Adamson University
College of Engineering
Computer Engineering Department

Linear Algebra

Laboratory Activity No. 6

Matrices

Submitted by:

Parco, Jovian Asher G.

Instructor:

Engr. Dylan Josh D. Lopez

November 28, 2020

I. Objectives

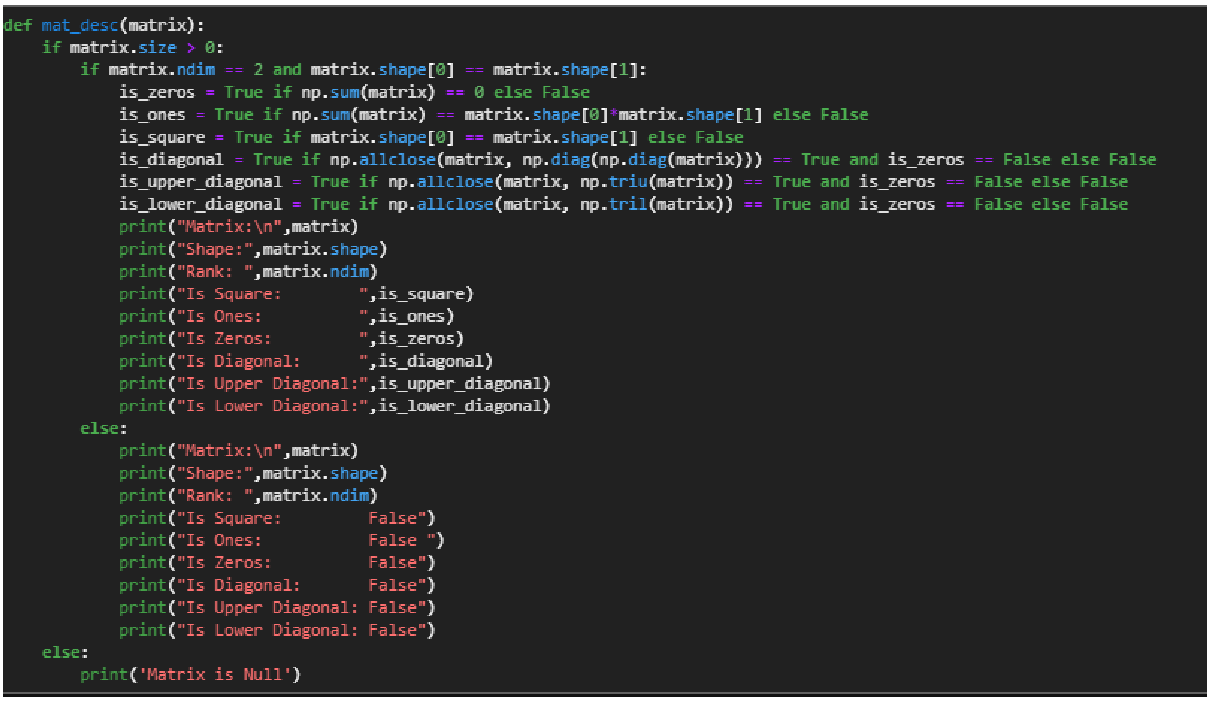
This laboratory activity aims to implement matrix operations and matrix identities using Python's Numpy module.

II. Methods

In this laboratory activity, the practices consist of creating a function that describes a matrix identity such as shape() function, size() function, ndim() function and, combinations of functions to identify different identity, also matrix operations such as numpy.sum() & numpy.diff() and element wise operation such as numpy.multiply() & numpy.divide().

