



Adamson University  
College of Engineering  
Computer Engineering Department



Linear Algebra

Laboratory Activity No. 2

---

# Intro to Vectors And Numpy

---

*Submitted by:*

Casiño, Christian Carlos H.

*Instructor:*

Engr. Dylan Josh D. Lopez

February 19, 2021

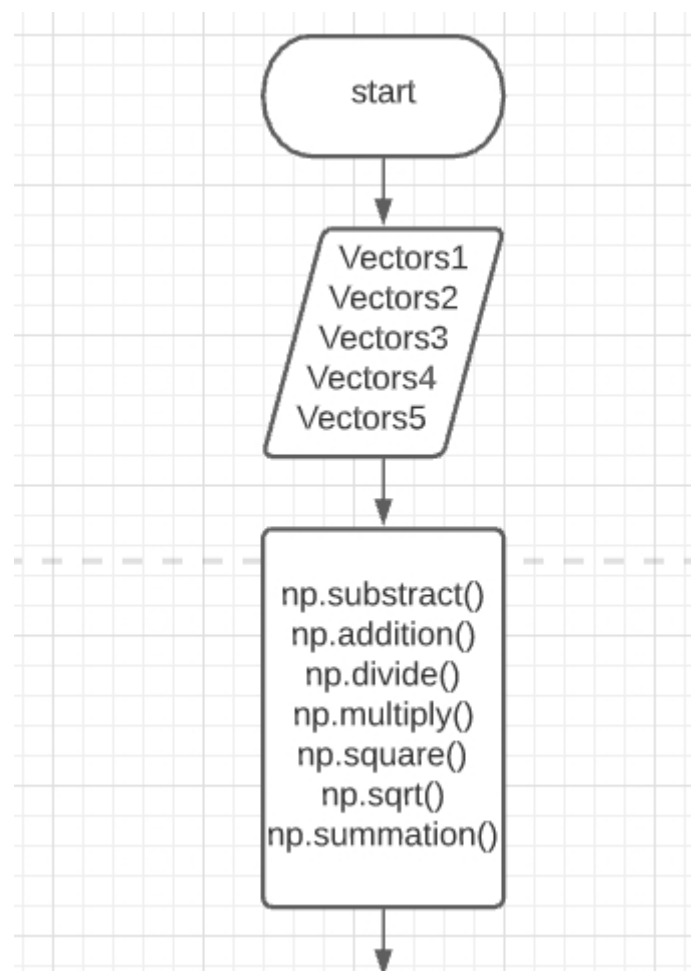
---

## I. Objectives

This laboratory activity aims to be familiar with the libraries in Python for numerical and scientific programming, visualize vectors through Python programming, and perform simple vector operations through code. This laboratory activity aims to create tasks with the use of the libraries Numpy and Matplotlib, the vectors will perform the following operations in any combination and any number of trials: addition, subtraction, multiplication, division, squaring, square root, and summation.

## II. Methods

First, the programmer should draw a flow chart as a reference in our program which will teach us the flow of the program. Secondly, the programmer should discuss and review how to manipulate the vectors, arithmetic, scalars, libraries, scaling, and formulas for filling up the flow of the program. The deliverance of this activity is to help the programmer to familiarize the use of numpy and matplotlib libraries.



