Adamson University
College of Engineering
Computer Engineering Department

Linear Algebra

Laboratory Activity No. 6

Matrices

Submitted by:
Chipongian, John Patrick Ryan J.

Instructor:
Engr. Dylan Josh D. Lopez

December 1, 2020

I. Objectives

This laboratory activity aims to implement the principles and techniques of using operators in matrices and other higher dimensions. It also shows the relationship of matrices in linear equations.

II. Methods

The methods used in completing this activity came from the numpy library and the built in functions of python. The functions that are used are np.array(), np.sum(), shape(), and np.ndim. By using these functions and other built in functions, a new function is created to determine the values being fined.

$$matrixA = \begin{bmatrix} 1 & 2 & 4 \\ 4 & 8 & 0 \\ 6 & 7 & 5 \end{bmatrix}$$

$$matrixB = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

$$matrixC = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

$$matrixD = \begin{bmatrix} 2 & 3 & 6 \\ 5 & 9 & 7 \\ 1 & 2 & 3 \end{bmatrix}$$

$$matrixE = \begin{bmatrix} 9 & 9 & 9 \\ 9 & 9 & 9 \\ 9 & 9 & 9 \end{bmatrix}$$

Figure 1 Matrices to be used in the function in task 1

Figure 1 shows the matrices that would be defined and be used in showing the results using the function that would be created in task 1.