III. Results

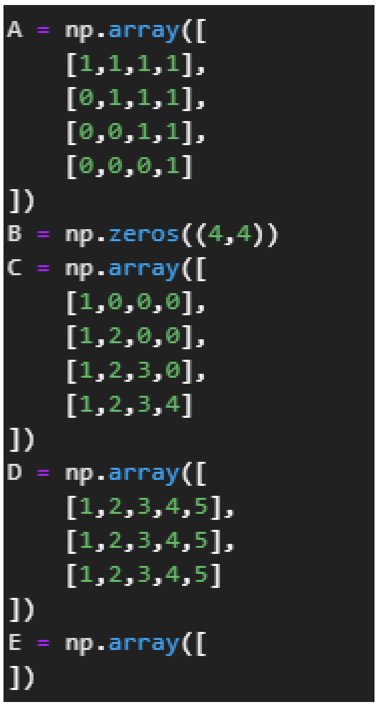
The complete repository for the activity is available at Github (https://bit.ly/2JfWTVd). The lab activity consists of 2 parts; the first part is creating a function that describes a matrix, while the second part takes two arguments (matrix/scalar) and run it through different types of matrix operation.



```python
def mat_desc(matrix):
    if matrix.size > 0:
        if matrix.ndim == 2 and matrix.shape[0] == matrix.shape[1]:
            is_zeros = True if np.sum(matrix) == 0 else False
            is_ones = True if np.sum(matrix) == matrix.shape[0]*matrix.shape[1] else False
            is_square = True if matrix.shape[0] == matrix.shape[1] else False
            is_diagonal = True if np.allclose(matrix, np.diag(np.diag(matrix))) == True and is_zeros == False else False
            is_upper_diagonal = True if np.allclose(matrix, np.triu(matrix)) == True and is_zeros == False else False
            is_lower_diagonal = True if np.allclose(matrix, np.tril(matrix)) == True and is_zeros == False else False
            print("Matrix:\n",matrix)
            print("Shape:",matrix.shape)
            print("Rank: ",matrix.ndim)
            print("Is Square:        ",is_square)
            print("Is Ones:          ",is_ones)
            print("Is Zeros:         ",is_zeros)
            print("Is Diagonal:      ",is_diagonal)
            print("Is Upper Diagonal:",is_upper_diagonal)
            print("Is Lower Diagonal:",is_lower_diagonal)
        else:
            print("Matrix:\n",matrix)
            print("Shape:",matrix.shape)
            print("Rank: ",matrix.ndim)
            print("Is Square:         False")
            print("Is Ones:           False ")
            print("Is Zeros:          False")
            print("Is Diagonal:       False")
            print("Is Upper Diagonal: False")
            print("Is Lower Diagonal: False")
    else:
        print('Matrix is Null')
```

Figure 1 Function for Part 1

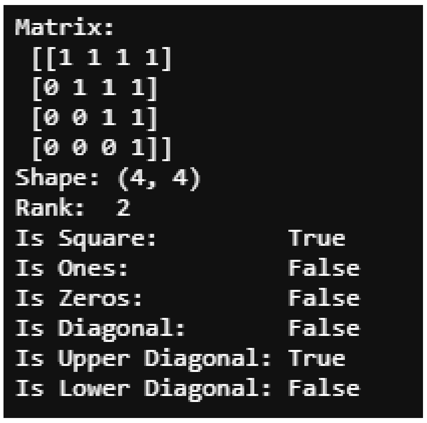
For part 1 of the lab activity, the researcher created a function called "mat_desc" (Figure 1); it is a simple function that takes one parameter that is a system of linear equation in a matrix form and identifies its shapes, rank, size, different identities such as upper diagonal, lower diagonal and many more. For the code in the second line, the researcher created an “if / else” condition where the matrix checks if the size of the matrix is more than one by using the size() function, if the matrix is confirmed to have more than one size of matrix it run to the next code if not the function outputs “Matrix is Null” & terminates the program, for the third line of the code is also an “if / else” where the matrix is checked if the matrix dimension is two using ndim() function and check if the shape is a square matrix using the shape() function, if the function failed to meet the argument it goes to the else statement where it only outputs its matrix shape, matrix dimension, and all the matrix identities are set to false, but if the statement is met it runs through all the arguments that describe matrix identity and form using different combinations of Numpy functions.



```
A = np.array([
    [1,1,1,1],
    [0,1,1,1],
    [0,0,1,1],
    [0,0,0,1]
])
B = np.zeros((4,4))
C = np.array([
    [1,0,0,0],
    [1,2,0,0],
    [1,2,3,0],
    [1,2,3,4]
])
D = np.array([
    [1,2,3,4,5],
    [1,2,3,4,5],
    [1,2,3,4,5]
])
E = np.array([
])
```

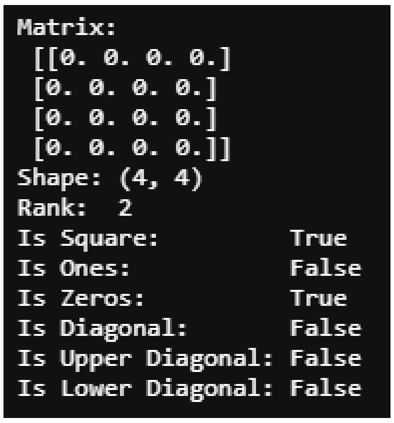
Figure 2 Sample Matrices for part 1

Using the function that the researcher created it describes a set of matrices (Figure 2) that have different identities and form, which if the function takes one argument in the set of matrices (Figure 2) it outputs all the following:



