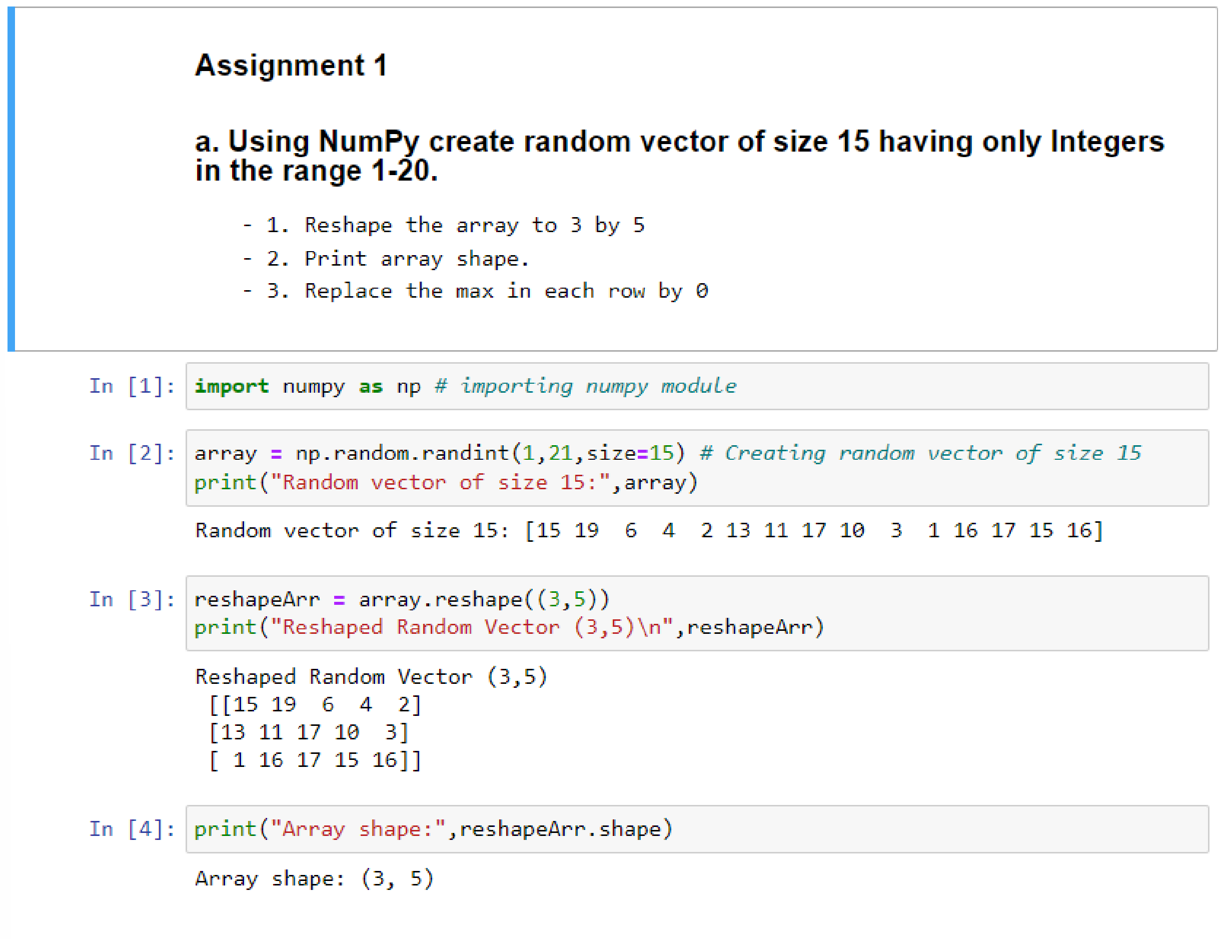
Name: Deepika Rani Gokavarapu
700#: 700740599
Programming Assignment 1

GitHub Link: https://github.com/GDeepikarani59/MachineLearning_Assignment1
Video Link: https://drive.google.com/file/d/15ajNWyxPjz1axvE-LH2AWMFMrpCLXmIi/view?usp=drive_link



Assignment 1

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
```

In [1]:
```python
import numpy as np # importing numpy module
```

In [2]:
```python
array = np.random.randint(1,21,size=15) # Creating random vector of size 15
print("Random vector of size 15:",array)
```

```
Random vector of size 15: [15 19  6  4  2 13 11 17 10  3  1 16 17 15 16]
```

In [3]:
```python
reshapeArr = array.reshape((3,5))
print("Reshaped Random Vector (3,5)\n",reshapeArr)
```

```
Reshaped Random Vector (3,5)
 [[15 19  6  4  2]
 [13 11 17 10  3]
 [ 1 16 17 15 16]]
```

In [4]:
```python
print("Array shape:",reshapeArr.shape)
```

