# Numpy:

* 1. Using NumPy create random vector of size 15 having only Integers in the range 1-20.
     1. Reshape the array to 3 by 5
     2. Print array shape.
     3. Replace the max in each row by 0

Create a 2-dimensional array of size 4 x 3 (composed of 4-byte integer elements), also print the shape, type and data type of the array.

import numpy as np  
 rand = np.random.randint(1,20,15) #create random vector of size 15 having only Integers in the range 1-20.  
 print("Original array:")  
 print(rand)  
 rand=np.reshape(rand,(3,5),order='c') #Reshape the array to 3 by 5  
 print(rand)  
 print(np.shape(rand)) #Print array shape  
 maxNum = np.amax(rand, axis=1)  
 rand=np.where(np.isin(rand,maxNum), 0, rand) #Replace the max in each row by 0  
 print("Maximum value of each row replaced by 0:")  
 print(rand)  
}

**Original array:**

**[ 2 3 14 9 8 11 2 3 19 5 5 12 14 11 6]**

**[[ 2 3 14 9 8]**

**[11 2 3 19 5]**

**[ 5 12 14 11 6]]**

**(3, 5)**

**Maximum value of each row replaced by 0:**

**[[ 2 3 0 9 8]**

**[11 2 3 0 5]**

**[ 5 12 0 11 6]]**

* 1. Write a program to compute the eigenvalues and right eigenvectors of a given square array given below: [[ 3 -2]

[ 1 0]]

import numpy as np  
 A = np.array([[3, -2],  
 [1, 0]])  
 eigenvalues, eigenvectors = np.linalg.eig(A)  
 print("Eigenvalues:")  
 for eigenvalue in eigenvalues:  
 print(eigenvalue)  
 print("\nRight Eigenvectors:")  
 for i in range(len(eigenvectors)):  
 print(f"Eigenvector {i+1}: {eigenvectors[:, i]}")

print(result)

Original matrix:

[[ 3 -2]

[ 1 0]]

Eigenvalues of the said matrix [ 2. 1.]

Eigenvectors of the said matrix [[ 0.89442719 0.70710678]

[ 0.4472136 0.70710678]]

* 1. Compute the sum of the diagonal element of a given array. [[0 1 2]

[3 4 5]]

import numpy as np  
 A = np.array([[0, 1, 2],  
 [3, 4, 5]])  
 diagonal\_sum = np.trace(A)  
 print("Sum of diagonal elements:", diagonal\_sum)

print(result)

Original matrix:

[[0 1 2]

[3 4 5]]

Condition number of the said matrix: 4

* 1. Write a NumPy program to create a new shape to an array without changing its data. Reshape 3x2:

[[1 2]

[3 4]

[5 6]]

Reshape 2x3:

[[1 2 3]

[4 5 6]]

import numpy as np  
 A = np.array([[1, 2],  
 [3, 4],  
 [5, 6]])  
 reshape\_3x2 = A.reshape(3, 2)  
 print("Reshape 3x2:")  
 print(reshape\_3x2)  
 reshape\_2x3 = A.reshape(2, 3)  
 print("\nReshape 2x3:")  
 print(reshape\_2x3)

output the reshaped arrays:

Reshape 3x2:

[[1 2]

[3 4]

[5 6]]

Reshape 2x3:

[[1 2 3]

[4 5 6]]

# Matplotlib

1. Write a Python programming to create a below chart of the popularity of programming Languages.
2. Sample data:

Programming languages: Java, Python, PHP, JavaScript, C#, C++ Popularity: 22.2, 17.6, 8.8, 8, 7.7, 6.7

import matplotlib.pyplot as plt  
 # Data to plot  
 languages = 'Java', 'Python', 'PHP', 'JavaScript', 'C#', 'C++'  
 popuratity = [22.2, 17.6, 8.8, 8, 7.7, 6.7]  
 colors = ["#1f77b4", "#ff7f0e", "#2ca02c", "#d62728", "#9467bd", "#8c564b"]  
 # explode 1st slice  
 explode = (0.1, 0, 0, 0,0,0)  
 # Plot  
 plt.pie(popuratity, explode=explode, labels=languages, colors=colors,  
 autopct='%1.1f%%', shadow=True, startangle=140)  
  
 plt.axis('equal')  
 plt.show()

