Script:

import pandas as pd

import os

from nltk.translate.bleu\_score import sentence\_bleu

df = pd.read\_csv("Input file")

reference = df.loc[ref\_rownum, 'response'].split()

candidate = df.loc[cand\_rownum, 'response'].split()

print('BlEU score -> {}'.format(sentence\_bleu(reference, candidate)))

|  |  |
| --- | --- |
| Input | CodeBleu Score |
| Reference:  def solve(cost, m, n):  # initialize dp matrix with 0s  dp = [[0 for i in range(n+1)] for j in range(m+1)]    # fill first row and column with cumulative costs  for i in range(1, m+1):  dp[i][0] = dp[i-1][0] + cost[i][0]  for j in range(1, n+1):  dp[0][j] = dp[0][j-1] + cost[0][j]    # fill rest of dp matrix with minimum cost path to reach each position  for i in range(1, m+1):  for j in range(1, n+1):  dp[i][j] = min(dp[i-1][j], dp[i][j-1]) + cost[i][j]    # return minimum cost to reach (m, n)  return dp[m][n]  Candidate:  def solve(cost, m, n):  # initialize a 2D array to store the minimum cost path  path = [[0 for i in range(n+1)] for j in range(m+1)]  # base case for starting position (0,0)  path[0][0] = cost[0][0]  # fill the first row  for j in range(1, n+1):  path[0][j] = path[0][j-1] + cost[0][j]  # fill the first column  for i in range(1, m+1):  path[i][0] = path[i-1][0] + cost[i][0]  # fill the rest of the path  for i in range(1, m+1):  for j in range(1, n+1):  path[i][j] = min(path[i-1][j], path[i][j-1]) + cost[i][j]  # return the minimum cost path to reach (m,n)  return path[m][n] | 9.641193013181824e-232 |
| Candidate:  def solve(n):  return n \* (n + 1)  Reference:  def solve(n):  return n \* (n + 1) | 1.0 |