Assignment-8 | Numpy

Akash Duttachowdhury

21052386

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

1. Create a null vector of size 10 but the fifth value is 1

In [ ]:
```
# Akash Duttachowdhury | 21052386
arr = np.arange(10)
arr[4] = 1
print(arr)
```

```
[0 1 2 3 1 5 6 7 8 9]
```

2. Create a vector with values ranging from 10 to 49

In [ ]:
```
# Akash Duttachowdhury | 21052386
arr2 = np.arange(10,49,2)
print(arr2)
```

```
[10 12 14 16 18 20 22 24 26 28 30 32 34 36 38 40 42 44 46 48]
```

3. Reverse a vector (first element becomes last)

In [ ]:
```
# Akash Duttachowdhury | 21052386
arr = np.arange(20)
rev = arr[::-1]
arr = rev
print(arr)
```

```
[19 18 17 16 15 14 13 12 11 10  9  8  7  6  5  4  3  2  1  0]
```

4. Create a 3x3 matrix with values ranging from 0 to 8. (hint: reshape)

In [ ]:
```
# Akash Duttachowdhury | 21052386
arr = np.arange(9).reshape(3, 3)
print(arr)
```

```
[[0 1 2]
 [3 4 5]
 [6 7 8]]
```

5. Find indices of non-zero elements from [1,2,0,0,4,0] (hint: np.nonzero)

In [ ]:
```
# Akash Duttachowdhury | 21052386
given = [1, 2, 0, 0, 4, 0]
arr = np.array(given)
print(arr.nonzero())
```

```
(array([0, 1, 4]),)
```

6. Create a 3x3x3 array with random values (hint: np.random.random)

In [ ]:
```
# Akash Duttachowdhury | 21052386
arr = np.random.rand(27).reshape(3, 3, 3)
print(arr)
```

```
[[[0.00115433 0.75412212 0.77752014]
  [0.96128665 0.82868792 0.64411909]
  [0.39842924 0.6867075  0.20686412]]

 [[0.44240242 0.11640725 0.82415998]
  [0.25677848 0.13017644 0.48627273]
  [0.83058356 0.81136106 0.74832649]]

 [[0.27182788 0.24208836 0.31891555]
  [0.53766339 0.89354396 0.97865404]
  [0.42673964 0.85415905 0.26070612]]]
```

7. Create a 10x10 array with random values and find the minimum and maximum values (hint: min, max)

In [ ]:
```
# Akash Duttachowdhury | 21052386
arr = np.random.randint(0,100, size = (10, 10))
print(arr)
print(f"Max: {arr.max()}\nMin: {arr.min()}")
```

```
[[36 44 61 62 47 23 90 93 76 96]
 [41 20 35  8 86 50  4 19 46 43]
 [46 26 38 11 83  2  6 61  3 18]
 [65 43 20 59 45 99 16 47 67 25]
 [19 10 21 99 13 50 98 90 10 83]
 [15 21 49 33 65 17 93 55 60 36]
 [69  0 19 19 68  5 26  6 15 17]
 [26 55  5 63 12 22 95 16 70 90]
 [11 24 41 34 26 30 65 70 96 56]
 [94 12 41 82 26 70 89 71 97 84]]
Max: 99
Min: 0
```

8. Create a random vector of size 30 and find the mean value (hint: mean)

In [ ]:
```
# Akash Duttachowdhury | 21052386
arr = np.random.rand(30)
print(f"Vector = {arr}")
avg = np.mean(arr)
print(f"Mean value is {avg}")
```

```
Vector = [0.21143845 0.97922954 0.14658811 0.45883546 0.19861757 0.2685863
5
 0.1341218  0.78679763 0.64878047 0.65678762 0.39494587 0.90194302
 0.66260697 0.93515837 0.71224246 0.70223679 0.64871188 0.50400108
 0.88625851 0.50010124 0.29128623 0.00400353 0.33029933 0.10147764
 0.8655343  0.80562755 0.13516984 0.6723518  0.04741446 0.46873708]
Mean value is 0.5019963645928762
```

