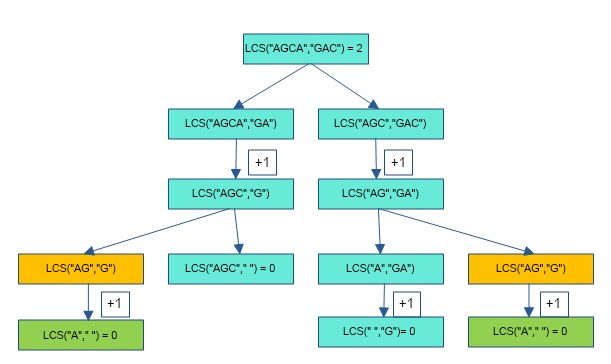
## Practical 5

### **Aim:** Write a program to implement Longest Common Subsequence (LCS) algorithm

**Theory:**

1. The longest common subsequence (LCS) is defined as the longest subsequence that is common to all the given sequences, provided that the elements of the subsequence are not required to occupy consecutive positions within the original sequences.
2. LCS can be performed by Two methods – Naïve method and DP

**Example:**



**Algorithm:**

LCS(X,Y,i,j)

X and Y be two given sequences

Initialize a table LCS of dimension X.length \* Y.length

X.label = X

Y.label = Y

LCS[0][] = 0

LCS[][0] = 0

Start from LCS[1][1]

Compare X[i] and Y[j]

If X[i] = Y[j]

LCS[i][j] = 1 + LCS[i-1, j-1]

Point an arrow to LCS[i][j]

Else

LCS[i][j] = max(LCS[i-1][j], LCS[i][j-1])

Point an arrow to max(LCS[i-1][j], LCS[i][j-1])

**Code:**

# A Naive recursive Python implementation of LCS problem

def lcs(X, Y, m, n):

if m == 0 or n == 0:

return 0;

elif X[m-1] == Y[n-1]:

return 1 + lcs(X, Y, m-1, n-1);

else:

return max(lcs(X, Y, m, n-1), lcs(X, Y, m-1, n));

# Driver program to test the above function

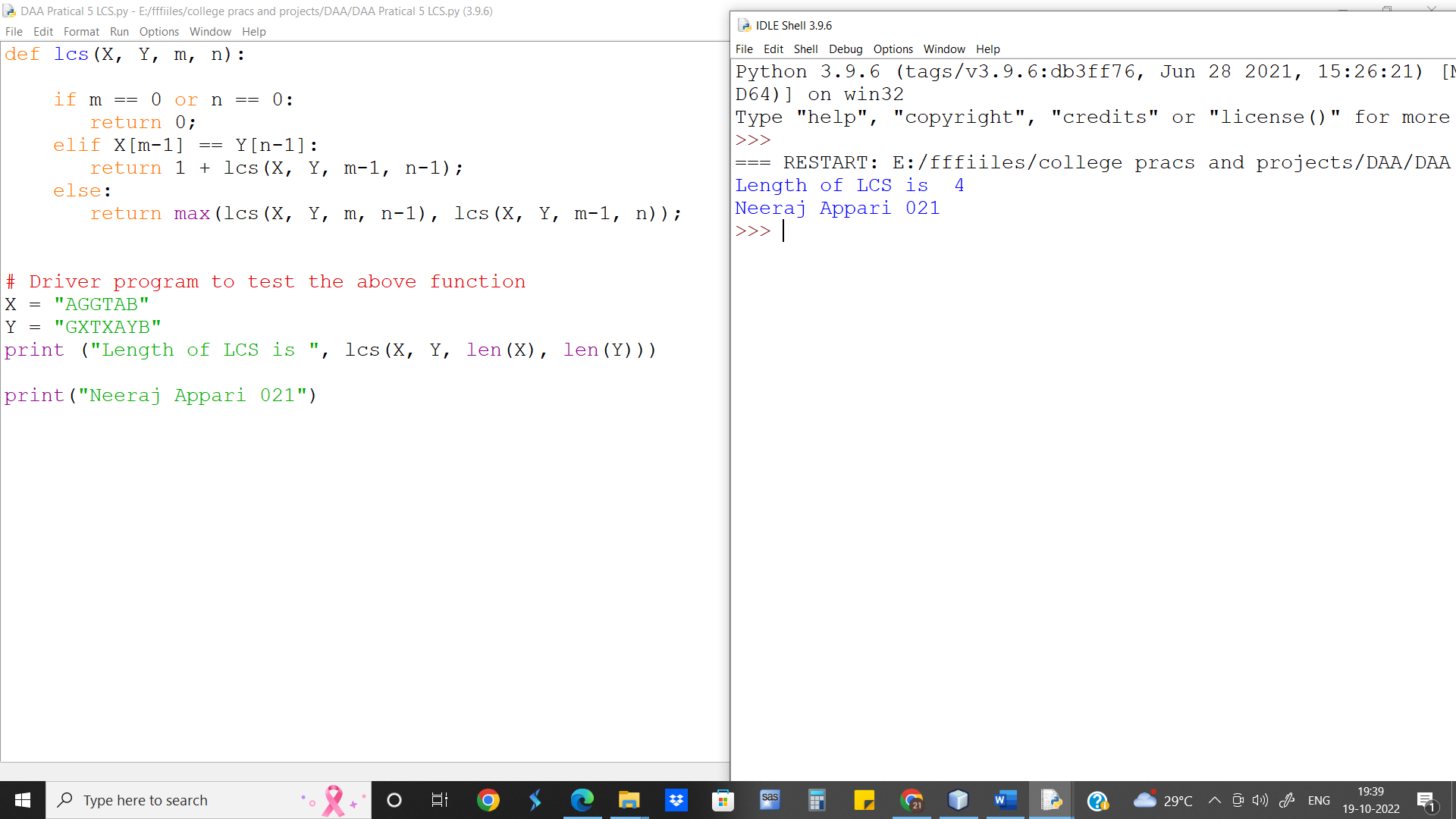
X = "AGGTAB"

Y = "GXTXAYB"

print ("Length of LCS is ", lcs(X, Y, len(X), len(Y)))

print("Neeraj Appari 021")

**Output**:



Length of LCS is 4

Neeraj Appari 021

**Runtime for Longest Common Subsequence Recursive approach is O(2n)**

**Conclusion:** We observe the difference in time complexities from O(2n) and O(m.n) for Longest Common Subsequence.