NUMPY - CODING Q&A

BY - SHUBHAM WADEKAR

NUMPY CODING QUESTIONS FOR DATA ENGINEERS

1. Create a 2D array and find the sum of all elements.

Scenario:
You have a 2D array. Calculate the sum of all its elements.

Logic:

- Use np.sum() to calculate the sum.

Program:

```
import numpy as np
arr = np.array([[1, 2], [3, 4]])
total_sum = np.sum(arr)
print(total_sum)
```

2. Find the mean of each row in a 2D array.

Scenario:
You have a 2D array. Calculate the mean of each row.

Logic:

- Use np.mean() with axis=1.

Program:

```
import numpy as np
arr = np.array([[1, 2], [3, 4]])
row_means = np.mean(arr, axis=1)
print(row_means)
```

3. Find the maximum value in each column of a 2D array.

Scenario:
You have a 2D array. Find the maximum value in each column.

Logic:

- Use np.max() with axis=0.

Program:

```
import numpy as np
arr = np.array([[1, 2], [3, 4]])
col_max = np.max(arr, axis=0)
print(col_max)
```

4. Reshape a 1D array into a 2D array.

Scenario:
You have a 1D array. Reshape it into a 2D array of shape (2, 3).

Logic:

- Use np.reshape().

Program:

```
import numpy as np
arr = np.array([1, 2, 3, 4, 5, 6])
reshaped_arr = arr.reshape(2, 3)
print(reshaped_arr)
```

5. Flatten a 2D array into a 1D array.

Scenario:
You have a 2D array. Flatten it into a 1D array.

Logic:

- Use .flatten().

Program:

```
import numpy as np
arr = np.array([[1, 2], [3, 4]])
flattened_arr = arr.flatten()
print(flattened_arr)
```

6. Concatenate two 1D arrays.

Scenario:
You have two 1D arrays. Concatenate them.

Logic:

- Use np.concatenate().

Program:

```
import numpy as np
arr1 = np.array([1, 2, 3])
arr2 = np.array([4, 5, 6])
concatenated_arr = np.concatenate((arr1, arr2))
print(concatenated_arr)
```

7. Stack two 1D arrays vertically.

Scenario:
You have two 1D arrays. Stack them vertically.

Logic:

- Use np.vstack().

Program:

```
import numpy as np
arr1 = np.array([1, 2, 3])
arr2 = np.array([4, 5, 6])
stacked_arr = np.vstack((arr1, arr2))
print(stacked_arr)
```

8. Stack two 1D arrays horizontally.

Scenario:
You have two 1D arrays. Stack them horizontally.

Logic:

- Use np.hstack().

Program:

```
import numpy as np
arr1 = np.array([1, 2, 3])
arr2 = np.array([4, 5, 6])
stacked_arr = np.hstack((arr1, arr2))
print(stacked_arr)
```

9. Split a 1D array into 3 equal parts.

Scenario:
You have a 1D array. Split it into 3 equal parts.

Logic:

- Use np.split().

Program:

```
import numpy as np
arr = np.array([1, 2, 3, 4, 5, 6])
split_arr = np.split(arr, 3)
print(split_arr)
```

10. Find the indices of non-zero elements in an array.

Scenario:
You have an array. Find the indices of non-zero elements.

Logic:

- Use np.nonzero().

Program:

```python
import numpy as np
arr = np.array([0, 1, 0, 2, 3])
nonzero_indices = np.nonzero(arr)
print(nonzero_indices)
```

11. Create a 3D array and find the sum along a specific axis.

Scenario:
You have a 3D array. Calculate the sum along the second axis.

Logic:

- Use np.sum() with axis=1.

Program:

```python
import numpy as np
arr = np.array([[[1, 2], [3, 4]], [[5, 6], [7, 8]]])
sum_axis_1 = np.sum(arr, axis=1)
print(sum_axis_1)
```

12. Perform matrix multiplication on two 2D arrays.

Scenario:
You have two 2D arrays. Perform matrix multiplication.

Logic:

- Use np.dot() or @.

Program:

```python
import numpy as np
arr1 = np.array([[1, 2], [3, 4]])
arr2 = np.array([[5, 6], [7, 8]])
result = np.dot(arr1, arr2)
print(result)
```

13. Find the determinant of a 2D array.

Scenario:
You have a 2D array. Calculate its determinant.

Logic:

- Use np.linalg.det().

Program:

```
import numpy as np
arr = np.array([[1, 2], [3, 4]])
determinant = np.linalg.det(arr)
print(determinant)
```

14. Find the inverse of a 2D array.

Scenario:
You have a 2D array. Calculate its inverse.

Logic:

- Use np.linalg.inv().

Program:

```
import numpy as np
arr = np.array([[1, 2], [3, 4]])
inverse = np.linalg.inv(arr)
```

15. Solve a system of linear equations.

Scenario:
You have a system of linear equations. Solve it using NumPy.

Logic:

- Use np.linalg.solve().