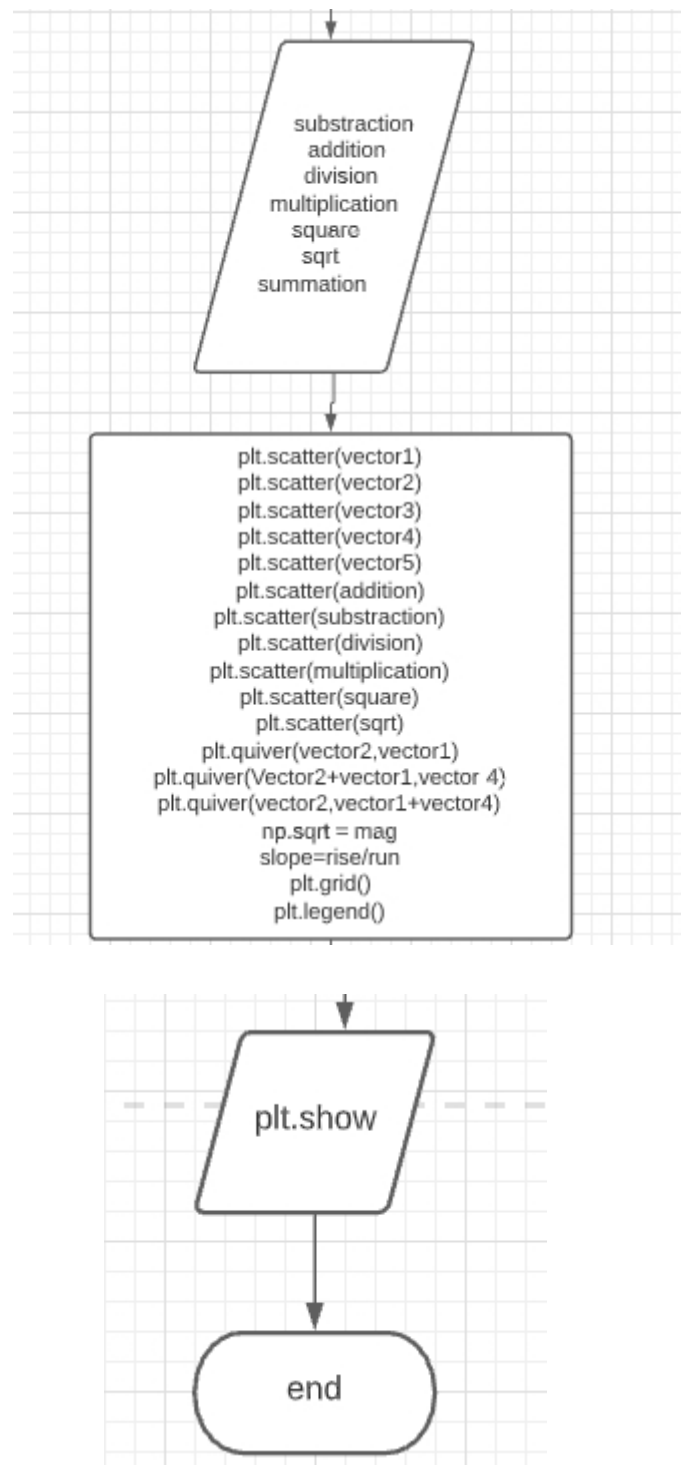


Figure 1.0 flowchart by Casiño

### III. Results

#### **Scalars**

Scalars are numerical entities that are represented by a single value.

```
In [5]: import numpy as np

x = np.array(-0.5)
x
```

```
Out[5]: array(-0.5)
```

Figure 1.1 Scalars by Dylan Josh Domingo Lopez

In this figure we can learn about scalars. Scalars can be determined by its singular represented value. Additional information, the data that will be stored by the scalar variable can be of the different type like string, character, floating point, a large group of strings.

#### **Vectors**

Vectors are array of numerical values or scalars that would represent any feature space function.

#### **Representing Vectors**

Now that you know how to represent vectors using their component and matrix form we can

$$A = 4\hat{x} + 3\hat{y} B = :$$

In which it's matrix equivalent is:

$$A = \begin{bmatrix} 4 \\ 3 \end{bmatrix}, B = \begin{bmatrix} 2 \\ -5 \end{bmatrix} A = \begin{bmatrix} 4 \\ 3 \end{bmatrix}$$

We can then start doing numpy code with this by:

```
In [6]: A = np.array([4,3])
B = np.array([2, -5])

print('Vector A is ', A)
print('Vector B is ', B)

Vector A is [4 3]
Vector B is [ 2 -5]
```

Figure 1.2 Vectors by Dylan Josh Domingo Lopez

In this Figure, Vectors are arrays of numerical values or with the help of scalars it can represent any point of space. Vector in the numpy library are the components, which make ordinary numbers combine together. The idea is that when a vector is in a list of numbers, we can use vector algebra as operations performed on the numbers in the list. In other words vector is the numpy 1-D array but may become 2-D depending in the given vectors.