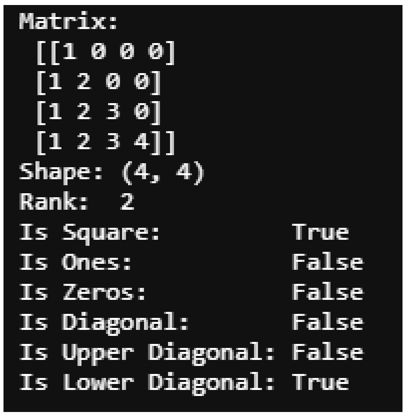
```
Matrix:
 [[1 1 1 1]
 [0 1 1 1]
 [0 0 1 1]
 [0 0 0 1]]
Shape: (4, 4)
Rank:  2
Is Square:          True
Is Ones:            False
Is Zeros:           False
Is Diagonal:        False
Is Upper Diagonal: True
Is Lower Diagonal: False
```

Figure 3 Output for Part 1 Using the “A” Arguments



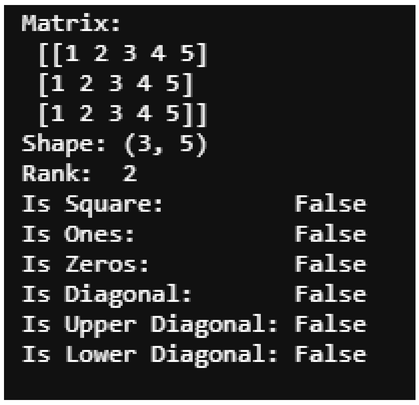
```
Matrix:
 [[0. 0. 0. 0.]
 [0. 0. 0. 0.]
 [0. 0. 0. 0.]
 [0. 0. 0. 0.]]
Shape: (4, 4)
Rank:  2
Is Square:          True
Is Ones:            False
Is Zeros:           True
Is Diagonal:        False
Is Upper Diagonal: False
Is Lower Diagonal: False
```

Figure 4 Output for Part 1 Using the “B” Arguments



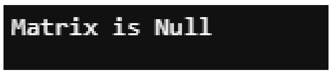
```
Matrix:
 [[1 0 0 0]
 [1 2 0 0]
 [1 2 3 0]
 [1 2 3 4]]
Shape: (4, 4)
Rank:  2
Is Square:          True
Is Ones:            False
Is Zeros:           False
Is Diagonal:        False
Is Upper Diagonal: False
Is Lower Diagonal: True
```

Figure 5 Output for Part 1 Using the “C” Arguments



```
Matrix:
 [[1 2 3 4 5]
 [1 2 3 4 5]
 [1 2 3 4 5]]
Shape: (3, 5)
Rank:  2
Is Square:         False
Is Ones:           False
Is Zeros:          False
Is Diagonal:       False
Is Upper Diagonal: False
Is Lower Diagonal: False
```

Figure 6 Output for Part 1 Using the “D” Arguments



```
Matrix is Null
```

Figure 7 Output for Part 1 Using the “E” Arguments

Using the sample matrices, the researcher describes all the matrices with the preview of the matrix in the top to confirm it manually.

