Matrix Multiplication

Definition

Matrix Multiplication is used to multiply to matrices of the order m $\times$ n and k $\times$ l if the number of columns in the first matrix is the same as the number of rows in the second, i.e., if n = k, and results in a matrix of order m $\times$ l. The product of matrices *A* and *B* is denoted as *AB*.

If the matrices *A* and *B* are as follows

$$A = \begin{bmatrix} a_{11} & a_{12} & \cdots & nbsp; a_{1n} \\ a_{21} & a_{22} & \cdots & nbsp; a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & nbsp; a_{mn} \end{bmatrix}, \quad B = \begin{bmatrix} b_{11} & b_{12} & \cdots & nbsp; b_{1l} \\ b_{21} & b_{22} & \cdots & nbsp; b_{2l} \\ \vdots & \vdots & \ddots & \vdots \\ b_{n1} & b_{n2} & \cdots & nbsp; b_{nl} \end{bmatrix}$$

Then the product matrix *AB* is defined to be

$$AB = \begin{bmatrix} a_{11}b_{11} + \cdots + a_{1n}b_{n1} & a_{11}b_{12} + \cdots + a_{1n}b_{n2} & \cdots & a_{11}b_{1l} + \cdots + a \\ & nbsp; a_{21}b_{11} + \cdots + a_{2n}b_{n1} & a_{21}b_{12} + \cdots + a_{2n}b_{n2} & \cdots \\ \vdots & \vdots & \ddots & \vdots \\ & nbsp; a_{m1}b_{11} + \cdots + a_{mn}b_{n1} & a_{m1}b_{12} + \cdots + a_{mn}b_{n2} & \cdots \end{bmatrix}$$

Solution

A program to multiply two matrices of compatible sizes has been implemented in the following way:

```
def matmul(m1, m2):

#  Checking whether the matrices m1 and m2 are nested iterables

for i in range(len(m1)):

    if (not hasattr(m1[i], '__iter__')):

        raise TypeError  #  If not raising a Type Error

for i in range(len(m2)):

    if (not hasattr(m1[2], '__iter__')):

        raise TypeError  #  If not raising a Type Error

#  Checking for Axis Length Mismatch
```

```
    if(len(m1[0]) != len(m2)):

        raise ValueError  #  If so raising a Value Error

    #  Creating an empty matrix(nested list with elements zero) with the number
of rows of matrix m1 and the number of columns of matrix m2

    productMatrix = [[0 for j in range(len(m2))] for i in range(len(m1))]



    #  Iterating over every row in matrix m1

    for row in range(len(m1)):

        #  Iterating over every column in matrix m2

        for column in range(len(m2[0])):

            #  Iterating over every element in each row of m1 and element of m2,
which have the same number of elements

            for current in range(len(m2)):

                #  Multiplying each element in a given row of m1(row) with the
corresponding elements in a given column(column) of m2

                #  and incrementing the element in the given row and column (in
the 'row'th row and the 'column'th column)of the product matrix by the product
calculated

                productMatrix[row][column] += m1[row][current] * m2[current]
[column]

    #  Returning the product matrix

    return productMatrix
```

After testing for invalid inputs the program iterates over every row in the 1st matrix, every column in the 2nd matrix and every element in each of the rows of the 1st matrix and columns of the 2nd, multiplying and adding them to be stored as each element of the product matrix.

$$AB_{ij} = A_{i1}B_{1j} + A_{i2}B_{2j} + \cdots + A_{in}B_{nj} = \sum_{k=1}^{n} A_{ik}B_{kj},$$

Tested Cases

- If the input are not matrices (2D Nested iterable), lines 3 to 9 check for such inputs and raise a TypeError.
- If any element of the input is non-numeric, Python will automatically raise a TypeError.
- If the axis of the matrices do not match a ValueError is raised at line 14.
- If the lengths of all of the sub elements of either of the matrices do not match a IndexError will be raised by the interpreter when trying to perform multiplication at line 27.
- Testing every single possible extreme input case was not done.