1. def list\_operation(start,end,divisor):

out\_list = []

for ele in range(start,end+1):

if ele%divisor == 0:

out\_list.append(ele)

print(f'Output: ➞ {out\_list}')

list\_operation(1, 10, 3)

list\_operation(7, 9, 2)

list\_operation(15, 20, 7)

Output:

Output: ➞ [3, 6, 9]

Output: ➞ [8]

Output: ➞ []

1. def simon\_says(in\_list\_1,in\_list\_2):

if len(in\_list\_1) == len(in\_list\_1) and len(in\_list\_1) >=2 and len(in\_list\_1) >=2:

if(in\_list\_1[:-1] == in\_list\_2[1:]):

print(f'{in\_list\_1,in\_list\_2} ➞ {True}')

else:

print(f'{in\_list\_1,in\_list\_2} ➞ {False}')

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])

Output:

([1, 2], [5, 1]) ➞ True

([1, 2], [5, 5]) ➞ False

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

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

1. def society\_name(in\_list):

out\_string = []

for ele in in\_list:

out\_string.append(ele[0])

output = ''.join(sorted(out\_string))

print(f'{in\_list} ➞ {output}')

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

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

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

Output:

['Adam', 'Sarah', 'Malcolm'] ➞ AMS

['Harry', 'Newt', 'Luna', 'Cho'] ➞ CHLN

['Phoebe', 'Chandler', 'Rachel', 'Ross', 'Monica', 'Joey'] ➞ CJMPRR

1. def is\_isogram(in\_string):

lower\_in\_string = in\_string.lower()

if len(lower\_in\_string) == len(set(lower\_in\_string)):

print(f'{in\_string} ➞ {True}')

else:

print(f'{in\_string} ➞ {False}')

is\_isogram("Algorism")

is\_isogram("PasSword")

is\_isogram("Consecutive")

Output:

Algorism ➞ True

PasSword ➞ False

Consecutive ➞ False

1. def is\_in\_order(in\_string):

in\_string\_sorted = ''.join(sorted(in\_string))

if in\_string == in\_string\_sorted:

print(f'{in\_string} ➞ {True}')

else:

print(f'{in\_string} ➞ {False}')

is\_in\_order("abc")

is\_in\_order("edabit")

is\_in\_order("123")

is\_in\_order("xyzz")

Output:

abc ➞ True

edabit ➞ False

123 ➞ True

xyzz ➞ True