Advanced Programming Assignment

# Advanced Programming Assignment 1

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.

Examples

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:

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

2. Odd positions will be represented as 1 and 0 otherwise.

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

4. 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]

# Advanced Programming Assignment 2

1. 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

2. 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"

3. 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

4. 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

5. 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" }

# Advanced Programming Assignment 3

1. 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

2. 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

3. 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

4. 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

5. 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

# Advanced Programming Assignment 4

1. 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:

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

2. 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"

3. 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"

4. 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"

5. 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

# Advanced Programming Assignment 5

1. 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

2. 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

3. 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 \*\*\*\*?"

4. 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

5. 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

# Advanced Programming Assignment 6

1. 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"

2. 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]

3. 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"

4. 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"

5. 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

# Advanced Programming Assignment 7

1. 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

2. 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

3. 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"

4. 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

5. 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

# Advanced Programming Assignment 8

1. 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

2. 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

3. 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"]

4. 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

5. 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 = "100111110110110101111"

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"

# Advanced Programming Assignment 9

1. 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"

2. 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

3. 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

4. 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

5. 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

# Advanced Programming Assignment 10

1. 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.

2. 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

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

Examples

make\_box(5) ➞ [

"#####",

"# #",

"# #",

"# #",

"#####"

]

make\_box(3) ➞ [

"###",

"# #",

"###"

]

make\_box(2) ➞ [

"##",

"##"

]

make\_box(1) ➞ [

"#"

]

4. 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".

5. 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"]

# Advanced Programming Assignment 11

1. Create a function that takes a list and returns a new list containing only prime numbers.

Examples

filter\_primes([7, 9, 3, 9, 10, 11, 27]) ➞ [7, 3, 11]

filter\_primes([10007, 1009, 1007, 27, 147, 77, 1001, 70]) ➞ [10007, 1009]

filter\_primes([1009, 10, 10, 10, 3, 33, 9, 4, 1, 61, 63, 69, 1087, 1091, 1093, 1097]) ➞ [1009, 3, 61, 1087, 1091, 1093, 1097]

2. Once a water balloon pops, is soaks the area around it. The ground gets drier the further away you travel from the balloon.

The effect of a water balloon popping can be modeled using a list. Create a function that takes a list which takes the pre-pop state and returns the state after the balloon is popped. The pre-pop state will contain at most a single balloon, whose size is represented by the only non-zero element.

Examples

pop([0, 0, 0, 0, 4, 0, 0, 0, 0]) ➞ [0, 1, 2, 3, 4, 3, 2, 1, 0]

pop([0, 0, 0, 3, 0, 0, 0]) ➞ [0, 1, 2, 3, 2, 1, 0]

pop([0, 0, 2, 0, 0]) ➞ [0, 1, 2, 1, 0]

pop([0]) ➞ [0]

3. "Loves me, loves me not" is a traditional game in which a person plucks off all the petals of a flower one by one, saying the phrase "Loves me" and "Loves me not" when determining whether the one that they love, loves them back.

Given a number of petals, return a string which repeats the phrases "Loves me" and "Loves me not" for every alternating petal, and return the last phrase in all caps. Remember to put a comma and space between phrases.

Examples

loves\_me(3) ➞ "Loves me, Loves me not, LOVES ME"

loves\_me(6) ➞ "Loves me, Loves me not, Loves me, Loves me not, Loves me, LOVES ME NOT"

loves\_me(1) ➞ "LOVES ME"

4. Write a function that sorts each string in a list by the letter in alphabetic ascending order (a-z).

Examples

sort\_by\_letter(["932c", "832u32", "2344b"])

➞ ["2344b", "932c", "832u32"]

sort\_by\_letter(["99a", "78b", "c2345", "11d"])

➞ ["99a", "78b", "c2345", "11d"]

sort\_by\_letter(["572z", "5y5", "304q2"])

➞ ["304q2", "5y5", "572z"]

sort\_by\_letter([])

➞ []

5. There are three cups on a table, at positions A, B, and C. At the start, there is a ball hidden under the cup at position B.

However, I perform several swaps on the cups, which is notated as two letters. For example, if I swap the cups at positions A and B, I could notate this as AB or BA.

Create a function that returns the letter position that the ball is at, once I finish swapping the cups. The swaps will be given to you as a list.

Example

cup\_swapping(["AB", "CA", "AB"]) ➞ "C"

# Ball begins at position B.

# Cups A and B swap, so the ball is at position A.

# Cups C and A swap, so the ball is at position C.

# Cups A and B swap, but the ball is at position C, so it doesn't move.

# Advanced Programming Assignment 12

1. For this challenge, forget how to add two numbers together. The best explanation on what to do for this function is this meme:

Examples

meme\_sum(26, 39) ➞ 515

# 2+3 = 5, 6+9 = 15

# 26 + 39 = 515

meme\_sum(122, 81) ➞ 1103

# 1+0 = 1, 2+8 = 10, 2+1 = 3

# 122 + 81 = 1103

meme\_sum(1222, 30277) ➞ 31499

2. Given an integer, create a function that returns the next prime. If the number is prime, return the number itself.

Examples

next\_prime(12) ➞ 13

next\_prime(24) ➞ 29

next\_prime(11) ➞ 11

# 11 is a prime, so we return the number itself.

3. If a person traveled up a hill for 18mins at 20mph and then traveled back down the same path at 60mph then their average speed traveled was 30mph.

Write a function that returns the average speed traveled given an uphill time, uphill rate and a downhill rate. Uphill time is given in minutes. Return the rate as an integer (mph). No rounding is necessary.

Examples

ave\_spd(18, 20, 60) ➞ 30

ave\_spd(30, 10, 30) ➞ 15

ave\_spd(30, 8, 24) ➞ 12

4. The Kempner Function, applied to a composite number, permits to find the smallest integer greater than zero whose factorial is exactly divided by the number.

kempner(6) ➞ 3

1! = 1 % 6 > 0

2! = 2 % 6 > 0

3! = 6 % 6 === 0

kempner(10) ➞ 5

1! = 1 % 10 > 0

2! = 2 % 10 > 0

3! = 6 % 10 > 0

4! = 24 % 10 > 0

5! = 120 % 10 === 0

A Kempner Function applied to a prime will always return the prime itself.

kempner(2) ➞ 2

kempner(5) ➞ 5

Given an integer n, implement a Kempner Function.

Examples

kempner(6) ➞ 3

kempner(10) ➞ 5

kempner(2) ➞ 2

5. You work in a factory, and your job is to take items from a conveyor belt and pack them into boxes. Each box can hold a maximum of 10 kgs. Given a list containing the weight (in kg) of each item, how many boxes would you need to pack all of the items?

Example

boxes([2, 1, 2, 5, 4, 3, 6, 1, 1, 9, 3, 2]) ➞ 5

# Box 1 = [2, 1, 2, 5] (10kg)

# Box 2 = [4, 3] (7kg)

# Box 3 = [6, 1, 1] (8kg)

# Box 4 = [9] (9kg)

# Box 5 = [3, 2] (5kg)

# Advanced Programming Assignment 13

1. Create a function that takes a list and string. The function should remove the letters in the string from the list, and return the list.

