MACHINE LEARNING LAB EXERCISE

## DATE: 30-01-2025

# Exercise on numpy and Pandas(without using functions)

Code:

import numpy as np

import pandas as pd

**1. Create a 1D and 2D using Numpy , print its datatype, shape,dimension.\**

**Ans:**

print("question 1")

print("Krithika Ravishankar - 2022503035")

print("\nintegers")

arr1D = np.array([10, 20, 30, 40, 50, 60])

print("1D Array:", arr1D)

print("Datatype of 1D Array:", arr1D.dtype)

print("Shape of 1D Array:", arr1D.shape)

print("Dimension of 1D Array:", arr1D.ndim)

print("\n")

arr2D = np.array([[10, 20, 30], [40, 50, 60]])

print("2D Array:")

print( arr2D)

print("Datatype of 2D Array:", arr2D.dtype)

print("Shape of 2D Array:", arr2D.shape)

print("Dimension of 2D Array:", arr2D.ndim)

print("\nfloats")

float1D = np.array([10.2, 20.3, 30.4, 40.5, 50.6, 60.7])

print("1D Array:", float1D)

print("Datatype of 1D Array:", float1D.dtype)

print("Shape of 1D Array:", float1D.shape)

print("Dimension of 1D Array:",float1D.ndim)

print("\n")

float2D = np.array([[10.12, 20.13, 30.14], [40.15, 50.16, 60.17]])

print("2D Array:")

print( float2D)

print("Datatype of 2D Array:", float2D.dtype)

print("Shape of 2D Array:", float2D.shape)

print("Dimension of 2D Array:",float2D.ndim)

print("\ncomplexnumbers")

complex1D = np.array([15 + 8j, 1 + 8j, 25 + 9j, 9 + 24j ])

print("1D Array:", complex1D)

print("Datatype of 1D Array:", complex1D.dtype)

print("Shape of 1D Array:", complex1D.shape)

print("Dimension of 1D Array:",complex1D.ndim)

print("\n")

complex2D = np.array([[15 + 8j, 1 + 8j], [25 + 9j, 9 + 24j]])

print("2D Array:")

print( complex2D)

print("Datatype of 2D Array:",complex2D.dtype)

print("Shape of 2D Array:", complex2D.shape)

print("Dimension of 2D Array:",complex2D.ndim)

print("\nboolean")

bool1D = np.array([True, False, False, True ])

print("1D Array:", bool1D)

print("Datatype of 1D Array:",bool1D.dtype)

print("Shape of 1D Array:", bool1D.shape)

print("Dimension of 1D Array:",bool1D.ndim)

print("\n")

bool2D = np.array([[True, False], [ False, True]])

print("2D Array:")

print( bool2D)

print("Datatype of 2D Array:",bool2D.dtype)

print("Shape of 2D Array:", bool2D.shape)

print("Dimension of 2D Array:",bool2D.ndim)

print("\nstring")

str1D = np.array(['krithika','krithikaravishankar','krithikar','kritz'])

print("1D Array:", str1D)

print("Datatype of 1D Array:",str1D.dtype)

print("Shape of 1D Array:", str1D.shape)

print("Dimension of 1D Array:",str1D.ndim)

print("\n")

str2D = np.array([['krithika', 'krithikaravishankar', 'krithikar', 'kritz'],

['machine', 'learning', 'lab', 'exercise']])

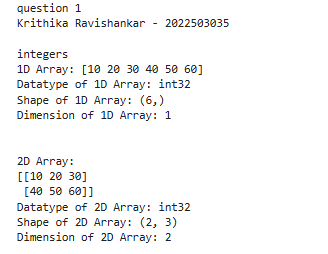
print("2D Array:")

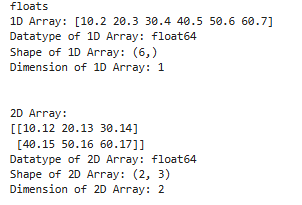
print(str2D)

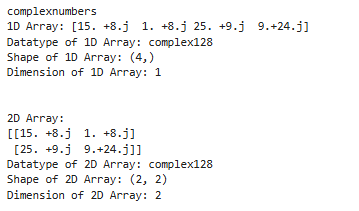
print("Datatype of 2D Array:", str2D.dtype)

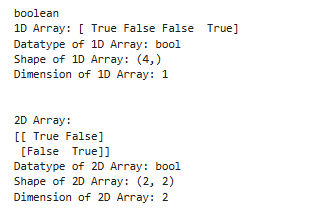
print("Shape of 2D Array:", str2D.shape)

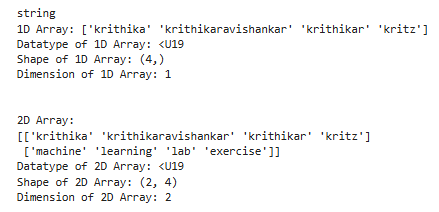
print("Dimension of 2D Array:", str2D.ndim)