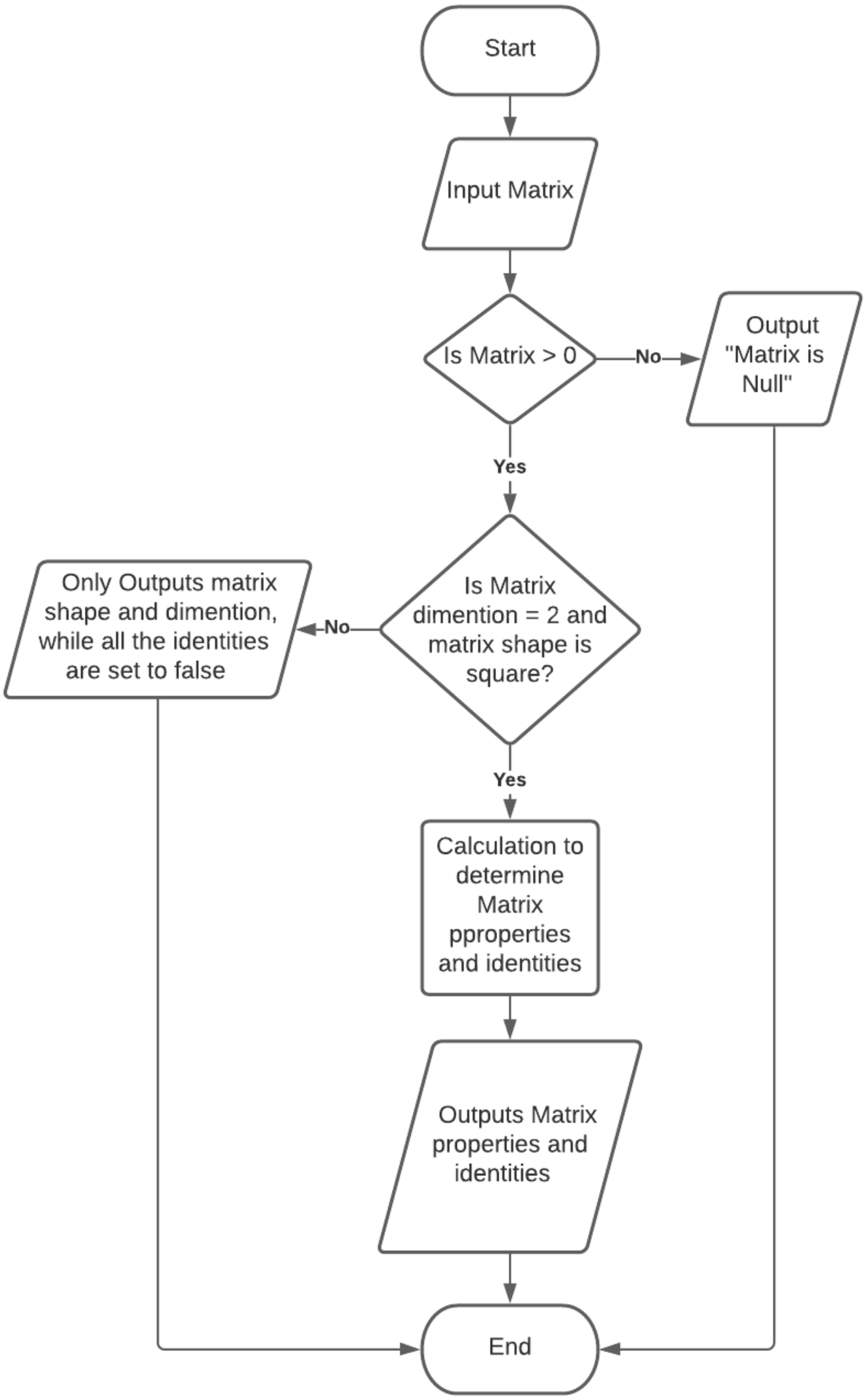

Start
Input Matrix
Is Matrix > 0
No
Output "Matrix is Null"
Yes
Is Matrix dimention = 2 and matrix shape is square?
No
Only Outputs matrix shape and dimention, while all the identities are set to false
Yes
Calculation to determine Matrix pproperties and identities
Outputs Matrix properties and identities
End

Figure 8 Flowchart for Part1

The researcher created a simple flowchart to describe or summarize the function created.

