**NumPy**

1. Wap to Create an Empty and a Full NumPy Array?

**Input**

**import numpy as np**

**empty\_array = np.empty((3, 3))**

**filled\_array = np.ones((2, 2))**

**print("Empty Array:")**

**print(empty\_array)**

**print("\nFilled Array:")**

**print(filled\_array)**

**Output**

**Empty Array:**

**[[6.92732734e-310 6.92732734e-310 2.10966031e-321]**

**[1.29477285e-272 6.32040375e-316 0.00000000e+000]**

**[0.00000000e+000 1.25990589e-300 0.00000000e+000]]**

**Filled Array:**

**[[1. 1.]**

**[1. 1.]]**

1. **Wap to remove rows in numpy array that contains non-numeric values?**

**Import numpy as np?**

**Def remove\_non\_numeric\_rows(arr):**

**Mask = np.all(np.isreal(arr), axis=1)**

**Return arr[mask]**

**Original\_array = np.array([[1, 2, 3],**

**[4, 5, 6],**

**[‘a’, ‘b’, ‘c’],**

**[7, 8, 9]])**

**Cleaned\_array = remove\_non\_numeric\_rows(original\_array)**

**Print(“Original Array:”)**

**Print(original\_array)**

**Print(“\nCleaned Array:”)**

**Print(cleaned\_array)**

**Output**

**Original Array:**

**[['1' '2' '3']**

**['4' '5' '6']**

**['a' 'b' 'c']**

**['7' '8' '9']]**

**Cleaned Array:**

**[]**

**>**

**3.wap to find the number of occurrences of a sequence in a numpy array ?**

**Input**

**Import numpy as np**

**def count\_occurrences(sequence, array):**

**occurrences = np.where(np.correlate(array, sequence, 'valid') == np.sum(sequence))**

**return len(occurrences[0])**

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

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

**result = count\_occurrences(sequence, array)**

**print(f"The sequence {sequence} occurs {result} times in the array**

**Output**

**Original Array:**

**[['1' '2' '3']**

**['4' '5' '6']**

**['a' 'b' 'c']**

**['7' '8' '9']]**

**Cleaned Array:**

**[]**

**>**

**4.wap to find the most frequent value in a numpy array ?**

**Input**

**import numpy as np**

**def most\_frequent\_value(arr):**

**unique\_values, counts = np.unique(arr, return\_counts=True)**

**max\_count\_index = np.argmax(counts)**

**most\_frequent\_value = unique\_values[max\_count\_index]**

**return most\_frequent\_value**

**my\_array = np.array([1, 2, 3, 4, 2, 2, 3, 1, 4, 4, 4])**

**result = most\_frequent\_value(my\_array)**

**print(f"The most frequent value is: {result}")**

Output

The most frequent value is: 4

**5.wap to bulid an array of add of two numpy arrays ?**

**Input**

import numpy as np

array1 = np.array([1, 2, 3])

array2 = np.array([4, 5, 6])

sum\_array = array1 + array2

print("Resultant Array:", sum\_array)

**Output**

**Resultant array :[5 7 9]**

**6.wap to return the indices of element where the given condition Is satisfied ?**

**Input**

**import numpy as np**

**sample\_array = np.array([1, 2, 3, 4, 5, 6])**

**condition = sample\_array > 3**

**indices = np.where(condition)**

**print("Indices where condition is satisfied:", indices)**

**Output**

**Indices where condition is satisfied: (array([3, 4, 5]),)**

**7.wap to multiplying a matrix(numpy array) by a scalar ?**

**Input**

**import numpy as np**

**def multiply\_matrix\_by\_scalar(matrix, scalar):**

**result = matrix \* scalar**

**return result**

**matrix = np.array([[1, 2], [3, 4]])**

**scalar = 2**

**result\_matrix = multiply\_matrix\_by\_scalar(matrix, scalar)**

**print(result\_matrix)**

**output**

**[[2 4]**

**[6 8]]**

**8.wap for array de dimensioning of giving data ?**

**A=np.array ([[2.5,3.8,1.5])**

**[4.7,2.9,1.56]])**

**Input**

**import numpy as np**

**def redimension\_array(data, new\_shape):**

**result = data.reshape(new\_shape)**

**return result**

**a = np.array([[2.5, 3.8, 1.5], [4.7, 2.9, 1.56]])**

**new\_shape = (3, 2)**

**result\_array = redimension\_array(a, new\_shape)**

**Print(result\_array)**

**Output**

**[[2.5 3.8 ]**

**[1.5 4.7 ]**

**[2.9 1.56]]**

**9.wap to obtaining boolean array from binary array of given data?**

**A=np.array ([1,0,0],**

**[1,1,1,],**

**[0,0,0]])**

**Input**

**Import numpy as np**

**def get\_boolean\_array(binary\_array):**

**boolean\_array = binary\_array.astype(bool)**

**return boolean\_array**

**a = np.array([[1, 0, 0],**

**[1, 1, 1],**

**[0, 0, 0]])**

**result\_boolean\_array = get\_boolean\_array(a)**

**print(result\_boolean\_array)**

**Output**

**[[ True False False]**

**[ True True True]**

**[False False False]]**

**10. Wap to horizontal stackingof numpy arrays?**

**Input**

**import numpy as np**

**def horizontal\_stack\_arrays(arrays):**

**result = np.hstack(arrays)**

**return result**

**array1 = np.array([1, 2, 3])**

**array2 = np.array([4, 5, 6])**

**array3 = np.array([7, 8, 9])**

**result\_array = horizontal\_stack\_arrays([array1, array2, array3])**

**print(result\_array)**

**Output:[1 2 3 4 5 6 7 8 9]**

**11.array generation of random integers within a specified range ?**

**Input**

**import numpy as np**

**random\_integers = np.random.randint(1, 11, 5)**

**print(random\_integers)**

**Output**

**[33893]**

**12. Matrix Multiplication of Given Data?**

**a = np.array([[1,2,3],**

**[4,5,6],**

**[7,8,9]])**

**b = np.array([[2,3,4],**

**[5,6,7],**

**[8,9,10]])**

**Input**

**Import numpy as np**

**a = np.array([[1, 2, 3],**

**[4, 5, 6],**

**[7, 8, 9]])**

**b = np.array([[2, 3, 4],**

**[5, 6, 7],**

**[8, 9, 10]])**

**result = np.dot(a, b)**

**print(result)**

**Output**

**[ 81 96 111]**

**[126 150 174]]**