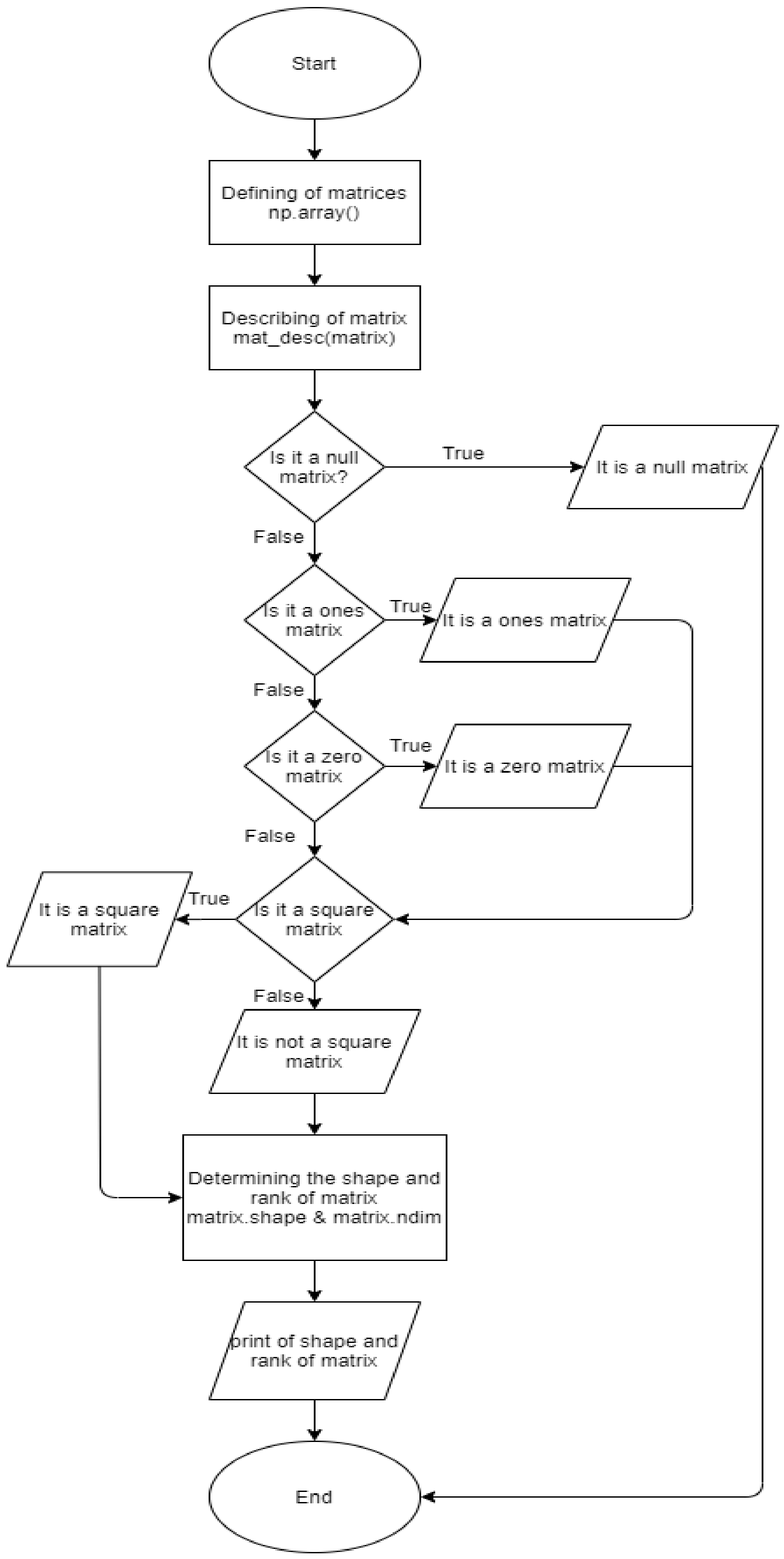


Figure 2 Flowchart of the function mat_desc()

$$matrix1 = \begin{bmatrix} 3 & 1 & 6 \\ 8 & 0 & 2 \\ 9 & 7 & 6 \end{bmatrix}$$

$$matrix2 = \begin{bmatrix} 1 & 1 & 1 \\ 2 & 2 & 2 \\ 3 & 3 & 3 \end{bmatrix}$$

$$matrix3 = \begin{bmatrix} 2 & 1 & 1 \\ 2 & 2 & 2 \\ 2 & 1 & 1 \end{bmatrix}$$

$$matrix4 = \begin{bmatrix} 6 & 5 & 4 & 2 \\ 5 & 4 & 1 & 2 \\ 7 & 5 & 3 & 2 \end{bmatrix}$$

$$matrix5 = \begin{bmatrix} 5 & 6 & 7 \\ 5 & 6 & 7 \\ 5 & 6 & 7 \\ 5 & 6 & 7 \end{bmatrix}$$

Figure 3 Matrices to be used in the function in task 2

Figure 1 shows the matrices that would be defined and be used in showing the results using the function that would be created in task 2.