### Describing vectors in NumPy

Describing vectors is very important if we want to perform basic to advanced operations with them. T size and dimensions.

```
In [13]: ### Checking shapes
### Shapes tells us how many rows and columns are there
ball1 = np.array([1,2,3])
ball2 = np.array([0,1,-1])
pool = np.array([J,K]) ## Matrix
pool.shape
```

Out[13]: (2, 3)

```
In [12]: U = np.array([
    [1, 2],
    [2, 3]
])
U.shape
```

Out[12]: (2, 2)

```
In [16]: ### Checking size
### Array/Vector sizes tells us many total number of elements are there in the
U.size
```

Out[16]: 4

```
In [17]: ### Checking dimensions
### The dimensions or rank of a vector tells us how many dimensions are there f
A.ndim
```

Out[17]: 1

```
In [18]: pool.ndim
```

Out[18]: 2

Figure 1.3 Describing vectors in Numpy by Dylan Josh Domingo Lopez

In this figure, vectors can be describe by its shape, size or dimension using Numpy. Using this feature will be valuable to check the shape and size of an array both for further

calculations and simply as a guide checking after some operation because when two vectors have unequal set of elements and forming them to a operation, it would say that it is in error.

## Scaling

Scaling or scalar multiplication takes a scalar

We can do this in numpy through:

```
In [35]: A = np.array([1,5,8,9])
         S = 5*A
         S
```

```
Out[35]: array([ 5, 25, 40, 45])
```

```
In [36]: S = np.multiply(5,A)
         S
```

```
Out[36]: array([ 5, 25, 40, 45])
```

```
In [38]: X = np.array([1, 1])
         Y = np.array([2, -3])
         R = np.subtract(np.multiply(3,X),Y)
         # R = 3*X - Y
         R
```

```
Out[38]: array([1, 6])
```

Figure 1.4 Scaling by Dylan Josh Domingo Lopez

In this figure, scaling or scalar multiplication can help us with the arithmetic of the vectors. The simple operations using the symbol operations has the same approach with matrix operation for vectors. Each operations with the library numpy is explained in the table.

Np.add()	This function is used to perform element wise matrix addition.
Np.substract()	This function is used to perform element wise matrix subtraction.
Np.divide()	This function is used to perform element wise matrix division.

Np.multiply()	This function is used to perform element wise matrix multiplication.
Np.dot	This function is used to compute the matrix multiplication, rather than element wise multiplication.
Np.sqrt()	This function is used to compute the square root of each element of matrix.
Np.sum()	This function is used to add all the elements in matrix. Optional “axis” argument computes the column sum if axis is 0 and row sum if axis is 1.
Np.t()	This argument is used to transpose the specified matrix.

Table 1 Matrix Operations by <https://www.geeksforgeeks.org/matrix-manipulation-python/>

$$R = 6\hat{x} - 2\hat{y} \quad \text{or} \quad R = \begin{bmatrix} 6 & -2 \end{bmatrix}$$

So let's try to do that in NumPy in several number of ways:

```
In [29]: position1 = np.array([0, 0, 0])
position2 = np.array([1, 1, 0])
position3 = np.array([-1, 2, 0])
position4 = np.array([2, 5, 3])

R = position1 + position2 + position3 + position4 #Eager execution
R
```

```
Out[29]: array([2, 8, 3])
```

```
In [25]: R1 = np.add(position1, position2) #functional method
R2 = np.add(R1, position3)
R3 = np.add(R2, position4)
R3
```

```
Out[25]: array([2, 7, 3])
```

```
In [30]: Rm = np.multiply(position3, position4)
Rm
```

```
Out[30]: array([-2, 10, 0])
```

Figure 1.5 Addition Rule by Dylan Josh Domingo Lopez

In this figure, we can figure out how we can add, subtract, multiply and divide elements of the matrices using the library of Numpy and also how to manually add vectors without the use of Numpy.

