Linear Algebra

Laboratory Activity No.

**Matrices**

|  |  |
| --- | --- |
| *Submitted by:* | *Instructor:* |
| Guy, Lawrence Adrian B. | Engr. Dylan Josh D. Lopez |

November 01, 2020

# Objectives

The laboratory aims to be familiar with matrices and their relation to linear equations. The activity shows the declaration of matrices in Python and the different categories of matrices. The activity also shows basic matrix operations using Python.

# Methods

The practices of the activity are performing basic matrix operations and programming of different matrix equations and operations in Python. The matrices in the activity are classified into two, shape and elemental values. The matrix operations included in the activity are addition, subtraction, and element-wise multiplication.

The first deliverable of the activity is to create a function named mat\_desc(). The function describes the matrix’s shape, size, and rank. The function also categorizes the matrix if it is a square or non-square, empty, identity, ones, or zeros matrix.

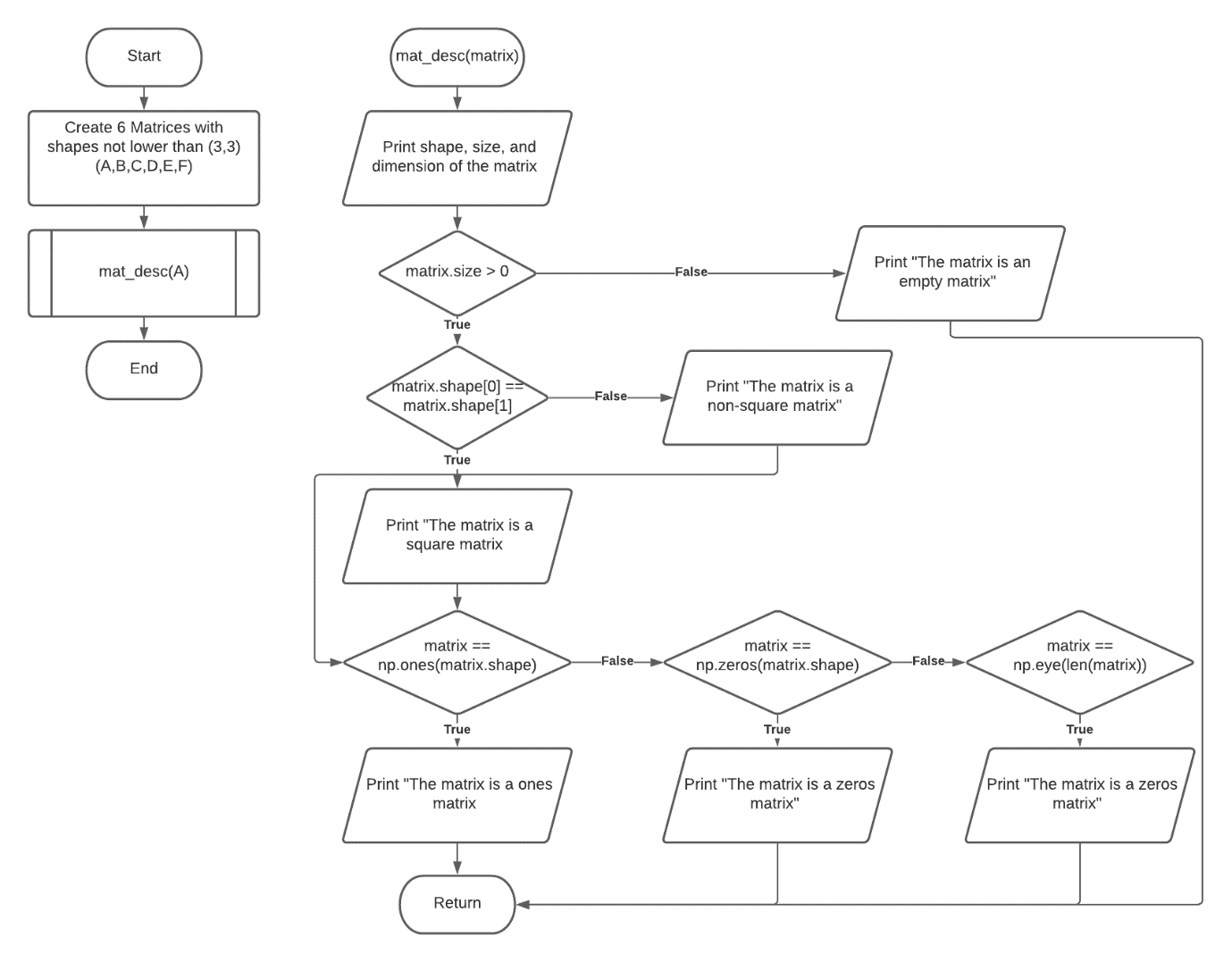


Figure 1 Flowchart for task 1

The mat\_desc() function has a parameter named matrix. The function uses np.ndarray.shape, np.ndarray.size, and np.ndarray.ndim to describe the shape, size, and dimensions or rank of the matrix [1][2][3]. Next, the function checks whether the matrix size is greater than 0 if it isn’t then it will print that the matrix is an empty matrix then return. If it is, then it will check if the rows and columns of the matrix are equal by using np.ndarray.shape. If the shape of the rows and columns are equal, it will print that the matrix is a square matrix. If the shape of the rows is not equal then it will print that the matrix is a non-square matrix. Next, the function will check if the matrix is an identity, zeros, or ones matrix using np.eye(), np.zeros(), and np.ones(). The function np.eye() returns an array with zeros and ones, wherein the ones are diagonally aligned [4]. The function np.zeros() returns an array with all zeros as elements [5]. The function np.ones() returns an array with all ones as elements [6]. The function will do this by using an if-else statement if the matrix is equal to np.ones(matrix.shape) then it will print that the matrix is a ones matrix. If it is not then it will check if the matrix is equal to the np.zeros(matrix.shape). If it is then it will print that the matrix is a zeros matrix. Lastly, the function will check if the matrix is equal to the np.eye(len(matrix)). If it is then it will print that the matrix is an identity matrix. If the parameter isn’t a matrix then the function will output an error.

The second deliverable is to create a function named mat\_operations(). The function determines if the first parameter and the second parameter is a matrix or a scalar. Then the function will describe the matrix if the parameter is a matrix. Lastly, the function will perform matrix operations on the first and second parameters.

