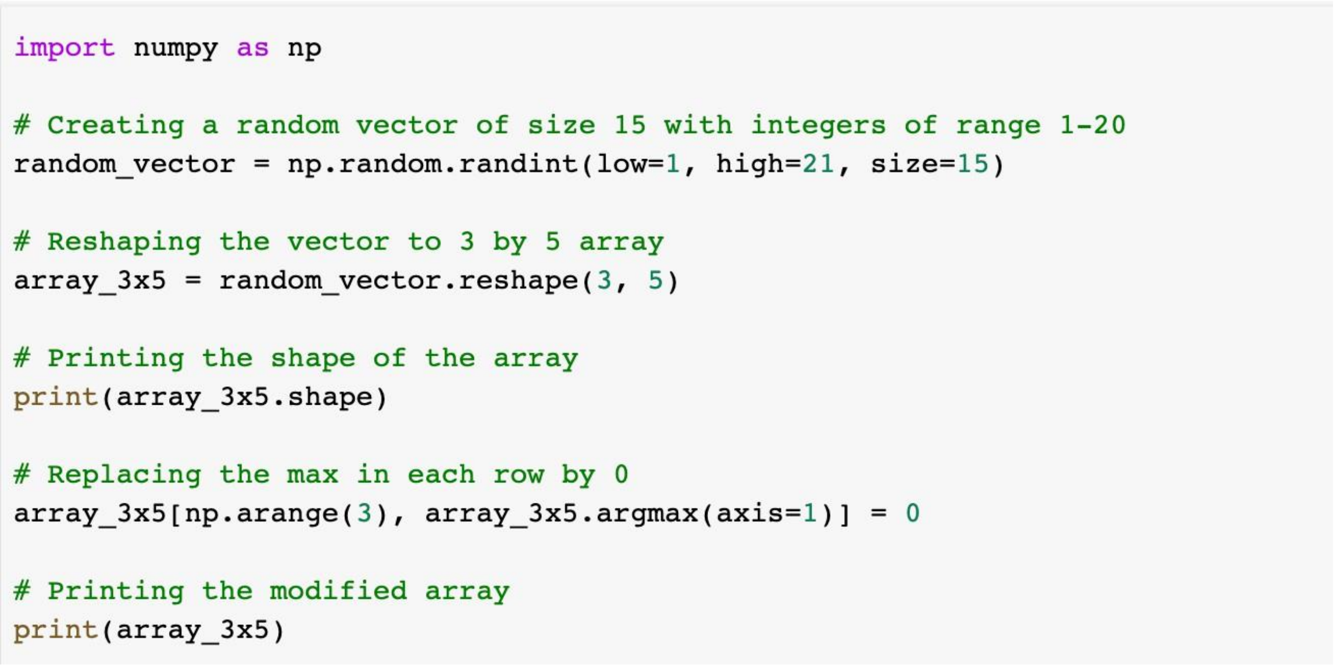
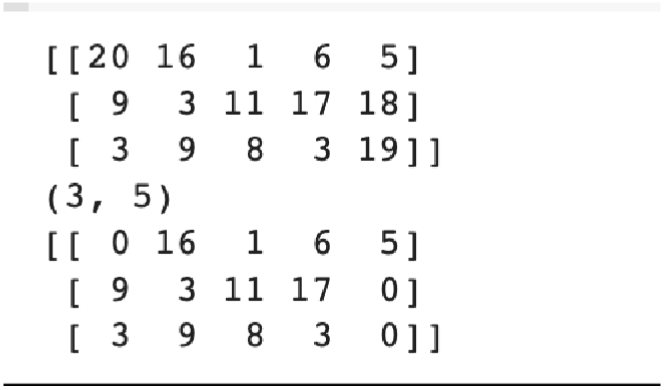
Solution a:



```
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

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

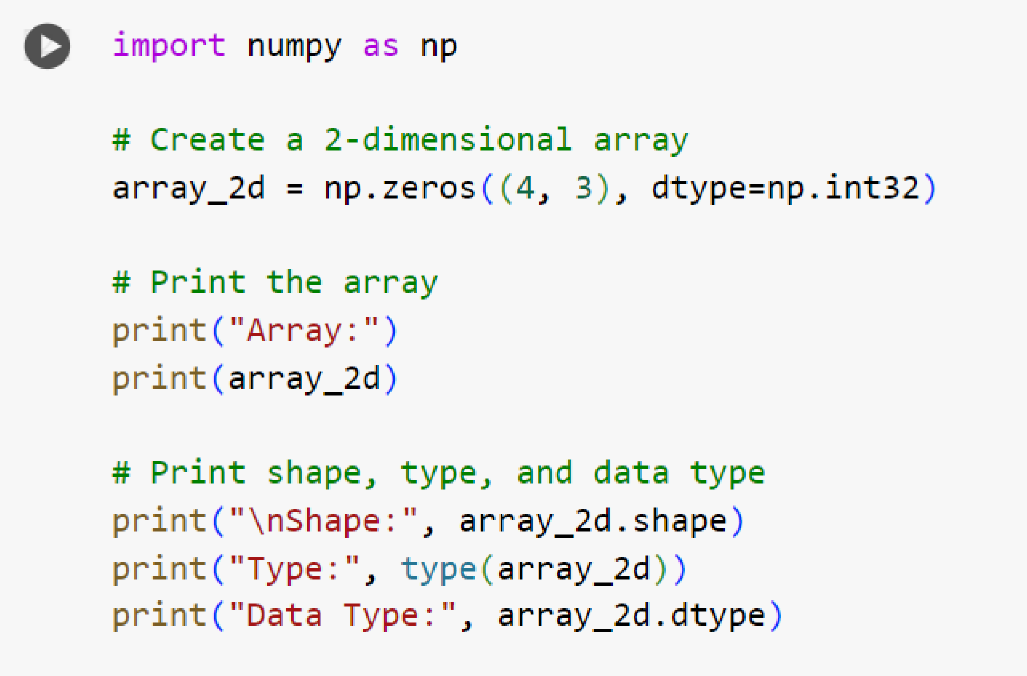
Output:



