Summer 2023: CS5710 – Machine Learning

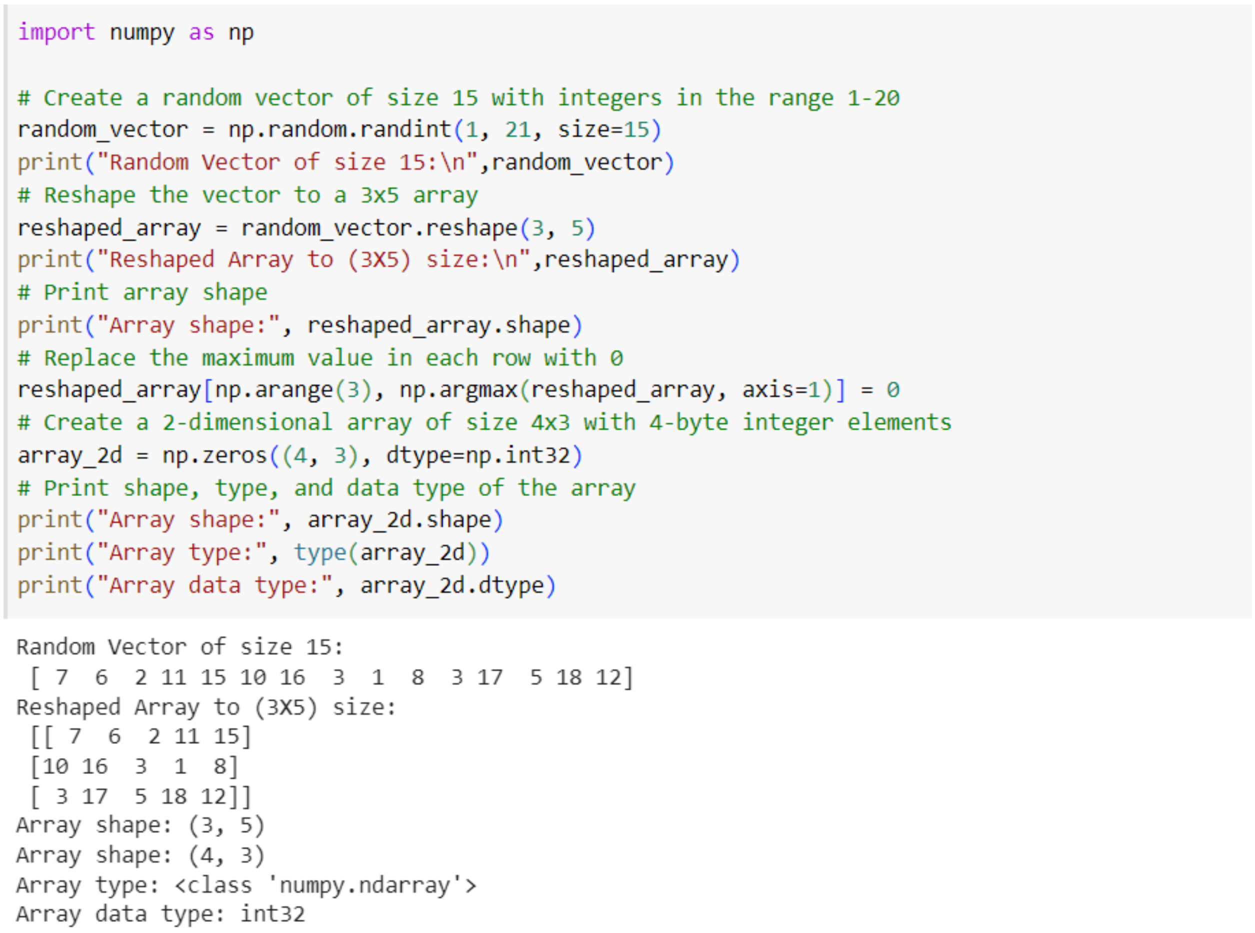
Assignment-1

Hima Bindu Desalanka - 700739704

a. 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.

