MACHINE LEARNING
ASSIGNMENT3

NAME:ANJALI ERRA(700740323)
Github link: https://github.com/Anjali555-erra/ML-Assignment3.git

Video Link:
https://drive.google.com/file/d/1Ba4EGYlou3hSsmZcUwb6sTxBlUEO3wzh/view?usp=share_link

1. Numpy:

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.

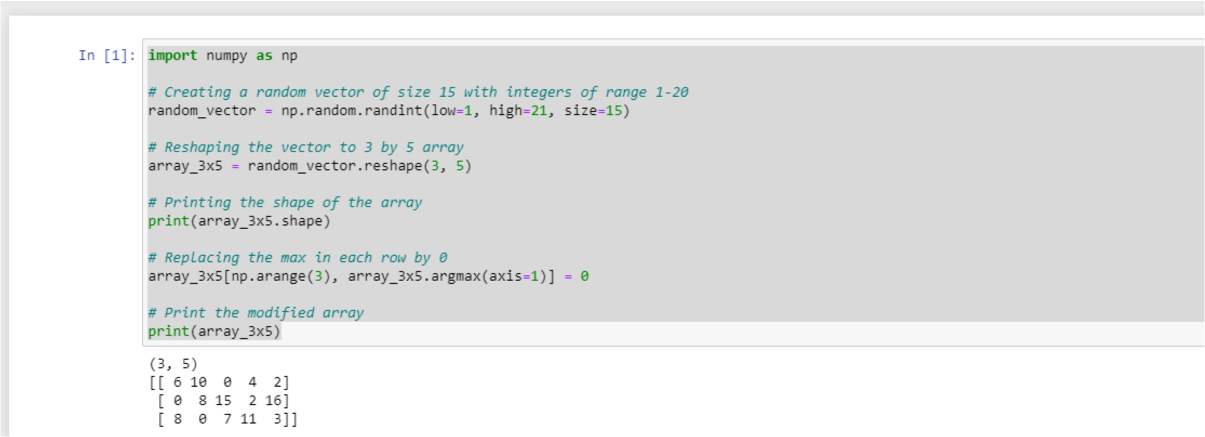
3. Replace the max in each row by 0

Source Code:

```
import numpy as np


# Creating a random vector of size 15 with integers of range 1-20

random_vector = np.random.randint(low=1, high=21, size=15)


# Reshaping the vector to 3 by 5 array

array_3x5 = random_vector.reshape(3, 5)


# Printing the shape of the array

print(array_3x5.shape)


# Replacing the max in each row by 0

array_3x5[np.arange(3), array_3x5.argmax(axis=1)] = 0


# Print the modified array
```

print(array_3x5)

Output:



```
In [1]: import numpy as np

        # Creating a random vector of size 15 with integers of range 1-20
        random_vector = np.random.randint(low=1, high=21, size=15)

        # Reshaping the vector to 3 by 5 array
        array_3x5 = random_vector.reshape(3, 5)

        # Printing the shape of the array
        print(array_3x5.shape)

        # Replacing the max in each row by 0
        array_3x5[np.arange(3), array_3x5.argmax(axis=1)] = 0

        # Print the modified array
        print(array_3x5)
```

```
(3, 5)
[[ 6 10  0  4  2]
 [ 0  8 15  2 16]
 [ 8  0  7 11  3]]
```

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.

b. Write a program to compute the eigenvalues and right eigenvectors of a given square array given below:

[[3 -2]

[1 0]]

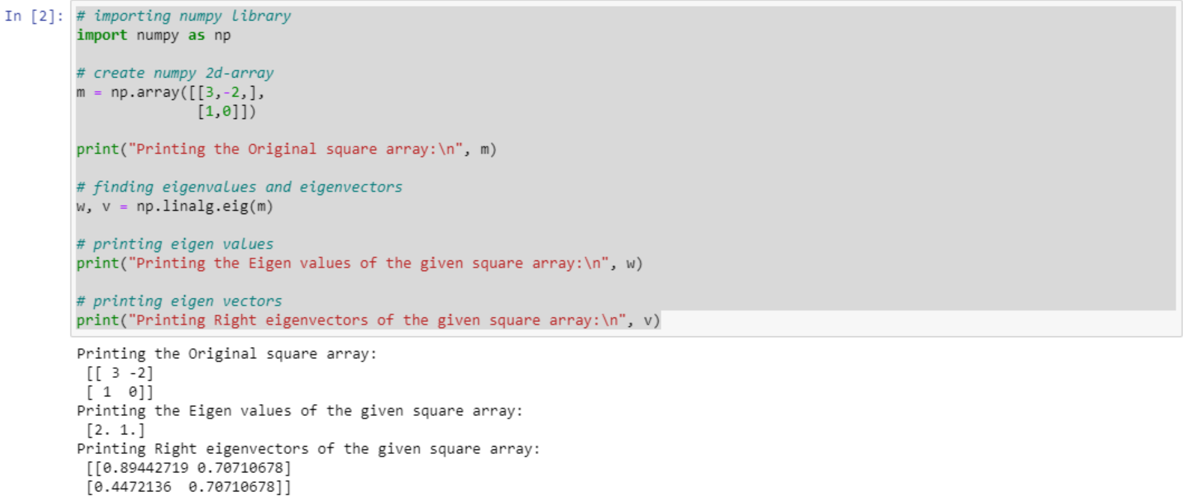
Source Code:

```
# importing numpy library

import numpy as np


# create numpy 2d-array

m = np.array([[3,-2,],

           [1,0]])


print("Printing the Original square array:\n", m)


# finding eigenvalues and eigenvectors

w, v = np.linalg.eig(m)


# printing eigen values

print("Printing the Eigen values of the given square array:\n", w)
```