### Matplotlib

Matplotlib or MATLAB Plotting library is Python's take on MATLAB's plotting feature. Matplotlib can be used vastly from graphing values to visualizing several dimensions of data.

### Visualizing Data

It's not enough just solving these vectors so might need to visualize them. So we'll use Matplotlib for that. We'll need to import it first.

```
In [39]: import matplotlib.pyplot as plt  ## use this one if not in jupyterlab/notebook
# from matplotlib import pyplot as plt
import matplotlib
```

Figure 1.6 Matplotlib by Dylan Josh Domingo Lopez

In this figure, we can familiarize ourselves in the library of Matplotlib. This helps to plot the vectors we can obtain in the graph visualizing it in x and y dimensions.

```
In [54]: A = [2,-1]
B = [5,2]
plt.scatter(A[0],A[1], label='A', c='magenta')
plt.scatter(B[0],B[1], label='B', c='mediumspringgreen')

plt.grid()
plt.legend()
plt.show()
```

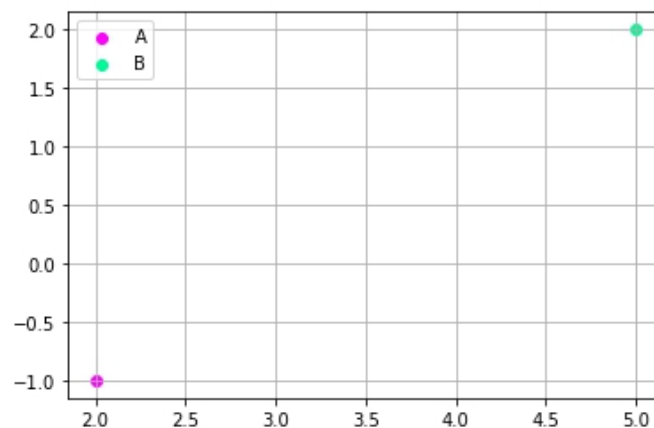


Figure 1.7 Visualization by Dylan Josh Domingo Lopez

In this figure, We will learn how to plot the vectors given in A and B. Using `plt.scatter` we can declare the first and second element of the vector and label it with whatever we want, and declare a color so we can differentiate it with another vector.



```
In [72]: A = np.array([-5,0])
B = np.array([0,5])

plt.title("Resultant Vector\nMagnitude:{:.2f}".format(R_mag))

plt.xlim(-15, 15)
plt.ylim(-15, 15)
# print(B)
plt.quiver(0,0, A[0], A[1], angles='xy', scale_units='xy',scale=1, color='red') # Red --> A
plt.quiver(A[0], A[1], B[0], B[1],angles='xy', scale_units='xy',scale=1, color='green')
R = A+B
plt.quiver(0, 0, R[0], R[1],angles='xy', scale_units='xy',scale=1, color='orange')
plt.grid()
plt.show()
```

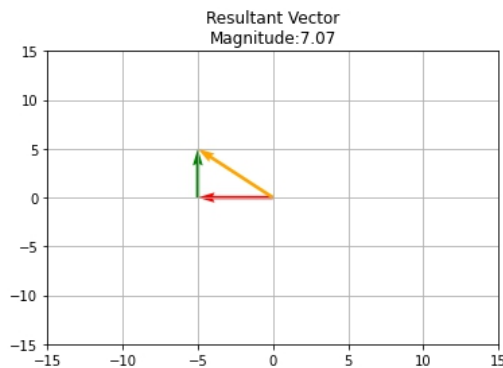


Figure 1.8 Resultant Vector by Dylan Josh Domingo Lopez

In this figure, we can identify the resultant vector using the Pythagorean theorem.