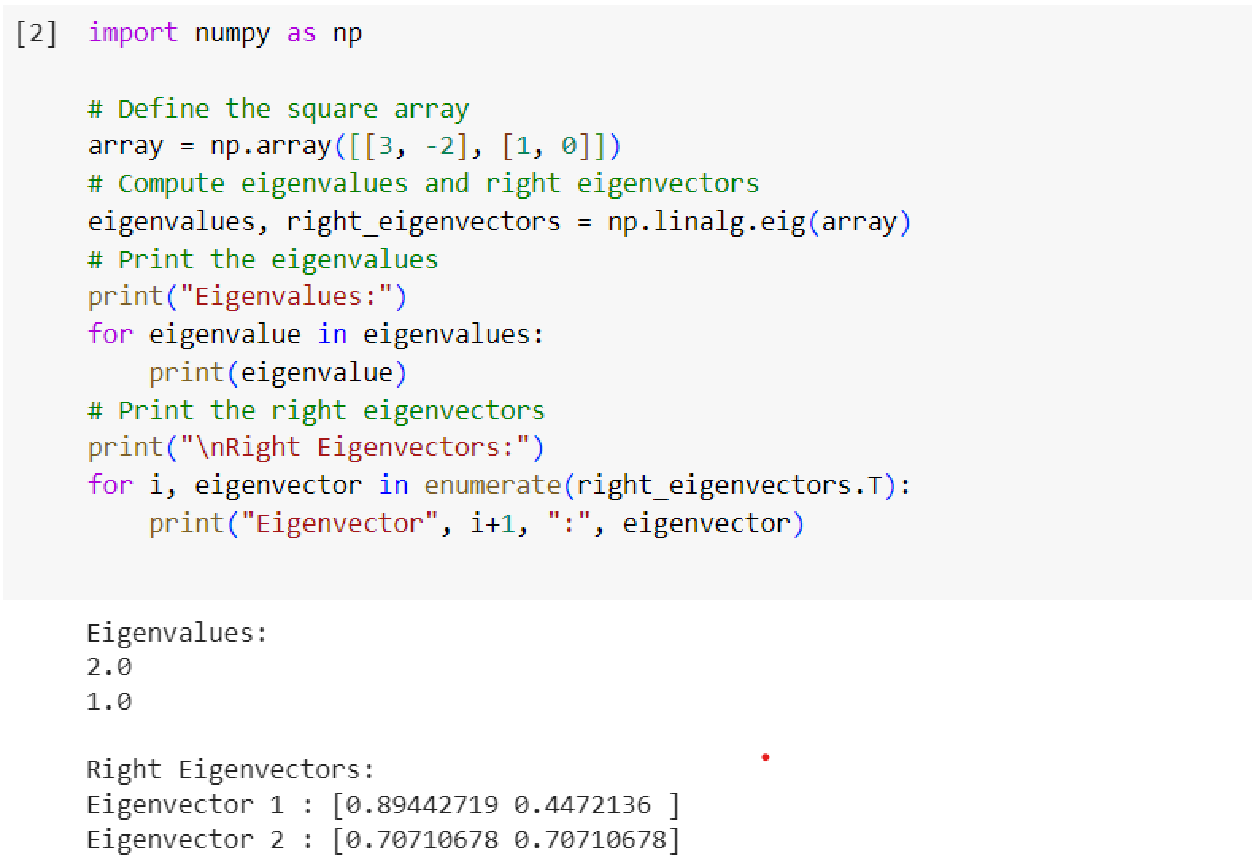


```python
import numpy as np

# Create a random vector of size 15 with integers in the range 1-20
random_vector = np.random.randint(1, 21, size=15)
print("Random Vector of size 15:\n",random_vector)
# Reshape the vector to a 3x5 array
reshaped_array = random_vector.reshape(3, 5)
print("Reshaped Array to (3X5) size:\n",reshaped_array)
# Print array shape
print("Array shape:", reshaped_array.shape)
# Replace the maximum value in each row with 0
reshaped_array[np.arange(3), np.argmax(reshaped_array, axis=1)] = 0
# Create a 2-dimensional array of size 4x3 with 4-byte integer elements
array_2d = np.zeros((4, 3), dtype=np.int32)
# Print shape, type, and data type of the array
print("Array shape:", array_2d.shape)
print("Array type:", type(array_2d))
print("Array data type:", array_2d.dtype)
```

```
Random Vector of size 15:
 [ 7  6  2 11 15 10 16  3  1  8  3 17  5 18 12]
Reshaped Array to (3X5) size:
 [[ 7  6  2 11 15]
 [10 16  3  1  8]
 [ 3 17  5 18 12]]
Array shape: (3, 5)
Array shape: (4, 3)
Array type: <class 'numpy.ndarray'>
Array data type: int32
```

