3rd Batch exercises

Exercise 1 - skip multiples

Define a function `mysum(m, n)` which sums the integers from `m` (included) to `n` (excluded) but skipping the multiples of 3 or 5.

Exercise 2 - Hamming Distance

Write a function called `hammingdist` that takes two lists of equal length as its arguments, `l1` and `l2` , and returns their Hamming distance (defined below).

The two arguments must be both lists, otherwise an error should be produced.

Also, an exception should be raised if their length differ.

The Hamming distance is simply defined in this way: it's the number of differences between the two lists, when you compare them element-by-element. For example, these two lists have Hamming distance 3, since they differ in the three middle positions (while the first and the last elements coincide):

```
[0, 3, 5, 7, 1]
[0, 1, 2, 3, 1]
    ↑  ↑  ↑
```

Exercise 3 - Local Minima

Write a function called `localmin` that takes a list of numbers as input and returns the positions of "local minima" of the list.

A "local minimum" of a list happens when the element at that position is smaller or equal to its neighbors on the left and the right. At the border positions, only one neighbor needs to be checked.

Examples with the positions of the local minima marked by arrows:

```
[3, 2, 0, 1, 5, 8, 7]
       ↑           ↑

[1, 3, 3, 5, 9]
 ↑     ↑
```

Your function must return a list with the indices of the local minima found. It is not necessary to check the argument of the function.

Notice that:

- every non-empty list has at least one local minimum
- in a constant list, all elements are local minima
- in a list of length 1, the only element is also a local minimum

- the elements at the border need a special treatment.

Exercise 4: Relative Primes

Write a function called `relprime` that takes two positive integer arguments, `a` and `b` , and decides whether they are relatively prime. The function should return `True` if they are and `False` otherwise.

To write the algorithm, use the following simple definition: `a` and `b` are **not** relatively prime if there is any number `k` between `2` and `min (a, b)` (both extremes included) that divides both `a` and `b`. Otherwise, if no such number exists, they are relatively prime.

Examples:

```
In [5]: relprime(3, 5)
Out[5]: True

In [6]: relprime(6, 4)
Out[6]: False

In [7]: relprime(15, 26)
Out[7]: True

In [8]: relprime(27, 9)
Out[8]: False

In [9]: relprime(1, 6)
Out[9]: True
```

Exercise 5: issubblist

Write a function called `issublist` that takes two lists `l1` and `l2` , and returns `True` if `l1` is a sublist of `l2` , and `False` otherwise. To be more precise, in order for a list `l1` to be considered a sublist of `l2` , there should be a contiguous chunk of `l2` which is equal to `l1` . For example, `[2,3,2,1]` is a sublist of `[1,2,3,2,1,0]`, but not of `[2,3,4,3,2,1]`.

Some extreme cases that you will want to make sure about when testing: an empty list is always a sublist of any other list. Also, any list is a sublist of itself.

IMPORTANT: You cannot use any special list method for this exercise, nor you can use slice indexing (i.e. indexing expressions with the colon like `l[0:3]` are not allowed). Only loops and conditionals, and accessing one position at a time. If in doubt about what you can use, ask.