

Summer 2023: CS5710 – Machine Learning

In-Class Programming Assignment-1

Anudeep Allamsetty - 700741171

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

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.

```
+ Code + Text

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:
[ 1  3 10 19 13 13 13 14  7 11 19  6  6 20  3]
Reshaped Array to (3X5) size:
[[ 1  3 10 19 13]
 [13 13 14  7 11]
 [19  6  6 20  3]]
Array shape: (3, 5)
Array shape: (4, 3)
Array type: <class 'numpy.ndarray'>
Array data type: int32
```

In the above code we import the numpy library which provides functions for working with arrays and mathematical problems. In order to generate the random number we use randint function which belongs to random module. Randint function generates the random numbers with the specified range with the specified size. Then we use reshape function to reshape the vector with the specified dimension without modifying the elements in the vector.

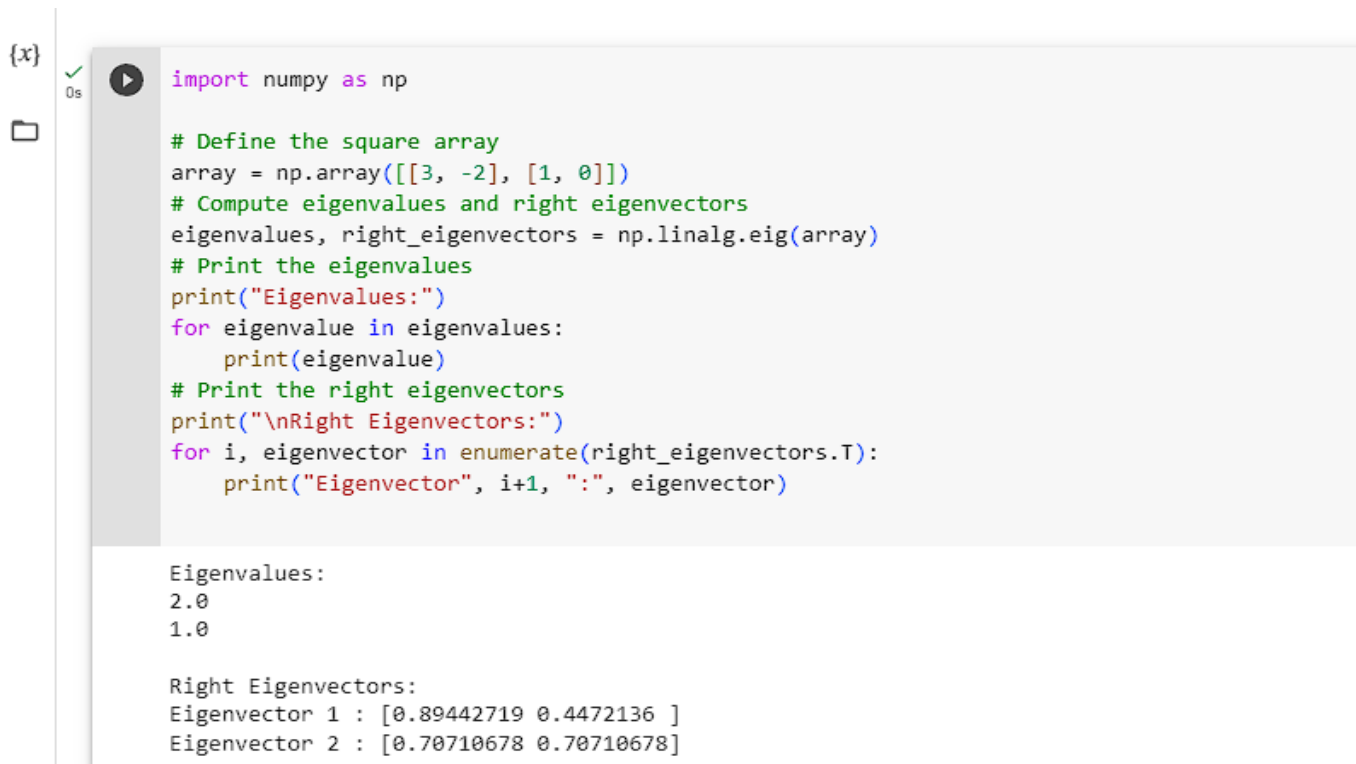
GITHUB ID: allamsettyanudeep@gmail.com

GITHUB LINK: <https://github.com/anudeep4c4/Programming-Asst-1/>

Using the shape attribute, we output the modified array's shape. Since the array in this example has 3 rows and 5 columns, the shape informs us of its dimensions, which in this case would be (3, 5). To get the indices of the highest value in each row of the reshaped array, we utilize the `np.argmax()` function. The function `argmax()` returns the index of the greatest element along a given axis. This tells us where the highest value in each row is located. The maximum values in each row are changed to 0 using the indices we acquired. The corresponding rows are chosen, and their maximum values are set to 0, using the `np.arange(3)` function and the acquired indices. Next, we use `np.zeros()` to construct a new 2-dimensional array called `array_2d`. This function generates an array that contains just zeros. The array will contain 4 rows and 3 columns if its size is specified as 4x3. We print the shape of `array_2d` using the shape attribute. This will give us the dimensions of the array, which is (4, 3) in this case. We also print the type of `array_2d` using the `type()` function. It tells us that the array belongs to the `ndarray` class, which is provided by NumPy. Finally, we print the data type of the array using the `dtype` attribute. In this case, the array has a data type of `int32`, which means it contains 4-byte integers.