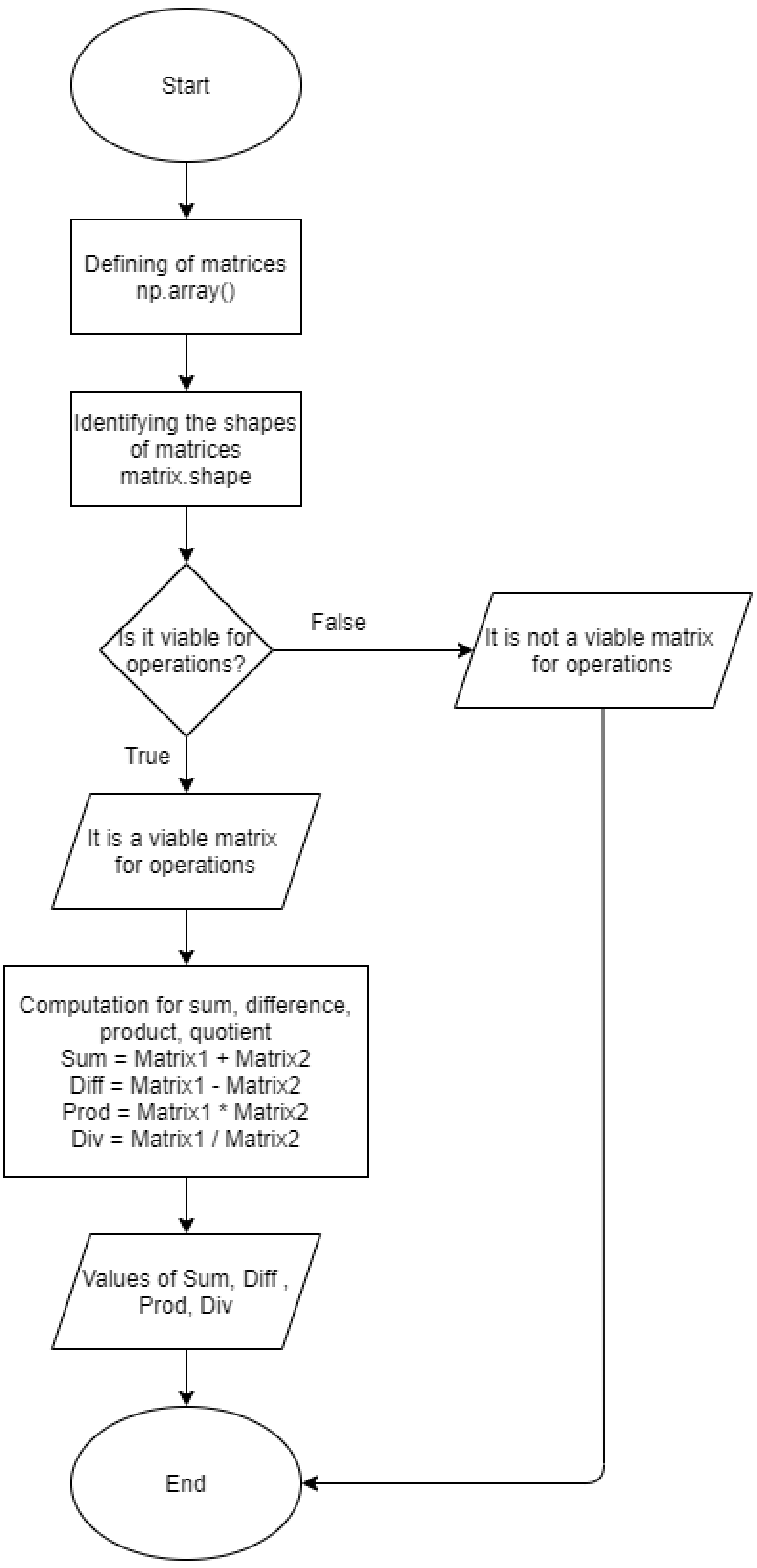
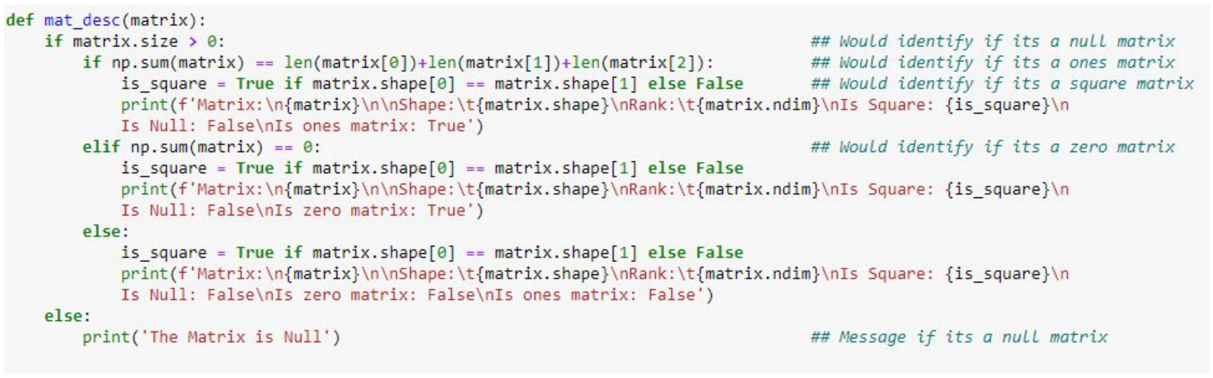

Start
Defining of matrices np.array()
Identifying the shapes of matrices matrix.shape
Is it viable for operations?
False
It is not a viable matrix for operations
True
It is a viable matrix for operations
Computation for sum, difference, product, quotient
Sum = Matrix1 + Matrix2
Diff = Matrix1 - Matrix2
Prod = Matrix1 * Matrix2
Div = Matrix1 / Matrix2
Values of Sum, Diff , Prod, Div
End