The numpy library is imported in the code above and offers functions for dealing with arrays and mathematical issues. We employ the random module's randint function to produce the random number. With the specified size and range, the Randint function generates random numbers. Then, without changing the elements of the vector, we reshape the vector with the specified dimension using the reshape function. We produce the modified array's shape using the shape attribute. This example's array has 3 rows and 5 columns, so the shape tells us what those dimensions are, which in this case would be (3, 5). We use the np.argmax() function to obtain the indices of the highest values in each row of the reshaped array. In this case, axis 1 (rows) is the given axis, and the function argmax() returns the index of the largest element along that axis. This reveals the location of each row's highest value. Using the acquired indices, the maximum values in each row are changed to 0. The np.arange(3) function and the acquired indices are used to select the corresponding rows and set their maximum values to 0. Next, we create a brand-new 2-dimensional array called array_2d using np.zeros(). The array produced by this function is entirely made up of zeros. If the array's size is specified as being 4x3, then it will have 4 rows and 3 columns. Using the shape attribute, we print the shape of the array_2d. This will reveal the array's dimensions, which in this instance are (4, 3).Using the type() function, we also print the type of the array_2d. It informs us that the array is a member of the NumPy ndarray class.Finally, we use the dtype attribute to print the array's data type. Since the array in this instance has an int32 data type, it contains 4-byte integers.

b. Write a program to compute the eigenvalues and right eigenvectors of a given square array given below: [[3 -2] [1 0]]



```
[2]  import numpy as np

     # Define the square array
     array = np.array([[3, -2], [1, 0]])
     # Compute eigenvalues and right eigenvectors
     eigenvalues, right_eigenvectors = np.linalg.eig(array)
     # Print the eigenvalues
     print("Eigenvalues:")
     for eigenvalue in eigenvalues:
         print(eigenvalue)
     # Print the right eigenvectors
     print("\nRight Eigenvectors:")
     for i, eigenvector in enumerate(right_eigenvectors.T):
         print("Eigenvector", i+1, ":", eigenvector)
```

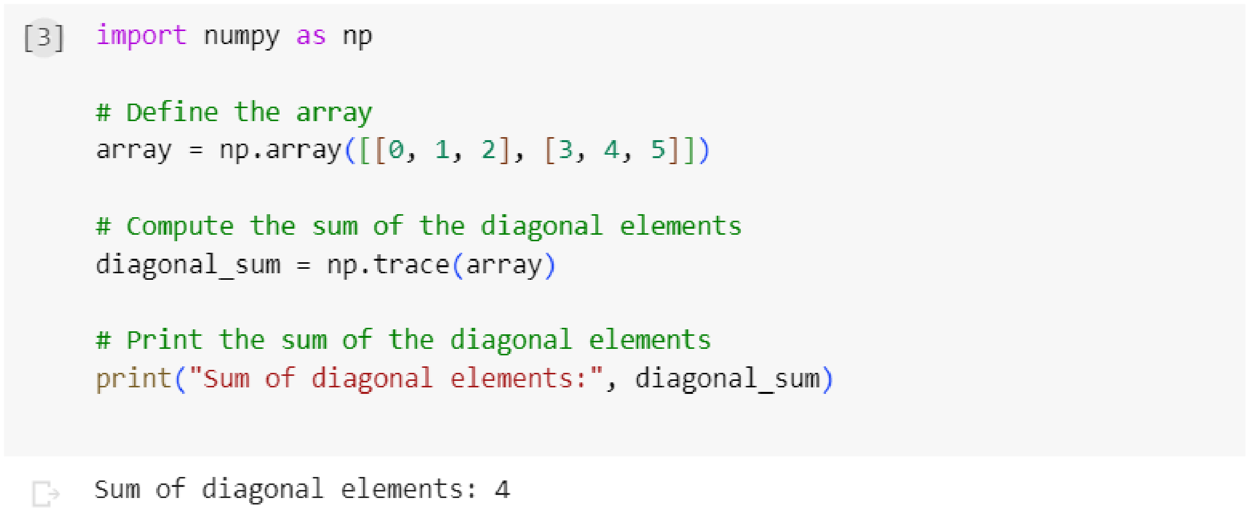
```
Eigenvalues:
2.0
1.0

Right Eigenvectors:
Eigenvector 1 : [0.89442719 0.4472136 ]
Eigenvector 2 : [0.70710678 0.70710678]
```