$$\sqrt{A^2 + B^2 + C^2}$$

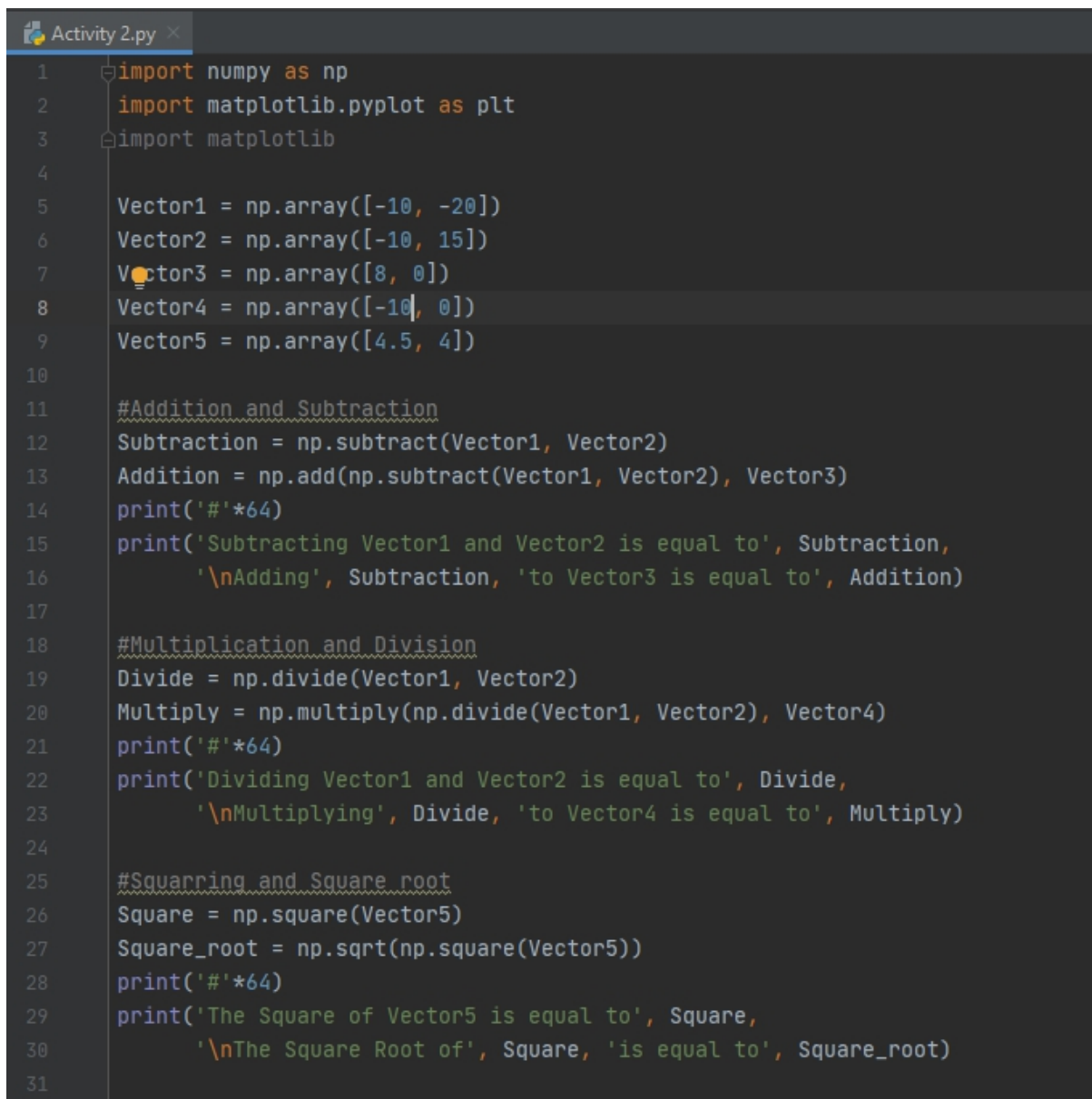
```
In [66]: R
Out[66]: array([-5,  5])

In [70]: R_mag = np.sqrt(np.sum(A**2+B**2)) ##Euclidean Distance / Euclidean Norm
rise = R[1]
run = R[0]
slope = rise/run
slope
## angle of the vector? arctan(rise/run)

Out[70]: -1.0
```