**2.Create a 1D and 2D with**

**i) random initialization**

**ii) Default initialization**

**iii) Print Diagonal values**

**vi) Print First Element**

**v) Print Last ELement**

**vi) Print 2nd row and 2nd column Element**

**vii) Print No. of Elements**

ans: print("question 2")

print("Krithika Ravishankar - 2022503035")

print("randomly initialising: \n")

randomarr\_1D = np.random.randint(5,60, 8)

randomarr\_2D = np.random.randint(5,60, (3,3))

#here, if i just give it as random.rand, the default value is taken as float

print("\nrandom 1D array is: \n")

print(randomarr\_1D)

print("\nrandom 2D array is: \n")

print(randomarr\_2D)

print("\ndefault initialising: \n")

zeros\_1D = np.zeros(5)

zeros\_2D = np.zeros((3, 3),dtype='int')

print("\nzeros 1D array is: \n")

print(zeros\_1D)

print("\nzeros 2D array is: \n")

print(zeros\_2D)

print("\ndiagonal values of 2d array\n")

print(np.diagonal(randomarr\_2D))

print("\n first element of the array:\n")

print(randomarr\_2D[0, 0])

print("\n last element of the array:\n")

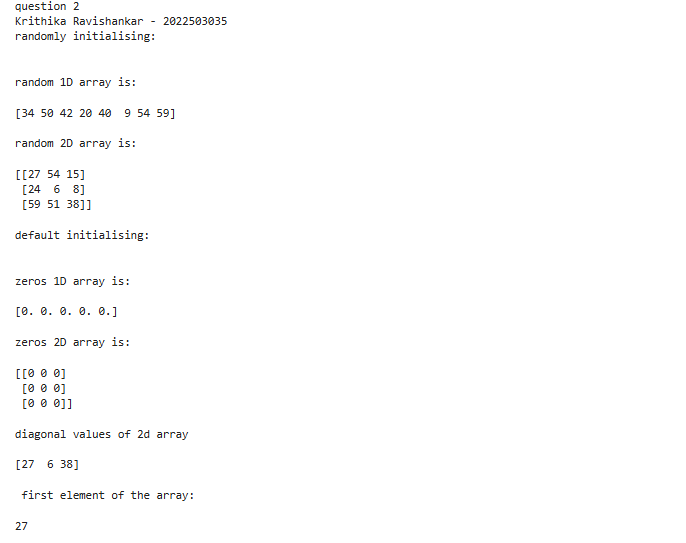
print(randomarr\_2D[-1, -1])

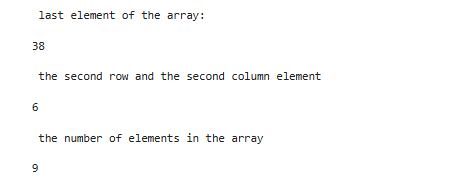
print("\n the second row and the second column element\n")

print(randomarr\_2D[1,1])

print("\n the number of elements in the array\n")

print(randomarr\_2D.size)





**3.Find the Determinant, Addition , Multiplication , Subraction , Division , Transpose, Cross Product , Dot product of a matrix**

Ans: print("question 3")

print("Krithika Ravishankar - 2022503035")

matrix1 = np.array([[15, 22], [3, 16]])

matrix2 = np.array([[1, 9], [12, 25]])

det1 = np.linalg.det(matrix1)

print("\nDeterminant of matrix 1:", det1)

det2 = np.linalg.det(matrix2)

print("\nDeterminant of matrix 2:", det2)

addition = matrix1 + matrix2

print("\nAddition:\n", addition)

multiplication =matrix1 \* matrix2

print("\n multiplication:\n", multiplication)

subtraction = matrix1- matrix2

print("Subtraction :\n", subtraction)

division =np.divide(matrix1, matrix2, where = matrix2!=0)

print("division:\n", np.round(division,2))

transpose1 = matrix1.T

print("Transpose of matrix 1:\n", transpose1)

transpose2 = matrix2.T

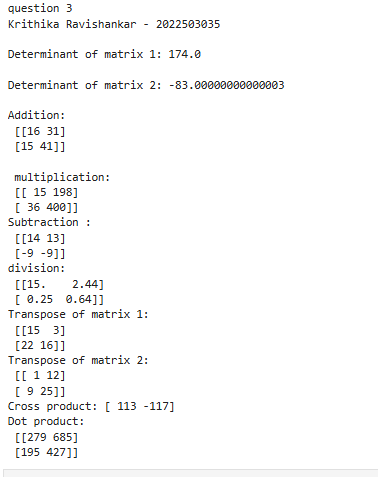
print("Transpose of matrix 2:\n", transpose2)

cross\_product = np.cross(matrix1, matrix2)

print("Cross product:", cross\_product)

dot\_product = np.dot(matrix1, matrix2)

print("Dot product:\n", dot\_product)



**4. Print range values in 1D and 2D.**

Ans: print("question 4")

