Adamson University
College of Engineering
Computer Engineering Department

Linear Algebra

Laboratory Activity No. 6

Matrices

Submitted by:
Vallarta, Troy Joaquin G.

Instructor:
Engr. Dylan Josh D. Lopez

November 28, 2020

I. Objectives

This laboratory activity aims to implement the principles and techniques of learning matrix algebra and categorizing matrices.

II. Methods

- The practices of the laboratory activity are to be familiar with how the matrix works.
    - The laboratory activity implies to teach how to categorize and how to compute for matrix algebra.
- The laboratory activity allows the students to declare their matrices and learn how to compute simple algebra in matrices.
    - The students achieve the learnings by using the Jupyter Notebook to compute faster and create the situation needed.

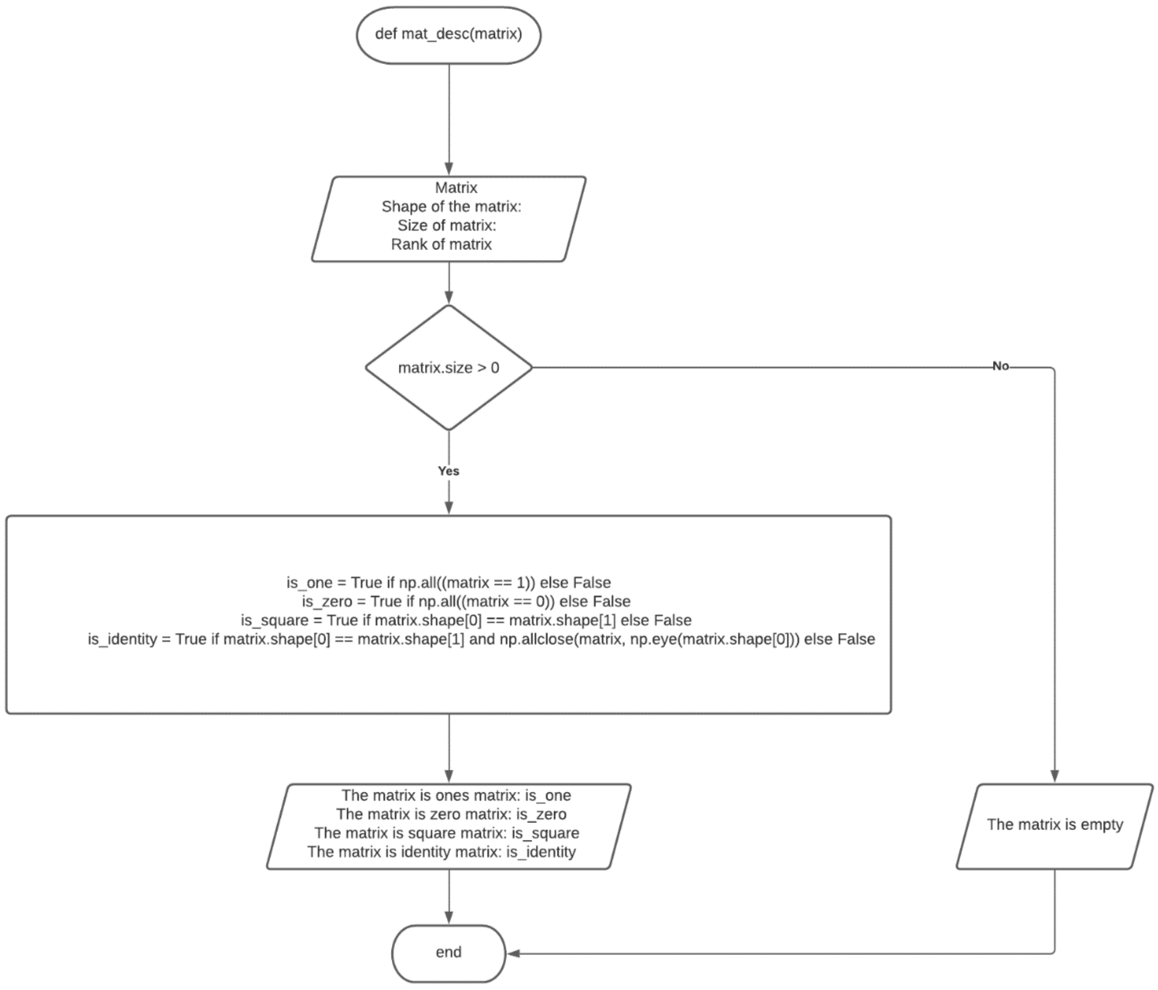
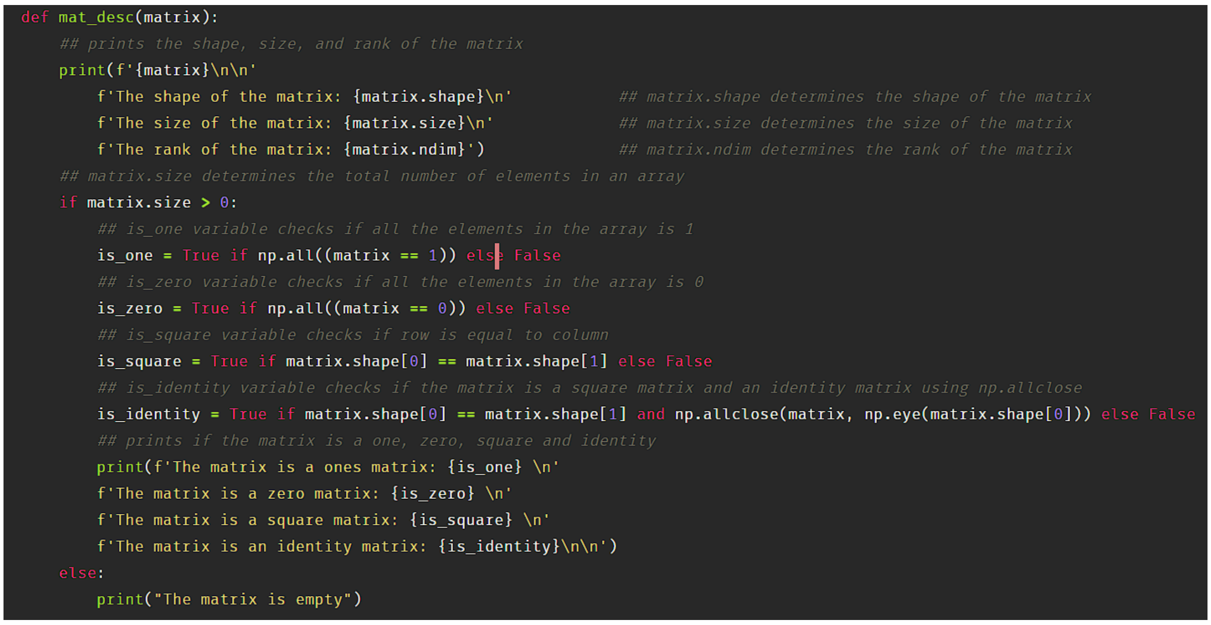

def mat_desc(matrix)
Matrix
Shape of the matrix:
Size of matrix:
Rank of matrix
matrix.size > 0
No
Yes
is_one = True if np.all((matrix == 1)) else False
is_zero = True if np.all((matrix == 0)) else False
is_square = True if matrix.shape[0] == matrix.shape[1] else False
is_identity = True if matrix.shape[0] == matrix.shape[1] and np.allclose(matrix, np.eye(matrix.shape[0])) else False
The matrix is ones matrix: is_one
The matrix is zero matrix: is_zero
The matrix is square matrix: is_square
The matrix is identity matrix: is_identity
The matrix is empty
end