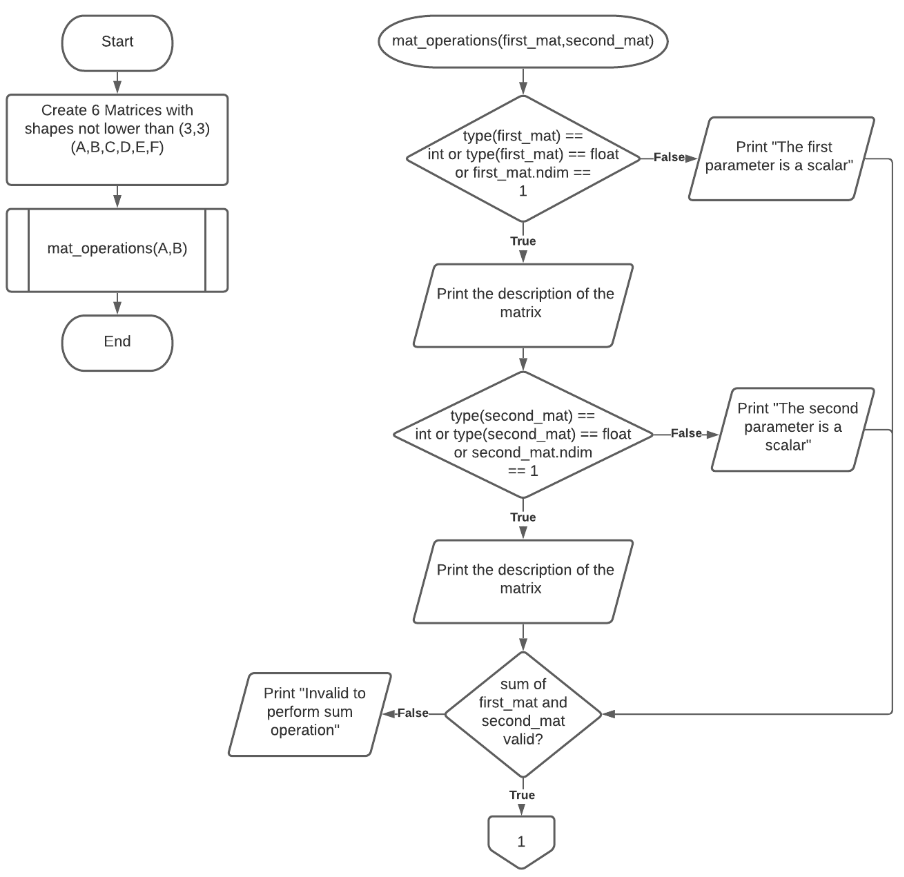


Figure 2.1 Flowchart for task 2

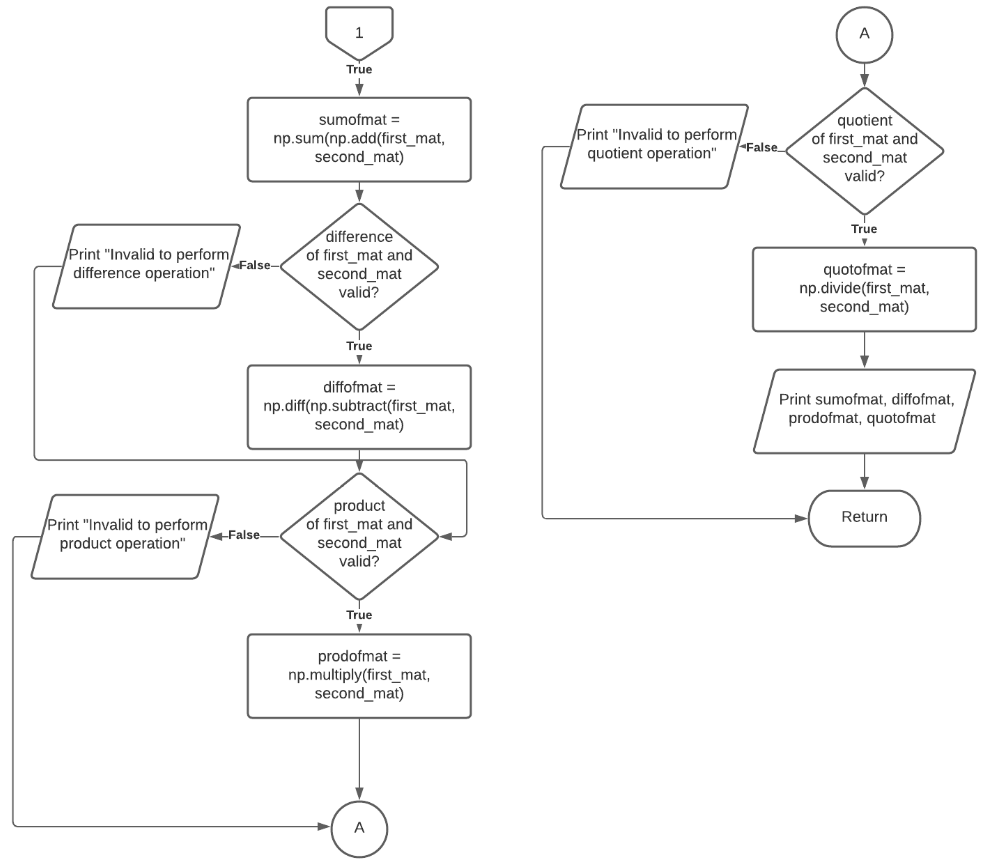


Figure 2.2 Flowchart for task 2

The function first checks if the given parameter is a scalar or a matrix. Using an if-else statement, the function checks if the parameter is an integer or a float or the rank of the parameter is equal to 1. If it is then it will print that the parameter is a scalar, if it is not then it will print the description of the matrix the same as task 1. After determining whether the parameter is a scalar or a matrix, the function will perform basic matrix operations between the first and second parameters. The function used np.sum() and np.add() to compute for the sum between the two parameters [7][8]. Then, the function used np.diff() and np.subtract() to compute for the difference [9][10]. Lastly, the function used np.multiply() and np.divide() to perform element-wise multiplication and element-wise division [11][12]. If the first and second parameters aren’t valid to perform matrix operations, instead of an error, it would print an error message using the print function. This is done by using the try and except in Python. First, the try clause will be executed, if the try clause is successful, it would skip the except clause [13]. But if the try clause is unsuccessful then instead of an error, it would output whatever is in the except clause [13].

# Results

The first task is to create a function named mat\_desc() with one parameter that describes the matrix. The codes are seen in figure 3.

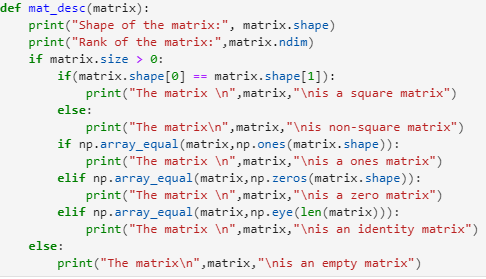


Figure 3 Codes for task 1

The function describes the matrix’s shape, size, and rank. The function also categorizes the matrix. It determines if the parameter is an empty matrix or not. If the parameter is not an empty matrix, it determines whether the matrix is a square or a non-square. Lastly, it determines whether the parameter is an identity, ones, or zeros matrix.

The matrices used for this activity is seen in figure 4. The programmer created 7 variables to test each condition in the created function.