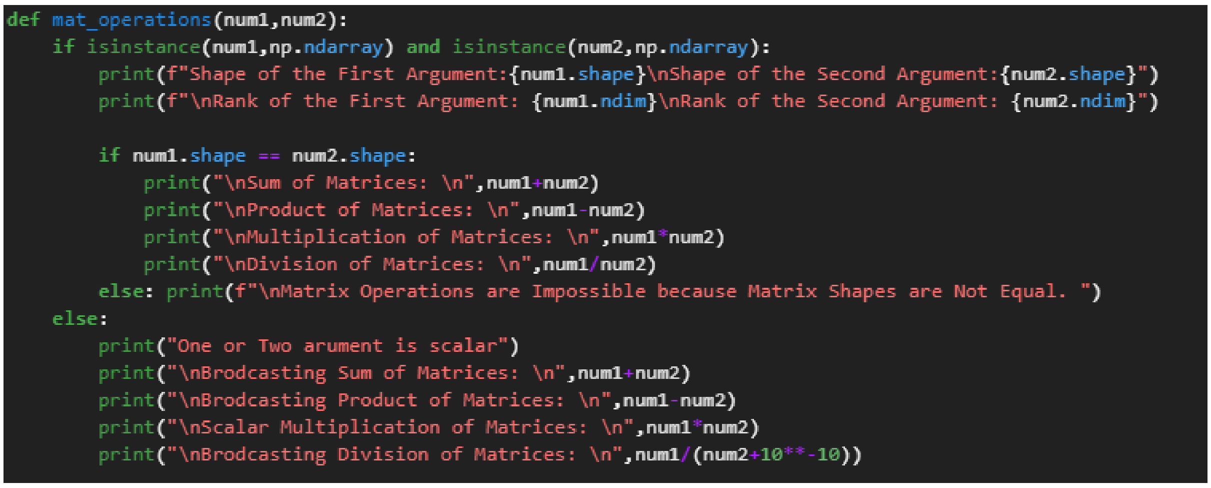

```
def mat_operations(num1,num2):
    if isinstance(num1,np.ndarray) and isinstance(num2,np.ndarray):
        print(f"Shape of the First Argument:{num1.shape}\nShape of the Second Argument:{num2.shape}")
        print(f"\nRank of the First Argument: {num1.ndim}\nRank of the Second Argument: {num2.ndim}")

        if num1.shape == num2.shape:
            print("\nSum of Matrices: \n",num1+num2)
            print("\nProduct of Matrices: \n",num1-num2)
            print("\nMultiplication of Matrices: \n",num1*num2)
            print("\nDivision of Matrices: \n",num1/num2)
        else: print(f"\nMatrix Operations are Impossible because Matrix Shapes are Not Equal. ")
    else:
        print("One or Two arument is scalar")
        print("\nBrodcasting Sum of Matrices: \n",num1+num2)
        print("\nBrodcasting Product of Matrices: \n",num1-num2)
        print("\nScalar Multiplication of Matrices: \n",num1*num2)
        print("\nBrodcasting Division of Matrices: \n",num1/(num2+10**-10))
```

Figure 9 Function for Part 2

For part 2 of the lab activity, the researcher created another function (Figure 8) this time; it takes two parameters that can be either matrices or scalars and will output matrix operation such as the sum of the matrix, difference of matrix, element-wise multiplication of the matrix, and element-wise division of the matrix. The second line of the researcher created an “if / else” statement where it uses the isinstance() function that checks if the type of the inputted argument is the same array type or matrix and will run a simple function describes the matrix. Simultaneously, if it is not the same, it will output “One or Two arguments is scalar” but still run all the operation as a broadcasting operation. If we continue, there is one last “if / else” statement, where it checks the shape of the matrix if it is equal using the shape()function; if not, it will output “Matrix Operations are Impossible because Matrix Shapes are Not Equal.” However, if it is equal, it will run all the matrix operations that are mention above.