Figure 1. Task 1 Flowchart

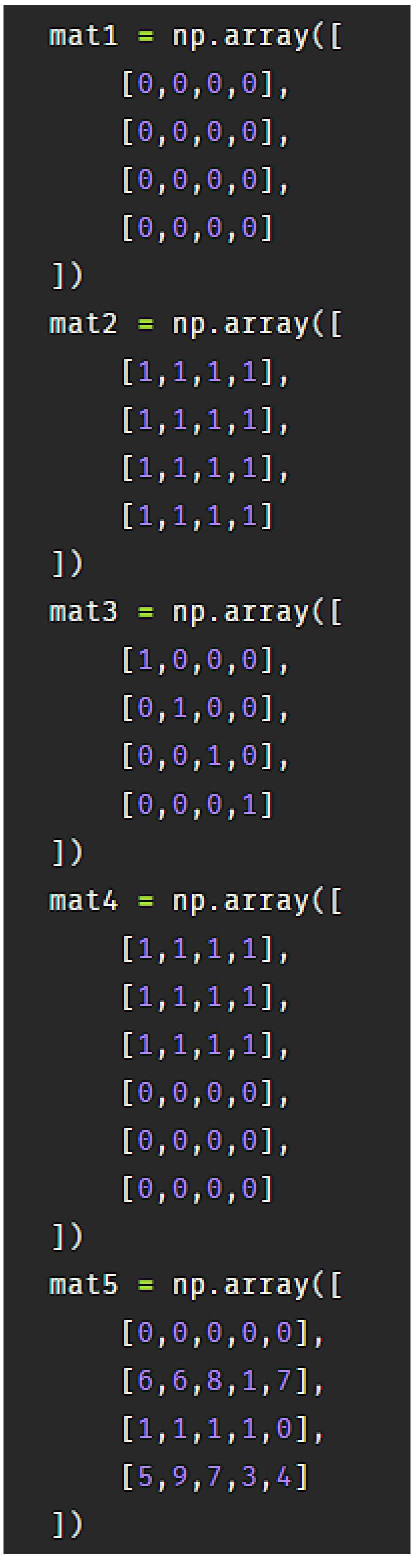
The created flowchart in Figure 1. is how the mat_desc() functions process. It shows the output of the matrix and its size, rank, and shape. In this task, it is necessary to know the matrix description to identify whether the created matrix is ones, zero, square, or identity matrix. It is also represented in this figure that the matrix should be more than zero in size. The process of finding the matrix's category is simple. To find if the matrix is one's matrix, all values inside the matrix should have only a value of one. The function np.all() is responsible for finding all the values inside the matrix. The same goes for the zero matrices. The matrix should have a value of only 0 to be called a zero matrix. On the other hand, the square matrix should have the same amount of columns and rows inside the matrix. Matrix.shape[0] is responsible for checking the columns' value and the matrix.shape[1] for the rows. Lastly, the matrix identity is finding the matrix if it is identity is using the numpy.allclose()[1] function by having two matrices inside if it has almost the same values in each other. The creation of diagonal "1" in the matrix is the NumPy.eye()[2] function.



```python
def mat_desc(matrix):
    ## prints the shape, size, and rank of the matrix
    print(f'{matrix}\n\n'
        f'The shape of the matrix: {matrix.shape}\n'          ## matrix.shape determines the shape of the matrix
        f'The size of the matrix: {matrix.size}\n'            ## matrix.size determines the size of the matrix
        f'The rank of the matrix: {matrix.ndim}')             ## matrix.ndim determines the rank of the matrix
    ## matrix.size determines the total number of elements in an array
    if matrix.size > 0:
        ## is_one variable checks if all the elements in the array is 1
        is_one = True if np.all((matrix == 1)) else False
        ## is_zero variable checks if all the elements in the array is 0
        is_zero = True if np.all((matrix == 0)) else False
        ## is_square variable checks if row is equal to column
        is_square = True if matrix.shape[0] == matrix.shape[1] else False
        ## is_identity variable checks if the matrix is a square matrix and an identity matrix using np.allclose
        is_identity = True if matrix.shape[0] == matrix.shape[1] and np.allclose(matrix, np.eye(matrix.shape[0])) else False
        ## prints if the matrix is a one, zero, square and identity
        print(f'The matrix is a ones matrix: {is_one} \n'
        f'The matrix is a zero matrix: {is_zero} \n'
        f'The matrix is a square matrix: {is_square} \n'
        f'The matrix is an identity matrix: {is_identity}\n\n')
    else:
        print("The matrix is empty")
```

Figure 1.2. Task 1 created function

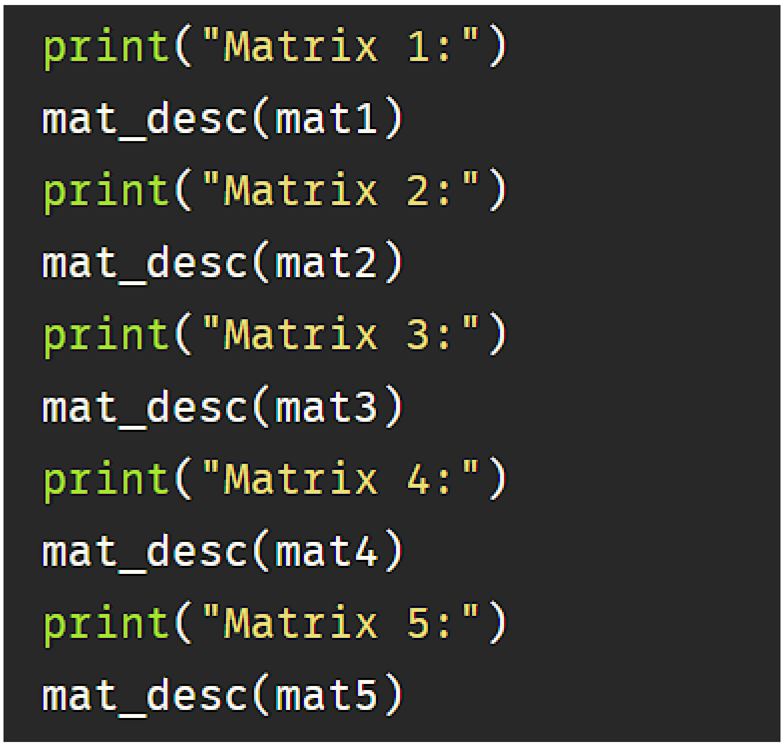
Figure 1.2 shows the researcher's created function to describe a specific matrix.



```
mat1 = np.array([
        [0,0,0,0],
        [0,0,0,0],
        [0,0,0,0],
        [0,0,0,0]
])
mat2 = np.array([
        [1,1,1,1],
        [1,1,1,1],
        [1,1,1,1],
        [1,1,1,1]
])
mat3 = np.array([
        [1,0,0,0],
        [0,1,0,0],
        [0,0,1,0],
        [0,0,0,1]
])
mat4 = np.array([
        [1,1,1,1],
        [1,1,1,1],
        [1,1,1,1],
        [0,0,0,0],
        [0,0,0,0],
        [0,0,0,0]
])
mat5 = np.array([
        [0,0,0,0,0],
        [6,6,8,1,7],
        [1,1,1,1,0],
        [5,9,7,3,4]
])
```

