**How to search the maximum and minimum element in the given array using NumPy?**

Searching is a technique that helps finds the place of a given element or value in the list. In Numpy, one can perform various searching operations using the various functions that are provided in the library like **argmax**, **argmin**, etc.

**numpy.argmax( )**

This function returns indices of the maximum element of the array in a particular axis.

**Example**:

import numpy as np

# Creating 5x4 array

array = np.arange(20).reshape(5, 4)

print(array)

print()

# If no axis mentioned, then it works on the entire array

print(np.argmax(array))

# If axis=1, then it works on each row

print(np.argmax(array, axis=1))

# If axis=0, then it works on each column

print(np.argmax(array, axis=0))

**Output**:

[[ 0 1 2 3]

[ 4 5 6 7]

[ 8 9 10 11]

[12 13 14 15]

[16 17 18 19]]

19

[3 3 3 3 3]

[4 4 4 4]

Similarly one can use **numpy.argmin( )** to return indices of the minimum element of the array in a particular axis.

**How to sort the elements in the given array using Numpy?**

Sorting refers to arrange data in a particular format. Sorting algorithm specifies the way to arrange data in a particular order. In Numpy, one can perform various sorting operations using the various functions that are provided in the library like **sort**, **argsort,** etc.

1. **numpy.sort( )**

This function returns a sorted copy of an array.

**Example**:

import numpy as np

array = np.array([

[3, 7, 1],

[10, 3, 2],

[5, 6, 7]

])

print(array)

print()

# Sort the whole array

print(np.sort(array, axis=None))

# Sort along each row

print(np.sort(array, axis=1))

# Sort along each column

print(np.sort(array, axis=0))

**Output**:

[[ 3 7 1]

[10 3 2]

[ 5 6 7]]

[ 1 2 3 3 5 6 7 7 10]

[[ 1 3 7]

[ 2 3 10]

[ 5 6 7]]

[[ 3 3 1]

[ 5 6 2]

[10 7 7]]

1. **numpy.argsort( )**

This function returns the indices that would sort an array.

**Example**:

import numpy as np

array = np.array([28, 13, 45, 12, 4, 8, 0])

print(array)

print(np.argsort(array))

**Output**:

[28 13 45 12 4 8 0]

[6 4 5 3 1 0 2]

**How to find the mean of every NumPy array in the given list?**

The problem statement is given a list of NumPy array, the task is to find mean of every NumPy array.

1. **Using np.mean**( )

import numpy as np

list = [

np.array([3, 2, 8, 9]),

np.array([4, 12, 34, 25, 78]),

np.array([23, 12, 67])

]

result = []

for i in range(len(list)):

result.append(np.mean(list[i]))

print(result)

**Output**:

[5.5, 30.6, 34.0]

**How to add rows and columns in NumPy array?**

The problem statement is given NumPy array, the task is to add rows/columns basis on requirements to numpy array.

1. **Adding Row** **using numpy.vstack( )**

import numpy as np

array = np.array([

[3, 2, 8],

[4, 12, 34],

[23, 12, 67]

])

newRow = np.array([2, 1, 8])

newArray = np.vstack((array, newRow))

print(newArray)

**Output**:

[[ 3 2 8]

[ 4 12 34]

[23 12 67]

[ 2 1 8]]

1. **Adding Column** **using numpy.column\_stack( )**

import numpy as np

array = np.array([

[3, 2, 8],

[4, 12, 34],

[23, 12, 67]

])

newColumn = np.array([2, 1, 8])

newArray = np.column\_stack((array, newColumn))

print(newArray)

**Output**:

[[ 3 2 8 2]

[ 4 12 34 1]

[23 12 67 8]]

**How to reverse a NumPy array?**

The problem statement is given NumPy array, the task is to reverse the NumPy array.

1. **Using numpy.flipud( )**

import numpy as np

array = np.array([3, 6, 7, 2, 5, 1, 8])

reversedArray = np.flipud(array)

print(reversedArray)

**Output**:

[8 1 5 2 7 6 3]

**How to multiply two matrices in a single line using NumPy?**

The problem statement is given two matrices and one has to multiply those two matrices in a single line using NumPy.

1. **Using numpy.dot( )**

import numpy as np

matrix1 = [

[3, 4, 2],

[5, 1, 8],

[3, 1, 9]

]

matrix2 = [

[3, 7, 5],

[2, 9, 8],

[1, 5, 8]

]

result = np.dot(matrix1, matrix2)

print(result)

**Output**:

[[19 67 63]

[25 84 97]

[20 75 95]]

**How to print the checkerboard pattern of nxn using NumPy?**

The problem statement is given n, print the checkerboard pattern for a nxn matrix considering that 0 for black and 1 for white.

**Solution**:

import numpy as np

n = 8

# Create a nxn matrix filled with 0

matrix = np.zeros((n, n), dtype=int)

# fill 1 with alternate rows and column

matrix[::2, 1::2] = 1

matrix[1::2, ::2] = 1

# Print the checkerboard pattern

for i in range(n):

for j in range(n):

print(matrix[i][j], end=" ")

print()

**Output**:

0 1 0 1 0 1 0 1

1 0 1 0 1 0 1 0

0 1 0 1 0 1 0 1

1 0 1 0 1 0 1 0

0 1 0 1 0 1 0 1

1 0 1 0 1 0 1 0

0 1 0 1 0 1 0 1

1 0 1 0 1 0 1 0