Examples

remove\_letters(["s", "t", "r", "i", "n", "g", "w"], "string") ➞ ["w"]

remove\_letters(["b", "b", "l", "l", "g", "n", "o", "a", "w"], "balloon") ➞ ["b", "g", "w"]

remove\_letters(["d", "b", "t", "e", "a", "i"], "edabit") ➞ []

2. A block sequence in three dimensions. We can write a formula for this one:

Create a function that takes a number (step) as an argument and returns the amount of blocks in that step.

Examples

blocks(1) ➞ 5

blocks(5) ➞ 39

blocks(2) ➞ 12

3. Create a function that subtracts one positive integer from another, without using any arithmetic operators such as -, %, /, +, etc.

Examples

my\_sub(5, 9) ➞ 4

my\_sub(10, 30) ➞ 20

my\_sub(0, 0) ➞ 0

4. Create a function that takes a string containing money in dollars and pounds sterling (seperated by comma) and returns the sum of dollar bills only, as an integer.

For the input string:

- Each amount is prefixed by the currency symbol: $ for dollars and £ for pounds.

- Thousands are represented by the suffix k.

i.e. $4k = $4,000 and £40k = £40,000

Examples

add\_bill("d20,p40,p60,d50") ➞ 20 + 50 = 70

add\_bill("p30,d20,p60,d150,p360") ➞ 20 + 150 = 170

add\_bill("p30,d2k,p60,d200,p360") ➞ 2 \* 1000 + 200 = 2200

5. Create a function that flips a horizontal list into a vertical list, and a vertical list into a horizontal list.

In other words, take an 1 x n list (1 row + n columns) and flip it into a n x 1 list (n rows and 1 column), and vice versa.

Examples

flip\_list([1, 2, 3, 4]) ➞ [[1], [2], [3], [4]]

# Take a horizontal list and flip it vertical.

flip\_list([[5], [6], [9]]) ➞ [5, 6, 9]

# Take a vertical list and flip it horizontal.

flip\_list([]) ➞ []

# Advanced Programming Assignment 14

1. Given a list of numbers, create a function that removes 25% from every number in the list except the smallest number, and adds the total amount removed to the smallest number.

Examples

show\_the\_love([4, 1, 4]) ➞ [3, 3, 3]

show\_the\_love([16, 10, 8]) ➞ [12, 7.5, 14.5]

show\_the\_love([2, 100]) ➞ [27, 75]

2. Create a function that takes in two words as input and returns a list of three elements, in the following order:

1.Shared letters between two words.

2.Letters unique to word 1.

3.Letters unique to word 2.

Each element should have unique letters, and have each letter be alphabetically sorted.

Examples

letters("sharp", "soap") ➞ ["aps", "hr", "o"]

letters("board", "bored") ➞ ["bdor", "a", "e"]

letters("happiness", "envelope") ➞ ["enp", "ahis", "lov"]

letters("kerfuffle", "fluffy") ➞ ["flu", "ekr", "y"]

# Even with multiple matching letters (e.g. 3 f's), there should

# only exist a single "f" in your first element.

letters("match", "ham") ➞ ["ahm", "ct", ""]

# "ham" does not contain any letters that are not found already

# in "match".

3. Write a function that pairs the first number in an array with the last, the second number with the second to last, etc.

Examples

pairs([1, 2, 3, 4, 5, 6, 7]) ➞ [[1, 7], [2, 6], [3, 5], [4, 4]]

pairs([1, 2, 3, 4, 5, 6]) ➞ [[1, 6], [2, 5], [3, 4]]

pairs([5, 9, 8, 1, 2]) ➞ [[5, 2], [9, 1], [8, 8]]

pairs([]) ➞ []

4. Write a function that adds two numbers. The catch, however, is that the numbers will be strings.

Examples

add\_str\_nums("4", "5") ➞ "9"

add\_str\_nums("abcdefg", "3") ➞ "-1"

add\_str\_nums("1", "") ➞ "1"

add\_str\_nums("1874682736267235927359283579235789257", "32652983572985729") ➞ "1874682736267235927391936562808774986"

5. lPaeesh le pemu mnxit ehess rtnisg! Oh, sorry, that was supposed to say: Please help me unmix these strings!

Somehow my strings have all become mixed up; every pair of characters has been swapped. Help me undo this so I can understand my strings again.

Examples

unmix("123456") ➞ "214365"

unmix("hTsii s aimex dpus rtni.g") ➞ "This is a mixed up string."

unmix("badce") ➞ "abcde"

# Advanced Programming Assignment 15

1. Write a function that returns True if a given name can generate an array of words.

Examples

anagram("Justin Bieber", ["injures", "ebb", "it"]) ➞ True

anagram("Natalie Portman", ["ornamental", "pita"]) ➞ True

anagram("Chris Pratt", ["chirps", "rat"]) ➞ False

# Not all letters are used

anagram("Jeff Goldblum", ["jog", "meld", "bluffs"]) ➞ False

# "s" does not exist in the original name

2. Given an array of users, each defined by an object with the following properties: name, score, reputation create a function that sorts the array to form the correct leaderboard.

The leaderboard takes into consideration the score of each user of course, but an emphasis is put on their reputation in the community, so to get the trueScore, you should add the reputation multiplied by 2 to the score.

Once you know the trueScore of each user, sort the array according to it in descending order.

Examples

leaderboards([

{ "name": "a", "score": 100, "reputation": 20 },

{ "name": "b", "score": 90, "reputation": 40 },

{ "name": "c", "score": 115, "reputation": 30 },

]) ➞ [

{ "name": "c", "score": 115, "reputation": 30 }, # trueScore = 175

{ "name": "b", "score": 90, "reputation": 40 }, # trueScore = 170

{ "name": "a", "score": 100, "reputation": 20 } # trueScore = 140

]

3. Create a function that, given a phrase and a number of letters guessed, returns a string with hyphens - for every letter of the phrase not guessed, and each letter guessed in place.

Examples

hangman("helicopter", ["o", "e", "s"]) ➞ "-e---o--e-"

hangman("tree", ["r", "t", "e"]) ➞ "tree"

hangman("Python rules", ["a", "n", "p", "r", "z"]) ➞ "P----n r----"

hangman("He"s a very naughty boy!", ["e", "a", "y"]) ➞ "-e"- a -e-y -a----y –y!"

4. The Collatz sequence is as follows:

- Start with some given integer n.

- If it is even, the next number will be n divided by 2.

- If it is odd, multiply it by 3 and add 1 to make the next number.

- The sequence stops when it reaches 1.

According to the Collatz conjecture, it will always reach 1. If that's true, you can construct a finite sequence following the aforementioned method for any given integer.

Write a function that takes in an integer n and returns the highest integer in the corresponding Collatz sequence.

