Date 19-sept

21118

Assignment No: 03
Matrix operations

Problem Statement:

Write a python program to compute following computation on matrix:

a> addition of two matrices
b> Substraction of two matrices
c> Multiplication of two matrices
d> Transpose of a matrix

Objectives:

1> Understand concept & operations of matrices
2> Understand use of 2D list to represent matrix & perform various operations

Outcomes:

1> To implement string operations using list data structure in python.
2> To write menu driven, modular program in python
3> To implement user defined functions in the python

Hardware Requirements:

Manufacturer: Acer
Model: Swift SF 314-55G
Processor: Intel(R) Core(TM) i5-8265U CPU @ 1.60 GHz 1.8 GH.
Installed memory (RAM): 8.00 GB (7.85 GB Available)
System Type: 64-bit OS, x 64-based architecture.

Software Requirements:

OS: Windows 10, Home Single language (Version: 1903)
Python Version: 3.8.5
Vs Code (text editor) : Version: 1.49.1 (Aug. 2020 version)

Theory:

Concept:

- Matrix: 2D array of elements ~~having non-zero~~ number of rows & ~~non-zero~~ ~~columns number of~~ colums.

- Matrix operations:

A) Addition: (row major)

i) addition of two matrices is possible only if two matrices have same rows & same columns.

ii) the addition matrix is a one whose element at location (i,j) is the sum of element at location (i,j) for the two matrices.

eg. $A = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$ $B = \begin{bmatrix} 2 & 3 \\ 4 & 5 \end{bmatrix}$

then $A+B = \begin{bmatrix} 3 & 5 \\ 7 & 9 \end{bmatrix}$ = Addition matrix

B) Substraction (row major)

i) Substraction of two matrices is possible only if they have same ~~row~~ rows & same columns.

ii) the substraction matrix is a one whose element at location (i,j) is the substraction of element at location (i,j) for the two matrices

eg $A = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$ $B = \begin{bmatrix} 2 & 3 \\ 4 & 5 \end{bmatrix}$

$A - B = \begin{bmatrix} -1 & -1 \\ -1 & -1 \end{bmatrix}$ = Substraction matrix

c) Multiplication (row major):

i) The multiplication of two matrices A & B is possible only if columns of A & rows of B are equal.

ii) consider two matrices A & B as follows,

$$A = \begin{bmatrix} a_{11} & a_{12} & a_{13} & \cdots & a_{1n} \\ a_{21} & \cdot & \cdot & \cdot & \cdot \\ a_{31} & & & & \\ \vdots & & & & \\ a_{m1} & \cdot & \cdot & \cdot & a_{mn} \end{bmatrix}_{m \times n} \qquad B = \begin{bmatrix} b_{11} & b_{12} & \cdots & b_{1k} \\ b_{21} & \cdot & \cdot & \cdot \\ b_{31} & & & \vdots \\ \vdots & & & \\ b_{n1} & \cdots & \cdots & b_{nk} \end{bmatrix}_{n \times k}$$

the product matrix M is given as.

$$M = \begin{bmatrix} a_{11} b_{11} + a_{12} b_{21} \cdots + a_{m1} \cdot b_{n1} & \cdots \\ & \\ a_{m1} b_{11} + \cdots + a_{mn} \cdot b_{n1} & a_{m1} \cdot b_{1k} + \cdots + a_{mn} b_{nk} \end{bmatrix}_{m \times k}$$

eg. $A = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$ $B = \begin{bmatrix} 3 & 4 \\ 1 & 2 \end{bmatrix}$

product matrix $M = \begin{bmatrix} 5 & 8 \\ 13 & 20 \end{bmatrix}$

d> Transpose:

The transpose of a given matrix is a matrix obtained by interchanging rows & colums.

eg $A = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}$

the transpose of A, $A^T = \begin{bmatrix} 1 & 4 \\ 2 & 5 \\ 3 & 6 \end{bmatrix}$

- class: class is a blueprint of object It provides way to implement various oop concepts.
  eg. data hiding, abstraction.
- object: Object is an instance of class. for same class there can be multiple objects.
- oop concepts:

data hiding: work through function, don't play with actual data.

Abstraction: showing imp details & hiding unneccenary things

- Also basic knowledge of python language, matrix sub data type is required.

ADT:

ADT matrix is

Data Object: A set of rows & columns r, c = {0, ... x} x belongs to tre integer.

Represented as array M of two dimensions with indices as m, & n

$m = \{0, \ldots, ur-1\}$ $n = \{0 \ldots ; uc-1\}$

for each Matrix M, m & n are rows & columns ur, uc are upper bounds of rows & columns.

Create Matrix (ur, uc) : Matrix
// Create matrix of ur x uc.

DisplayMatrix (ur, uc) : Matrix
// ~~Create~~ Display matrix of ur x uc.

AddMatrices (A, ur_A, uc_A, B, ur_B, uc_B) : Matrix
// Adds matrices A & B if possible.

SubMatrices (A , ur_A, uc_A , B , ur_B, uc_B) : Matrix
// Substract matrix B from A if possible.

Prod. Matrix (A, ur_A, uc_A, B, ur_B, uc_B) : Matrix
// Multiply two matrices A & B if possible

Transpose (A, ~~u1~~, u2): Matrix
// finds transpose of matrix A.