Figure 4 Flowchart of the function mat_operations()

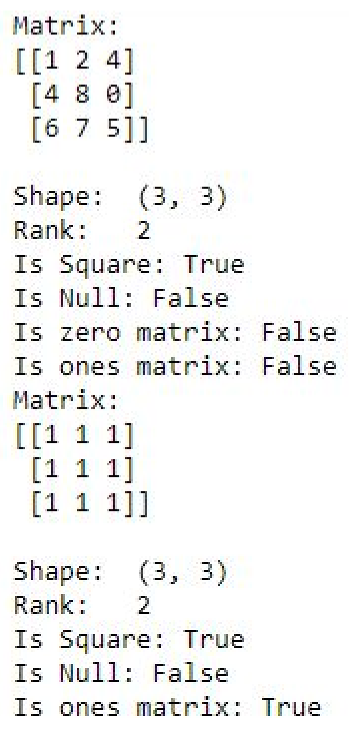
III. Results



```python
def mat_desc(matrix):
    if matrix.size > 0:                                                       ## Would identify if its a null matrix
        if np.sum(matrix) == len(matrix[0])+len(matrix[1])+len(matrix[2]):    ## Would identify if its a ones matrix
            is_square = True if matrix.shape[0] == matrix.shape[1] else False  ## Would identify if its a square matrix
            print(f'Matrix:\n{matrix}\n\nShape:\t{matrix.shape}\nRank:\t{matrix.ndim}\nIs Square: {is_square}\n
            Is Null: False\nIs ones matrix: True')
        elif np.sum(matrix) == 0:                                             ## Would identify if its a zero matrix
            is_square = True if matrix.shape[0] == matrix.shape[1] else False
            print(f'Matrix:\n{matrix}\n\nShape:\t{matrix.shape}\nRank:\t{matrix.ndim}\nIs Square: {is_square}\n
            Is Null: False\nIs zero matrix: True')
        else:
            is_square = True if matrix.shape[0] == matrix.shape[1] else False
            print(f'Matrix:\n{matrix}\n\nShape:\t{matrix.shape}\nRank:\t{matrix.ndim}\nIs Square: {is_square}\n
            Is Null: False\nIs zero matrix: False\nIs ones matrix: False')
    else:
        print('The Matrix is Null')                                           ## Message if its a null matrix
```

Figure 5 Code of the function mat_desc() in task 1

Figure 5 shows the codes of the function mat_desc() that was used in describing the given matrices. Firstly, defining of matrices is done using the function np.array(), then proceed in describing the matrices using the function. The matrix.size > 0 would determine if the matrix is a null or not, if that argument is true it would proceed to the next step while if its false then it would instantly end the sequence and say that it is a null matrix. Next is, identifying if the matrix is a ones matrix. This is done by comparing if the sum of the matrix, which is done by [1] np.sum(), is equal to the number of elements that the matrix has. If the argument is true then it would continue to the statements inside it while if it is false then it would go to the other statement. The other statement is determining if it is a zero matrix. This would be determined by getting the sum of the matrix, which is done by [1] np.sum(), and compare it if its equal to 0. If this is true then it would be identified as a zero matrix and continue in the statements inside but if it is false then it would go to the next statement. The next statement would not be defined as a one or zero matrix since it produces a false in the statement in determining if it is. The conditions in the inner if statements determine if it is a square matrix by comparing the shape of the matrix using matrix.shape. Lastly, The other conditions print the [2] shape, by matrix.shape, [3] rank, by using matrix.ndim, and if it is a null, one or zero matrix.