Examples

max\_collatz(10) ➞ 16

# Collatz sequence: 10, 5, 16, 8, 4, 2, 1

max\_collatz(32) ➞ 32

# Collatz sequence: 32, 16, 8, 4, 2, 1

max\_collatz(85) ➞ 256

# Collatz sequence: 85, 256, 128, 64, 32, 16, 8, 4, 2, 1

5. Write a function that sorts a list of integers by their digit length in descending order, then settles ties by sorting numbers with the same digit length in ascending order.

Examples

digit\_sort([77, 23, 5, 7, 101])

➞ [101, 23, 77, 5, 7]

digit\_sort([1, 5, 9, 2, 789, 563, 444])

➞ [444, 563, 789, 1, 2, 5, 9]

digit\_sort([53219, 3772, 564, 32, 1])

➞ [53219, 3772, 564, 32, 1]

# Advanced Programming Assignment 16

1. Rondo Form is a type of musical structure, in which there is a recurring theme/refrain, notated as A. Here are the rules for valid rondo forms:

- Rondo forms always start and end with an A section.

- In between the A sections, there should be contrasting sections notated as B, then C, then D, etc... No letter should be skipped.

- There shouldn't be any repeats in the sequence (such as ABBACCA).

Create a function which validates whether a given string is a valid Rondo Form.

Examples

valid\_rondo("ABACADAEAFAGAHAIAJA") ➞ True

valid\_rondo("ABA") ➞ True

valid\_rondo("ABBACCA") ➞ False

valid\_rondo("ACAC") ➞ False

valid\_rondo("A") ➞ False

2. Create a function that returns the whole of the first sentence which contains a specific word. Include the full stop at the end of the sentence.

Examples

txt = "I have a cat. I have a mat. Things are going swell."

sentence\_searcher(txt, "have") ➞ "I have a cat."

sentence\_searcher(txt, "MAT") ➞ "I have a mat."

sentence\_searcher(txt, "things") ➞ "Things are going swell."

sentence\_searcher(txt, "flat") ➞ ""

3. Given a number, find the "round "of each digit of the number. An integer is called "round" if all its digits except the leftmost (most significant) are equal to zero.

- Round numbers: 4000, 1, 9, 800, 90

- Not round numbers: 110, 707, 222, 1001

Create a function that takes a number and returns the "round" of each digit (except if the digit is zero) as a string. Check out the following examples for more clarification.

Examples

sum\_round(101) ➞ "1 100"

sum\_round(1234) ➞ "4 30 200 1000"

sum\_round(54210) ➞ "10 200 4000 50000"

4. Your task, is to create N x N multiplication table, of size n provided in parameter.

For example, when n is 5, the multiplication table is:

- 1, 2, 3, 4, 5

- 2, 4, 6, 8, 10

- 3, 6, 9, 12, 15

- 4, 8, 12, 16, 20

- 5, 10, 15, 20, 25

This example will result in:

[[1, 2, 3, 4, 5], [2, 4, 6, 8, 10], [3, 6, 9, 12, 15], [4, 8, 12, 16, 20], [5, 10, 15, 20, 25]]

Examples

multiplication\_table(1) ➞ [[1]]

multiplication\_table(3) ➞ [[1, 2, 3], [2, 4, 6], [3, 6, 9]]

5. Create a function that returns True if two lines rhyme and False otherwise. For the purposes of this exercise, two lines rhyme if the last word from each sentence contains the same vowels.

Examples

does\_rhyme("Sam I am!", "Green eggs and ham.") ➞ True

does\_rhyme("Sam I am!", "Green eggs and HAM.") ➞ True

# Capitalization and punctuation should not matter.

does\_rhyme("You are off to the races", "a splendid day.") ➞ False

does\_rhyme("and frequently do?", "you gotta move.") ➞ False

# Advanced Programming Assignment 17

1. Create a function that transposes a 2D matrix.

Examples

transpose\_matrix([

[1, 1, 1],

[2, 2, 2],

[3, 3, 3]

]) ➞ [

[1, 2, 3],

[1, 2, 3],

[1, 2, 3]

]

transpose\_matrix([

[5, 5],

[6, 7],

[9, 1]

]) ➞ [

[5, 6, 9],

[5, 7, 1]

]

2. Create a function that determines whether a string is a valid hex code.

A hex code must begin with a pound key # and is exactly 6 characters in length. Each character must be a digit from 0-9 or an alphabetic character from A-F. All alphabetic characters may be uppercase or lowercase.

Examples

is\_valid\_hex\_code("#CD5C5C") ➞ True

is\_valid\_hex\_code("#EAECEE") ➞ True

is\_valid\_hex\_code("#eaecee") ➞ True

is\_valid\_hex\_code("#CD5C58C") ➞ False

# Length exceeds 6

is\_valid\_hex\_code("#CD5C5Z") ➞ False

# Not all alphabetic characters in A-F

is\_valid\_hex\_code("#CD5C&C") ➞ False

# Contains unacceptable character

is\_valid\_hex\_code("CD5C5C") ➞ False

# Missing #

3. Given a list of math equations (given as strings), return the percentage of correct answers as a string. Round to the nearest whole number.

Examples

mark\_maths(["2+2=4", "3+2=5", "10-3=3", "5+5=10"]) ➞ "75%"

mark\_maths(["1-2=-2"]), "0%"

mark\_maths(["2+3=5", "4+4=9", "3-1=2"]) ➞ "67%"

4. There are two players, Alice and Bob, each with a 3-by-3 grid. A referee tells Alice to fill out one particular row in the grid (say the second row) by putting either a 1 or a 0 in each box, such that the sum of the numbers in that row is odd. The referee tells Bob to fill out one column in the grid (say the first column) by putting either a 1 or a 0 in each box, such that the sum of the numbers in that column is even.

Alice and Bob win the game if Alice’s numbers give an odd sum, Bob’s give an even sum, and (most important) they’ve each written down the same number in the one square where their row and column intersect.

Examples

magic\_square\_game([2, "100"], [1, "101"]) ➞ False

magic\_square\_game([2, "001"], [1, "101"]) ➞ True

magic\_square\_game([3, "111"], [2, "011"]) ➞ True

magic\_square\_game([1, "010"], [3, "101"]) ➞ False

# Two lists, Alice [row, "her choice"], Bob [column, "his choice"]

5. From point A, an object is moving towards point B at constant velocity va (in km/hr). From point B, another object is moving towards point A at constant velocity vb (in km/hr). Knowing this and the distance between point A and B (in km), write a function that returns how much time passes until both objects meet.

Format the output like this:

"2h 23min 34s"

Examples

lets\_meet(100, 10, 30) ➞ "2h 30min 0s"

lets\_meet(280, 70, 80) ➞ "1h 52min 0s"

lets\_meet(90, 75, 65) ➞ "0h 38min 34s"

# Advanced Programming Assignment 18

1. A robot has been given a list of movement instructions. Each instruction is either left, right, up or down, followed by a distance to move. The robot starts at [0, 0]. You want to calculate where the robot will end up and return its final position as a list.

To illustrate, if the robot is given the following instructions:

["right 10", "up 50", "left 30", "down 10"]

It will end up 20 left and 40 up from where it started, so we return [-20, 40].

