Programming_Assingment22

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) ➞ []

In [1]:

```
def list_operation(x,y,n):

    divisor = []
    nonDiv = []
    for i in range(x,y+1):
        if i % n == 0:
            divisor.append(i)
    return divisor
```

In [2]:

```
list_operation(1, 10, 3)
```

Out[2]:

```
[3, 6, 9]
```

In [3]:

```
list_operation(7, 9, 2)
```

Out[3]:

```
[8]
```

In [4]:

```
list_operation(15, 20, 7)
```

Out[4]:

```
[]
```

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.

In [5]:

```
def simon_says(ls,ls2):
    if ls[0:len(ls)-1] == ls2[1:len(ls2)]:
        return True
    return False
```

In [6]:

```
simon_says([1, 2], [5, 1])
```

Out[6]:

```
True
```

In [7]:

```
simon_says([1, 2], [5, 5])
```

Out[7]:

```
False
```

In [8]:

```
simon_says([1, 2, 3, 4, 5], [0, 1, 2, 3, 4])
```

Out[8]:

```
True
```

In [111]:

```
simon_says([1, 2, 3, 4, 5], [5, 5, 1, 2, 3])
```

Out[111]:

```
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'])

In [9]:

```
def society_name(lst):
    return ''.join([i[0] for i in sorted(lst)])
```

In [10]:

```
society_name(['Adam', 'Sarah', 'Malcolm'])
```

Out[10]:

```
'AMS'
```

In [11]:

```
society_name(['Harry', 'Newt', 'Luna', 'Cho'])
```

Out[11]:

```
'CHLN'
```

In [12]:

```
society_name(['Phoebe', 'Chandler', 'Rachel', 'Ross', 'Monica', 'Joey'])
```

Out[12]:

```
'CJMPRR'
```

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.

In [13]:

```
def is_isogram(s):
    for i in range(len(s)):
        if s.lower().count(s[i]) > 1:
            return False
    else:
        return True
```

In [14]:

```
is_isogram('Algorism')
```

Out[14]:

```
True
```

In [15]:

```
is_isogram('PasSword')
```

Out[15]:

```
False
```

In [16]:

```
is_isogram('Consecutive')
```

Out[16]:

```
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.

In [17]:

```
def is_in_order(s):
    n = len(s)
    c = [s[i] for i in range(len(s))]
    c.sort()
    for i in range(n):
        if c[i] != s[i]:
            return False
    return True
```

In [18]:

```
is_in_order('abc')
```

Out[18]:

```
True
```

In [19]:

```
is_in_order('edabit')
```

Out[19]:

```
False
```

In [20]:

```
is_in_order('123')
```

Out[20]:

```
True
```

In [21]:

```
is_in_order('xyzz')
```

Out[21]:

```
True
```