printing eigen vectors

print("Printing Right eigenvectors of the given square array:\n", v)

Output:



```
In [2]: # importing numpy library
        import numpy as np

        # create numpy 2d-array
        m = np.array([[3,-2,],
                      [1,0]])

        print("Printing the Original square array:\n", m)

        # finding eigenvalues and eigenvectors
        w, v = np.linalg.eig(m)

        # printing eigen values
        print("Printing the Eigen values of the given square array:\n", w)

        # printing eigen vectors
        print("Printing Right eigenvectors of the given square array:\n", v)
```

```
Printing the Original square array:
 [[ 3 -2]
 [ 1  0]]
Printing the Eigen values of the given square array:
 [2. 1.]
Printing Right eigenvectors of the given square array:
 [[0.89442719 0.70710678]
 [0.4472136  0.70710678]]
```

c. Compute the sum of the diagonal element of a given array. [[0 1 2] [3 4 5]]

source code:

import numpy as np

m = np.arange(6).reshape(2,3)

print("Original matrix:")

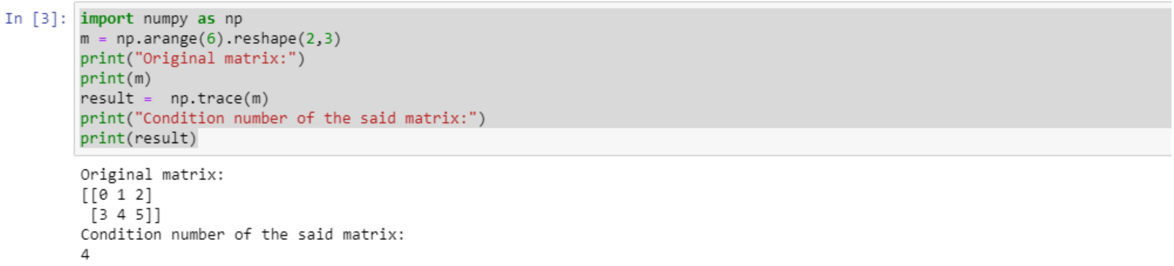
print(m)

result = np.trace(m)

print("Condition number of the said matrix:")

print(result)

Output:



```
In [3]: import numpy as np
        m = np.arange(6).reshape(2,3)
        print("Original matrix:")
        print(m)
        result =  np.trace(m)
        print("Condition number of the said matrix:")
        print(result)
```

```
Original matrix:
[[0 1 2]
 [3 4 5]]
Condition number of the said matrix:
4
```

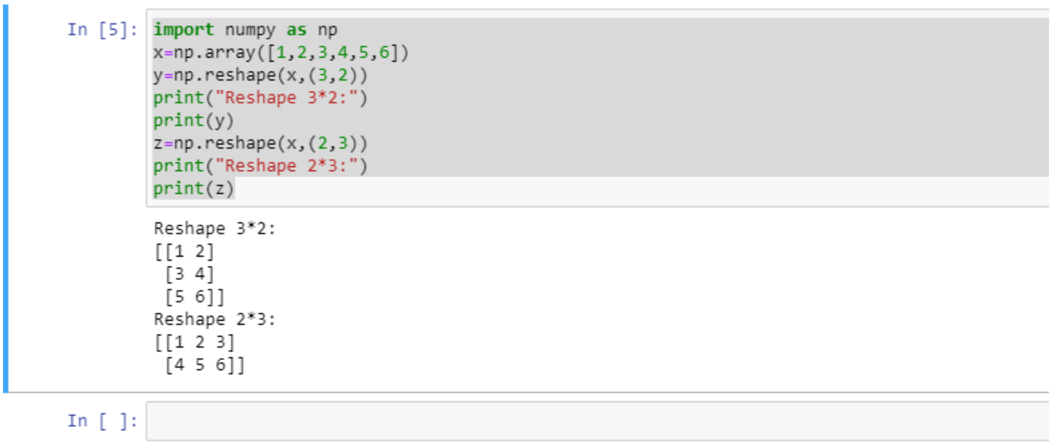
d. Write a NumPy program to create a new shape to an array without changing its data. Reshape 3x2: [[1 2] [3 4] [5 6]] Reshape 2x3: [[1 2 3] [4 5 6]]