The square array is defined using the np.array() function. The array is being assigned by [[3, -2], [1, 0]].We use the np.linalg.eig() function to find the right eigenvectors and eigenvalues of the array. The eig() function takes an array as input and returns right eigenvectors and eigenvalues.We print the eigenvalues utilizing a loop. Each eigenvalue in the array of eigenvalues is printed separately.The appropriate eigenvectors are present in the resulting 2-dimensional array. Each of the array's columns represents a different eigenvector.To print the appropriate eigenvectors, we use another loop. We print the array of right eigenvectors' individual columns (or eigenvectors) one at a time.The magnitudes of the eigenvectors are scaled to unit length and normalized to 1.

c. Compute the sum of the diagonal element of a given array.

[[0 1 2]

[3 4 5]]



```
[3] import numpy as np

    # Define the array
    array = np.array([[0, 1, 2], [3, 4, 5]])

    # Compute the sum of the diagonal elements
    diagonal_sum = np.trace(array)

    # Print the sum of the diagonal elements
    print("Sum of diagonal elements:", diagonal_sum)
```

```
Sum of diagonal elements: 4
```

We employ the np.trace() function to determine the total of the diagonal array's constituents. The diagonal element of an array is the main diagonal, which extends from top-left to bottom-right.The sum of a 2-dimensional array's diagonal elements is calculated using the np.trace() function. It determines the total number of items in the diagonal array in this case.We store the sum of the diagonal elements in the variable diagonal_sum.We show the sum of the diagonal elements using print(). The total in this case is 4.

d. Write a NumPy program to create a new shape to an array without changing its data.

Reshape 3x2:

[Type here]

[[1 2]