Figure 1.3. Task 1 Declaring the matrix

The created matrix in figure 1.3 is not lower than the shape of (3,3). In this case, the created matrix has a minimum size of (4,4). This figure shows the different matrix categories.



```
print("Matrix 1:")
mat_desc(mat1)
print("Matrix 2:")
mat_desc(mat2)
print("Matrix 3:")
mat_desc(mat3)
print("Matrix 4:")
mat_desc(mat4)
print("Matrix 5:")
mat_desc(mat5)
```

Figure 1.4. Printing the Matrix

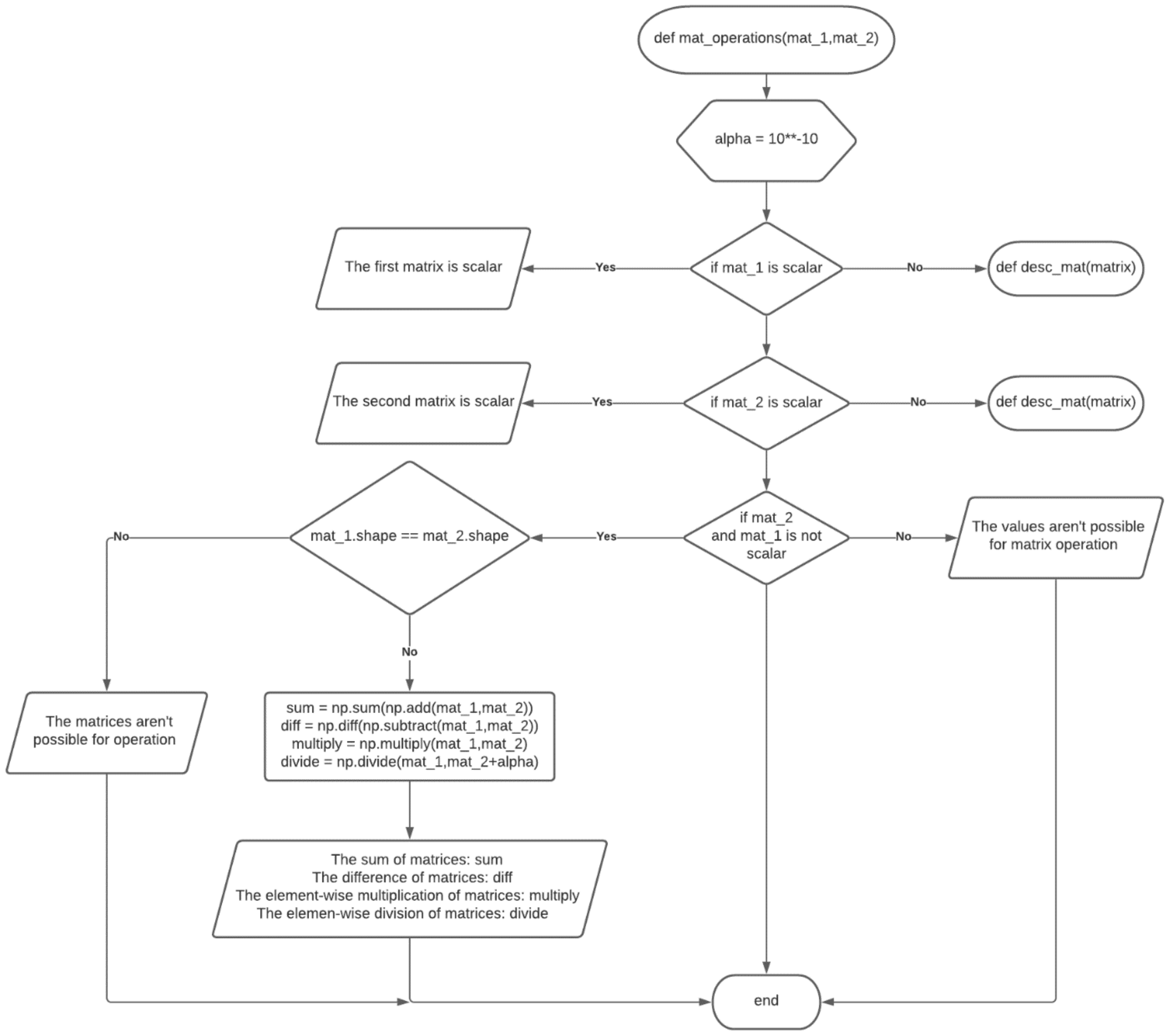
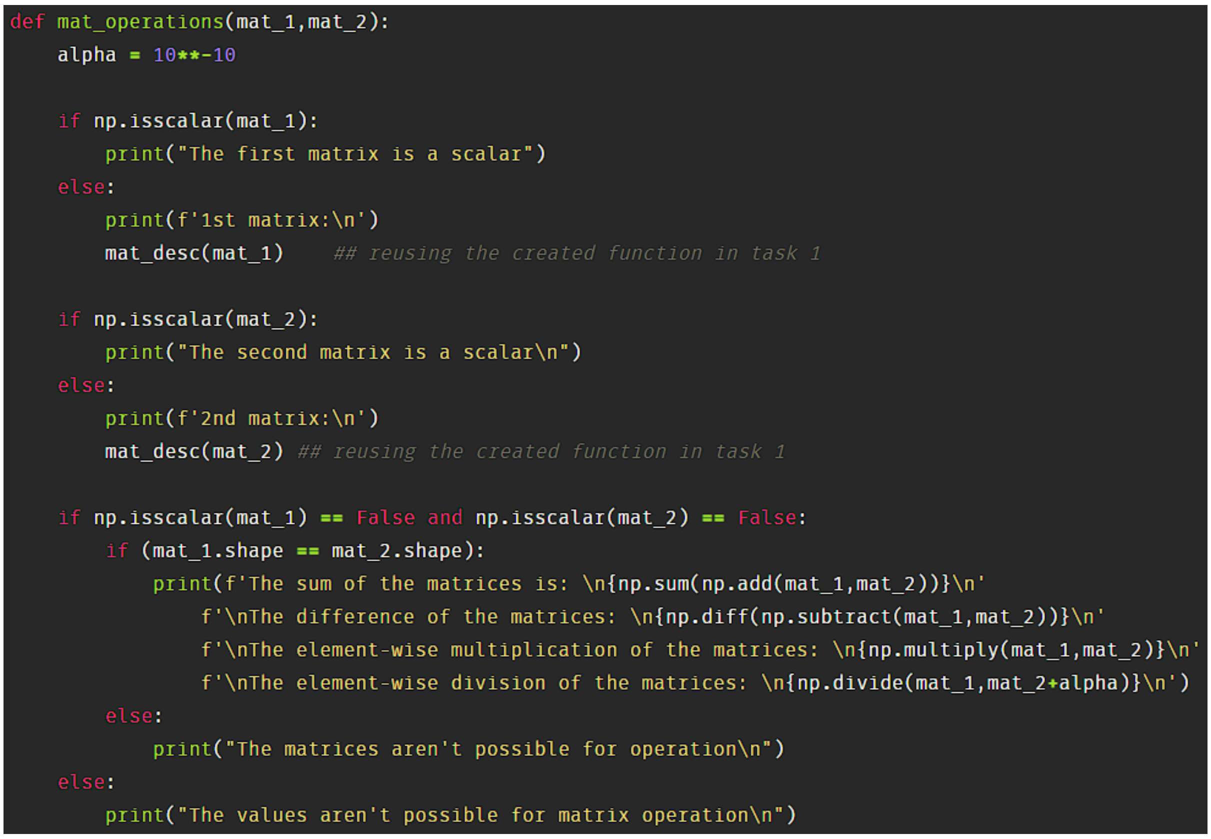

def mat_operations(mat_1,mat_2)
alpha = 10**-10
The first matrix is scalar
Yes
if mat_1 is scalar
No
def desc_mat(matrix)
The second matrix is scalar
Yes
if mat_2 is scalar
No
def desc_mat(matrix)
No
mat_1.shape == mat_2.shape
Yes
if mat_2 and mat_1 is not scalar
No
The values aren't possible for matrix operation
No
The matrices aren't possible for operation
sum = np.sum(np.add(mat_1,mat_2))
diff = np.diff(np.subtract(mat_1,mat_2))
multiply = np.multiply(mat_1,mat_2)
divide = np.divide(mat_1,mat_2+alpha)
The sum of matrices: sum
The difference of matrices: diff
The element-wise multiplication of matrices: multiply
The elemen-wise division of matrices: divide
end