Class Declaration:

- class Matrix is declared with number of rows, columns & corresponding ~~&~~ 2D list.
- Constructor used for class is parameterized. if value for rows & colums are provided it creates zero matrix of that particular size.
  else creates empty list.
- Other methods in class are as below:

```
class Matrix():
    def __init__(self, rows=0, columns=0):
        self.rows = rows
        self.colums = colums
        self.mymatrix = [[0] * self.columns for _ in range(self.rows)]

    def readmatrix(self):
        # reads the matrix.

    def addMatrices(self, matrix B): addition Matrix
        # adds matrix ~~with~~ matrix B

    def SubstractMatrices(self, matrix.B): Subtraction Matrix
        # Substract matrixB from matrix (ie. mymatrix)

    def MultiplyMatrices(self, matrix B): product matrix
        # Multiply matrix with matrix_B.

    def Transpose (self) ~~matrix B~~ : Transpose Matrix
        # returns transpose of matrix (ie. mymatrix)
```

Algorithm for each operation

Algorithm addMatrices (A, B)

1. check if rows & columns ~~are~~ ~~eq~~ of A & B are equal or not, if not equal return from method.
2. create matrix C. of order same as A
3. for each element at index (i, j) of matrix C, assign this element with sum of elements at same index in A & B.
4. return matrix C.

Algorithm subMatrices (A, B)

1. check if rows & columns of A & B are equal or not if not equal return from method.
2. create matrix C of order same as A.
3. for each element at index (i, j) of matrix C assign this element with difference of elements at same index in A & B
4. return matrix C.

Algorithm Mult.Matrices (A, B):

1. check if columns of A are equals to rows of B ~~or~~ If equal proceed to step 2 else return from ~~message~~ method.
2. create matrix C of order ~~same~~ as row of A X column of
3. multiply ith row (of A) with jth column of B & assign the result to the ~~eleme~~ index (i, j) in matrix C.
4. return matrix C.

Algorithm Transpose (A):

1. create matrix AT of order column of A X row of A

2. for each element at index (i,j) in A, assign it to the index (j,i) in matrix A^T.
3. return matrix A^T.

Analysis of Algorithms:

Algorithm	Time Complexity	Space Complexity
1) addMatrices()	~~T(m,n)=~~ O(mxn) where m,n are rows & columns respectively.	space is required to store the elements of addition matrix. ∴ space complexity is O(mxn) where m,n are rows, columns resp.
2) SubMatrices()	O(mxn) where m,n are rows & columns respectively	space complexity is O(mxn) where m,n are rows & columns respectively
3) MultMatrices()	let $A_{m \times n}$ & $B_{n \times p}$ then time complexity will be O(mnp)	space complexity is O(mxp) where $A_{m \times n}$ & $B_{n \times p}$.
4) Transpose()	O(mxn) where m,n are rows & columns of matrix	O(mxn) where m,n are rows & columns of matrix.

Test cases:

A, B are input matrices, C is output matrix.

Test case no.	Input given	expected output	Actual output
1)	$A = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}$ $B = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}$ for addition	$C = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}$	$C = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}$
2)	$A = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$ $B = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}$ for addition	Addition not possible.	Addition not possible
3)	$A = \begin{bmatrix} -1 & 2 & 3 \\ -4 & 5 & -6 \end{bmatrix}$ $B = \begin{bmatrix} 4 & 5 & -6 \\ 1 & 2 & -3 \end{bmatrix}$ for addition	$C = \begin{bmatrix} 3 & 7 & -3 \\ -3 & 7 & -9 \end{bmatrix}$	$C = \begin{bmatrix} 3 & 7 & -3 \\ -3 & 7 & -9 \end{bmatrix}$
4)	for transpose $A = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$	$A^T = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$	$A^T = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$
5)	for transpose $A = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 3 \end{bmatrix}$	$A^T = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 3 \end{bmatrix}$	$A^T = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 3 \end{bmatrix}$
6)	for transpose $A = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}$	$A^T = \begin{bmatrix} 1 & 4 \\ 2 & 5 \\ 3 & 6 \end{bmatrix}$	$A^T = \begin{bmatrix} 1 & 4 \\ 2 & 5 \\ 3 & 6 \end{bmatrix}$
7)	for multiplication		

$A = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$ $B = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$	$C = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$	$C = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$
8) $A = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}_{2\times2}$ $B = \begin{bmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{bmatrix}_{3\times2}$	Multiplication not possible	Multiplication not possible
9) $A = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}_{2\times3}$ $B = \begin{bmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{bmatrix}_{3\times2}$	$C = \begin{bmatrix} 22 & 28 \\ 49 & 64 \end{bmatrix}_{2\times2}$	$C = \begin{bmatrix} 22 & 28 \\ 49 & 64 \end{bmatrix}$

Applications

Applications of matrices are found in most scientific fields.

In every branch of physics, including classical mechanics optics, electromagnetism, quantum mechanics & quantum electrodynamics, they are used to study physical phenomenon such as the motion of rigid bodies.

Conclusion:

At the end of this assignment I'm able to perform various matrix operations using the computer with help of python program. Got better understanding of matrix operations & developed programming logic.