3a) lst \* 3: creates a list of the elements 1, 2, 3 three times, giving [1, 2, 3, 1, 2, 3, 1, 2, 3]

[ lst ] \* 3: creates a list of lists that contain the elements 1, 2, 3, giving three lists of [1, 2, 3]: [[1,2,3],[1,2,3],[1,2,3]]

b) The result of arr changes as the values in lst change. It is because for mutable data types, assignment copies references, not values. Same for arr[1][1] = 7, changing the values in arr will also change the values in lst, since they reference each other.

c) If you use a slice copy, that will fix the copy by reference problem. Now, if you change an element in the array, it will not affect the elements in lst, so it stayed the same.

d) arr = [[1,2,3],[1,2,3],[1,2,3]]

4a) If n = 0 and r = 0, it goes to line#8 and executes continue, which ignores the statements after and goes back to the for loop in line#4 and continues the for loop.

b) If n = 1 and r = 2, executes line#10 and continues the for loop

c) If n = 2 and r = 0, executes line#9, breaks out of the for loop and checks the if statement in line#14.

d) If n = 3 and r = 3, continues the for loop, but n is now 4, line#4 is false, so it goes to the else statement in line#11 and prints “Wow, you are lucky.”

5a)

i) ZeroDivisionError: division by zero: 5 / 0

ii) KeyError: dct = {}, dct[‘Tim’]

iii) TypeError: list indices must be integers or slices, not str: lst = [1,2,3] lst[“abc”]

8c)

The expression

[i for i in range(2,n) if len([x for x in range(2,i) if i%x==0]) == 0]

is a list comprehension.

What the exact expression does is the following:

-It returns a list of all the elements i in the range from 2...n that passes the if statement: len([x for x in range(2,i) if i%x==0]) == 0], which essentially means if the element i is not divisible by every element x in the range from 2...i.

🡪In other words, it returns a list of all the elements in the range of 2…n that are not divisible by any of the elements in the range of 2…i.