```
Array shape: (3, 5)
```

With the help of NumPy, the following code generates an array of 15 numbers, resizes it into a 3x5 array, then substitutes the maximum value for each row with 0. It creates a vector using random.randint(), reshapes it using reshape(), and then uses print() to show the shape of the array. Using argmax() and NumPy's indexing syntax, the maximum value in each row is changed to 0 for each row. After that, print() is used to display the output array.

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

[[3 -2] [1 0]]

In [9]:
```
a2 = np.array([[3,-2],[1,0]])
print("Given Array\n",a2)
```

```
Given Array
 [[ 3 -2]
 [ 1  0]]
```

In [10]:
```
eigenValue, eigenVector = np.linalg.eig(a2)

print("Eigen Values:\n",eigenValue)
print("Eigen Vectors:\n",eigenVector)
```

```
Eigen Values:
 [2. 1.]
Eigen Vectors:
 [[0.89442719 0.70710678]
 [0.4472136  0.70710678]]
```

The eigenvalues and right eigenvectors of a given 2x2 square array are calculated using NumPy in this code. The right eigenvectors and eigenvalues are calculated using np.linalg.eig(), and the array is constructed using np.array(). Print() is used to show the resulting eigenvalues and eigenvectors.

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

[[0 1 2] [3 4 5]]

In [11]:
```
a3 = np.array([[0,1,2],[3,4,5]])
print("Given Array:\n",a3)
```

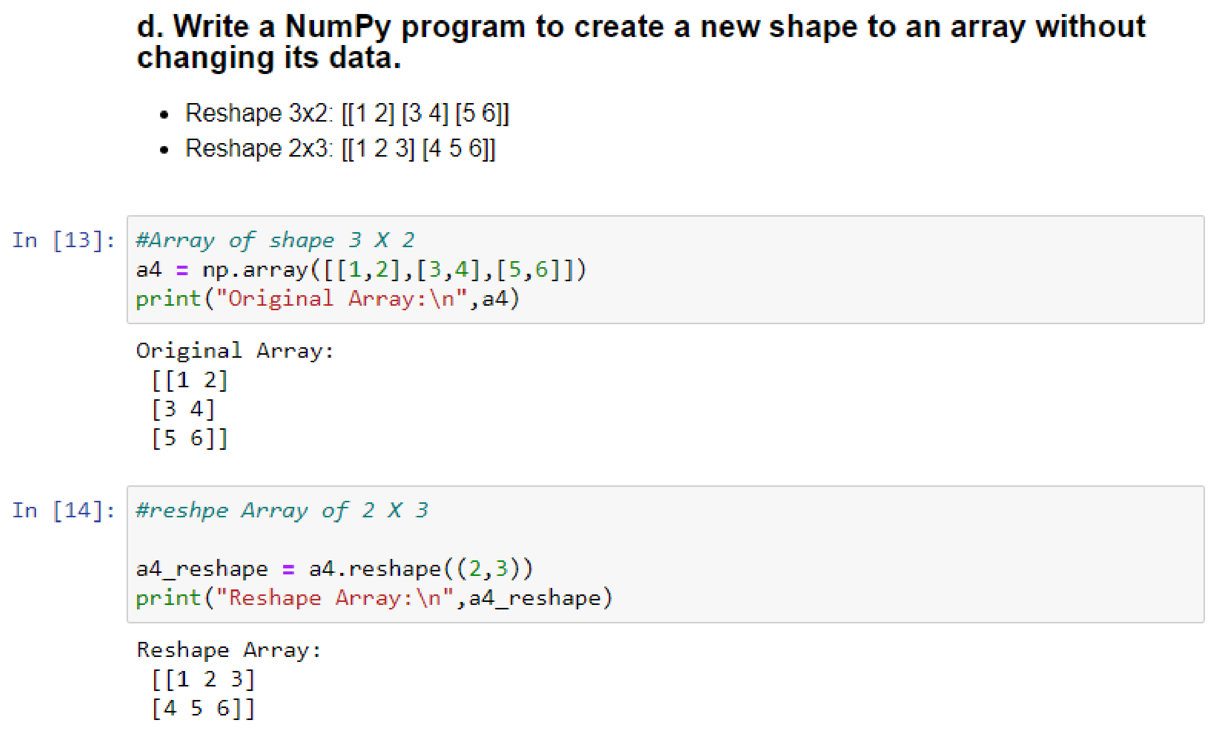
```
Given Array:
 [[0 1 2]
 [3 4 5]]
```

In [12]:
```
diagArray = np.diag(a3)
diagSum = np.sum(diagArray)

