1. create the following two dimensional arrays

[[1., 0., 0., 0., 0.],

[0., 1., 0., 0., 0.],

[0., 0., 1., 0., 0.],

[0., 0., 0., 1., 0.],

[0., 0., 0., 0., 1.]]

Explanation:

import numpy as np: This line imports the NumPy library and gives it the alias "np." NumPy is a powerful library for numerical operations in Python, and it provides support for working with arrays and matrices.

two\_dimensional\_array = np.eye(5): This line creates a 5x5 identity matrix using the np.eye function. The np.eye function generates a 2D identity matrix with ones on the diagonal and zeros elsewhere. The argument 5 specifies the size of the matrix (in this case, a 5x5 matrix).

print(two\_dimensional\_array): This line prints the resulting 5x5 identity matrix to the console. The print function is used to display the content of the variable two\_dimensional\_array.

2. [[1, 0, 0, 0],

[0, 2, 0, 0],

[0, 0, 3, 0],

[0, 0, 0, 4]]

Explanation:

import numpy as np: This line imports the NumPy library and gives it the alias "np."

two\_dimensional\_array = np.diag([1, 2, 3, 4]): This line creates a 4x4 diagonal matrix using the np.diag function. The np.diag function constructs a diagonal matrix with the specified values along its main diagonal. In this case, the specified values are [1, 2, 3, 4], resulting in a diagonal matrix with these values on the main diagonal and zeros elsewhere.

print(two\_dimensional\_array): This line prints the resulting 4x4 diagonal matrix to the console. The print function is used to display the content of the variable two\_dimensional\_array.

3. Create a checkerboard pattern in an array using row and col slicing

Explanation:

import numpy as np: This line imports the NumPy library and gives it the alias "np."

rows, cols = 8, 8: This line sets the variables rows and cols to 8, defining the size of the checkerboard.

checkerboard = np.zeros((rows, cols), dtype=int): This line creates an 8x8 NumPy array filled with zeros. The dtype=int specifies that the array should contain integer values.

checkerboard[1::2, ::2] = 1: This line uses NumPy array slicing to set every second element in odd rows (starting from the second row) and every element in even columns to 1. This creates the black squares in the checkerboard.

checkerboard[::2, 1::2] = 1: This line uses NumPy array slicing to set every second element in even rows and every element in odd columns to 1. This creates the black squares in the remaining positions of the checkerboard.

print(checkerboard): This line prints the resulting checkerboard array to the console.

4. Compute the no of bytes occupied in the memory for an array of numbers from 1 to 10

Explanation:

import numpy as np: This line imports the NumPy library and gives it the alias "np."

arr = np.array([1, 2, 3, 4, 5, 6, 7, 8, 9, 10]): This line creates a NumPy array named arr containing the numbers from 1 to 10.

bytes\_occupied = arr.nbytes: This line uses the nbytes attribute of the NumPy array to compute the number of bytes occupied by the array in memory. The nbytes attribute returns the total number of bytes used by the array data.

print(f"Number of bytes occupied in memory: {bytes\_occupied} bytes"): This line prints the result, indicating the number of bytes occupied by the array in memory.

The actual number of bytes may vary depending on the system architecture and the data type used in the array (e.g., whether it's an array of integers or floating-point numbers). In this example, the default data type for integers in NumPy is used, which is typically 4 bytes per integer on most systems.

5. img=[[200,210,209],[213,0,214],[214,215,217]]

Given a snap shot of the image, estimate the centre pixel with mean,median values.

Explanation:

import numpy as np: This line imports the NumPy library and gives it the alias "np."

img = np.array([[200, 210, 209], [213, 0, 214], [214, 215, 217]]): This line creates a 3x3 NumPy array named img representing an image snapshot with pixel intensities.

mean\_value = np.mean(img): This line calculates the mean value of all pixel intensities in the image using the np.mean function.

median\_value = np.median(img): This line calculates the median value of all pixel intensities in the image using the np.median function.

center\_pixel\_value = img[1, 1]: This line extracts the value of the center pixel in the image (at row 1, column 1) using array indexing.

print(f"Mean value: {mean\_value}"): This line prints the calculated mean value of the pixel intensities.

print(f"Median value: {median\_value}"): This line prints the calculated median value of the pixel intensities.

print(f"Center pixel value: {center\_pixel\_value}"): This line prints the value of the center pixel in the image.

