**Note 1:**

If I specified the except statement as:

|  |  |
| --- | --- |
| 20 | **except** IndexError **and** RecursionError**:** |

then the programme, when executed, would have first encountered an error that would be both an IndexError and RecursionError, by first identifying the IndexError and rendering it mute without the user's awareness and then proceeding to the identifying recursion error as it would also be the case.

As a solution I substituted to above statement to:

|  |  |
| --- | --- |
| 20 | **except** IndexError **or** RecursionError**:** |

The solution is mainly based on a test run of the programme and entering a negative number for the lower range bound. The attempt was carried out with the except statement written before the amendment, as shown in the first instance, and when running the programme, the stack report had returned an IndexError regardless of the written except statement. This led to the second instance as the solution, which indicates that the programme would the code would cause either error instead of both, thus finally displaying the expected result.

**Note 2:**

Initial analysis concluded that the time taken for some of the code is constant, yet when the first condition (base case) is not met and the programme executes the code in lines 14 or 16, the function calls itself thus triggering a recursion. Due to this, the programme re-executes the code before the call until the base case is met, which arguably therefore alters the time complexity of the said code to **O(log n)**. Nevertheless, after much consideration, the worst-case scenario is taken in account so the complexity of the programme algorithm would result to O(log n) regardless.