Examples

track\_robot(["right 10", "up 50", "left 30", "down 10"]) ➞ [-20, 40]

track\_robot([]) ➞ [0, 0]

// If there are no instructions, the robot doesn't move.

track\_robot(["right 100", "right 100", "up 500", "up 10000"]) ➞ [200, 10500]

2. Write a function that will return the longest word in a sentence. In cases where more than one word is found, return the first one.

Examples

find\_longest("A thing of beauty is a joy forever.") ➞ "forever"

find\_longest("Forgetfulness is by all means powerless!") ➞ "forgetfulness"

find\_longest("\"Strengths\" is the longest and most commonly used word that contains only a single vowel.") ➞ "strengths"

3. Create a function to check if a candidate is qualified in an imaginary coding interview of an imaginary tech startup.

The criteria for a candidate to be qualified in the coding interview is:

1. The candidate should have complete all the questions.

2. The maximum time given to complete the interview is 120 minutes.

3. The maximum time given for very easy questions is 5 minutes each.

4. The maximum time given for easy questions is 10 minutes each.

5. The maximum time given for medium questions is 15 minutes each.

6. The maximum time given for hard questions is 20 minutes each.

If all the above conditions are satisfied, return "qualified", else return "disqualified".

You will be given a list of time taken by a candidate to solve a particular question and the total time taken by the candidate to complete the interview.

Given a list , in a true condition will always be in the format [very easy, very easy, easy, easy, medium, medium, hard, hard].

The maximum time to complete the interview includes a buffer time of 20 minutes.

Examples

interview([5, 5, 10, 10, 15, 15, 20, 20], 120) ➞ "qualified"

interview([2, 3, 8, 6, 5, 12, 10, 18], 64) ➞ "qualified"

interview([5, 5, 10, 10, 25, 15, 20, 20], 120) ➞ "disqualified"

# Exceeded the time limit for a medium question.

interview([5, 5, 10, 10, 15, 15, 20], 120) ➞ "disqualified"

# Did not complete all the questions.

interview([5, 5, 10, 10, 15, 15, 20, 20], 130) ➞ "disqualified"

# Solved all the questions in their respected time limits but exceeded the total time limit of the interview.

4. Write a function that divides a list into chunks of size n, where n is the length of each chunk.

Examples

chunkify([2, 3, 4, 5], 2) ➞ [[2, 3], [4, 5]]

chunkify([2, 3, 4, 5, 6], 2) ➞ [[2, 3], [4, 5], [6]]

chunkify([2, 3, 4, 5, 6, 7], 3) ➞ [[2, 3, 4], [5, 6, 7]]

chunkify([2, 3, 4, 5, 6, 7], 1) ➞ [[2], [3], [4], [5], [6], [7]]

chunkify([2, 3, 4, 5, 6, 7], 7) ➞ [[2, 3, 4, 5, 6, 7]]

5. You are given a list of strings consisting of grocery items, with prices in parentheses. Return a list of prices in float format.

Examples

get\_prices(["salad ($4.99)"]) ➞ [4.99]

get\_prices([

"artichokes ($1.99)",

"rotiserrie chicken ($5.99)",

"gum ($0.75)"

])

➞ [1.99, 5.99, 0.75]

get\_prices([

"ice cream ($5.99)",

"banana ($0.20)",

"sandwich ($8.50)",

"soup ($1.99)"

])

➞ [5.99, 0.2, 8.50, 1.99]

# Advanced Programming Assignment 19

1. Create a checker board generator, which takes as inputs n and 2 elements to generate an n x n checkerboard with those two elements as alternating squares.

Examples

checker\_board(2, 7, 6) ➞ [

[7, 6],

[6, 7]

]

checker\_board(3, "A", "B") ➞ [

["A", "B", "A"],

["B", "A", "B"],

["A", "B", "A"]

]

checker\_board(4, "c", "d") ➞ [

["c", "d", "c", "d"],

["d", "c", "d", "c"],

["c", "d", "c", "d"],

["d", "c", "d", "c"]

]

checker\_board(4, "c", "c") ➞ "invalid"

2. A string is an almost-palindrome if, by changing only one character, you can make it a palindrome. Create a function that returns True if a string is an almost-palindrome and False otherwise.

Examples

almost\_palindrome("abcdcbg") ➞ True

# Transformed to "abcdcba" by changing "g" to "a".

almost\_palindrome("abccia") ➞ True

# Transformed to "abccba" by changing "i" to "b".

almost\_palindrome("abcdaaa") ➞ False

# Can't be transformed to a palindrome in exactly 1 turn.

almost\_palindrome("1234312") ➞ False

3. Create a function that finds how many prime numbers there are, up to the given integer.

Examples

prime\_numbers(10) ➞ 4

# 2, 3, 5 and 7

prime\_numbers(20) ➞ 8

# 2, 3, 5, 7, 11, 13, 17 and 19

prime\_numbers(30) ➞ 10

# 2, 3, 5, 7, 11, 13, 17, 19, 23 and 29

4. If today was Monday, in two days, it would be Wednesday.

Create a function that takes in a list of days as input and the number of days to increment by. Return a list of days after n number of days has passed.

Examples

after\_n\_days(["Thursday", "Monday"], 4) ➞ ["Monday", "Friday"]

after\_n\_days(["Sunday", "Sunday", "Sunday"], 1) ➞ ["Monday", "Monday", "Monday"]

after\_n\_days(["Monday", "Tuesday", "Friday"], 1) ➞ ["Tuesday", "Wednesday", "Saturday"]

5. You are in the process of creating a chat application and want to add an anonymous name feature. This anonymous name feature will create an alias that consists of two capitalized words beginning with the same letter as the users first name.

Create a function that determines if the list of users is mapped to a list of anonymous names correctly.

Examples

is\_correct\_aliases(["Adrian M.", "Harriet S.", "Mandy T."], ["Amazing Artichoke", "Hopeful Hedgehog", "Marvelous Mouse"]) ➞ True

is\_correct\_aliases(["Rachel F.", "Pam G.", "Fred Z.", "Nancy K."], ["Reassuring Rat", "Peaceful Panda", "Fantastic Frog", "Notable Nickel"]) ➞ True

is\_correct\_aliases(["Beth T."], ["Brandishing Mimosa"]) ➞ False

# Both words in "Brandishing Mimosa" should begin with a "B" - "Brandishing Beaver" would do the trick.

# Advanced Programming Assignment 20

1. Create a function based on the input and output. Look at the examples, there is a pattern.

Examples

secret("p.one.two.three") ➞ "<p class='one two three'></p>"

secret("p.one") ➞ "<p class='one'></p>"

secret("p.four.five") ➞ "<p class='four five'></p>"

2. Create a function which counts how many lone 1s appear in a given number. Lone means the number doesn't appear twice or more in a row.

Examples

count\_lone\_ones(101) ➞ 2

count\_lone\_ones(1191) ➞ 1

count\_lone\_ones(1111) ➞ 0

count\_lone\_ones(462) ➞ 0

3. Write a method that accepts two integer parameters rows and cols. The output is a 2d array of numbers displayed in column-major order, meaning the numbers shown increase sequentially down each column and wrap to the top of the next column to the right once the bottom of the current column is reached.