```
[[20 16  1  6  5]
 [ 9  3 11 17 18]
 [ 3  9  8  3 19]]
(3, 5)
[[ 0 16  1  6  5]
 [ 9  3 11 17  0]
 [ 3  9  8  3  0]]
```

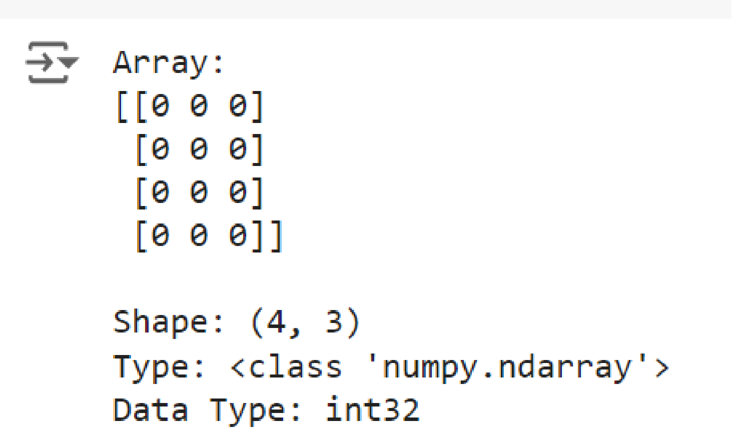
Explanation:

Firstly created a random vector using np.random.randint. Used ‘reshape’ method to reshape the vector to 3 by 5 array. The ‘shape’ attribute is used to know the shape of vector. To determine the index of the largest member in each row of the array, use the argmax method. The maximum element is set to 0 by combining the index of the maximum element for each row with the array of indices for the rows created by the np.arange(3) function.



```python
import numpy as np

# Create a 2-dimensional array
array_2d = np.zeros((4, 3), dtype=np.int32)

# Print the array
print("Array:")
print(array_2d)

# Print shape, type, and data type
print("\nShape:", array_2d.shape)
print("Type:", type(array_2d))
print("Data Type:", array_2d.dtype)
```

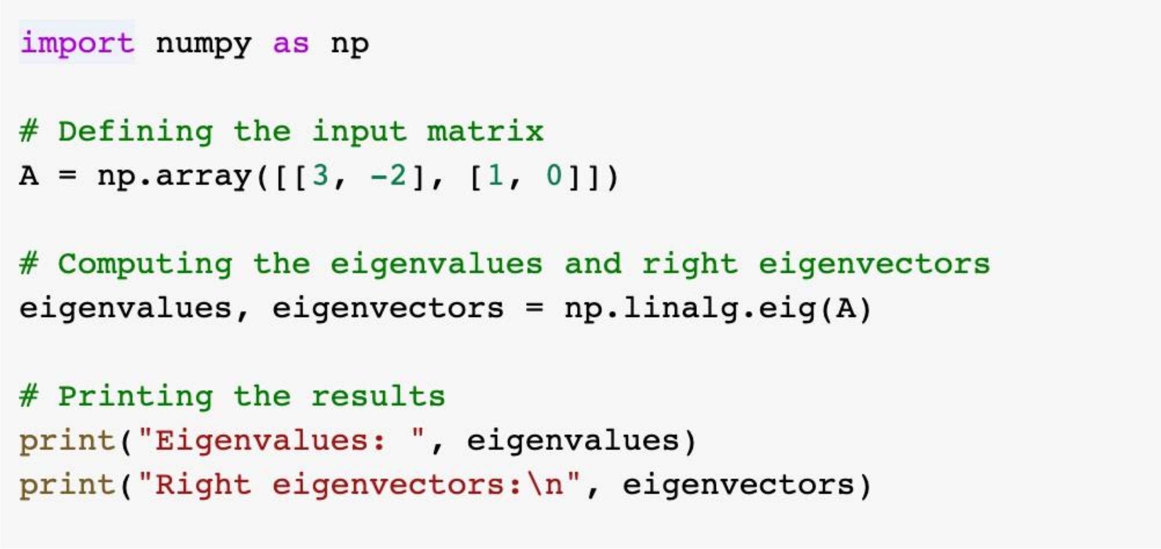
Output:



