

Lab report 01

**Numpy**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**CSE-422L-Data Analytics Lab**

**Department of Computer System Engineering**

**University of Engineering and Technology Peshawar**

Submitted to: **Dr. Abeer Irfan**

Submitted by: **Muhammad Saad**

Reg ID: **21PWCSE1997**

**DCSE, Batch 23, Section “B”**

**Lab 1: Tasks (NumPy)**

**Task 1:**

**Objective:** Learn the basics of NumPy arrays and simple operations.

1. **Create a 1D NumPy array with numbers from 1 to 20.**

*# 1. Create a 1D NumPy array with numbers from 1 to 20.*

**import** **numpy** **as** **np**

arr\_1d = np.arange(1, 21)

print(arr\_1d )

Array([ 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 19, 20])

1. **Create a 2D array of shapes (3, 5) using random integers between 10 and 50.**

*# 2. Create a 2D array of shape (3, 5) using random integers between 10 and 50.*

arr\_2d = np.random.randint(10, 50, size=(3, 5))

arr\_2d

array([[42, 28, 24, 45, 26],

[16, 40, 32, 37, 14],

[19, 12, 41, 11, 14]], dtype=int32)

1. **Extract the first and third rows of the 2D array.**

*# 3. Extract the first and third rows of the 2D array.*

extracted\_rows = arr\_2d[[0, 2], :]

extracted\_rows

array([[42, 28, 24, 45, 26],

[19, 12, 41, 11, 14]], dtype=int32)

1. **Calculate the sum, mean, minimum, and maximum of the entire 2D array.**

*# 4. Calculate the sum, mean, minimum, and maximum of the entire 2D array*

total\_sum = arr\_2d.sum()

mean\_value = arr\_2d.mean()

min\_value = arr\_2d.min()

max\_value = arr\_2d.max()

total\_sum, mean\_value, min\_value, max\_valued.min()max\_value = arr\_2d.max()total\_sum, mean\_value, min\_value, max\_value x

*#summeanminmax*

(np.int64(401), np.float64(26.733333333333334), np.int32(11), np.int32(45))

1. **Reshape the 1D array into a 4x5 matrix.**

*# 5. Reshape the 1D array into a 4x5 matrix.*

reshaped\_1d = arr\_1d.reshape(4, 5)

reshaped\_1d

Array([[ 1, 2, 3, 4, 5], [ 6, 7, 8, 9, 10], [11, 12, 13, 14, 15], [16, 17, 18, 19, 20]])

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Task 2:**

**Objective:** Practice element-wise operations and array manipulations.

1. **Create two random 3x3 matrices using [Hint: numpy.random.rand().]**

*# 1. Create two random 3x3 matrices.*

matrix\_a = np.random.rand(3, 3)

matrix\_b = np.random.rand(3, 3)

matrix\_a, matrix\_b

(array([[0.78451914, 0.58146057, 0.89148833], [0.76029478, 0.36375358, 0.21922538], [0.98803456, 0.82342979, 0.5340467 ]]),

array([[0.12180932, 0.71946139, 0.9236956 ], [0.50533536, 0.03961328, 0.4156349 ],

[0.63873981, 0.87737221, 0.32935445]]))

1. **Perform the following element-wise operations on these matrices:**
   1. **Addition**
   2. **Subtraction**
   3. **Multiplication**
   4. **Division**

*# 2. Perform element-wise addition, subtraction, multiplication, and division.*

add\_matrices = matrix\_a + matrix\_b

subtract\_matrices = matrix\_a - matrix\_b

multiply\_matrices = matrix\_a \* matrix\_b

divide\_matrices = matrix\_a / matrix\_b

add\_matrices, subtract\_matrices, multiply\_matrices, divide\_matrices

(array([[0.90632845, 1.30092196, 1.81518393], [1.26563014, 0.40336686, 0.63486028],

[1.62677437, 1.700802 , 0.86340115]]),---> Addition

array([[ 0.66270982, -0.13800082, -0.03220727], [ 0.25495943, 0.32414029, -0.19640952],

[ 0.34929475, -0.05394243, 0.20469225]]), --->Subtraction

array([[0.09556174, 0.41833843, 0.82346385], [0.38420384, 0.01440947, 0.09111772],

[0.63109701, 0.72245441, 0.17589066]]),--->Multiplication

array([[6.4405512 , 0.80818871, 0.96513217], [1.50453511, 9.1826162 , 0.527447 ],

[1.54684982, 0.93851819, 1.62149531]]))--->Division

1. **Multiply one of the matrices by a scalar (e.g., multiply by 5).**

*# 3. Multiply one of the matrices by a scalar (e.g., multiply by 5).*

matrix\_a\_scaled = matrix\_a \* 5

matrix\_a\_scaledx

array([[3.92259568, 2.90730285, 4.45744167],

[3.80147392, 1.81876788, 1.0961269 ],

[4.9401728 , 4.11714893, 2.67023349]])

1. **Transpose of one of the matrices**.