Figure 2.1. Task 2 Flowchart

Figure 2.1 shows the created mat_operations(mat_1,mat_2) function. It has three main conditional statements in order to finish the program. The first condition is if the first parameter is a scalar. The function for finding if the matrix is scalar is the numpy.isscalar() as seen in Figure 2.2. The stated function returns a Boolean data type; the first two condition statement have the same process; the only difference is the parameter. If the matrix is not a scalar value, the program's flow will jump to the previous function stated in Figure 1. Using a modular approach to lessen the repeated codes when finding the matrix description is the fundamentals of Object-oriented programming. The last main condition statement checks whether the first and second matrix is not scalar. As seen in Figure 2.1, it shows that the third conditional statement will pass through another condition if the matrices provided are possible to perform matrix operations.

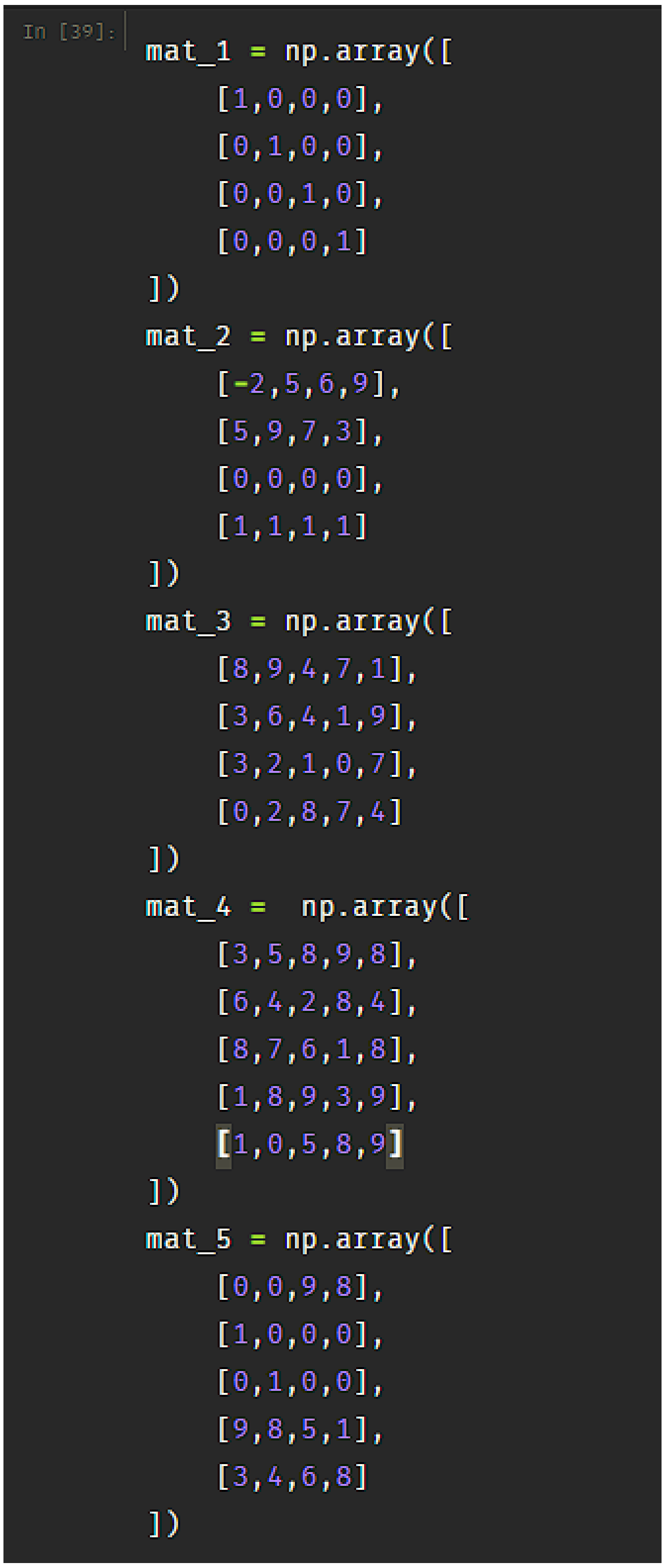
The matrix operations in matrices are one of the most important things when dealing with matrices. The sum of the matrices has two stages in order to work. Using Vector operations by using numpy.add() to get the addition of the two matrices and after the matrices turned into a one-dimensional array, the numpy.sum() function combines all the value and gets the sum of it. The same goes for subtracting matrices, the numpy.subtract(), and the numpy.diff() function respectively. Multiplication of matrices uses a numpy.multiply() function to produce element-wise multiplication of the matrices and in the element-wise division uses the numpy.divide() function.



```python
def mat_operations(mat_1,mat_2):
    alpha = 10**-10

    if np.isscalar(mat_1):
        print("The first matrix is a scalar")
    else:
        print(f'1st matrix:\n')
        mat_desc(mat_1)    ## reusing the created function in task 1

    if np.isscalar(mat_2):
        print("The second matrix is a scalar\n")
    else:
        print(f'2nd matrix:\n')
        mat_desc(mat_2) ## reusing the created function in task 1

    if np.isscalar(mat_1) == False and np.isscalar(mat_2) == False:
        if (mat_1.shape == mat_2.shape):
            print(f'The sum of the matrices is: \n{np.sum(np.add(mat_1,mat_2))}\n'
                f'\nThe difference of the matrices: \n{np.diff(np.subtract(mat_1,mat_2))}\n'
                f'\nThe element-wise multiplication of the matrices: \n{np.multiply(mat_1,mat_2)}\n'
                f'\nThe element-wise division of the matrices: \n{np.divide(mat_1,mat_2+alpha)}\n')
        else:
            print("The matrices aren't possible for operation\n")
    else:
        print("The values aren't possible for matrix operation\n")
```

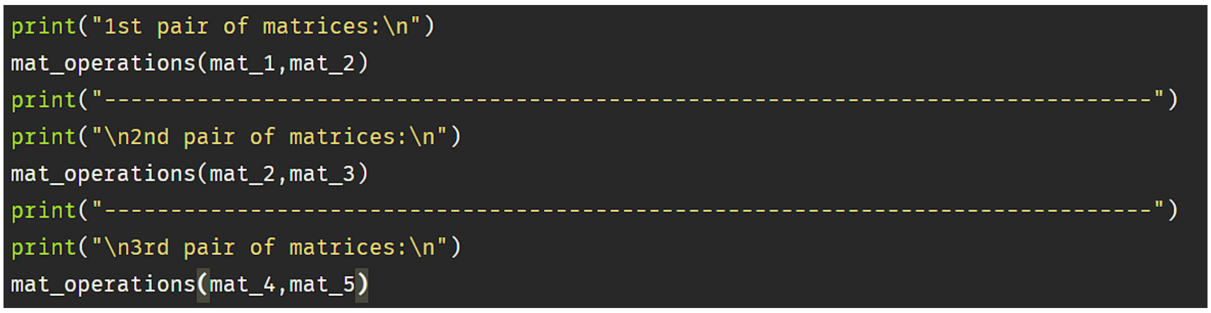
Figure 2.2. Creating the Matrix algebra function