```
Array:
[[0 0 0]
 [0 0 0]
 [0 0 0]
 [0 0 0]]

Shape: (4, 3)
Type: <class 'numpy.ndarray'>
Data Type: int32
```

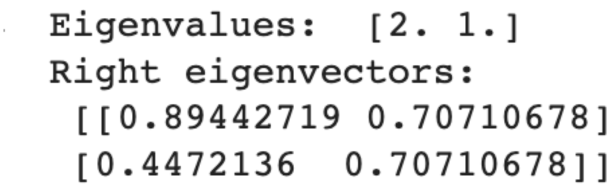
Firstly created a random vector using np.random.randint. It will create a 4x3 mtarix and by using shape, type, and dtype functions we can print the shape, type and data type of the matrix as shown in the output.

Solution b:



```
import numpy as np

# Defining the input matrix
A = np.array([[3, -2], [1, 0]])

# Computing the eigenvalues and right eigenvectors
eigenvalues, eigenvectors = np.linalg.eig(A)

# Printing the results
print("Eigenvalues: ", eigenvalues)
print("Right eigenvectors:\n", eigenvectors)
```

Output:

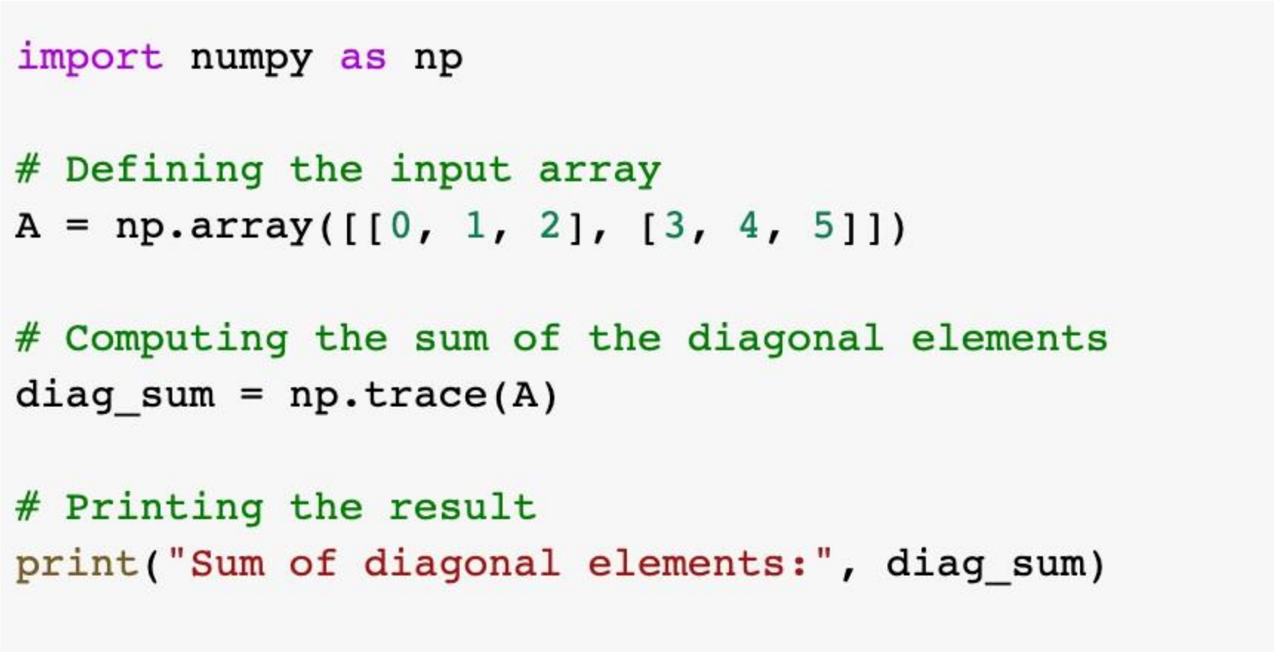