Examples

printGrid(3, 6) ➞ [

[1, 4, 7, 10, 13, 16],

[2, 5, 8, 11, 14, 17],

[3, 6, 9, 12, 15, 18]

]

printGrid(5, 3) ➞ [

[1, 6, 11],

[2, 7, 12],

[3, 8, 13],

[4, 9, 14],

[5, 10, 15]

]

printGrid(4, 1) ➞ [

[1],

[2],

[3],

[4]

]

4. Given a list of integers, return the smallest positive integer not present in the list.

Here is a representative example. Consider the list:

[-2, 6, 4, 5, 7, -1, 7, 1, 3, 6, 6, -2, 9, 10, 2, 2]

After reordering, the list becomes:

[-2, -2, -1, 1, 2, 2, 3, 4, 5, 6, 6, 6, 7, 7, 9, 10]

from which we see that the smallest missing positive integer is 8.

Examples

min\_miss\_pos([-2, 6, 4, 5, 7, -1, 1, 3, 6, -2, 9, 10, 2, 2]) ➞ 8

# After sorting, list becomes [-2, -2, -1, 1, 2, 2, 3, 4, 5, 6, 6, 7, 9, 10]

# So the smallest missing positive integer is 8

min\_miss\_pos([5, 9, -2, 0, 1, 3, 9, 3, 8, 9]) ➞ 2

# After sorting, list becomes [-2, 0, 1, 3, 3, 5, 8, 9, 9, 9]

# So the smallest missing positive integer is 2

min\_miss\_pos([0, 4, 4, -1, 9, 4, 5, 2, 10, 7, 6, 3, 10, 9]) ➞ 1

# After sorting, list becomes [-1, 0, 2, 3, 4, 4, 4, 5, 6, 7, 9, 9, 10, 10]

# So the smallest missing positive integer is 1

5. Google is launching a network of autonomous pizza delivery drones and wants you to create a flexible rewards system (Pizza Points™) that can be tweaked in the future. The rules are simple: if a customer has made at least N orders of at least Y price, they get a FREE pizza!

Create a function that takes a dictionary of customers, a minimum number of orders and a minimum order price. Return a list of customers that are eligible for a free pizza.

Examples

customers = {

"Batman": [22, 30, 11, 17, 15, 52, 27, 12],

"Spider-Man": [5, 17, 30, 33, 40, 22, 26, 10, 11, 45]

}

pizza\_points(customers, 5, 20) ➞ ["Spider-Man"]

pizza\_points(customers, 3, 10) ➞ ["Batman", "Spider-Man"]

pizza\_points(customers, 5, 100) ➞ []

# Advanced Programming Assignment 21

1. Given a sentence, return the number of words which have the same first and last letter.

Examples

count\_same\_ends("Pop! goes the balloon") ➞ 1

count\_same\_ends("And the crowd goes wild!") ➞ 0

count\_same\_ends("No I am not in a gang.") ➞ 1

2. The Atbash cipher is an encryption method in which each letter of a word is replaced with its "mirror" letter in the alphabet: A <=> Z; B <=> Y; C <=> X; etc.

Create a function that takes a string and applies the Atbash cipher to it.

Examples

atbash("apple") ➞ "zkkov"

atbash("Hello world!") ➞ "Svool dliow!"

atbash("Christmas is the 25th of December") ➞ "Xsirhgnzh rh gsv 25gs lu Wvxvnyvi"

3. Create a class Employee that will take a full name as argument, as well as a set of none, one or more keywords. Each instance should have a name and a lastname attributes plus one more attribute for each of the keywords, if any.

Examples

john = Employee("John Doe")

mary = Employee("Mary Major", salary=120000)

richard = Employee("Richard Roe", salary=110000, height=178)

giancarlo = Employee("Giancarlo Rossi", salary=115000, height=182, nationality="Italian")

john.name ➞ "John"

mary.lastname ➞ "Major"

richard.height ➞ 178

giancarlo.nationality ➞ "Italian"

4. Create a function that determines whether each seat can "see" the front-stage. A number can "see" the front-stage if it is strictly greater than the number before it.

Everyone can see the front-stage in the example below:

# FRONT STAGE

[[1, 2, 3, 2, 1, 1],

[2, 4, 4, 3, 2, 2],

[5, 5, 5, 5, 4, 4],

[6, 6, 7, 6, 5, 5]]

# Starting from the left, the 6 > 5 > 2 > 1, so all numbers can see.

# 6 > 5 > 4 > 2 - so all numbers can see, etc.

Not everyone can see the front-stage in the example below:

# FRONT STAGE

[[1, 2, 3, 2, 1, 1],

[2, 4, 4, 3, 2, 2],

[5, 5, 5, 10, 4, 4],

[6, 6, 7, 6, 5, 5]]

# The 10 is directly in front of the 6 and blocking its view.

The function should return True if every number can see the front-stage, and False if even a single number cannot.

Examples

can\_see\_stage([

[1, 2, 3],

[4, 5, 6],

[7, 8, 9]

]) ➞ True

can\_see\_stage([

[0, 0, 0],

[1, 1, 1],

[2, 2, 2]

]) ➞ True

can\_see\_stage([

[2, 0, 0],

[1, 1, 1],

[2, 2, 2]

]) ➞ False

can\_see\_stage([

[1, 0, 0],

[1, 1, 1],

[2, 2, 2]

]) ➞ False

# Number must be strictly smaller than

# the number directly behind it.

5. Create a Pizza class with the attributes order\_number and ingredients (which is given as a list). Only the ingredients will be given as input.

You should also make it so that its possible to choose a ready made pizza flavour rather than typing out the ingredients manually! As well as creating this Pizza class, hard-code the following pizza flavours.

Name Ingredients

hawaiian ham, pineapple

meat\_festival beef, meatball, bacon

garden\_feast spinach, olives, mushroom

Examples

p1 = Pizza(["bacon", "parmesan", "ham"]) # order 1

p2 = Pizza.garden\_feast() # order 2

p1.ingredients ➞ ["bacon", "parmesan", "ham"]

p2.ingredients ➞ ["spinach", "olives", "mushroom"]

p1.order\_number ➞ 1

p2.order\_number ➞ 2

# Advanced Programming Assignment 22

1. Create a class Smoothie and do the following:

- Create an instance attribute called ingredients.

- Create a get\_cost method which calculates the total cost of the ingredients used to make the smoothie.

- Create a get\_price method which returns the number from get\_cost plus the number from get\_cost multiplied by 1.5. Round to two decimal places.

- Create a get\_name method which gets the ingredients and puts them in alphabetical order into a nice descriptive sentence. If there are multiple ingredients, add the word "Fusion" to the end but otherwise, add "Smoothie". Remember to change "-berries" to "-berry". See the examples below.

