#####"

]

make_box(3) ⟶ [

"###",

"# #",

"###"

]

make_box(2) ⟶ [

"##",

"##"

]

make_box(1) ⟶ [

"#"

]

49. Given a common phrase, return False if any individual word in the phrase contains duplicate letters. Return True otherwise.

Examples

no_duplicate_letters("Fortune favours the bold.") ⟶ True

no_duplicate_letters("You can lead a horse to water, but you can't make him drink.") ⟶ True

no_duplicate_letters("Look before you leap.") ⟶ False

Duplicate letters in "Look" and "before".

no_duplicate_letters("An apple a day keeps the doctor away.") ⟶ False

Duplicate letters in "apple", "keeps", "doctor", and "away".

50. Write a regular expression that will match the states that voted yes to President Trump's impeachment. You must use RegEx positive lookahead.

Example

txt = "Texas = no, California = yes, Florida = yes, Michigan = no"

pattern = "yourregularexpressionhere"

re.findall(pattern, txt) ⟶ ["California", "Florida"]