The following is pseudo code created for a recursive solution to the z-transformation. The overall logic works; however, extra work will be required to get it working in Python. For example, the initialization that occurs in z_transformation has been left out. Please note that in Python, indexing of a two-dimensional array can be accomplished through the following: array[:,i] where column i and all of the rows of the array will be returned. Lastly, the pseudo code begins indexing at 0.

z_helper$(X, Z, i, degrees, l_{i-1}, B_{i-1})$

Input: The original input X, the Z array in progress, the bucket index i, the degrees calculated, the lexicographic array of the previous bucket, l_{i-1}, and the previous bucket B_{i-1}. Additionally, let d be the number of features in X.

Output: The resulting Z matrix of the concatenated buckets.

1: **if** $i \geq degrees$ **then** //Base Case
2: return Z
3: **end if**
4: $l_X = [0...d-1]$ //Assigning the l array for X. Can be a parameter instead.
5: $q = 0$ // Index for the new column in B_i.
6: **for** j = 0 ... $|l_{i-1}|$ **do** //For each element in the previous bucket
7: **for** k = 0 ... length of l_X **do** //For each element in original input
8: **if** $l_{i-1}[j] \leq l_X[k]$ **then** //If the lexicographic value of the bucket is less than or equal to the X value...
9: $l[q] = l_X[k]$ //Update the current l value with the respective lexicographic value in X.
10: $temp = B_{i-1}[:, j] * X[:, k]$ //Create the column vector from the element-wise multiplication of the respective elements in $B_{i-1} and X$
11: B.append($temp$,$axis = 1$) //Append the result to Z.
12: $q = q + 1$ //Increment the index of B_i.
13: **end if**
14: **end for**
15: **end for**
16: Z.append(B,$axis = 1$) //Append the bucket to the final result.
17: **return** z_helper$(X, Z, i + 1, degrees, l, B)$ //Recursively calculate the next bucket(s).