Figure 2.2 shows the program form of the flowchart created in task 2. Reusing the functions created in this task can save time and effort in creating the same process. The mat_desc() function is the programer's creation in this laboratory activity, which produces the matrix given description.



```
In [39]:
mat_1 = np.array([
    [1,0,0,0],
    [0,1,0,0],
    [0,0,1,0],
    [0,0,0,1]
])
mat_2 = np.array([
    [-2,5,6,9],
    [5,9,7,3],
    [0,0,0,0],
    [1,1,1,1]
])
mat_3 = np.array([
    [8,9,4,7,1],
    [3,6,4,1,9],
    [3,2,1,0,7],
    [0,2,8,7,4]
])
mat_4 =  np.array([
    [3,5,8,9,8],
    [6,4,2,8,4],
    [8,7,6,1,8],
    [1,8,9,3,9],
    [1,0,5,8,9]
])
mat_5 = np.array([
    [0,0,9,8],
    [1,0,0,0],
    [0,1,0,0],
    [9,8,5,1],
    [3,4,6,8]
])
```

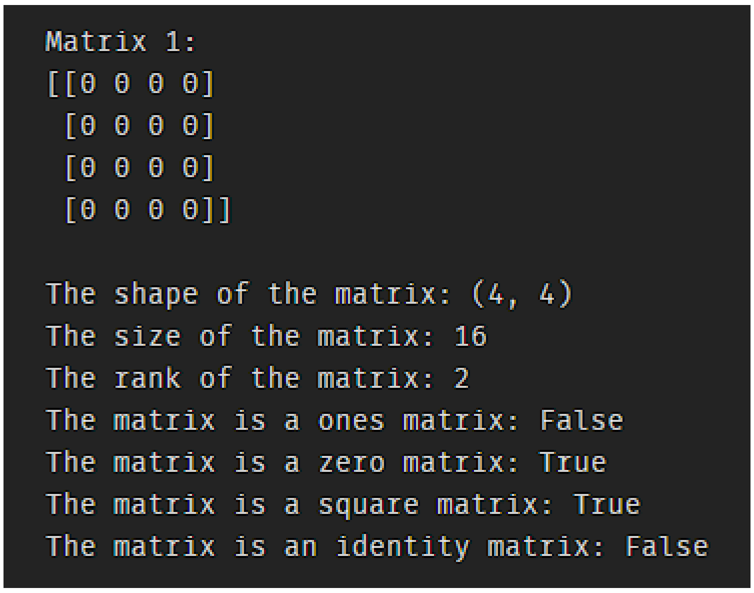
Figure 2.3. Declaring Matrices



```
print("1st pair of matrices:\n")
mat_operations(mat_1,mat_2)
print("-------------------------------------------------------------------------------")
print("\n2nd pair of matrices:\n")
mat_operations(mat_2,mat_3)
print("-------------------------------------------------------------------------------")
print("\n3rd pair of matrices:\n")
mat_operations(mat_4,mat_5)
```

Figure 2.4. Printing matrices operation

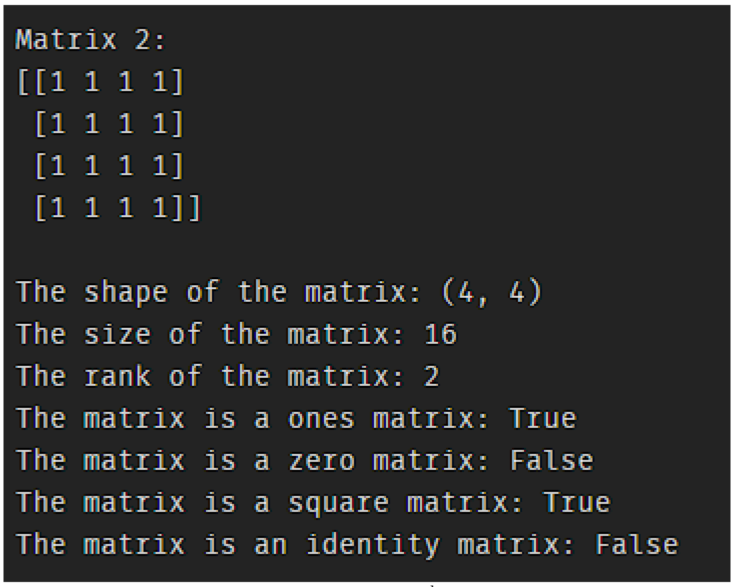
III. Results



```
Matrix 1:
[[0 0 0 0]
 [0 0 0 0]
 [0 0 0 0]
 [0 0 0 0]]

The shape of the matrix: (4, 4)
The size of the matrix: 16
The rank of the matrix: 2
The matrix is a ones matrix: False
The matrix is a zero matrix: True
The matrix is a square matrix: True
The matrix is an identity matrix: False
```

Figure 3.1. Task 1 1st Matrix

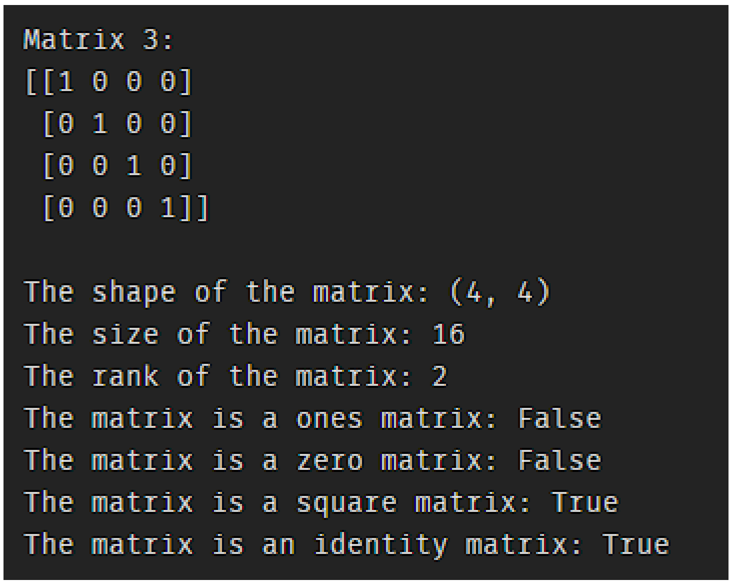
Figure 3.1 matrix's description has a shape of (4, 4), and its size is 16 because of I * j or 4*4. The matrix's rank is a two-dimensional array, and it is a zero matrix that contains all zero values and a square matrix with the same column and row count.



```
Matrix 2:
[[1 1 1 1]
 [1 1 1 1]
 [1 1 1 1]
 [1 1 1 1]]

The shape of the matrix: (4, 4)
The size of the matrix: 16
The rank of the matrix: 2
The matrix is a ones matrix: True
The matrix is a zero matrix: False
The matrix is a square matrix: True
The matrix is an identity matrix: False
```