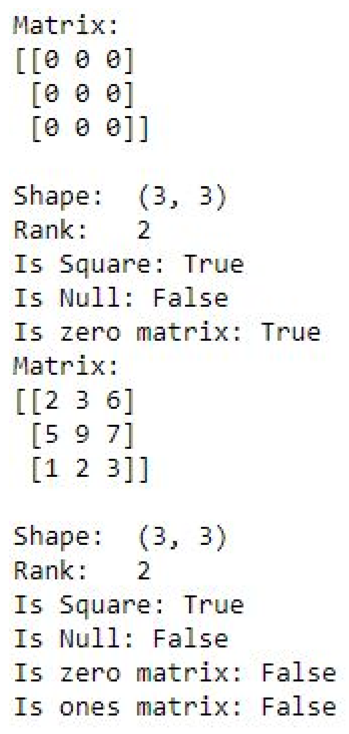


```
Matrix:
[[1 2 4]
 [4 8 0]
 [6 7 5]]

Shape:  (3, 3)
Rank:   2
Is Square: True
Is Null: False
Is zero matrix: False
Is ones matrix: False
Matrix:
[[1 1 1]
 [1 1 1]
 [1 1 1]]

Shape:  (3, 3)
Rank:   2
Is Square: True
Is Null: False
Is ones matrix: True
```

Figure 6 Output of function mat_desc() 1

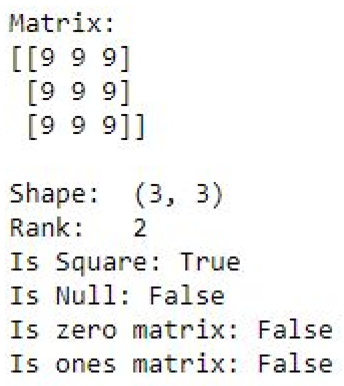


```
Matrix:
[[0 0 0]
 [0 0 0]
 [0 0 0]]

Shape:  (3, 3)
Rank:   2
Is Square: True
Is Null: False
Is zero matrix: True
Matrix:
[[2 3 6]
 [5 9 7]
 [1 2 3]]

Shape:  (3, 3)
Rank:   2
Is Square: True
Is Null: False
Is zero matrix: False
Is ones matrix: False
```

Figure 7 Output of function mat_desc() 2

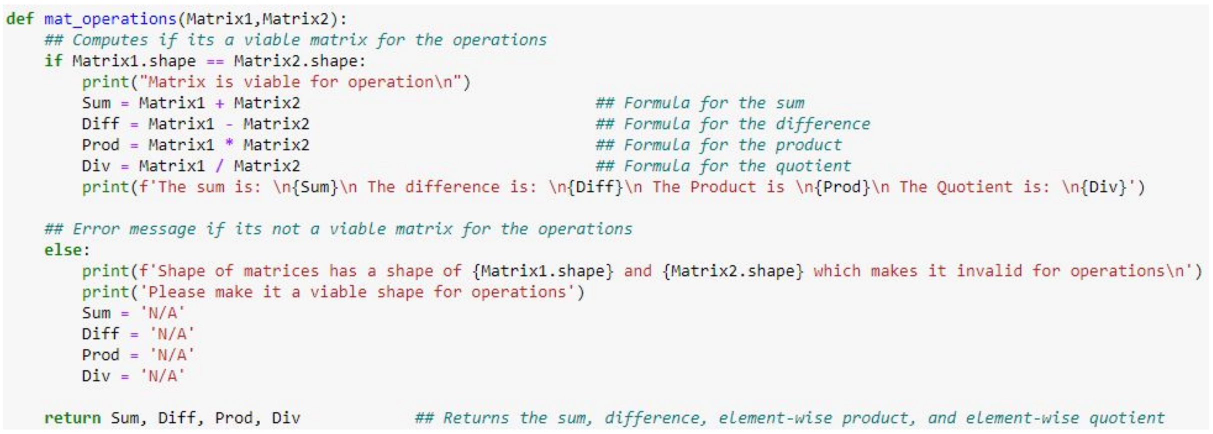


```
Matrix:
[[9 9 9]
 [9 9 9]
 [9 9 9]]

Shape:  (3, 3)
Rank:   2
Is Square: True
Is Null: False
Is zero matrix: False
Is ones matrix: False
```

Figure 8 Output of function mat_desc() 3

Figures 6, 7, and 8 show the output in using the function mat_desc() on the defined matrices earlier on. This shows the matrix itself, the shape of the matrix, the rank of the matrix, if it is a square matrix, if it is a null matrix, if it is a zero matrix, if it is a one matrix.