Figure 1.9 Slope by Dylan Josh Domingo Lopez

In this figure, we can determine the slope with the use of the rise and run formula.



```

1  import numpy as np
2  import matplotlib.pyplot as plt
3  import matplotlib
4
5  Vector1 = np.array([-10, -20])
6  Vector2 = np.array([-10, 15])
7  Vector3 = np.array([8, 0])
8  Vector4 = np.array([-10, 0])
9  Vector5 = np.array([4.5, 4])
10
11  #Addition and Subtraction
12  Subtraction = np.subtract(Vector1, Vector2)
13  Addition = np.add(np.subtract(Vector1, Vector2), Vector3)
14  print('#'*64)
15  print('Subtracting Vector1 and Vector2 is equal to', Subtraction,
16        '\nAdding', Subtraction, 'to Vector3 is equal to', Addition)
17
18  #Multiplication and Division
19  Divide = np.divide(Vector1, Vector2)
20  Multiply = np.multiply(np.divide(Vector1, Vector2), Vector4)
21  print('#'*64)
22  print('Dividing Vector1 and Vector2 is equal to', Divide,
23        '\nMultiplying', Divide, 'to Vector4 is equal to', Multiply)
24
25  #Squaring and Square root
26  Square = np.square(Vector5)
27  Square_root = np.sqrt(np.square(Vector5))
28  print('#'*64)
29  print('The Square of Vector5 is equal to', Square,
30        '\nThe Square Root of', Square, 'is equal to', Square_root)
31

```

Figure 2.1 Codes by Casio

In this figure, the programmer have declared different vectors with their own respective value. With the help of the vectors that we have, the programmer will use them to portray how to add, subtract, multiply, divide, square, square root, and summation. Addition and Subtraction will add and subtract arrays and requires two array with the same shape and dimension. Multiplication and Division will multiply and divide arrays and also requires two arrays to return a product or quotient. Squaring and Square root requires only one array, this will return as a square of the array and root of the array. Summation will give the sum of arrays over a specified axis.

```

33 #Summation
34 Summation = np.sum(Vector5)
35 print('#'*64)
36 print('The Summation of Vector5 is equal to', Summation)
37
38
39 #Visualizing Data
40 print('#'*64)
41 plt.title('Visualization of Data')
42 plt.xlim(-60, 60)
43 plt.ylim(-60, 60)
44
45 #Vector
46 plt.scatter(Vector1[0], Vector1[1], label='Vector1', c='Yellow')
47 plt.scatter(Vector2[0], Vector2[1], label='Vector2', c='OliveDrab')
48 plt.scatter(Vector3[0], Vector3[1], label='Vector3', c='YellowGreen')
49 plt.scatter(Vector4[0], Vector4[1], label='Vector4', c='darkolivegreen')
50 plt.scatter(Vector5[0], Vector5[1], label='Vector5', c='greenYellow')
51
52 #Results
53 plt.scatter(Addition[0], Addition[1], label='Addition', c='teal')
54 plt.scatter(Subtraction[0], Subtraction[1], label='Subtraction', c='cyan')
55 plt.scatter(Divide[0], Divide[1], label='Division', c='cadetblue')
56 plt.scatter(Multiply[0], Multiply[1], label='Multiplication', c='powderblue')
57 plt.scatter(Square[0], Square[1], label='Square', c='deeppskyblue')
58 plt.scatter(Square_root[0], Square_root[1], label='Squareroot', c='skyblue')
59
60 #Resultant Vector
61 plt.quiver(Vector2[0], Vector2[1], Vector1[0], Vector1[1], angles='xy', scale_units='xy', scale=1, color='Pink')
62 plt.quiver(-20, -5, Vector4[0], Vector4[1], angles='xy', scale_units='xy', scale=1, color='red')
63 Vector6 = Vector1 + Vector4
64 plt.quiver(Vector2[0], Vector2[1], Vector6[0], Vector6[1], angles='xy', scale_units='xy', scale=1, color='blue')
65
66
67 R_mag = np.sqrt(np.sum(Vector1**2+Vector4**2)) ##Euclidean Distance / Euclidean Norm
68 rise = Vector6[1]
69 run = Vector6[0]
70 slope = rise/run
71 print('The slope is: ',slope)
72
73
74
75 plt.grid()
76 plt.legend()
77 plt.show()
78
79
80

