```
numpy as np is python numpy library to converts to array a matrix
matrix_function.soltion:
Method in matrix_function that do a progressive or backward substitutionv
to find an array

Input: matrix matrix
Output: solution array x

begin steppedMethod

dict dictionary
auxiliary_matrix <- np.array(matriz)
dictionary[0] <- matrix
int count - 0
array temporal_array
while (count < matrix.shape[0]-1) do:
        float pivot_number <- auxiliary_matrix
        if (count == 0):
                for row in auxiliary_matrix:
                        pivot_column <- np.abs(row[:-1])
                        temporal_maxpivot <- np.max(pivot_column)
                        temporal_array.add(temporal_maxpivot)
                end for
        end if
        sub_matrix <- auxiliary_matrix.T[0]
        division_colum = np.abs(sub_matrix)/temporal_array[count:]
        posmax_pivot <- np.where(division_colum ==
                        np.max(division_colum))[0][0]
        if (posmax_pivot != 0):
                pivot_number <- auxiliary_matrix[posmax_pivot][0]
        temporal_matrix <- np.array(auxiliary_matrix[0])
        auxiliary_matrix[0] <- np.array(auxiliary_matrix[posmax_pivot])
        auxiliary_matrix[posmax_pivot] <- temporal_matrix
        temporal_matrix <- np.array(matrix[i])
        matrix[i] <- np.array(matrix[i+posmax_pivot])
        matrix[i+posmax_pivot] <- temporal_matrix
        end if
        if (pivot_number==0 and i == matrix.shape[0]-2):
                break;
        end if
        fj <- auxiliary_matrix[0]
    column_vector <- np.reshape(auxiliary_matrix.T[0][1:],
    (auxiliary_matrix.T[0][1:].shape[0], 1))
    multiplier <- column_vector/pivot_number
    fi <- auxiliary_matrix[1:]
    fi <- fi - (multiplier*fj)
        if(count == 0):
                matrix[i+1:] <- fi
        else:
                axiliary_fi <- fi
                while (axiliary_fi.shape[1]+1 <matrix[i+1:].shape[1]):
                        axiliary_fi <-  np.insert(axiliary_fi,
                        0, np.zeros(1), axis=1)
                matrix[i+1:] <- np.insert(axiliary_fi, 0,
                            np.zeros(1), axis=1)
        auxiliary_matrix <- fi.T[1:].T
    dictionary[count+1] <- np.array(matrix)
end while
a <- np.delete(matrix, matrix.shape[1]-1, axis=1)
b <- matrix.T[matrix.shape[1]-1]
return matrix_function.soltion(a,b)

end steppedMethod
```