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) ➞ []

Ans.

import logging as lg

# importing logging so every function call of

lg.basicConfig(filename ='C:\\Users\\Home\\Johns python talent\\logging\\testlog1.log', level =lg.INFO , format = '%(asctime)s %(message)s')

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

my\_final\_list = []

for i in range(x,y+1):

if i%n == 0:

my\_final\_list.append(i)

print(my\_final\_list)

return my\_final\_list

try:

list\_operation(1, 10, 3)

list\_operation(7, 9, 2)

list\_operation(15, 20, 7)

lg.info("""Class list\_operation() has been called has been called""")

except Exception as e:

print("There was an error called: ",e)

else:

pass

finally:

pass

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.

Ans.

import logging as lg

# importing logging so every function call of

lg.basicConfig(filename ='C:\\Users\\Home\\Johns python talent\\logging\\testlog1.log', level =lg.INFO , format = '%(asctime)s %(message)s')

def simon\_says(list\_1, list\_2):

if len(list\_1) >= 2:

if list\_1[-2] == list\_2[-1]:

print(True)

return True

else:

print(False)

return False

else:

print("Atleast 2 elements required in each list in argument to use this function")

try:

simon\_says([1, 2], [5, 1])

simon\_says([1, 2], [5, 5])

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

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

lg.info("""Class list\_operation() has been called has been called""")

except Exception as e:

print("There was an error called: ",e)

else:

pass

finally:

pass

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"])

Ans.

import logging as lg

# importing logging so every function call of

lg.basicConfig(filename ='C:\\Users\\Home\\Johns python talent\\logging\\testlog1.log', level =lg.INFO , format = '%(asctime)s %(message)s')

def society\_name(names):

alphas = []

society = ""

for i in range(len(names)):

alphas.append(names[i][0])

alphas.sort()

for j in alphas:

society += j

print(society)

try:

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

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

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

lg.info("""Class society\_name() has been called has been called""")

except Exception as e:

print("There was an error called: ",e)

else:

pass

finally:

pass

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.

Ans.

import logging as lg

# importing logging so every function call of

lg.basicConfig(filename ='C:\\Users\\Home\\Johns python talent\\logging\\testlog1.log', level =lg.INFO , format = '%(asctime)s %(message)s')

def is\_isogram(word):

test = []

for i in word.lower():

count = word.lower().count(i)

if count >1:

print(False)

break

else:

test.append(count)

if len(test) == len(word):

print(True)

try:

is\_isogram("Algorism")

is\_isogram("PasSword")

is\_isogram("Consecutive")

lg.info("""Class is\_isogram() has been called has been called""")

except Exception as e:

print("There was an error called: ",e)

else:

pass

finally:

pass

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.

import logging as lg

# importing logging so every function call of

lg.basicConfig(filename ='C:\\Users\\Home\\Johns python talent\\logging\\testlog1.log', level =lg.INFO , format = '%(asctime)s %(message)s')

def is\_in\_order(word):

decider = True

for i in range(len(word)-1):

if ord(word.lower()[i+1])- ord(word.lower()[i]) == 1 or ord(word.lower()[i+1])- ord(word.lower()[i]) == 0 :

decider = True

else:

decider = False

print(decider)

try:

is\_in\_order("abc")

is\_in\_order("edabit")

is\_in\_order("123")

is\_in\_order("xyzz")

lg.info("""Class is\_in\_order() has been called has been called""")

except Exception as e:

print("There was an error called: ",e)

else:

pass

finally:

pass