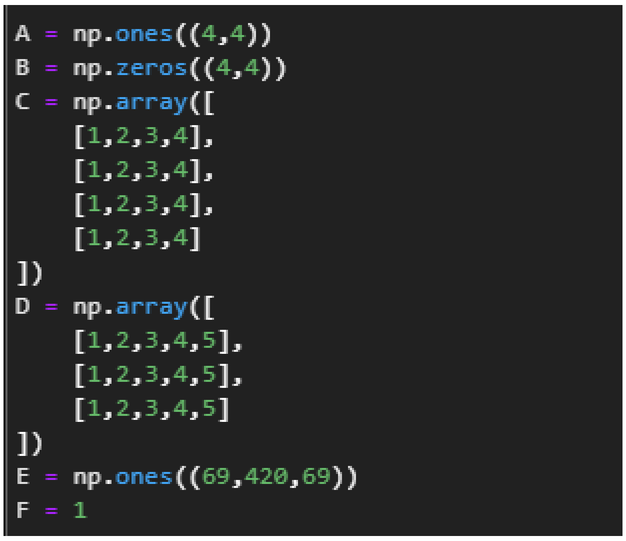
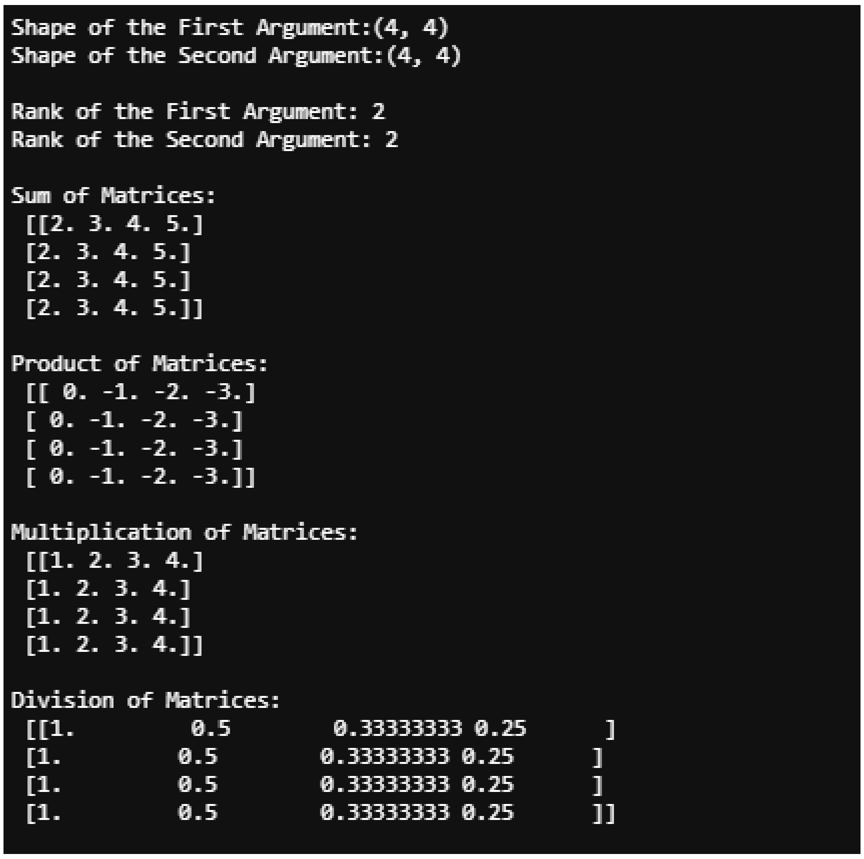

```
A = np.ones((4,4))
B = np.zeros((4,4))
C = np.array([
    [1,2,3,4],
    [1,2,3,4],
    [1,2,3,4],
    [1,2,3,4]
])
D = np.array([
    [1,2,3,4,5],
    [1,2,3,4,5],
    [1,2,3,4,5]
])
E = np.ones((69,420,69))
F = 1
```

Figure 10 Sample Matrices and Scalar for part 2

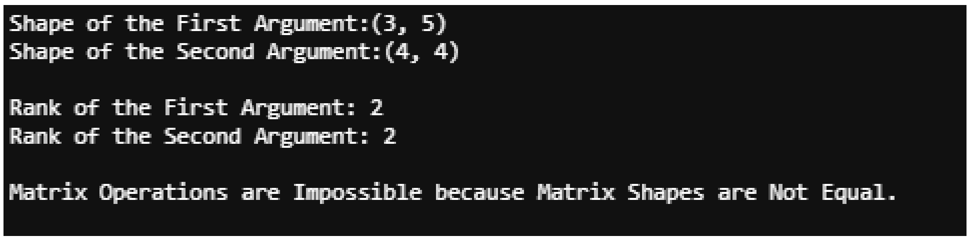
Using the function that the researcher created it describes a set of matrices and scalar (Figure 8) that have different identities and form, which if the function takes two arguments in the set of matrices or scalar (Figure 8).



```
Shape of the First Argument:(4, 4)
Shape of the Second Argument:(4, 4)

Rank of the First Argument: 2
Rank of the Second Argument: 2

Sum of Matrices:
 [[2. 3. 4. 5.]
 [2. 3. 4. 5.]
 [2. 3. 4. 5.]
 [2. 3. 4. 5.]]

Product of Matrices:
 [[ 0. -1. -2. -3.]
 [ 0. -1. -2. -3.]
 [ 0. -1. -2. -3.]
 [ 0. -1. -2. -3.]]

Multiplication of Matrices:
 [[1. 2. 3. 4.]
 [1. 2. 3. 4.]
 [1. 2. 3. 4.]
 [1. 2. 3. 4.]]

Division of Matrices:
 [[1.         0.5        0.33333333 0.25      ]
 [1.         0.5        0.33333333 0.25      ]
 [1.         0.5        0.33333333 0.25      ]
 [1.         0.5        0.33333333 0.25      ]]
```