```

Figure 2.2 Codes by Casiño

This figure shows the plotting points of each vector and results in the graph. It also shows the resultant vector and the slope. `Plt.scatter` is used to pinpoint any given of x and y which is equal to index 0 and index 1 inside of a vector. `Plt.quiver` is used to determine where the arrow will go with a argument of based on my observation (x1,y1) for its origin and (x2,y2) for its destination.

```

Activity 2 x
C:\Users\Casino\AppData\Local\Programs\Python\Python37\python.exe "C:/Users/C
#####
Subtracting Vector1 and Vector2 is equal to [ 0 -35]
Adding [ 0 -35] to Vector3 is equal to [ 8 -35]
#####
Dividing Vector1 and Vector2 is equal to [ 1.          -1.33333333]
Multiplying [ 1.          -1.33333333] to Vector4 is equal to [-10.  -0.]
#####
The Square of Vector5 is equal to [20.25 16. ]
The Square Root of [20.25 16. ] is equal to [4.5 4. ]
#####
The Summation of Vector5 is equal to 8.5
#####
The slope is: 1.0

```

Figure 3.1 Output by Casio

This is the overall output printed(the graph is in the next figure). Results are presented here based on the codes in Figures 2.1 and 2.2.

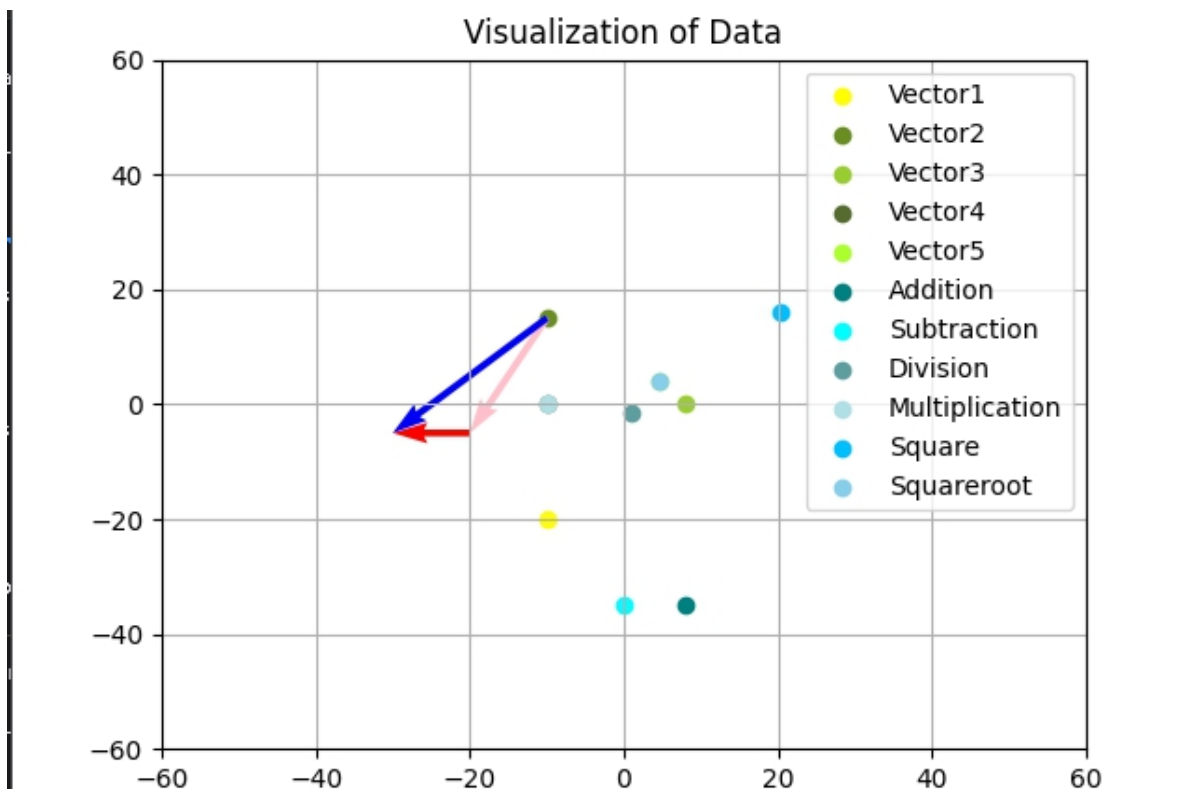


Figure 3.2 Output by Casio

In this figure, the graph shows all the plots of the vectors and their result based on figure 3.1. The resultant vector is also portrayed here.

## IV. Conclusion

The programmer have realized that NumPy is best used in numerical problems and with the combination of Matplotlib, it helps portray the plots of each vector and it also lets you perform a whole lot of complex scientific calculations on these arrays. The programmer also concluded that the library numpy and the python syntax “list” is almost the same but with a much better and wider area for numerical improvements. To the idea