Ingredient Price

Strawberries $1.50

Banana $0.50

Mango $2.50

Blueberries $1.00

Raspberries $1.00

Apple $1.75

Pineapple $3.50

Examples

s1 = Smoothie(["Banana"])

s1.ingredients ➞ ["Banana"]

s1.get\_cost() ➞ "$0.50"

s1.get\_price() ➞ "$1.25"

s1.get\_name() ➞ "Banana Smoothie"

s2 = Smoothie(["Raspberries", "Strawberries", "Blueberries"])

s2.ingredients ➞ ["Raspberries", "Strawberries", "Blueberries"]

s2.get\_cost() ➞ "$3.50"

s2.get\_price() ➞ "$8.75"

s2.get\_name() ➞ "Blueberry Raspberry Strawberry Fusion"

2. Your task is to write a program which allows teachers to create a multiple choice test in a class called Testpaper and to be also able to assign a minimum pass mark. The testpaper's subject should also be included. The attributes are in the following order:

1. subject

2. markscheme

3. pass\_mark

As well as that, we need to create student objects to take the test itself! Create another class called Student and do the following:

- Create an attribute called tests\_taken and set the default as 'No tests taken'.

- Make a method called take\_test(), which takes in the testpaper object they are completing and the student's answers. Compare what they wrote to the mark scheme, and append to the/create a dictionary assigned to tests\_taken in the way as shown in the point below.

- Each key in the dictionary should be the testpaper subject and each value should be a string in the format seen in the examples below (whether or not the student has failed, and their percentage in brackets).

Examples

paper1 = Testpaper("Maths", ["1A", "2C", "3D", "4A", "5A"], "60%")

paper2 = Testpaper("Chemistry", ["1C", "2C", "3D", "4A"], "75%")

paper3 = Testpaper("Computing", ["1D", "2C", "3C", "4B", "5D", "6C", "7A"], "75%")

student1 = Student()

student2 = Student()

student1.tests\_taken ➞ "No tests taken"

student1.take\_test(paper1, ["1A", "2D", "3D", "4A", "5A"])

student1.tests\_taken ➞ {"Maths" : "Passed! (80%)"}

student2.take\_test(paper2, ["1C", "2D", "3A", "4C"])

student2.take\_test(paper3, ["1A", "2C", "3A", "4C", "5D", "6C", "7B"])

student2.tests\_taken ➞ {"Chemistry" : "Failed! (25%)", "Computing" : "Failed! (43%)"}

3. Due to unforseen circumstances in Suburbia, the trains will be delayed by a further 10 minutes.

Create a function that will help to plan out and manage these delays! Create a function called manage\_delays that does the following:

- Parameters will be the train object, a destination and number of minutes the delay is.

- Increment to the train object's expected\_time by the delay, if the destination given is in the train object's destinations.

Examples

trains = [

Train(["Townsville", "Suburbia", "Urbantska"], "13:04"),

Train(["Farmsdale", "Suburbia", "Lakeside Valley"], "13:20"),

Train(["Suburbia", "Townsville", "Lakeside Valley"], "13:22")

]

for t in trains:

manage\_delays(t, "Lakeside Valley", 60)

trains[0].expected\_time ➞ "13:04"

trains[1].expected\_time ➞ "14:20"

trains[2].expected\_time ➞ "14:22"

4. Ted works as a computer programmer at Minecraft Inc. His boss has just given him an important assignment to update the code for the minecart tracks by the end of April. However, he has recently had to self-isolate due to Corvid-19 and has left the code for the tracks BACK AT WORK!! He has the shorthand for the tracks he's supposed to look at, and where the carts are suppost to end up, but not the actual code.

He knows that:

1. "-->" = "Speed-Up Track" ⁠— If a minecart interacts with this track, it's velocity increases by 2.67 BPS unless it's at its maximum speed of 8 BPS.

2. "<-->" = "Powered Track" ⁠— If a minecart interacts with this track, it's velocity remains the same.

3. "<--" = "Slow-Down Track" ⁠— If a minecart interacts with this track, it's velocity decreases by 2.67 BPS unless it's velocity equals 0, at which point it stops.

4. "---" = "Unpowered Track" ⁠— If a minecart interacts with this track, it's velocity decreases by 1 BPS unless it's velocity equals 0, at which point it stops.

Help Ted by writing a class for the tracks that interact with the provided Minecart class as shown above. And then write a function that will take a list of the shorthand tracks and:

- If the Minecart reaches the last peice of Track, return True.

- Else return the index of the Track where the Minecart stops.

Examples

mine\_run(["-->", "-->", "-->", "<--", "<--", "<--"]) ➞ True

mine\_run(["-->", "<--", "-->", "-->", "<-->", "---"]) ➞ 1

5. Make a Rectangle class with four parameters, an x, a y (representing the top-left corner of the rectangle), a width and a height exclusively in that order.

Lastly, make a function intersecting that takes two Rectangle objects and returns True if those objects are intersecting (colliding), else return False.

Examples

a = Rectangle(10, 20, 100, 20)

b = Rectangle(10, 40, 15, 20)

c = Rectangle(50, 50, 20, 30)

intersecting(a, b) ➞ True

intersecting(a, c) ➞ False

intersecting(b, c) ➞ True

# Advanced Programming Assignment 23

1. Create a class Sudoku that takes a string as an argument. The string will contain the numbers of a regular 9x9 sudoku board left to right and top to bottom, with zeros filling up the empty cells.

Attributes

An instance of the class Sudoku will have one attribute:

- board: a list representing the board, with sublits for each row, with the numbers as integers. Empty cell represented with 0.

Methods

An instance of the class Sudoku wil have three methods:

- get\_row(n): will return the row in position n.

- get\_col(n): will return the column in position n.

- get\_sqr([n, m]): will return the square in position n if only one argument is given, and the square to which the cell in position (n, m) belongs to if two arguments are given.

Example

game = Sudoku("417950030000000700060007000050009106800600000000003400900005000000430000200701580")

game.board ➞ [

[4, 1, 7, 9, 5, 0, 0, 3, 0],

[0, 0, 0, 0, 0, 0, 7, 0, 0],

[0, 6, 0, 0, 0, 7, 0, 0, 0],

[0, 5, 0, 0, 0, 9, 1, 0, 6],

[8, 0, 0, 6, 0, 0, 0, 0, 0],

[0, 0, 0, 0, 0, 3, 4, 0, 0],

[9, 0, 0, 0, 0, 5, 0, 0, 0],

[0, 0, 0, 4, 3, 0, 0, 0, 0],

[2, 0, 0, 7, 0, 1, 5, 8, 0]

]

game.get\_row(0) ➞ [4, 1, 7, 9, 5, 0, 0, 3, 0]

game.get\_col(8) ➞ [0, 0, 0, 6, 0, 0, 0, 0, 0]

game.get\_sqr(1) ➞ [9, 5, 0, 0, 0, 0, 0, 0, 7]

game.get\_sqr(1, 8) ➞ [0, 3, 0, 7, 0, 0, 0, 0, 0]

game.get\_sqr(8, 3) ➞ [0, 0, 5, 4, 3, 0, 7, 0, 1]

2. The function input is two non-empty linked lists representing two non-negative integers. The digits are stored in reverse order, and each of their nodes contains a single digit. Add the two numbers and return the sum as a linked list, in which the digits are also stored in reversed order. The class ListNode, building block of the linked list, is defined in the Tests tab.