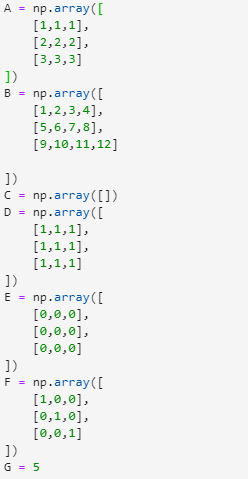


Figure 4 Arguments used for the function

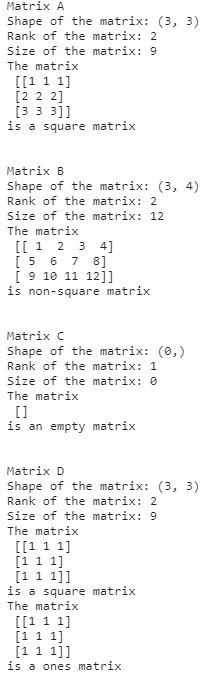


Figure 5.1 Output for task 1

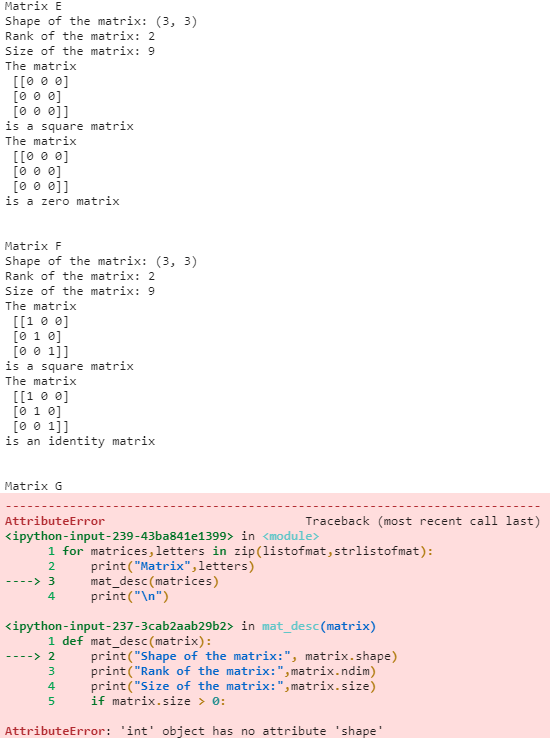


Figure 5.2 Output for task 1

As seen in figures 5.1 and 5.2, the function described each variable accurately. The variable G returned an error due to the value assigned to G being a scalar and not a matrix.

The second task is to create a function named mat\_opeartions() with two parameters.

Figure 6.1 Codes for task 2

The function determines whether the first or second parameter is a scalar or a matrix. Then it will describe the parameter if it is a matrix. If not then it will print that the parameter is a scalar.

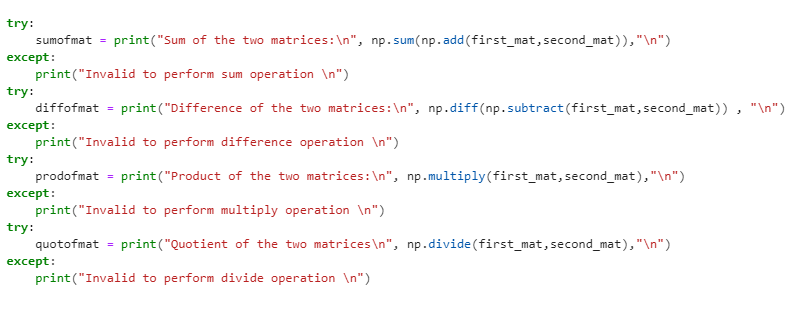




Figure 6.2 Codes for task 2

Next, the function will perform basic matrix operations. Using the try and except statement, if the operation is valid then it will execute and print the answer. If not, instead of returning an error the function will return its own error message “Invalid to perform operation”.