9. Create a 2d array with 1 on the border and 0 inside (hint: array[1:-1, 1:-1])

In [ ]:
```
# Akash Duttachowdhury | 21052386
r = int(input("Enter no. of rows: "))
c = int(input("Enter no. of columns: "))
arr = np.ones([r, c])
arr[1:-1,1:-1] = np.zeros([r-2, c-2])
print(arr)
```

```
Enter no. of rows: 5
Enter no. of columns: 5
[[1. 1. 1. 1. 1.]
 [1. 0. 0. 0. 1.]
 [1. 0. 0. 0. 1.]
 [1. 0. 0. 0. 1.]
 [1. 1. 1. 1. 1.]]
```

10. Normalize a 5x5 random matrix (hint: (x -mean)/std)

In [ ]:
```
# Akash Duttachowdhury | 21052386
matrix = np.random.rand(5, 5)
mean = np.mean(matrix)
std = np.std(matrix)

normalized_matrix = (matrix - mean) / std

print(normalized_matrix)
```

```
[[ 1.01716834 -0.73054755 -0.80060047 -0.75066574  1.17206494]
 [ 1.35694589 -1.59242764  0.12536523  0.32930149  1.03350286]
 [-0.08692671 -0.63955925 -1.15432456  0.1821099   1.30511352]
 [ 1.71611365 -0.7355395  -1.33203673 -1.22631731 -0.5345592 ]
 [ 0.09004639  1.45577078 -0.51568995  1.23147983 -0.91578822]]
```

11. Multiply a 5x3 matrix by a 3x2 matrix (real matrix product)

In [ ]:
```
# Akash Duttachowdhury | 21052386
```

```
m1 = np.random.rand(5, 3)
m2 = np.random.rand(3, 2)

product = np.dot(m1, m2)
print(f"{product}")
```

```
[[1.10845602 0.68000371]
 [0.63604916 0.69253053]
 [1.17253078 0.63109773]
 [1.45863633 0.94602637]
 [1.18539028 0.70403434]]
```

12. Given a 1D array, negate all elements which are between 3 and 8, in place.

In [ ]:
```
# Akash Duttachowdhury | 21052386
print("Enter 1D array elements: ")
lst = list(map(int, input().split()))
arr = np.array(lst)
mask = np.logical_and(arr >= 3, arr <=8)
arr[mask] *= -1
print(arr)
```

```
Enter 1D array elements:
1 2 3 4 5 6 7 8 9 10 11 12 13
[ 1  2 -3 -4 -5 -6 -7 -8  9 10 11 12 13]
```

13. Find the eigenvalues and eigenvectors of a square matrix.

In [ ]:
```
# Akash Duttachowdhury | 21052386
arr = np.random.randint(0,100, size = (3,3))
print(arr)
eigenvalues, eigenvectors = np.linalg.eig(arr)
print(f"Eigenvalues = {eigenvalues}\nEigenvectors = {eigenvectors}")
```

```
[[38 70 91]
 [60 90 24]
 [ 5 20 47]]
Eigenvalues = [146.69055084   6.93773637  21.37171279]
Eigenvectors = [[ 0.63573815  0.76269587  0.61108365]
 [ 0.75005406 -0.61165816 -0.67744319]
 [ 0.18236204  0.21016495  0.40944779]]
```

14. Find the inverse of a square matrix. (hint: np.linalg.inv)

In [ ]:
```
# Akash Duttachowdhury | 21052386
arr = np.random.randint(0,100, size = (3,3))
inverse_matrix = np.linalg.inv(arr)
print(f"The matrix is\n{arr}\nThe inverse of the matrix is \n{inverse_mat
```

```
The matrix is
[[21 78 12]
 [ 9 78 48]
 [ 6 43 10]]
The inverse of the matrix is
[[ 0.10278578  0.02113353 -0.22478386]
 [-0.01585014 -0.01104707  0.07204611]
 [ 0.00648415  0.03482229 -0.07492795]].
```