```
Eigenvalues:  [2. 1.]
Right eigenvectors:
 [[0.89442719 0.70710678]
 [0.4472136  0.70710678]]
```

Explanation:

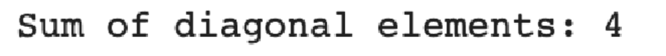
Input matrix is defined. Then used ‘np.linalg.eig’ function used to compute the eigenvalues and right eigenvectors. Finally printed the values by calling them.

Solution c:



```python
import numpy as np

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

# Computing the sum of the diagonal elements
diag_sum = np.trace(A)

# Printing the result
print("Sum of diagonal elements:", diag_sum)
```

Output:

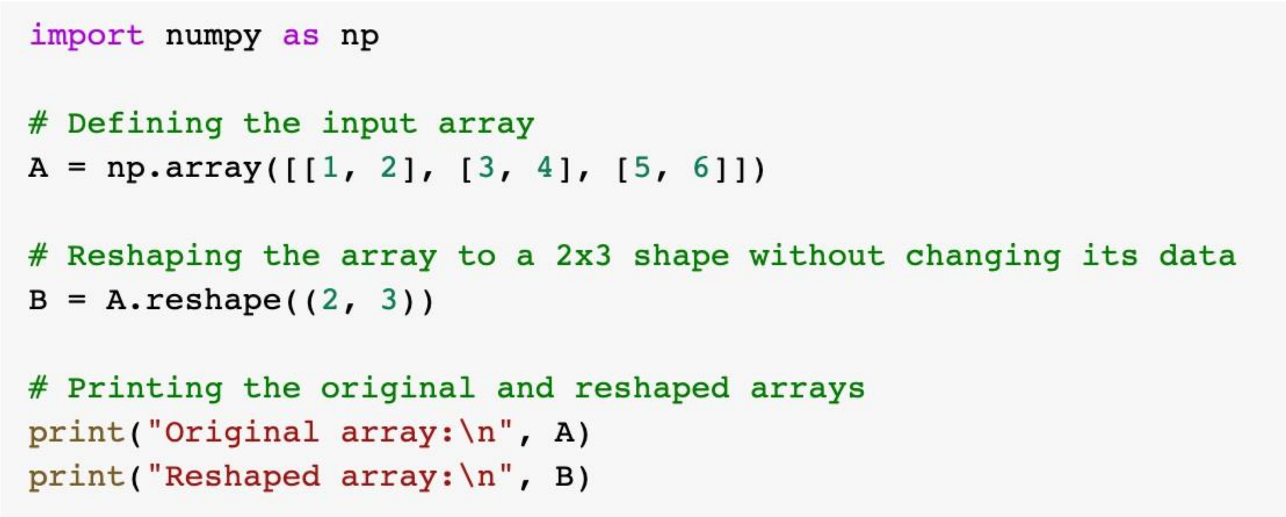


```
Sum of diagonal elements: 4
```

Explanation:

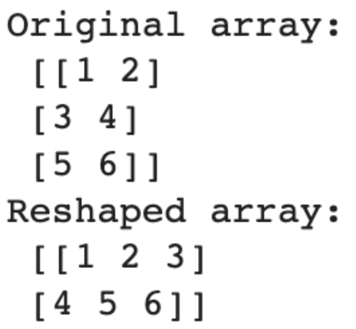
Defined an input array. The function ‘np.trace’ is used to compute the sum of diagonal elements and is being stored in diag_sum.

Solution d:



```
import numpy as np

# Defining the input array
A = np.array([[1, 2], [3, 4], [5, 6]])

# Reshaping the array to a 2x3 shape without changing its data
B = A.reshape((2, 3))

# Printing the original and reshaped arrays
print("Original array:\n", A)
print("Reshaped array:\n", B)
```

Output:



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

Explanation:

Defined an input array. Used ‘reshape’ function to reshape the array without changing the data then printed the reshaped arrays

Video Link:
https://drive.google.com/file/d/1oa5EkI6LHrR1YE7uLqMyfSvZzgzFWlVi/view?usp=drive_link
GitHub Link:
https://github.com/shruthikatkam26/MachinelearningICP3.git