Figure 11 Output for Part 2 Using the “A,C” Arguments

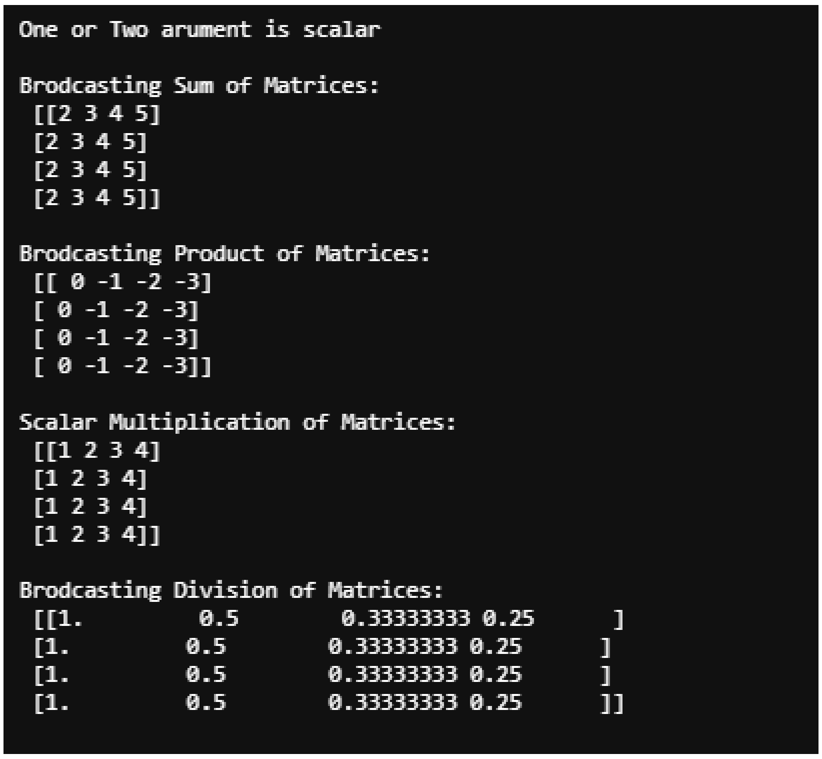
For the output(Figure 11) it shows that the inputted argument in the function is matrices, which first describes both the matrix by shape and dimensions and proceed to do al matric operations.



```
Shape of the First Argument:(3, 5)
Shape of the Second Argument:(4, 4)

Rank of the First Argument: 2
Rank of the Second Argument: 2

Matrix Operations are Impossible because Matrix Shapes are Not Equal.
```