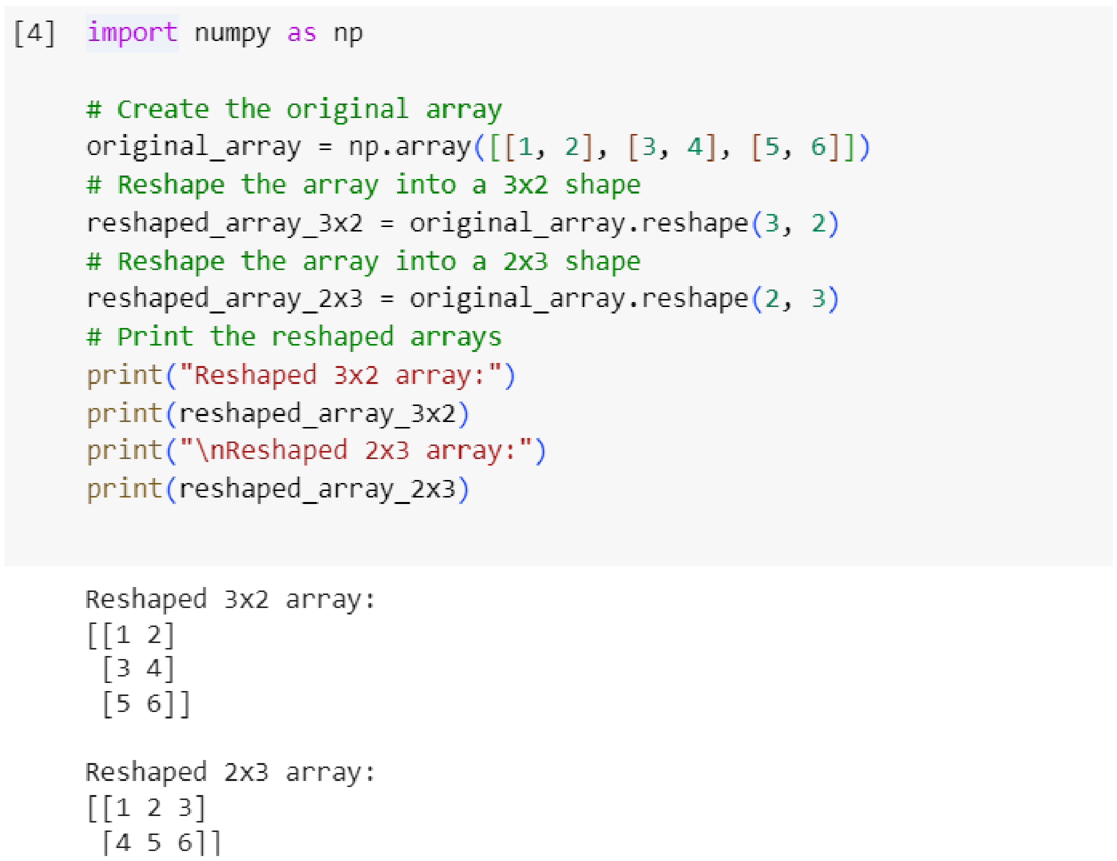
[3 4]

[5 6]]

Reshape 2x3:

[[1 2 3]

[[4 5 6]]

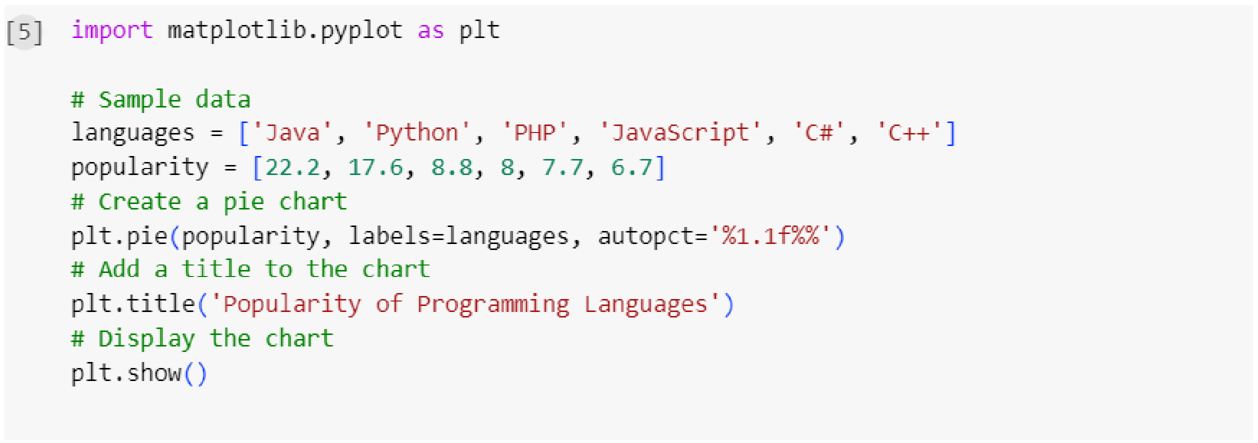


```
[4]  import numpy as np

     # Create the original array
     original_array = np.array([[1, 2], [3, 4], [5, 6]])
     # Reshape the array into a 3x2 shape
     reshaped_array_3x2 = original_array.reshape(3, 2)
     # Reshape the array into a 2x3 shape
     reshaped_array_2x3 = original_array.reshape(2, 3)
     # Print the reshaped arrays
     print("Reshaped 3x2 array:")
     print(reshaped_array_3x2)
     print("\nReshaped 2x3 array:")
     print(reshaped_array_2x3)
```