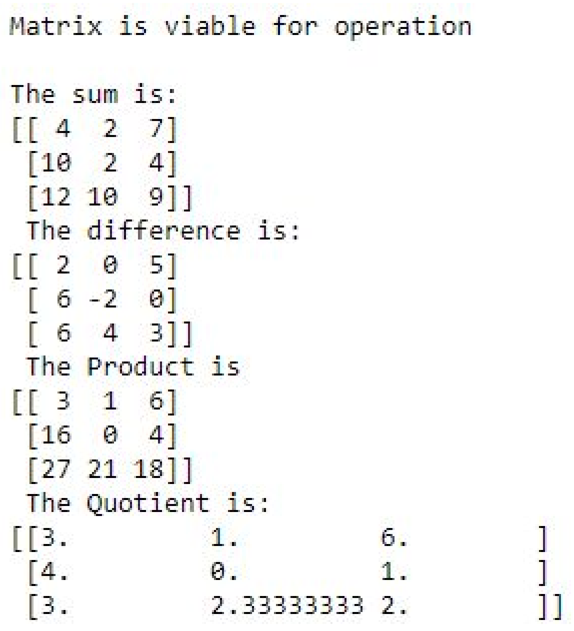
```python
def mat_operations(Matrix1,Matrix2):
    ## Computes if its a viable matrix for the operations
    if Matrix1.shape == Matrix2.shape:
        print("Matrix is viable for operation\n")
        Sum = Matrix1 + Matrix2                               ## Formula for the sum
        Diff = Matrix1 - Matrix2                              ## Formula for the difference
        Prod = Matrix1 * Matrix2                              ## Formula for the product
        Div = Matrix1 / Matrix2                               ## Formula for the quotient
        print(f'The sum is: \n{Sum}\n The difference is: \n{Diff}\n The Product is \n{Prod}\n The Quotient is: \n{Div}')

    ## Error message if its not a viable matrix for the operations
    else:
        print(f'Shape of matrices has a shape of {Matrix1.shape} and {Matrix2.shape} which makes it invalid for operations\n')
        print('Please make it a viable shape for operations')
        Sum = 'N/A'
        Diff = 'N/A'
        Prod = 'N/A'
        Div = 'N/A'

    return Sum, Diff, Prod, Div            ## Returns the sum, difference, element-wise product, and element-wise quotient
```

Figure 9 Code of the function mat_operations() in task 2

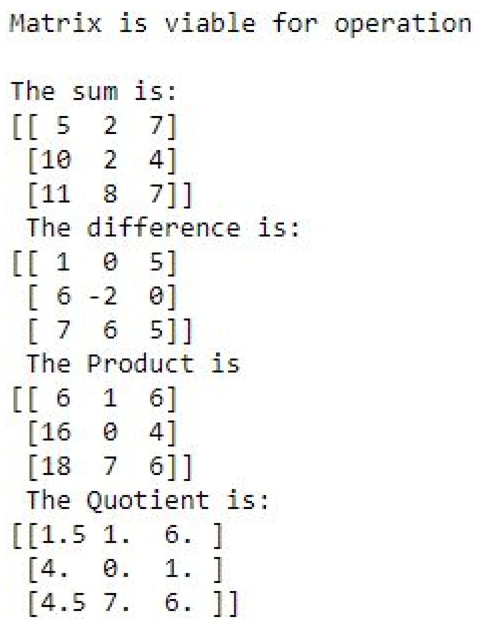
Figure 9 shows the codes of the function mat_operations() that is used in determining if the matrix is viable for operations then to perform operations into the two matrices. Firstly, the function determines if the matrices are both viable for operations. [2] This is known if the shape of the two matrices are equal, which is done using the shape function. If the matrix is both viable for operations, it says that it is viable for operations then proceeds to computing the sum, difference, product, and the quotient of the two matrices. After that, it would show the values of sum, difference, product, and the quotient and return the values of it. If it is not viable for operations, it says that the shape of the two matrices is not valid and would request to give a viable shape of the matrix. After that, it would return the values of sum, difference, product, and the quotient, which is zero or none since it is not viable for operations.



```
Matrix is viable for operation

The sum is:
[[ 4  2  7]
 [10  2  4]
 [12 10  9]]
 The difference is:
[[ 2  0  5]
 [ 6 -2  0]
 [ 6  4  3]]
 The Product is
[[ 3  1  6]
 [16  0  4]
 [27 21 18]]
 The Quotient is:
[[3.         1.         6.        ]
 [4.         0.         1.        ]
 [3.         2.33333333 2.        ]]
```