Class definition

class ListNode:

def \_\_init\_\_(self, val=0, next=None):

self.val = val

self.next = next

Examples

lt1 = ListNode(2)

lt1.add\_data([4, 3])

lt2 = ListNode(5)

lt2.add\_data([6, 4])

# print(lt1.get\_data()) # [2, 4, 3]

# print(lt2.get\_data()) # [5, 6, 4]

# print(342 + 465) # 807

add\_two\_numbers(lt1, lt2).get\_data() ➞ [7, 0, 8]

lt1 = ListNode(0)

lt2 = ListNode(0)

# print(lt1.get\_data()) # [0]

# print(lt2.get\_data()) # [0]

# print(0 + 0) # 0

add\_two\_numbers(lt1, lt2).get\_data() ➞ [0]

lt1 = ListNode(9)

lt1.add\_data([9,9,9,9,9,9])

lt2 = ListNode(9)

lt2.add\_data([9,9,9])

# print(lt1.get\_data()) # [9, 9, 9, 9, 9, 9, 9]

# print(lt2.get\_data()) # [9, 9, 9, 9]

# print(9999999 + 9999) # 10009998

add\_two\_numbers(lt1, lt2).get\_data() ➞ [8, 9, 9, 9, 0, 0, 0, 1]

3. Write a class called CoffeeShop, which has three instance variables:

1. name : a string (basically, of the shop)

2. menu : a list of items (of dict type), with each item containing the item (name of the item), type (whether a food or a drink) and price.

3. orders : an empty list

and seven methods:

1. add\_order: adds the name of the item to the end of the orders list if it exists on the menu, otherwise, return "This item is currently unavailable!"

2. fulfill\_order: if the orders list is not empty, return "The {item} is ready!". If the orders list is empty, return "All orders have been fulfilled!"

3. list\_orders: returns the item names of the orders taken, otherwise, an empty list.

4. due\_amount: returns the total amount due for the orders taken.

5. cheapest\_item: returns the name of the cheapest item on the menu.

6. drinks\_only: returns only the item names of type drink from the menu.

7. food\_only: returns only the item names of type food from the menu.

IMPORTANT: Orders are fulfilled in a FIFO (first-in, first-out) order.

Examples

tcs.add\_order("hot cocoa") ➞ "This item is currently unavailable!"

# Tesha's coffee shop does not sell hot cocoa

tcs.add\_order("iced tea") ➞ "This item is currently unavailable!"

# specifying the variant of "iced tea" will help the process

tcs.add\_order("cinnamon roll") ➞ "Order added!"

tcs.add\_order("iced coffee") ➞ "Order added!"

tcs.list\_orders ➞ ["cinnamon roll", "iced coffee"]

# all items of the current order

tcs.due\_amount() ➞ 2.17

tcs.fulfill\_order() ➞ "The cinnamon roll is ready!"

tcs.fulfill\_order() ➞ "The iced coffee is ready!"

tcs.fulfill\_order() ➞ "All orders have been fulfilled!"

# all orders have been presumably served

tcs.list\_orders() ➞ []

# an empty list is returned if all orders have been exhausted

tcs.due\_amount() ➞ 0.0

# no new orders taken, expect a zero payable

tcs.cheapest\_item() ➞ "lemonade"

tcs.drinks\_only() ➞ ["orange juice", "lemonade", "cranberry juice", "pineapple juice", "lemon iced tea", "vanilla chai latte", "hot chocolate", "iced coffee"]

tcs.food\_only() ➞ ["tuna sandwich", "ham and cheese sandwich", "bacon and egg", "steak", "hamburger", "cinnamon roll"]

4. In this challenge, write a funcion loneliest\_number to find the last Lonely number inside a sequence. A number is Lonely if the distance from its closest Prime sets a new record of the sequence.

Sequence = from 0 to 3

# Any number lower than 3 doesn't have a Prime preceeding it...

# ...so that you'll consider only its next closest Prime.

0 has distance 2 from its closest Prime (2)

# It's a new record! 0 It's the first lonely number of the sequence

1 has distance 1 from its closest Prime (2)

2 has distance 1 from 3

3 has distance 1 from 2

# The sequence 0 to 3 has only one Lonely number: 0

Sequence = Numbers from 5 to 10

5 has distance 2 from its closest Prime (3 or 7)

# It's a new record! 5 It's the first lonely number of the sequence

6 has distance 1 from 5 or 7

7 has distance 2 from 5

8 has distance 1 from 7

9 has distance 2 from 7 or 11

10 has distance 1 from 11

# The sequence 5 to 10 has only one Lonely number: 5

Sequence = Numbers from 19 to 24

19 has distance 2 from its closest Prime (17)

# It's a new record! 19 It's the first lonely number of the sequence

20 has distance 1 from 19

21 has distance 2 from 5

22 has distance 1 from 23

23 has distance 4 from 17 or 29

# It's a new record! 23 is the second lonely number of the sequence

24 has distance 1 from 23

# The sequence 19 to 24 has two Lonely numbers: 19 and 23

The function loneliest\_number must accept two integers lo and hi being the inclusive bounds of the sequence to analyze, and returns a dictionay (dict) object with the following keys and values:

- number: is the last Lonely number found in the given sequence;

- distance: is the distance of the number from its closest Prime;

- closest: is the Prime closest to number (if two Primes are equally distant from number, return the higher Prime).

Examples

loneliest\_number(0, 22) ➞ {

number: 0, distance: 2, closest: 2

}

loneliest\_number(8, 123) ➞ {

number: 53, distance: 6, closest: 59

}

loneliest\_number(938, 1190) ➞ {

number: 1140, distance: 11, closest: 1151

}

loneliest\_number(120, 1190) ➞ {

number: 211, distance: 12, closest: 223

}

5. Implement a class Selfie that can store the current state of the object in the form of binary string. It can take multiple pictures and then recover to a state it was before. During testing an object will be provided with new attributes and their values. It will store its state. Then the values will be changed. Then it will be given new attributes. It will store its state again. It will be repeated few times.

Later the states of the object will be recovered given an index. The return value should be a new Selfie with the requested historic state and the state history of the new object should be updated with a copy of current object's state history.

The object also knows how many states it has stored. If the index is not within the range of stored states, the object stays as is. If the argument is invalid, n < 0 or n >= self.n\_states(), the current object (or a copy thereof) should be returned.

Examples

p = Selfie()

p.x = 2

p.save\_state()

p.x = 5

p = p.recover\_state(0)

p.x ➞ 2

# Advanced Programming Assignment 24

1. Implement a class iterator to flatten a nested list of lists of integers. Each list element is either an integer or a list. There can be many levels of nested lists in lists.

The class initializes with a nested list. It also has two methods:

1. next() returns an integer in the order of appearance.

2. hasNext() returns True / False regarding if all integers have been retrieved or not.

Write the Class implementation for three required methods.