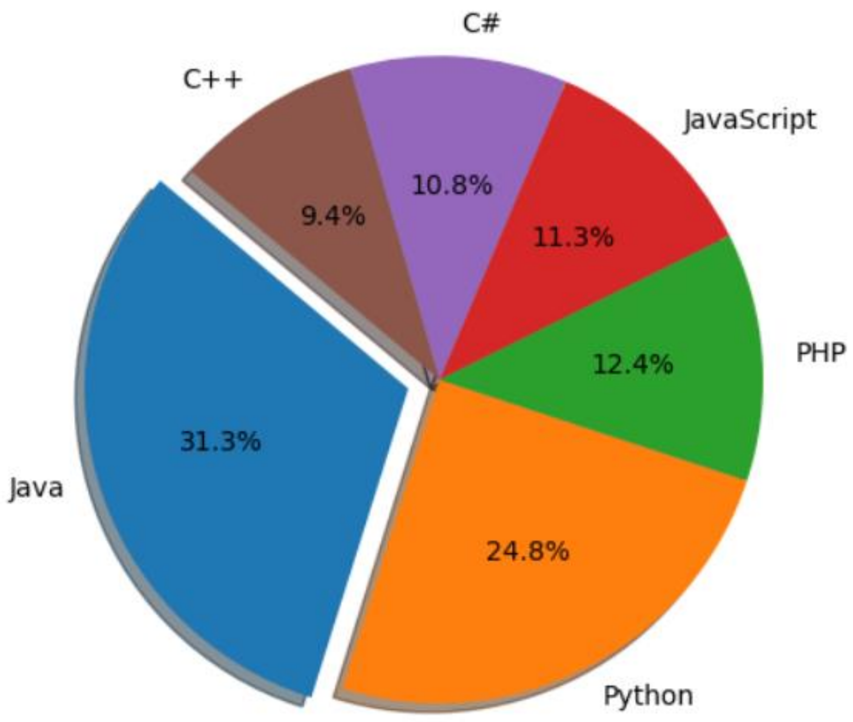
```
Reshaped 3x2 array:
[[1 2]
 [3 4]
 [5 6]]

Reshaped 2x3 array:
[[1 2 3]
 [4 5 6]]
```

The reshape() function is used to create two reshaped arrays, reshaped_array_3x2 and reshaped_array_2x3.Using the reshape() function, we can alter the shape of an array without altering its contents. To call this function, we use the original array and pass the necessary shape as an argument.The reshaped arrays are, respectively, contained in the variables reshaped_array_3x2 and reshaped_array_2x3.

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



```
[5] import matplotlib.pyplot as plt

    # Sample data
    languages = ['Java', 'Python', 'PHP', 'JavaScript', 'C#', 'C++']
    popularity = [22.2, 17.6, 8.8, 8, 7.7, 6.7]
    # Create a pie chart
    plt.pie(popularity, labels=languages, autopct='%1.1f%%')
    # Add a title to the chart
    plt.title('Popularity of Programming Languages')
    # Display the chart
    plt.show()
```



[Type here]

The Matplotlib library, which provides tools for creating visualizations, is first imported.These are the two lists that we define: languages and popularity. These lists display the programming languages along with the popularity rankings for each. For instance, "Java" is popular 22.2% of the time, "Python" 17.6% of the time, and so on.With the plt.pie() function, we can create a pie chart. For the chart data, this function accepts the popularity list as input, and for the names of the programming languages, it accepts the labels argument.The autopct parameter is used to format the percentage numbers shown on the chart. In this case, we use the notation "%1.1f%%" for the percentage numbers to indicate one decimal place.

https://drive.google.com/file/d/1Bzj38pngqzxPBiam7-nVOAyZCfmhDxxF/view?usp=drive_link --> this is video link

https://github.com/hbdesalanka/assignment1 --> this is public repository link