Question1

Create a function that takes three parameters where:

* x is the start of the range (inclusive).
* y is the end of the range (inclusive).
* n is the divisor to be checked against.

Return an ordered list with numbers in the range that are divisible by the third parameter n. Return an empty list if there are no numbers that are divisible by n.

**Examples**

list\_operation(1, 10, 3) ➞ [3, 6, 9]

list\_operation(7, 9, 2) ➞ [8]

list\_operation(15, 20, 7) ➞ []

def list\_operation(x,y,divisor):

div\_list = []

res\_lis = []

for i in range(1,10+1):

div\_list.append(i\*divisor)

for i in range(x,y):

if i in div\_list:

res\_lis.append(i)

print(res\_lis)

list\_operation(1, 10, 3) #[3, 6, 9]

list\_operation(7, 9, 2) # [8]

list\_operation(15, 20, 7) # []

Question2

Create a function that takes in two lists and returns True if the second list follows the first list by **one** element, and False otherwise. In other words, determine if the second list is the first list shifted to the right by 1.

**Examples**

simon\_says([1, 2], [5, 1]) ➞ True

simon\_says([1, 2], [5, 5]) ➞ False

simon\_says([1, 2, 3, 4, 5], [0, 1, 2, 3, 4]) ➞ True

simon\_says([1, 2, 3, 4, 5], [5, 5, 1, 2, 3]) ➞ False

**Notes**

* Both input lists will be of the same length, and will have a minimum length of 2.
* The values of the 0-indexed element in the second list and the n-1th indexed element in the first list do not matter.

def simon\_says(list1, list2):

# Check if all elements in list2, except the last one, are equal to the corresponding elements in list1

return list1[:-1] == list2[1:]

# Test cases

print(simon\_says([1, 2], [5, 1])) # ➞ True

print(simon\_says([1, 2], [5, 5])) # ➞ False

print(simon\_says([1, 2, 3, 4, 5], [0, 1, 2, 3, 4])) # ➞ True

print(simon\_says([1, 2, 3, 4, 5], [5, 5, 1, 2, 3])) # ➞ False

Question3

A group of friends have decided to start a secret society. The name will be the first letter of each of their names, sorted in alphabetical order.

Create a function that takes in a list of names and returns the name of the secret society.

### Examples

society\_name(["Adam", "Sarah", "Malcolm"]) ➞ "AMS"

society\_name(["Harry", "Newt", "Luna", "Cho"]) ➞ "CHLN"

society\_name(["Phoebe", "Chandler", "Rachel", "Ross", "Monica", "Joey"])

def society\_name(n):

l = []

for i in n:

l.append(i[0])

res= sorted(l)

res1= ''.join(res)

print(res1)

society\_name(["Adam", "Sarah", "Malcolm"])

society\_name(["Harry", "Newt", "Luna", "Cho"])

society\_name(["Phoebe", "Chandler", "Rachel", "Ross", "Monica", "Joey"])

Question4

An isogram is a word that has no duplicate letters. Create a function that takes a string and returns either True or False depending on whether or not it's an "isogram".

**Examples**

is\_isogram("Algorism") ➞ True

is\_isogram("PasSword") ➞ False

# Not case sensitive.

is\_isogram("Consecutive") ➞ False

**Notes**

* Ignore letter case (should not be case sensitive).
* All test cases contain valid one word strings.

def is\_isogram(n):

l = []

for i in n.lower():

if i in l:

return False # Found a duplicate, return False immediately

l.append(i) # Add the character to the list

return True # If no duplicates were found, return True

# Test cases

print(is\_isogram("Algorism")) # True

print(is\_isogram("PasSword")) # False

print(is\_isogram("Consecutive")) # False

Question5

Create a function that takes a string and returns True or False, depending on whether the characters are in order or not.

### Examples

is\_in\_order("abc") ➞ True

is\_in\_order("edabit") ➞ False

is\_in\_order("123") ➞ True

is\_in\_order("xyzz") ➞ True

### Notes

You don't have to handle empty strings.

def is\_in\_order(n):

return sorted(n) == list(n) # Convert n to a list for comparison

# Test cases

print(is\_in\_order("abc")) # True

print(is\_in\_order("zyx")) # False

print(is\_in\_order("aegmps")) # True

print(is\_in\_order("hello")) # False