**UIT2502---Data Analytics & Visualization**

**Ex 1 a: Basic Numpy Exercise**

**NAME: SHYAM SARAN P**

**REGNO:3122235002123**

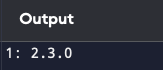
**AIM:**

To write python program using numpy for the given questions.

1. Import numpy and see version

import numpy as np

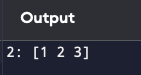
print("1:", np.\_\_version\_\_)



2. Create a 1D array

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

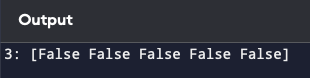
print("2:", a)



3. Create a boolean array

bool\_arr = np.full(5, False)

print("3:", bool\_arr)



4. Extract items that satisfy a condition

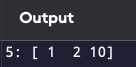
print("4:", a[a > 2])



5. Replace items that satisfy a condition

a[a > 2] = 10

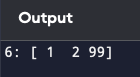
print("5:", a)



6. Replace items without affecting original array

b = np.where(a > 2, 99, a)

print("6:", b)

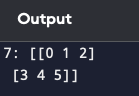


7. Reshape an array

a = np.arange(6)

a\_reshaped = a.reshape(2, 3)

print("7:", a\_reshaped)



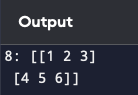
8. Stack two arrays vertically

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

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

v = np.vstack((x, y))

print("8:", v)



9. Stack two arrays horizontally

h = np.hstack((x, y))

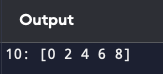
print("9:", h)



10. Generate custom sequences without hardcoding

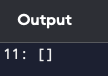
seq = np.arange(0, 10, 2)

print("10:", seq)



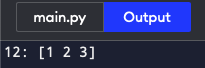
11. Get common items between two arrays

print("11:", np.intersect1d(x, y))



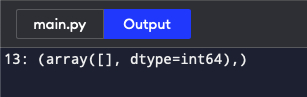
12. Remove from one array items existing in another

print("12:", np.setdiff1d(x, y))



13. Get positions where elements match

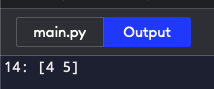
print("13:", np.where(x == y))



14. Extract numbers in a range

z = np.array([1, 4, 5, 8])

print("14:", z[(z >= 2) & (z <= 6)])

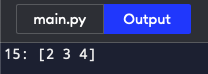


15. Make scalar function work on arrays

def add\_one(n): return n + 1

vfunc = np.vectorize(add\_one)

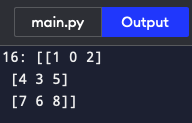
print("15:", vfunc(np.array([1, 2, 3])))

16. Swap two columns in 2D array

arr = np.arange(9).reshape(3, 3)

arr[:, [0, 1]] = arr[:, [1, 0]]

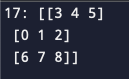
print("16:", arr)



17. Swap two rows in 2D array

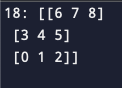
arr[[0, 1], :] = arr[[1, 0], :]

print("17:", arr)



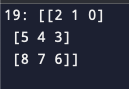
18. Reverse rows

print("18:", arr[::-1, :])



19. Reverse columns

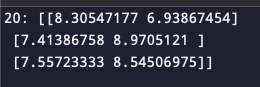
print("19:", arr[:, ::-1])



20. 2D array with random floats between 5 and 10

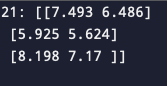
rand\_arr = 5 + (10 - 5) \* np.random.random((3, 2))

print("20:", rand\_arr)

21. Print only 3 decimal places

np.set\_printoptions(precision=3)

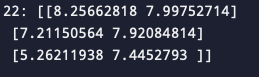
print("21:", rand\_arr)



22. Suppress scientific notation

np.set\_printoptions(suppress=True)

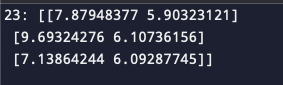
print("22:", rand\_arr)



23. Limit number of items printed

np.set\_printoptions(threshold=5)

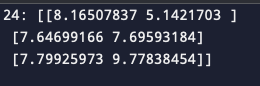
print("23:", rand\_arr)



24. Print full array without truncating

np.set\_printoptions(threshold=np.inf)

print("24:", rand\_arr)



25. Import dataset with numbers & text (skipped file read; showing placeholder)

# data = np.genfromtxt('file.csv', delimiter=',', dtype='object')

print("25: Skipped (requires external file)")



26. Extract particular column from 1D array of tuples

a = np.array([(1, 'a'), (2, 'b'), (3, 'c')])

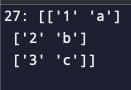
print("26:", a[:, 1])



27. Convert 1D array of tuples to 2D array

b = np.array(a.tolist())

print("27:", b)



28. Mean, median, std

nums = np.array([1, 2, 3, 4, 5])

print("28:", np.mean(nums), np.median(nums), np.std(nums))



29. Normalize 0–1

norm = (nums - np.min(nums)) / (np.max(nums) - np.min(nums))

print("29:", norm)



30. Compute softmax

e = np.exp(nums)

softmax = e / np.sum(e)

print("30:", softmax)



31. Percentile scores

print("31:", np.percentile(nums, [25, 50, 75]))



32. Insert at random positions

idx = np.random.randint(0, len(nums)+1, size=2)

print("32:", np.insert(nums, idx, [99, 100]))



33. Position of missing values

arr\_with\_nan = np.array([1, np.nan, 3, np.nan])

print("33:", np.where(np.isnan(arr\_with\_nan)))



34. Filter based on two conditions

print("34:", nums[(nums > 2) & (nums < 5)])



35. Drop rows with missing values (example with 2D)

arr2d = np.array([[1, 2], [np.nan, 3], [7, 6]])

print("35:", arr2d[~np.isnan(arr2d).any(axis=1)])



36. Correlation between two columns

arr2 = np.array([[1, 2], [3, 6], [5, 10]])

print("36:", np.corrcoef(arr2[:, 0], arr2[:, 1]))



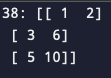
37. Second largest value grouped by another array

print("37: Skipped (needs pandas)")



38. Sort 2D array by a column

print("38:", arr2[arr2[:, 1].argsort()])



39. Most frequent value

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

print("39:", np.bincount(c).argmax())



40. Position of first value > given

print("40:", np.argmax(nums > 3))



41. Replace values greater than cutoff

nums\_copy = nums.copy()

nums\_copy[nums\_copy > 3] = 3

print("41:", nums\_copy)



42. Positions of top n values

n = 2

print("42:", np.argsort(nums)[-n:])



43. Row-wise counts of values (example)

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

counts = [np.bincount(row) for row in a]

print("43:", counts)



44. Flatten array of arrays

print("44:", a.flatten())



**Result:** Thus, we have successfully learned the basic operations of NumPy, including creating arrays, performing mathematical operations, and understanding array properties and behavior.