*# 4. Transpose one of the matrices.*

matrix\_a\_transpose = matrix\_a.T

matrix\_a\_transpose

array([[0.78451914, 0.76029478, 0.98803456],

[0.58146057, 0.36375358, 0.82342979],

[0.89148833, 0.21922538, 0.5340467 ]])

1. **Calculate the dot product of the two matrices using np.dot()**

*# 5. Calculate the dot product of the two matrices using np.dot().*

dot\_product = np.dot(matrix\_a, matrix\_b)

dot\_product

array([[0.95882341, 1.36963188, 1.25994783],

[0.41645651, 0.75375447, 0.92567248],

[0.87757689, 1.21202921, 1.43077999]])

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Task 3:**

**Objective:** Apply filtering and Boolean indexing techniques to extract and modify data.

1. **Create a NumPy array of 15 random integers between 1 and 100.**

*# 1. Create a NumPy array of 15 random integers between 1 and 100.*

arr\_random = np.random.randint(1, 100, 15)

arr\_random

array([92, 58, 73, 2, 22, 63, 4, 16, 23, 81, 23, 44, 40, 13, 19],

dtype=int32)

1. **Find all the values greater than 50 and return their indices.**

*# 2. Find all values greater than 50 and return their indices.*

indices\_greater\_than\_50 = np.where(arr\_random > 50)[0]

indices\_greater\_than\_50

array([0, 1, 2, 5, 9])

1. **Create a mask to filter out the elements that are divisible by 3.**

*# 3. Create a mask to filter out elements that are divisible by 3.*

divisible\_by\_3\_mask = arr\_random % 3 == 0

filtered\_elements = arr\_random[divisible\_by\_3\_mask]

filtered\_elements

array([63, 81], dtype=int32)

1. **Replace all values less than 20 in the array with -1.**

*# 4. Replace all values less than 20 in the array with -1.*

arr\_random[arr\_random < 20] = -1

arr\_random

array([92, 58, 73, -1, 22, 63, -1, -1, 23, 81, 23, 44, 40, -1, -1],

dtype=int32)

1. **Find the mean of all the values greater than 50.**

*# 5. Find the mean of all values greater than 50.*

mean\_greater\_than\_50 = arr\_random[arr\_random > 50].mean()

mean\_greater\_than\_50

np.float64(73.4)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Task 4:**

**Objective:** Explore NumPy's linear algebra capabilities and advanced operations.

1. **Create a random 4x4 matrix and compute its:**
   * **Determinant [Hint: np.linalg.det()]**
   * **Inverse [Hint: np.linalg.inv()]**
   * **Eigenvalues and eigenvectors [Hint: np.linalg.eig()]**

*# 1. Create a random 4x4 matrix.*

matrix\_4x4 = np.random.rand(4, 4)

matrix\_4x4

array([[0.19058793, 0.90380083, 0.72836261, 0.70856348],

[0.68473837, 0.38998782, 0.09900778, 0.3058579 ],

[0.27378436, 0.67815068, 0.42058899, 0.16832552],

[0.46933628, 0.98648037, 0.66590806, 0.47773844]])

*# 2. Compute the determinant.*

determinant = np.linalg.det(matrix\_4x4)

determinant

np.float64(-0.005685166778199702)

*# 3. Compute the inverse.*

inverse\_matrix = np.linalg.inv(matrix\_4x4)

inverse\_matrix

array([[ -5.3175312 , -1.53757993, -10.12048458, 12.43698352],

[ 8.98696047, 5.53737965, 22.2768616 , -24.72325108],

[-12.87533407, -8.94700127, -28.32234049, 34.80330147],

[ 4.61348477, 2.54743165, 3.42090724, -7.58558355]])

*# 4. Compute the eigenvalues and eigenvectors.*

eigenvalues, eigenvectors = np.linalg.eig(matrix\_4x4)

eigenvalues, eigenvectors

(array([ 1.95880984+0.j , -0.22395945+0.20143359j,

-0.22395945-0.20143359j, -0.03198778+0.j ]),----->EigenValues

array([[ 0.59100767+0.j , 0.42651213-0.25219366j,

0.42651213+0.25219366j, -0.30293475+0.j ],

[ 0.39863786+0.j , -0.68296465+0.j ,

-0.68296465-0.j , 0.59107676+0.j ],

[ 0.34758926+0.j , 0.39369039+0.2186626j ,

0.39369039-0.2186626j , -0.74051838+0.j ],----->EigenVecters

[ 0.60908087+0.j , 0.2886198 +0.04402488j,

0.2886198 -0.04402488j, 0.10242717+0.j ]]))

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_