that if we are given a large amount. These two combined make a wide range of methods in computation and graphical efficiency. As a programmer, these libraries best help in statistics because NumPy helps in assigning and computes values for getting the results in each vector operations then proceeding with the use of matplotlib for graphing data of the vectors and plotting all the points and quivers with its respective labels.

## References

- [1] Dyjdlopez, “dyjdlopez/linearAlgebra2021,” *GitHub*. [Online]. Available: <https://github.com/dyjdlopez/linearAlgebra2021>. [Accessed: 20-Feb-2021].
- [2] “numpy.sum() in Python,” *GeeksforGeeks*, 03-Apr-2020. [Online]. Available: <https://www.geeksforgeeks.org/numpy-sum-in-python/>. [Accessed: 20-Feb-2021].
- [3] “numpy.addition() in Python,” *GeeksforGeeks*, 03-Apr-2020. [Online]. Available: <https://www.geeksforgeeks.org/numpy-addition-in-python/>. [Accessed: 20-Feb-2021].
- [4] “numpy.subtraction() in Python,” *GeeksforGeeks*, 03-Apr-2020. [Online]. Available: <https://www.geeksforgeeks.org/numpy-subtraction-in-python/>. [Accessed: 20-Feb-2021].
- [5] “numpy.multiplication() in Python,” *GeeksforGeeks*, 03-Apr-2020. [Online]. Available: <https://www.geeksforgeeks.org/numpy-multiplication-in-python/>. [Accessed: 20-Feb-2021].
- [6] “numpy.division() in Python,” *GeeksforGeeks*, 03-Apr-2020. [Online]. Available: <https://www.geeksforgeeks.org/numpy-division-in-python/>. [Accessed: 20-Feb-2021].
- [7] “numpy.square() in Python,” *GeeksforGeeks*, 03-Apr-2020. [Online]. Available: <https://www.geeksforgeeks.org/numpy-square-in-python/>. [Accessed: 20-Feb-2021].
- [8] “numpy.squareroot in Python,” *GeeksforGeeks*, 03-Apr-2020. [Online]. Available: <https://www.geeksforgeeks.org/numpy-squareroot-in-python/>. [Accessed: 20-Feb-2021].