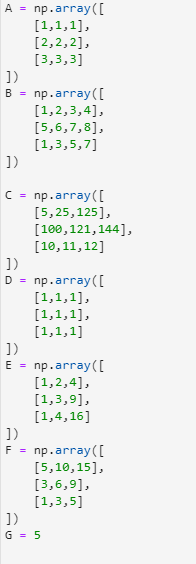


Figure 7 Arguments used for mat\_operations() function

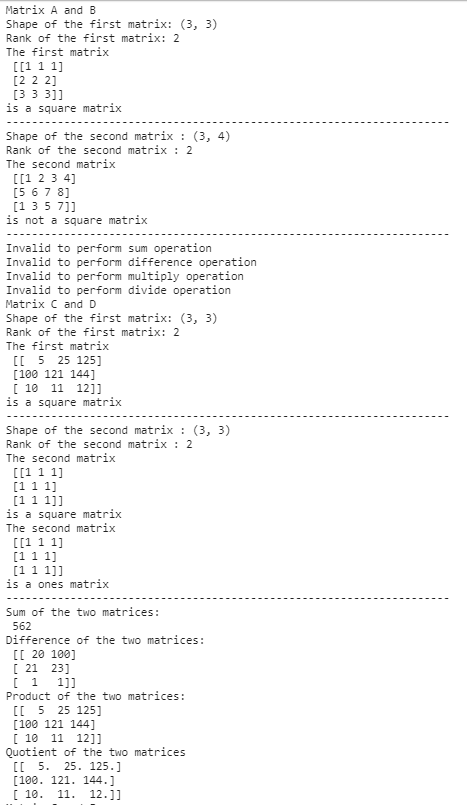


Figure 8.1 Output for task 2

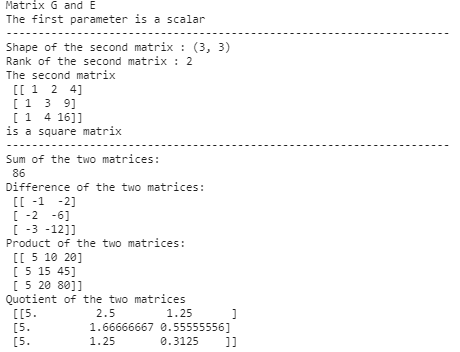


Figure 8.2 Output for task 2

Figure 8.2 shows the output using the created function mat\_operations(). The function categorized the parameter if it is a scalar or a matrix. The function also determined if the matrix is empty or not, square or non-square and identity, ones, or zeros matrix. Lastly, the function performed basic matrix operations using the two parameters and printed the result. If the operation is invalid, it would print its own error message.

# Conclusion

The laboratory activity discussed transforming equations into matrices. The activity also discussed the different categories of matrices. Lastly, the activity showed basic matrix operations using Python and NumPy.

Through the activity, the students were able to learn the different functions and techniques used to categorized matrices and do matrix operations using Python and NumPy.

Using the knowledge from this activity and past activities, matrices can be used to solve problems in the field of agriculture. A farmer can calculate the amount of time he needs to completely seed and harvest based on the size of his land and the materials he has. He can also compute the expenses that he will be spending and the profits that he will be able to gain.

**References**

[1] NumPy, “numpy.ndarray.shape,” 2020. https://numpy.org/doc/stable/reference/generated/numpy.ndarray.shape.html (accessed Mar. 11, 2020).

[2] NumPy, “np.ndarray.size,” 2020. https://numpy.org/doc/stable/reference/generated/numpy.ndarray.size.html (accessed Mar. 11, 2020).

[3] NumPy, “numpy.ndarray.ndim,” 2020. https://numpy.org/doc/stable/reference/generated/numpy.ndarray.ndim.html (accessed Mar. 11, 2020).

[4] NumPy, “numpy.eye,” 2020. .

[5] NumPy, “numpy.eye,” 2020. https://numpy.org/doc/stable/reference/generated/numpy.eye.html (accessed Mar. 11, 2020).

[6] NumPy, “numpy.zeros,” 2020. https://numpy.org/devdocs/reference/generated/numpy.zeros.html (accessed Mar. 11, 2020).

[7] NumPy, “numpy.ones,” 2020. https://numpy.org/doc/stable/reference/generated/numpy.ones.html#numpy-ones (accessed Mar. 11, 2020).

[8] NumPy, “numpy.sum,” 2020. https://numpy.org/doc/stable/reference/generated/numpy.sum.html#numpy-sum.

[9] NumPy, “numpy.add,” 2020. https://numpy.org/doc/stable/reference/generated/numpy.add.html (accessed Mar. 11, 2020).

[10] NumPy, “numpy.diff,” 2020. https://numpy.org/doc/stable/reference/generated/numpy.diff.html (accessed Mar. 11, 2020).

[11] NumPy, “numpy.subtract.” https://numpy.org/doc/stable/reference/generated/numpy.subtract.html (accessed Mar. 11, 2020).

[12] NumPy, “np.multiply,” 2020. https://numpy.org/doc/stable/reference/generated/numpy.multiply.html (accessed Mar. 11, 2020).

[13] NumPy, “np.divide,” 2020. https://numpy.org/doc/stable/reference/generated/numpy.divide.html (accessed Mar. 11, 2020).

[14] Python, “Errors and Exceptions,” 2020. https://docs.python.org/3/tutorial/errors.html (accessed Mar. 11, 2020).

**Appendix**

Github Repository Link: <https://github.com/Loreynszxc/Linear-Algebra-Lab-6>