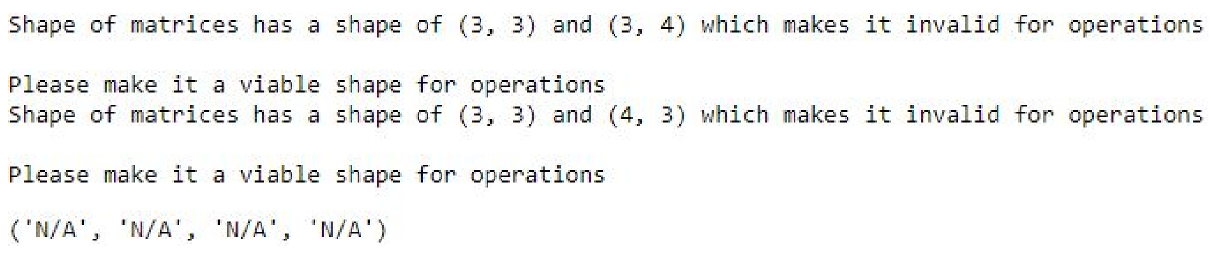
Figure 10 Output of function mat_operations() 1



```
Matrix is viable for operation

The sum is:
[[ 5  2  7]
 [10  2  4]
 [11  8  7]]
 The difference is:
[[ 1  0  5]
 [ 6 -2  0]
 [ 7  6  5]]
 The Product is
[[ 6  1  6]
 [16  0  4]
 [18  7  6]]
 The Quotient is:
[[1.5 1.  6. ]
 [4.  0.  1. ]
 [4.5 7.  6. ]]
```

Figure 11 Output of function mat_operations() 1



```
Shape of matrices has a shape of (3, 3) and (3, 4) which makes it invalid for operations

Please make it a viable shape for operations
Shape of matrices has a shape of (3, 3) and (4, 3) which makes it invalid for operations

Please make it a viable shape for operations

('N/A', 'N/A', 'N/A', 'N/A')
```

Figure 12 Output of function mat_operations() 1

Figures 10, 11, and 12 show the output in using the function mat_operations() on the new matrices that were defined for task 2. The function says if the matrices are viable for operations. It also returns the values of the sum, difference, product and quotient. And also shows its own error message when the matrix is not viable for operations.

IV. Conclusion

In conclusion, describing a matrix can be done by creating a new function that would show the rank, shape and other more values that are needed for a matrix. The same can be said in using operations in matrix. A different approach may be needed but it would produce the same values.

Matrix operations would be a huge help in solving the problems of agriculture. A sample of this is when determining what crop would be best to plant, the best soil for the crops to grow and many more. For example in crops, one of the matrices is the results of harvest of one crop while the other matrix is the results of harvest of the other crop. Using these matrices, we would be able to compare and to determine what crop is better and would produce more harvests.

References

[1] Numpy.org. 2020. *Numpy.Sum — Numpy V1.21.Dev0 Manual*. [online] Available at: <https://numpy.org/devdocs/reference/generated/numpy.sum.html> [Accessed 1 December 2020].
[2] Numpy.org. 2020. *Numpy.Shape — Numpy V1.21.Dev0 Manual*. [online] Available at: <https://numpy.org/devdocs/reference/generated/numpy.shape.html> [Accessed 1 December 2020].
[3] Numpy.org. 2020. *Numpy.Ndarray.Ndim — Numpy V1.19 Manual*. [online] Available at: <https://numpy.org/doc/stable/reference/generated/numpy.ndarray.ndim.html> [Accessed 1 December 2020].