Figure 3.2 Task 1 2nd Matrix

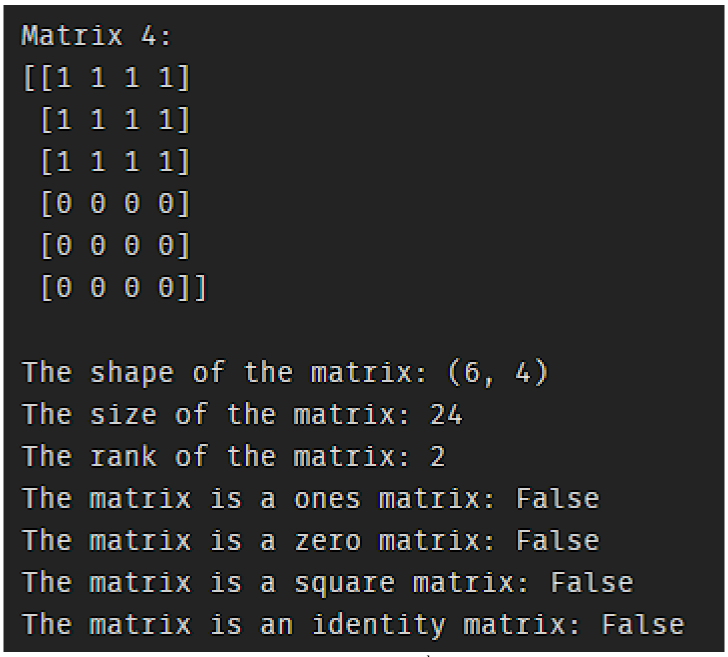
Figure 3.2 shows that the 2nd given matrix has a shape of (4,4) in a size of 16 and a two-dimensional array. The matrix's category is one's matrix consisting of all one values, and it also is a square matrix with the same columns and rows.



```
Matrix 3:
[[1 0 0 0]
 [0 1 0 0]
 [0 0 1 0]
 [0 0 0 1]]

The shape of the matrix: (4, 4)
The size of the matrix: 16
The rank of the matrix: 2
The matrix is a ones matrix: False
The matrix is a zero matrix: False
The matrix is a square matrix: True
The matrix is an identity matrix: True
```

Figure 3.3. Task 1 3rd Matrix

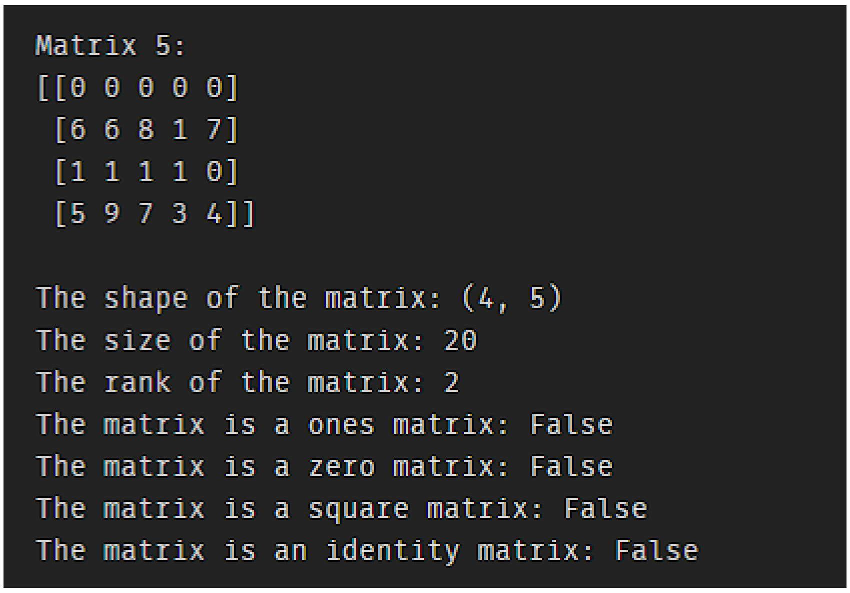
Figure 3.3 shows that the 3rd given matrix has a shape of (4,4) it is in a size of 16 and a two-dimensional array. The matrix's category is an identity matrix consisting of diagonal one value in the array, and the rest of the values are zero. It is also a square matrix.



```
Matrix 4:
[[1 1 1 1]
 [1 1 1 1]
 [1 1 1 1]
 [0 0 0 0]
 [0 0 0 0]
 [0 0 0 0]]

The shape of the matrix: (6, 4)
The size of the matrix: 24
The rank of the matrix: 2
The matrix is a ones matrix: False
The matrix is a zero matrix: False
The matrix is a square matrix: False
The matrix is an identity matrix: False
```

Figure 3.4. Task 1 4th Matrix

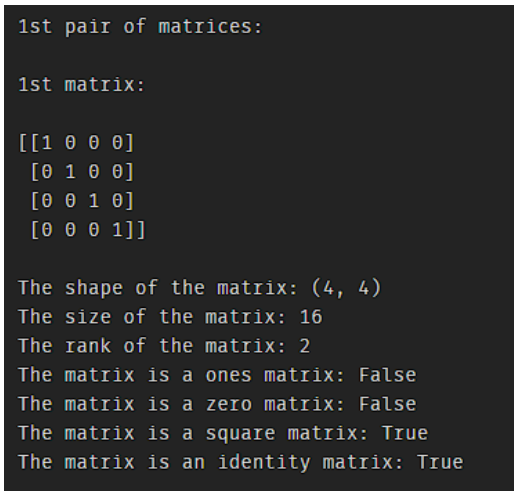
Figure 3.4 shows that the given matrix is in the shape of (6, 4), the size of this matrix is 24, it multiplies the value of the matrix I and j value in this case 6 and 4. Multiply it, and the result is 24. The matrix rank is also in a two-dimensional array. It is not a square matrix because the columns and the matrix's rows do not have the same count.



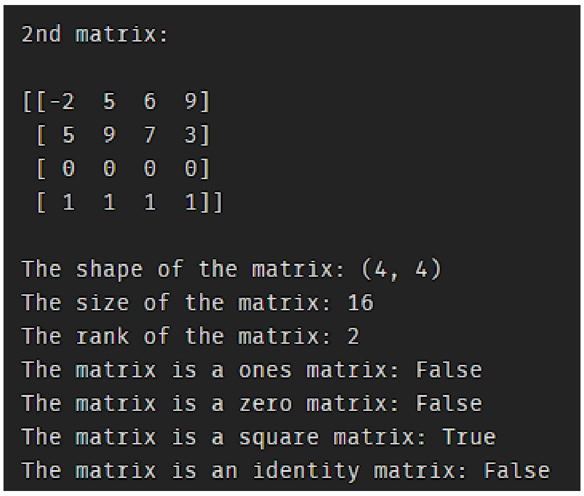
```
Matrix 5:
[[0 0 0 0 0]
 [6 6 8 1 7]
 [1 1 1 1 0]
 [5 9 7 3 4]]

The shape of the matrix: (4, 5)
The size of the matrix: 20
The rank of the matrix: 2
The matrix is a ones matrix: False
The matrix is a zero matrix: False
The matrix is a square matrix: False
The matrix is an identity matrix: False
```

Figure 3.5. Task 1 5th Matrix

Figure 3.5 shows that the given matrix is in the shape of (4, 5), the size of this matrix is 20, it multiplies the value of the matrix I and j value in this case 4 and 5. Multiply it, and the result is 20. The matrix rank is also in a two-dimensional array. It is not a square matrix because the columns and the matrix's rows do not have the same count.