Figure 12 Output for Part 2 Using the “D,C” Arguments

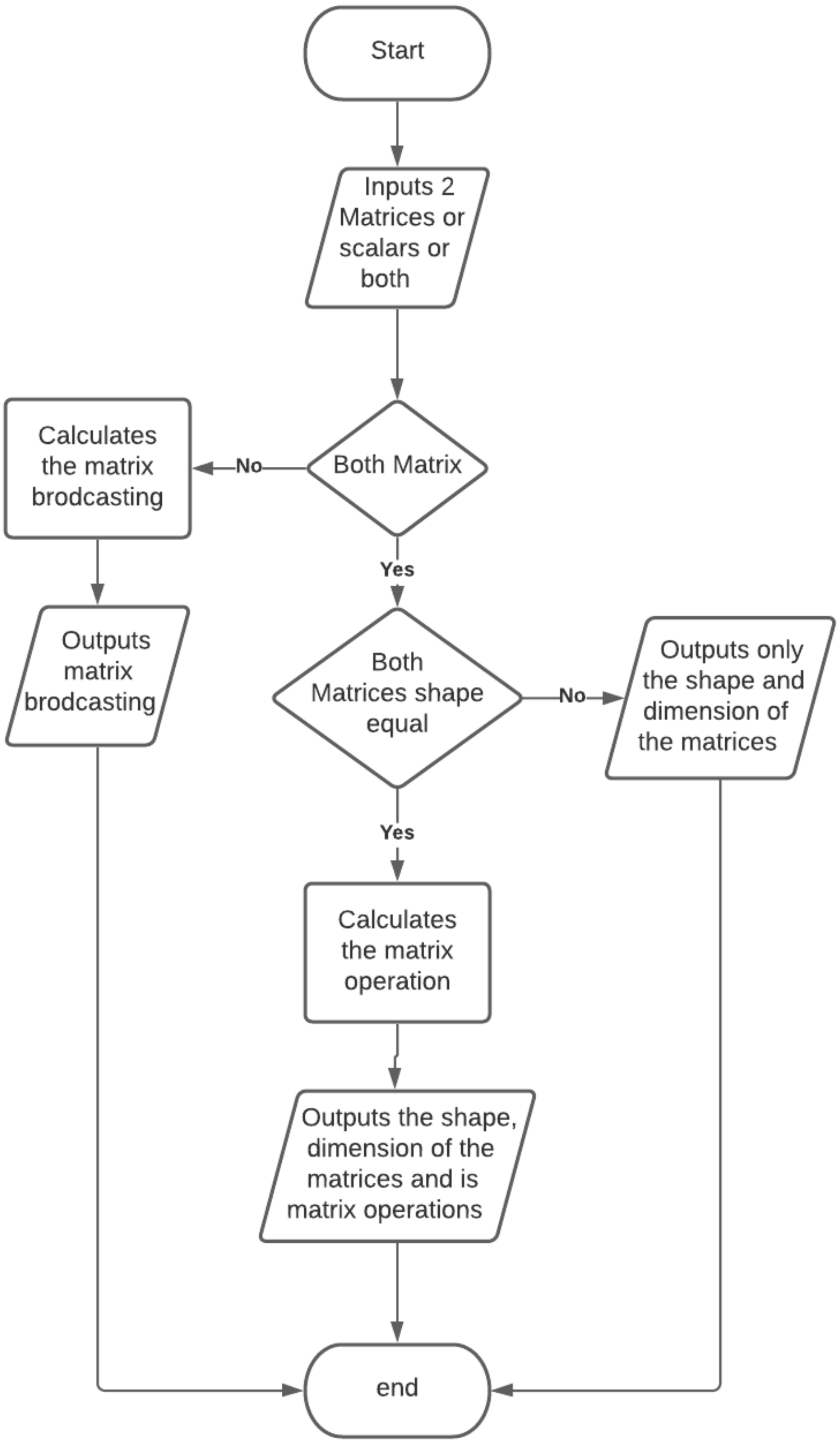
For the output(Figure 12), it shows that both inputted arguments are matrices, but it’s not the same shapes, so matrix operation is not possible.



```
One or Two arument is scalar

Brodcasting Sum of Matrices:
 [[2 3 4 5]
 [2 3 4 5]
 [2 3 4 5]
 [2 3 4 5]]

Brodcasting Product of Matrices:
 [[ 0 -1 -2 -3]
 [ 0 -1 -2 -3]
 [ 0 -1 -2 -3]
 [ 0 -1 -2 -3]]

Scalar Multiplication of Matrices:
 [[1 2 3 4]
 [1 2 3 4]
 [1 2 3 4]
 [1 2 3 4]]

Brodcasting Division of Matrices:
 [[1.         0.5        0.33333333 0.25      ]
 [1.         0.5        0.33333333 0.25      ]
 [1.         0.5        0.33333333 0.25      ]
 [1.         0.5        0.33333333 0.25      ]]
```

Figure 13 Output for Part 2 Using the “F,C” Arguments

The last unique output (Figure 13) shows one matrix and one scalar as the inputted argument in the function; it proceeded to do a broadcasting matrix instead of a matrix operation.



Start
Inputs 2 Matrices or scalars or both
Both Matrix
No
Calculates the matrix brodcasting
Outputs matrix brodcasting
Yes
Both Matrices shape equal
No
Outputs only the shape and dimension of the matrices
Yes
Calculates the matrix operation
Outputs the shape, dimension of the matrices and is matrix operations
end

Figure 14 Flowchart for Part2

The researcher created a simple flowchart to describe or summarize the function created.

IV. Conclusion

The researcher used Numpy functions and the “if \ else” statement to create two functions that describe a matrix also calculates the matrix operation or matric broadcasting, the researcher also learns that Discrete mathematics   subject is important to understand and efficiently created a function. Also, matrix operations can be used in farming by developing an algorithm that uses matrix operations or matrix identity to plot out the land   and maximize the given space.

References

[1] C. R. Harris *et al.*, “Array programming with NumPy,” *Nature*, vol. 585, no. 7825, pp. 357–362, 2020, doi: 10.1038/s41586-020-2649-2.

[2] S. van der Walt, S. C. Colbert, and G. Varoquaux, “The NumPy Array: A Structure for Efficient Numerical Computation,” *Comput. Sci. Eng.*, vol. 13, no. 2, pp. 22–30, 2011, doi: 10.1109/MCSE.2011.37.