Examples

ni, actual = NestedIterator([[1, 1], 2, [1, 1]]), []

while ni.hasNext():

actual.append(ni.next())

actual ➞ [1, 1, 2, 1, 1]

ni, actual = NestedIterator([1, [4, [6]]]), []

while ni.hasNext():

actual.append(ni.next())

actual ➞ [1, 4, 6]

ni, actual = NestedIterator([[[]], []]), []

while ni.hasNext():

actual.append(ni.next())

actual ➞ []

2. Implement the class Shape that receives perimeter and density function into \_\_init\_\_ method. The list of consecutive corners defines shape of a 2-dimensional object. The density function defines the mass distribution inside the shape. To compute mass in a certain point m(x, y) = small\_square \* density(x, y). The \_\_init\_\_ method calls other internal methods that compute three characteristics of the shape:

- area

- total mass

- center of mass (xc, yc)

The computational grid has distance between two neighboring points as 2 \* delta, the distance between a grid point and the perimeter wall is delta.

Examples

sh\_ex1 = Shape([(1, 1), (3, 1), (3, 2), (1, 2)], lambda x, y: 100 + 100 \* x)

sh\_ex1.area ➞ 2.0

sh\_ex1.mass ➞ 600.0

sh\_ex1.mass\_center ➞ (2.1, 1.5)

The example can be verified via analytical integration. Other shapes in Tests are slightly more complicated and require numerical integration as illustrated here:

3. Given a 3x3 matrix of a completed tic-tac-toe game, create a function that returns whether the game is a win for "X", "O", or a "Draw", where "X" and "O" represent themselves on the matrix, and "E" represents an empty spot.

Examples

tic\_tac\_toe([

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

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

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

]) ➞ "X"

tic\_tac\_toe([

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

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

["E", "X", "X"]

]) ➞ "O"

tic\_tac\_toe([

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

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

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

]) ➞ "Draw"

4. Your computer might have been infected by a virus! Create a function that finds the viruses in files and removes them from your computer.

Examples

remove\_virus("PC Files: spotifysetup.exe, virus.exe, dog.jpg") ➞ "PC Files: spotifysetup.exe, dog.jpg"

remove\_virus("PC Files: antivirus.exe, cat.pdf, lethalmalware.exe, dangerousvirus.exe ") ➞ "PC Files: antivirus.exe, cat.pdf"

remove\_virus("PC Files: notvirus.exe, funnycat.gif") ➞ "PC Files: notvirus.exe, funnycat.gif")

5. In a video game, a meteor will fall toward the main character's home planet. Given the meteor's trajectory as a string in the form y = mx + b and the character's position as a tuple of (x, y), return True if the meteor will hit the character and False if it will not.

Examples

will\_hit("y = 2x - 5", (0, 0)) ➞ False

will\_hit("y = -4x + 6", (1, 2)) ➞ True

will\_hit("y = 2x + 6", (3, 2)) ➞ False

# Advanced Programming Assignment 25

1. Write four functions that directly mutate a list:

1. repeat(lst, n): Repeat lst n times.

2. add(lst, x): Adds x to the end of the lst.

3. remove(lst, m, n): Removes all elements between indices m and n inclusive in lst.

4. concat(lst, x): concatenates lst with x (another list).

Examples

lst = [1, 2, 3, 4]

repeat(lst, 3) ➞ [1, 2, 3, 4, 1, 2, 3, 4, 1, 2, 3, 4]

add(lst, 1) ➞ [1, 2, 3, 4, 1, 2, 3, 4, 1, 2, 3, 4, 1]

remove(lst, 1, 12) ➞ [1]

concat(lst, [3, 4]) ➞ [1, 3, 4]

2. The classic game of Mastermind is played on a tray on which the Mastermind conceals a code and the Guesser has 10 tries to guess it. The code is a sequence of 4 (or 6, sometimes more) pegs of different colors. Each guess is a corresponding sequence of 4 (or more) pegs of different colors. A guess is "correct" when the color of every peg in the guess exactly matches the corresponding peg in the Mastermind's code.

After each guess by the Guesser, the Mastermind will give a score comprising black & white pegs, not arranged in any order:

- Black peg == guess peg matches the color of a code peg in the same position.

- White peg == guess peg matches the color of a code peg in another position.

Create a function that takes two strings, code and guess as arguments, and returns the score in a dictionary.

- The code and guess are strings of numeric digits

- The color of the pegs are represented by numeric digits

- no "peg" may be double-scored

Examples

guess\_score("1423", "5678") ➞ {"black": 0, "white": 0}

guess\_score("1423", "2222") ➞ {"black": 1, "white": 0}

guess\_score("1423", "1234") ➞ {"black": 1, "white": 3}

guess\_score("1423", "2211") ➞ {"black": 0, "white": 2}

3. Create a function that takes a list lst and a number N and returns a list of two integers from lst whose product equals N.

Examples

two\_product([1, 2, -1, 4, 5], 20) ➞ [4, 5]

two\_product([1, 2, 3, 4, 5], 10) ➞ [2, 5]

two\_product([100, 12, 4, 1, 2], 15) ➞ None

4. In this challenge, sort a list containing a series of dates given as strings. Each date is given in the format DD-MM-YYYY\_HH:MM:

"12-02-2012\_13:44"

The priority of criteria used for sorting will be:

- Year

- Month

- Day

- Hours

- Minutes

Given a list lst and a string mode, implement a function that returns:

- if mode is equal to "ASC", the list lst sorted in ascending order.

- if mode is equal to "DSC", the list lst sorted in descending order.

Examples

sort\_dates(["10-02-2018\_12:30", "10-02-2016\_12:30", "10-02-2018\_12:15"], "ASC") ➞ ["10-02-2016\_12:30", "10-02-2018\_12:15", "10-02-2018\_12:30"]

sort\_dates(["10-02-2018\_12:30", "10-02-2016\_12:30", "10-02-2018\_12:15"], "DSC") ➞ ["10-02-2018\_12:30", "10-02-2018\_12:15", "10-02-2016\_12:30"]

sort\_dates(["09-02-2000\_10:03", "10-02-2000\_18:29", "01-01-1999\_00:55"], "ASC") ➞ ["01-01-1999\_00:55", "09-02-2000\_10:03", "10-02-2000\_18:29"]

5. Write a function that selects all words that have all the same vowels (in any order and/or number) as the first word, including the first word.

Examples

same\_vowel\_group(["toe", "ocelot", "maniac"]) ➞ ["toe", "ocelot"]

same\_vowel\_group(["many", "carriage", "emit", "apricot", "animal"]) ➞ ["many"]

same\_vowel\_group(["hoops", "chuff", "bot", "bottom"]) ➞ ["hoops", "bot", "bottom"]

6. Create a function that takes a list of more than three numbers and returns the Least Common Multiple (LCM).

Examples

lcm\_of\_list([1, 2, 3, 4, 5, 6, 7, 8, 9, 10]) ➞ 2520

lcm\_of\_list([13, 6, 17, 18, 19, 20, 37]) ➞ 27965340

lcm\_of\_list([44, 64, 12, 17, 65]) ➞ 2333760