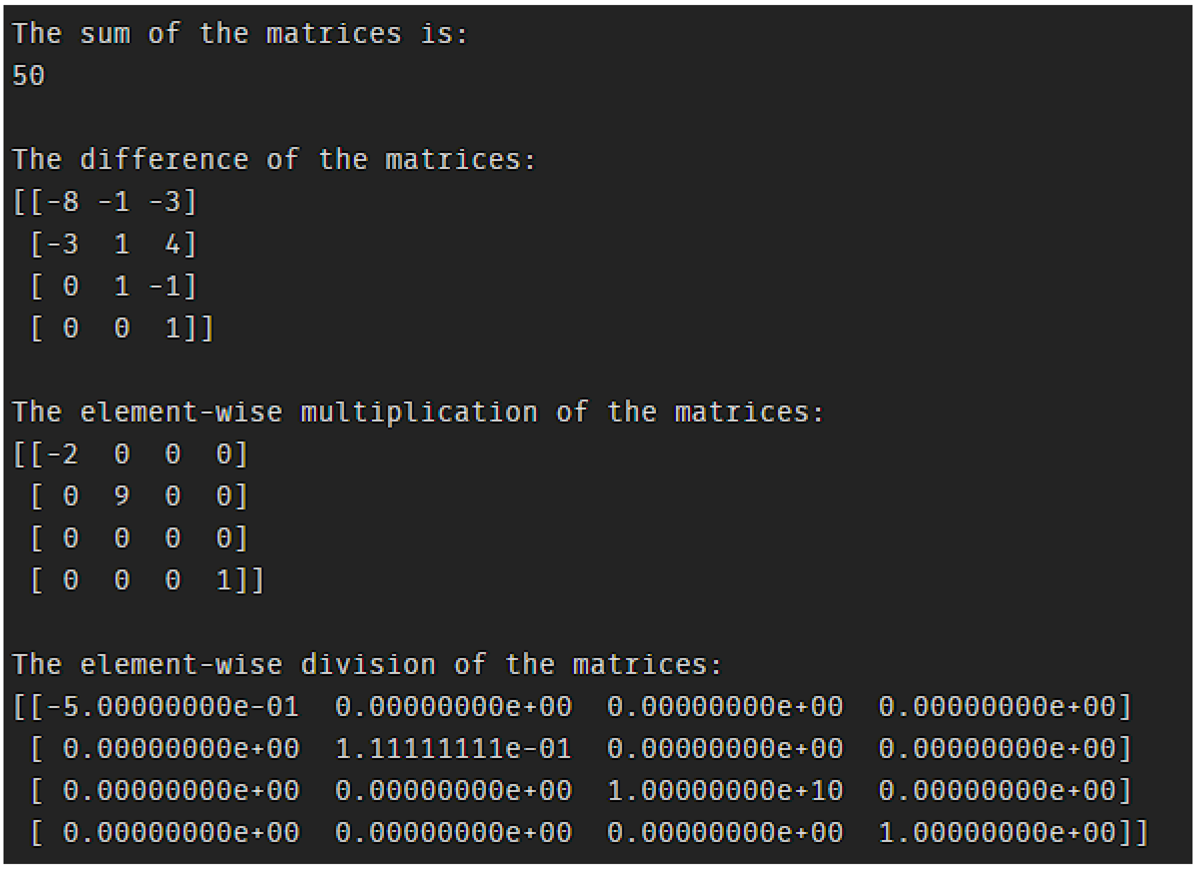


```
1st pair of matrices:

1st matrix:

[[1 0 0 0]
 [0 1 0 0]
 [0 0 1 0]
 [0 0 0 1]]

The shape of the matrix: (4, 4)
The size of the matrix: 16
The rank of the matrix: 2
The matrix is a ones matrix: False
The matrix is a zero matrix: False
The matrix is a square matrix: True
The matrix is an identity matrix: True
```



```
2nd matrix:

[[-2  5  6  9]
 [ 5  9  7  3]
 [ 0  0  0  0]
 [ 1  1  1  1]]

The shape of the matrix: (4, 4)
The size of the matrix: 16
The rank of the matrix: 2
The matrix is a ones matrix: False
The matrix is a zero matrix: False
The matrix is a square matrix: True
The matrix is an identity matrix: False
```

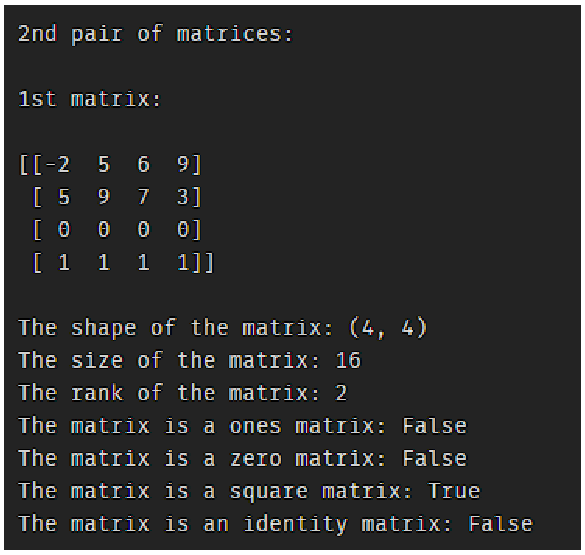
Figure 4. Task 2 1st pair Matrices



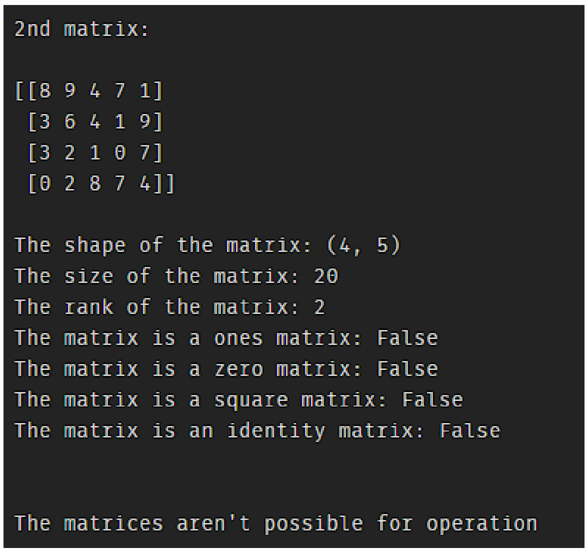
```
The sum of the matrices is:
50

The difference of the matrices:
[[-8 -1 -3]
 [-3  1  4]
 [ 0  1 -1]
 [ 0  0  1]]

The element-wise multiplication of the matrices:
[[-2  0  0  0]
 [ 0  9  0  0]
 [ 0  0  0  0]
 [ 0  0  0  1]]

The element-wise division of the matrices:
[[-5.00000000e-01  0.00000000e+00  0.00000000e+00  0.00000000e+00]
 [ 0.00000000e+00  1.11111111e-01  0.00000000e+00  0.00000000e+00]
 [ 0.00000000e+00  0.00000000e+00  1.00000000e+10  0.00000000e+00]
 [ 0.00000000e+00  0.00000000e+00  0.00000000e+00  1.00000000e+00]]
```

Figure 4.1. Task 2 1st pair Matrix Operation

Figure 4.1 shows that the sum of the paired matrices in Figure 4 is 50. The matrix's difference is reduced to a shape (4, 3) seen in the figure because subtraction reduces the matrix's size. Multiplication in this figure is multiplied by an identity matrix and a standard matrix. Therefore the answer only has values higher or lower than zero is in the diagonal form. Lastly, the matrices' element-wise division has a decimal place because of the alpha variable. Dividing a matrix can have difficulties. Having no value or NaN can cause a problem in some situations. Therefore the alpha is added to prevent no value arrays.