Source Code:

import numpy as np

```
x=np.array([1,2,3,4,5,6])

y=np.reshape(x,(3,2))

print("Reshape 3*2:")

print(y)

z=np.reshape(x,(2,3))

print("Reshape 2*3:")

print(z)
```

Output:



```
In [5]: import numpy as np
        x=np.array([1,2,3,4,5,6])
        y=np.reshape(x,(3,2))
        print("Reshape 3*2:")
        print(y)
        z=np.reshape(x,(2,3))
        print("Reshape 2*3:")
        print(z)

        Reshape 3*2:
        [[1 2]
         [3 4]
         [5 6]]
        Reshape 2*3:
        [[1 2 3]
         [4 5 6]]

In [ ]:
```

2. 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

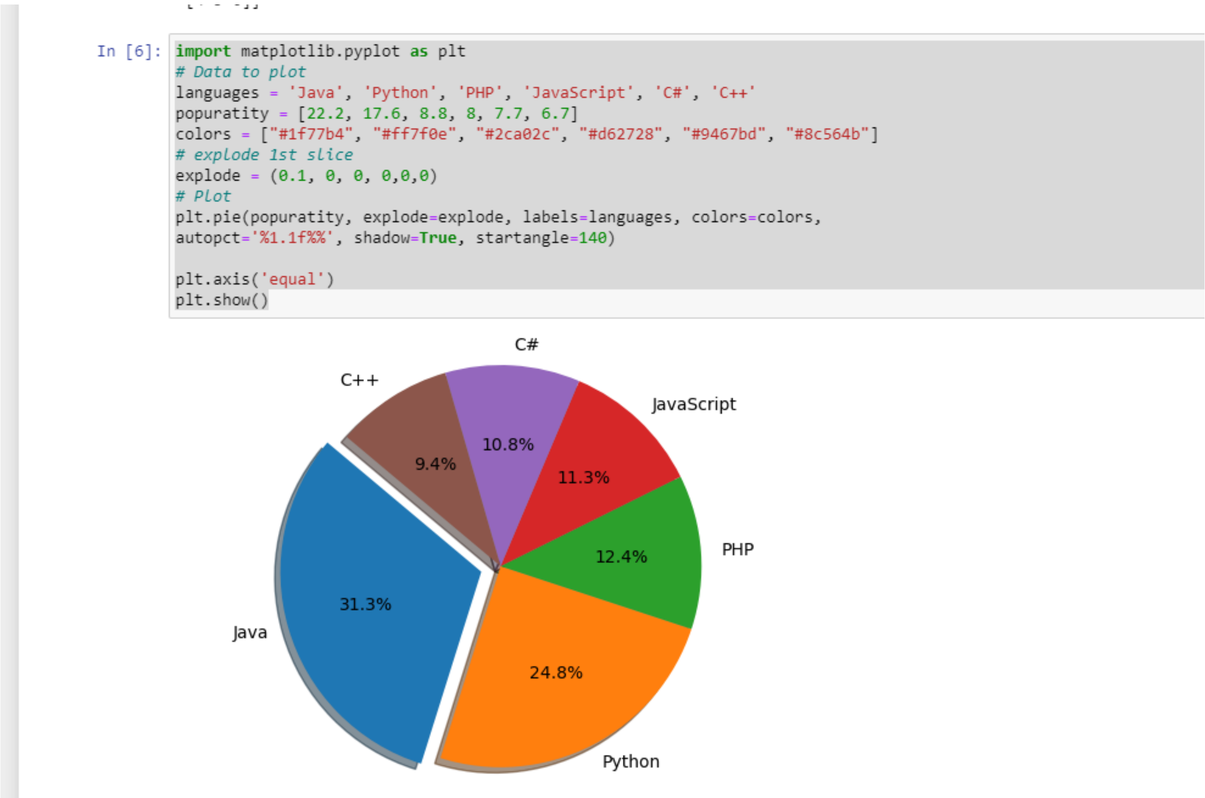
Source code:

```
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()

Output:



```
In [6]: 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()
```