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

$\begin{bmatrix} 3 & -2 \\ 1 & 0 \end{bmatrix}$



```
{x}
0s
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]
```

We use the `np.array()` function to define the square array. Assigning the array is $\begin{bmatrix} 3 & -2 \\ 1 & 0 \end{bmatrix}$. To determine the array's eigenvalues and right eigenvectors, we utilize the `np.linalg.eig()` function. The array is the input for the `eig()` function, which outputs right eigenvectors and eigenvalues. Using a loop, we print the eigenvalues. We print each eigenvalue from the array of eigenvalues separately. The resulting 2-dimensional array contains the right eigenvectors. The array's columns each correspond to a separate eigenvector. We employ another loop to print the appropriate eigenvectors. We print each column (or eigenvector) in the array of right eigenvectors separately. The eigenvectors are normalized to 1, with their magnitudes scaled to unit length.

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

```
[[0 1 2]
```

```
[3 4 5]]
```

✓
0s



```
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
```

To calculate the total of the diagonal array members, we utilize the `np.trace()` function. The primary diagonal of an array, which runs from top-left to bottom-right, is referred to as the diagonal element. The diagonal elements of a 2-dimensional array's sum are computed using the `np.trace()` function. In this instance, it calculates the diagonal array's sum of items. In the variable `diagonal_sum`, we keep the total of the diagonal elements. Using `print()`, we display the diagonal elements' total. In this instance, the total is 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]]
```

GITHUB ID: allamsettyanudeep@gmail.com

GITHUB LINK: <https://github.com/anudeep4c4/Programming-Asst-1/>

0s



```
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]]
```

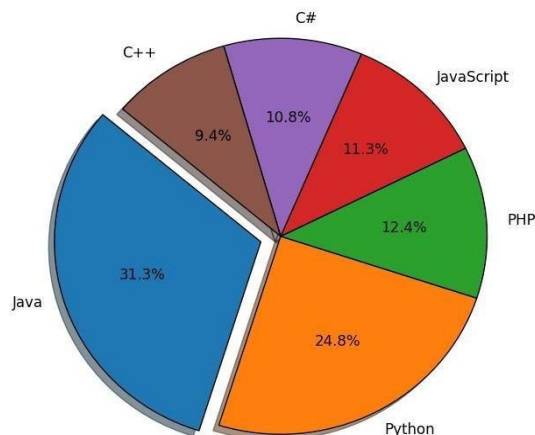
Two reshaped arrays, `reshaped_array_3x2` and `reshaped_array_2x3`, are made using the `reshape()` function. We can change the shape of an array without changing its data by using the `reshape()` function. We use the original array to invoke this function, passing the required shape as an argument. The variables `reshaped_array_3x2` and `reshaped_array_2x3` contain the reshaped arrays, accordingly.

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



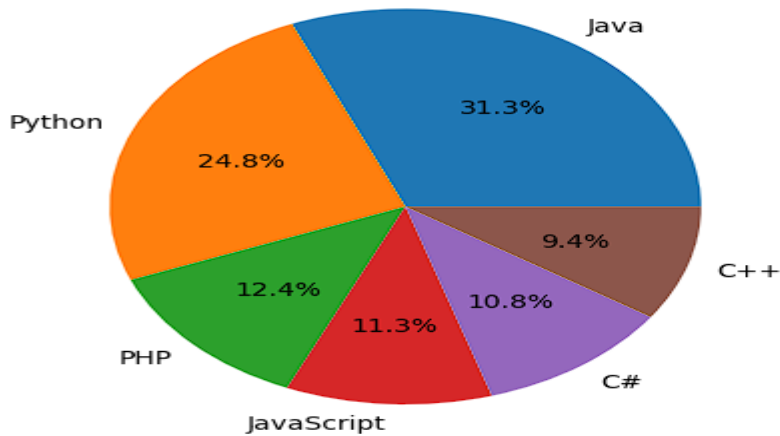
```

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()

```

Popularity of Programming Languages



We begin by importing the pyplot module from the Matplotlib library, which offers tools for making visualizations. Languages and popularity are the two lists that we define. These lists show the programming languages and their corresponding popularity scores. For instance, the popularity of "Java" is 22.2%, "Python" is 17.6%, and so on. To make a pie chart, we employ the `plt.pie()` function. This function accepts the popularity list as input for the chart data and the labels argument for the programming language names. The percentage numbers presented on the chart are formatted using the `autopct` parameter. To indicate one decimal place for the percentage numbers in this instance, we use the notation `"%1.1f%%"`.