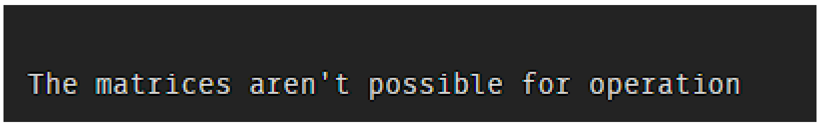


```
2nd pair of matrices:

1st matrix:

[[-2  5  6  9]
 [ 5  9  7  3]
 [ 0  0  0  0]
 [ 1  1  1  1]]

The shape of the matrix: (4, 4)
The size of the matrix: 16
The rank of the matrix: 2
The matrix is a ones matrix: False
The matrix is a zero matrix: False
The matrix is a square matrix: True
The matrix is an identity matrix: False
```



```
2nd matrix:

[[8 9 4 7 1]
 [3 6 4 1 9]
 [3 2 1 0 7]
 [0 2 8 7 4]]

The shape of the matrix: (4, 5)
The size of the matrix: 20
The rank of the matrix: 2
The matrix is a ones matrix: False
The matrix is a zero matrix: False
The matrix is a square matrix: False
The matrix is an identity matrix: False


The matrices aren't possible for operation
```

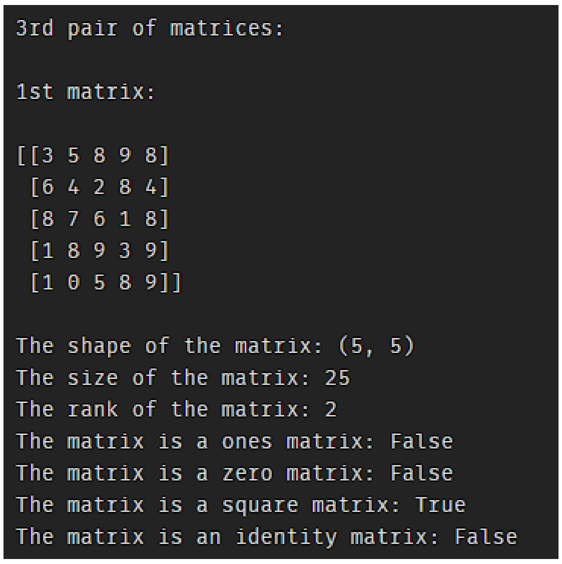
Figure 5. Task 2 2nd pair Matrices



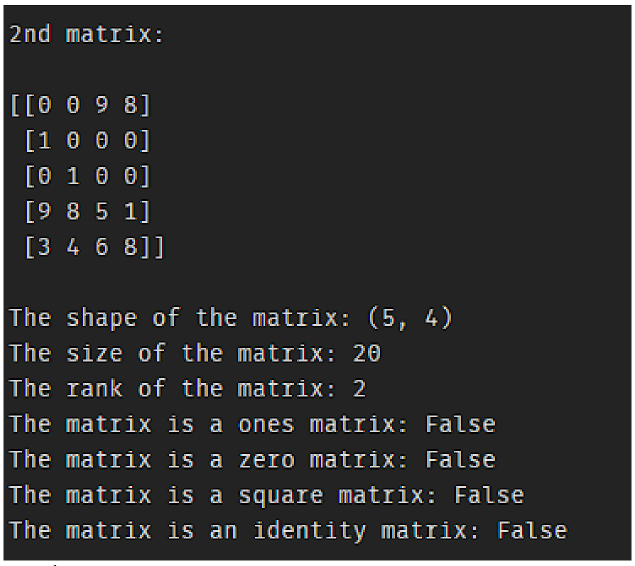
```
The matrices aren't possible for operation
```

Figure 5.1. Task 2 1st pair Matrix Operation

The created matrix in figure 5 is not possible for matrix operation because it does not have the same shape. One matrix should be reshaped into (4, 4) or (4, 5).

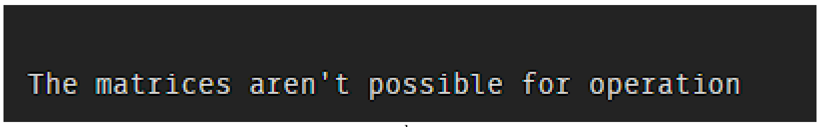


```
3rd pair of matrices:

1st matrix:

[[3 5 8 9 8]
 [6 4 2 8 4]
 [8 7 6 1 8]
 [1 8 9 3 9]
 [1 0 5 8 9]]

The shape of the matrix: (5, 5)
The size of the matrix: 25
The rank of the matrix: 2
The matrix is a ones matrix: False
The matrix is a zero matrix: False
The matrix is a square matrix: True
The matrix is an identity matrix: False
```



```
2nd matrix:

[[0 0 9 8]
 [1 0 0 0]
 [0 1 0 0]
 [9 8 5 1]
 [3 4 6 8]]

The shape of the matrix: (5, 4)
The size of the matrix: 20
The rank of the matrix: 2
The matrix is a ones matrix: False
The matrix is a zero matrix: False
The matrix is a square matrix: False
The matrix is an identity matrix: False
```

Figure 6. Task 3rd pair Matrices



```
The matrices aren't possible for operation
```

Figure 6.1. Task 2 3rd pair Matrix Operation

The created matrix in figure 6 is not possible for matrix operation because the matrices do not have the same shape. One matrix should be reshaped into (5, 4) or (5, 5).

IV. Conclusion

This laboratory activity teaches the students how to declare, categorize, and do matrix operations. In declaring the matrices, students must input a matrix shape not lower than (3, 3) to rely on the program more in computing for the matrices. The laboratory activity also taught the students how to categorize the matrix, from what is ones, squared, zero, and identity matrices. It teaches the students to know in an instant how to categorize a matrix. Lastly, the laboratory activity teaches the students how to do matrix operations, starting from addition, subtraction, element-wise multiplication, and division. Students should also know that to have a successful matrix operation method. The students should know how to determine the size of the matrix.

"[3] These techniques take due care of the constraints/restrictions confronted in the real-life situation in the agricultural sector. In the present day of growing global complexities, the operations research models are widely used in planning and management to achieve organizational and economic goals" In other words using matrix operation in agriculture can be optimized how our agriculture will work. The most critical use of matrix operation is in agricultural planning. The data are given to the programmers to optimize the land, and more seeds will be planted.

References

[1] “numpy.allclose¶,” *numpy.allclose - NumPy v1.19 Manual*. [Online]. Available: https://numpy.org/doc/stable/reference/generated/numpy.allclose.html. [Accessed: 28-Nov-2020].

[2] “numpy.eye¶,” *numpy.eye - NumPy v1.19 Manual*. [Online]. Available: https://numpy.org/doc/stable/reference/generated/numpy.eye.html. [Accessed: 28-Nov-2020].

[3] K. D. Sharma, *Application of operations research in agriculture*. Palampur: college of agriculture, 2016.

Appendix

https://github.com/Zofserif/Linear-Algebra/blob/master/Lab%205-6%20-%20Systems%20of%20Linear%20Equations/LinAlg%20Lab%206.ipynb