print("Sum of the diagonal elements:",diagSum)
```

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

This program utilizes NumPy to calculate a 2x3 array's diagonal elements' sum. The diagonal element total is calculated using np.diag() and then summed using np.sum() after the array has been defined using np.array(). Print() is used to display the total as a result.



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]]

In [13]:
```
#Array of shape 3 X 2
a4 = np.array([[1,2],[3,4],[5,6]])
print("Original Array:\n",a4)
```

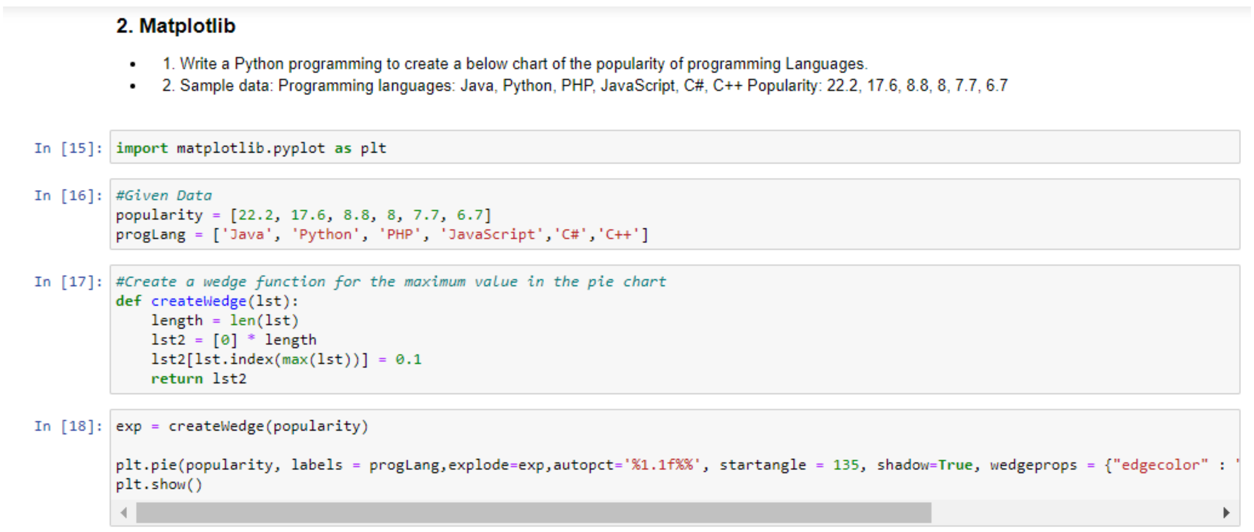
```
Original Array:
 [[1 2]
 [3 4]
 [5 6]]
```

In [14]:
```
#reshpe Array of 2 X 3

a4_reshape = a4.reshape((2,3))
print("Reshape Array:\n",a4_reshape)
```

```
Reshape Array:
 [[1 2 3]
 [4 5 6]]
```

In this program, a 2-dimensional array is transformed into a 2x3 array, using NumPy. The reshape() function is used to reshape the array to the appropriate shapes once the original array is defined using np.array(). The print() function is then used to display the generated arrays.



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

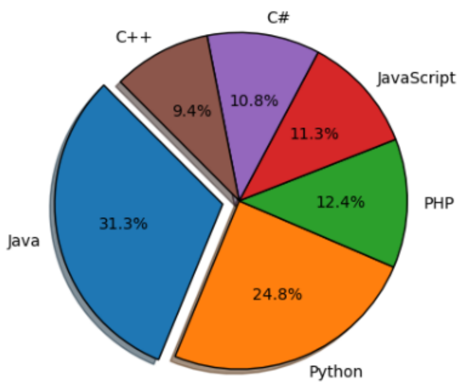
In [15]:
```
import matplotlib.pyplot as plt
```

In [16]:
```
#Given Data
popularity = [22.2, 17.6, 8.8, 8, 7.7, 6.7]
progLang = ['Java', 'Python', 'PHP', 'JavaScript','C#','C++']
```

In [17]:
```
#Create a wedge function for the maximum value in the pie chart
def createWedge(lst):
    length = len(lst)
    lst2 = [0] * length
    lst2[lst.index(max(lst))] = 0.1
    return lst2
```

In [18]:
```
exp = createWedge(popularity)

plt.pie(popularity, labels = progLang,explode=exp,autopct='%1.1f%%', startangle = 135, shadow=True, wedgeprops = {"edgecolor" : '
plt.show()
```



This Python code snippet creates a pie chart depicting the usage of different programming languages by using the matplotlib.pyplot package. The data to plot, the colors to use for each slice of the chart, and the amount to "explode" each slice are all specified in the code. The chart is then made using the pie function, with many options for labeling, formatting, and appearance. Finally, the chart is displayed using the show() function, and the aspect ratio is configured to be equal.