6. create a Pandas series to store the marks of students and filter the marks >50,>70,>90

Explanation:

import pandas as pd: This line imports the Pandas library and gives it the alias "pd."

marks = pd.Series([85, 92, 45, 60, 75, 88, 70, 95, 50]): This line creates a Pandas Series named marks with student marks.

marks\_above\_50 = marks[marks > 50]: This line filters the marks that are greater than 50 and creates a new Series named marks\_above\_50 containing those filtered marks.

marks\_above\_70 = marks[marks > 70]: This line filters the marks that are greater than 70 and creates a new Series named marks\_above\_70 containing those filtered marks.

marks\_above\_90 = marks[marks > 90]: This line filters the marks that are greater than 90 and creates a new Series named marks\_above\_90 containing those filtered marks.

print("Original marks:"): This line prints a label for the original marks.

print(marks): This line prints the original marks.

print("\nMarks > 50:"): This line prints a label for marks greater than 50.

print(marks\_above\_50): This line prints the marks greater than 50.

print("\nMarks > 70:"): This line prints a label for marks greater than 70.

print(marks\_above\_70): This line prints the marks greater than 70.

print("\nMarks > 90:"): This line prints a label for marks greater than 90.

print(marks\_above\_90): This line prints the marks greater than 90.

7.Create data frame given below to store student info from the numpy arrays defined

Explanation:

import numpy as np: This line imports the NumPy library and gives it the alias "np."

import pandas as pd: This line imports the Pandas library and gives it the alias "pd."

student\_names = np.array(['Alice', 'Bob', 'Charlie', 'David']): This line creates a NumPy array named student\_names containing the names of students.

student\_scores = np.array([85, 92, 78, 65]): This line creates a NumPy array named student\_scores containing the scores of students.

student\_grades = np.array(['A', 'A', 'B', 'C']): This line creates a NumPy array named student\_grades containing the grades of students.

student\_df = pd.DataFrame({...}): This line creates a Pandas DataFrame named student\_df by combining the NumPy arrays into a dictionary within the pd.DataFrame constructor. Each key-value pair in the dictionary corresponds to a column in the DataFrame.

print(student\_df): This line prints the resulting DataFrame, displaying the student information in tabular form.

name=np.array(['kevin','john','hari'])

age=np.array([23,21,22])

gender=np.array(['m','m','f'])

state=np.array(['TN','KL','KN'])

cgpa=np.array([9,8,7])

'''

1.Filter the age

2.filter age,cgpa

3.Filter age>22

4.names starts with j

5.names contains 'n'

Explanation:

age\_filter = df['Age']: This line creates a Series named age\_filter by selecting the 'Age' column from the DataFrame.

age\_cgpa\_filter = df[['Age', 'CGPA']]: This line creates a DataFrame named age\_cgpa\_filter by selecting the 'Age' and 'CGPA' columns from the original DataFrame.

age\_above\_22\_filter = df[df['Age'] > 22]: This line creates a DataFrame named age\_above\_22\_filter by filtering rows where the 'Age' is greater than 22.

names\_starts\_with\_j = df[df['Name'].str.startswith('j')]: This line creates a DataFrame named names\_starts\_with\_j by filtering rows where the 'Name' column starts with the letter 'j'.

names\_contain\_n = df[df['Name'].str.contains('n')]: This line creates a DataFrame named names\_contain\_n by filtering rows where the 'Name' column contains the letter 'n'.

The print statements are used to display the original DataFrame and the results of each filtering operation.

8. Given the two data frames,

dfr1=pd.DataFrame({'Id':[1,2,3,5,9],

'Col1':[1,2,3,4,5],

'col2':[6,7,8,9,10],

'col4':['apple','orange','banana','strawberry','rasberry']})

dfr2=pd.DataFrame({'Id':[1,1,3,5],

'ColA':[8,9,10,11],

'colB':[12,13,15,17],

'col4':['apple','orange','banana','kiwi']})

Perform left, right and inner join on the dataframes

Explanation:

pd.merge(dfr1, dfr2, on='Id', how='left', suffixes=('\_dfr1', '\_dfr2')): This line performs a left join on the 'Id' column, and the suffixes \_dfr1 and \_dfr2 are added to the column names from the left and right DataFrames to distinguish them.

pd.merge(dfr1, dfr2, on='Id', how='right', suffixes=('\_dfr1', '\_dfr2')): This line performs a right join on the 'Id' column with the same suffixes.

pd.merge(dfr1, dfr2, on='Id', how='inner', suffixes=('\_dfr1', '\_dfr2')): This line performs an inner join on the 'Id' column with the same suffixes.

print("Left Join:"), print(left\_join): These lines print the result of the left join.

print("\nRight Join:"), print(right\_join): These lines print the result of the right join.

print("\nInner Join:"), print(inner\_join): These lines print the result of the inner join.

9. Apply sigmoid function to the array [2,5,6]

1.define sigmoid function

2.create the np array

3.Apply the function to the array

Inference:

Sigmoid Function Definition: A sigmoid function is defined with the formula: 1 / (1 + np.exp(-x))

NumPy Array Creation: A NumPy array input\_array is created with the values [2, 5, 6].

Sigmoid Function Application: The sigmoid function is applied to each element of the array, resulting in a new array named result.

Display Results: The original array and the array after applying the sigmoid function are printed.

This code demonstrates the application of a sigmoid function to a NumPy array, commonly used in machine learning for converting values to a range between 0 and 1.