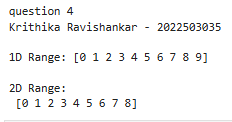
print("Krithika Ravishankar - 2022503035")

range\_1D = np.arange(0, 10)

print("\n1D Range:", range\_1D)

range\_2D = np.arange(0, 9)

print("\n2D Range:\n", range\_2D)

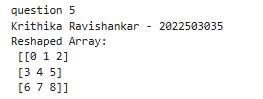


**5. Reshape an array.**

Ans: print("question 5")

print("Krithika Ravishankar - 2022503035")

print("Reshaped Array:\n", range\_2D.reshape(3, 3))



**6. Find the min, max sum, mean, median mode and SD**

Ans: import statistics as stats

print("question 6")

print("Krithika Ravishankar - 2022503035")

data = ([1,12,23,34,45,56,67,78,89,90])

print(" Minimum: " ,np.min(data))

print(" Maximum: " ,np.max(data))

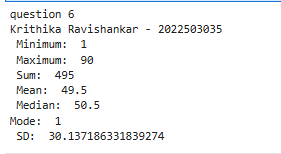
print(" Sum: " ,np.sum(data))

print(" Mean: " ,np.mean(data))

print(" Median: ", np.median(data))

print( "Mode: ", stats.mode(data))

print(" SD: ", np.std(data))



**7. Create a boolean mask for values greater than 5.**

**Use the mask to filter out values less than or equal to 5.**

**Count the number of values greater than 5.**

**A**ns: print("question 7")

print("Krithika Ravishankar - 2022503035")

bool\_mask= np.array([2, 4, 6, 8, 10])

mask = bool\_mask > 5

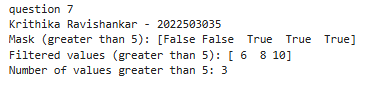
print("Mask (greater than 5):", mask)

filtered\_values = bool\_mask[mask]

print("Filtered values (greater than 5):", filtered\_values)

count\_greater\_than\_5 = np.sum(mask)

print("Number of values greater than 5:", count\_greater\_than\_5)



**8.Fancy Indexing**

**Use fancy indexing to access elements at specific indices.**

**Create a new array by selecting every other element.**

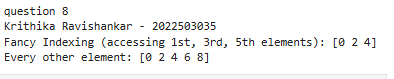
Ans: print("question 8")

print("Krithika Ravishankar - 2022503035")

print("Fancy Indexing (accessing 1st, 3rd, 5th elements):", range\_1D[[0, 2, 4]])

new\_arr = range\_1D[::2]

print("Every other element:", new\_arr)



**9. structured array:**

**Create a structured array to store information about students (name, age, and grade).**

**Create a few sample students.**

**Access and modify specific fields of the structured array.**

Ans: print("question 9")

print("Krithika Ravishankar - 2022503035")

dtype = [('Name', 'U10'), ('Age', 'i4'), ('Grade', 'f4')]

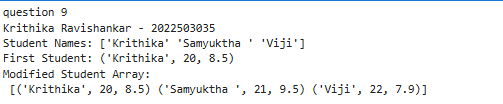
students = np.array([('Krithika', 20, 8.5), ('Samyuktha ', 21, 9.3), ('Viji', 22, 7.9)], dtype=dtype)

print("Student Names:", students['Name'])

print("First Student:", students[0])

students[1]['Grade'] = 9.5

print("Modified Student Array:\n", students)



**10.Linear Regression**

**Study Hours (x₁) Practice Tests (x₂) Score (y)**

**2 1 60**

**3 2 65**

**4 2 70**

**5 3 75**

**6 3 80**

**Find the best fit line without using function**

print("question 10")

print("Krithika Ravishankar - 2022503035")

from sympy import symbols, Eq

X1 = np.array([2, 3, 4, 5, 6])

X2 = np.array([1, 2, 2, 3, 3])

y = np.array([60, 65, 70, 75, 80])

X = np.column\_stack((np.ones(len(X1)), X1, X2))

XtX = np.dot(X.T, X)

XtX\_inverse = np.linalg.inv(XtX)

XtY = np.dot(X.T, y)

theta = np.dot(XtX\_inverse, XtY)

intercept, slope1, slope2 = theta

print(f"Regression Equation: y = {intercept:.2f} + ({slope1:.2f}) \* x1 + ({slope2:.2f}) \* x2")

print(f"Intercept (b): {intercept}")

print(f"Slope for study hours (x1): {slope1}")

print(f"Slope for practice tests (x2): {slope2}")

xnew = np.array([1, 7, 4])

predicted = np.dot(xnew, theta)

print(f"Predicted Score for Study Hours = 7, Practice Tests = 4: {predicted}")

x1, x2 = symbols('x1 x2')

y\_symbolic = intercept + slope1 \* x1 + slope2 \* x2

hours = float(input("Enter study hours: "))

practice\_Test = float(input("Enter practice tests: "))

predicted\_score = y\_symbolic.subs([(x1, hours), (x2, practice\_Test)])

print("Best fit line:", y\_symbolic)

print("Predicted score:", predicted\_score)