Program:

```
import numpy as np
A = np.array([[3, 1], [1, 2]])
b = np.array([9, 8])
solution = np.linalg.solve(A, b)
print(solution)
```

16. Find the eigenvalues and eigenvectors of a 2D array.

Scenario:
You have a 2D array. Find its eigenvalues and eigenvectors.

Logic:

- Use np.linalg.eig().

Program:

```
import numpy as np
arr = np.array([[1, 2], [3, 4]])
eigenvalues, eigenvectors = np.linalg.eig(arr)
print(eigenvalues, eigenvectors)
```

17. Perform element-wise multiplication of two arrays.

Scenario:
You have two arrays. Perform element-wise multiplication.

Logic:

- Use * operator.

Program:

```
import numpy as np
arr1 = np.array([1, 2, 3])
arr2 = np.array([4, 5, 6])
result = arr1 * arr2
print(result)
```

18. Perform element-wise division of two arrays.

Scenario:
You have two arrays. Perform element-wise division.

Logic:

- Use / operator.

Program:

```
import numpy as np
arr1 = np.array([1, 2, 3])
arr2 = np.array([4, 5, 6])
result = arr1 / arr2
print(result)
```

19. Find the dot product of two 1D arrays.

Scenario:
You have two 1D arrays. Calculate their dot product.

Logic:

- Use np.dot().

Program:

```
import numpy as np
arr1 = np.array([1, 2, 3])
arr2 = np.array([4, 5, 6])
dot_product = np.dot(arr1, arr2)
print(dot_product)
```

20. Find the cross product of two 1D arrays.

Scenario:
You have two 1D arrays. Calculate their cross product.

Logic:

- Use np.cross().

Program:

```
import numpy as np
arr1 = np.array([1, 2, 3])
arr2 = np.array([4, 5, 6])
cross_product = np.cross(arr1, arr2)
print(cross_product)
```

21. Normalize an array to a range of 0 to 1.

Scenario:
You have an array. Normalize its values to a range of 0 to 1.

Logic:

- Use (arr - min) / (max - min).

Program:

```
import numpy as np
arr = np.array([1, 2, 3, 4, 5])
normalized_arr = (arr - np.min(arr)) / (np.max(arr) - np.min(arr))
print(normalized_arr)
```

22. Calculate the cumulative sum of an array.

Scenario:
You have an array. Calculate its cumulative sum.

Logic:

- Use np.cumsum().

Program:

```python
import numpy as np
arr = np.array([1, 2, 3, 4, 5])
cumulative_sum = np.cumsum(arr)
print(cumulative_sum)
```

23. Calculate the cumulative product of an array.

Scenario:
You have an array. Calculate its cumulative product.

Logic:

- Use np.cumprod().

Program:

```python
import numpy as np
arr = np.array([1, 2, 3, 4, 5])
cumulative_product = np.cumprod(arr)
print(cumulative_product)
```

24. Find the unique elements in an array.

Scenario:
You have an array. Find its unique elements.

Logic:

- Use np.unique().

Program:

```python
import numpy as np
arr = np.array([1, 2, 2, 3, 3, 3])
unique_elements = np.unique(arr)
print(unique_elements)
```

25. Sort an array along a specific axis.

Scenario:
You have a 2D array. Sort it along the second axis.

Logic:

- Use np.sort() with axis=1.

Program:

```
import numpy as np
arr = np.array([[3, 1], [2, 4]])
sorted_arr = np.sort(arr, axis=1)
print(sorted_arr)
```

26. Find the indices that would sort an array.

Scenario:
You have an array. Find the indices that would sort it.

Logic:

- Use np.argsort().

Program:

```
import numpy as np
arr = np.array([3, 1, 2])
sorted_indices = np.argsort(arr)
print(sorted_indices)
```

27. Reverse an array.

Scenario:
You have an array. Reverse its elements.

Logic:

- Use slicing [::-1].

Program:

```
import numpy as np
arr = np.array([1, 2, 3, 4, 5])
reversed_arr = arr[::-1]
print(reversed_arr)
```

28. Find the index of the maximum value in an array.

Scenario:
You have an array. Find the index of the maximum value.

Logic:

- Use np.argmax().

Program:

```
import numpy as np
arr = np.array([1, 2, 3, 4, 5])
max_index = np.argmax(arr)
print(max_index)
```

29. Find the index of the minimum value in an array.

Scenario:
You have an array. Find the index of the minimum value.

Logic:

- Use np.argmin().

Program:

```
import numpy as np
arr = np.array([1, 2, 3, 4, 5])
min_index = np.argmin(arr)
print(min_index)
```

30. Replace all occurrences of a value in an array.

Scenario:
You have an array. Replace all occurrences of a value (e.g., 2) with another value (e.g., 10).

Logic:

- Use np.where().

Program:

```
import numpy as np
arr = np.array([1, 2, 3, 2, 5])
replaced_arr = np.where(arr == 2, 10, arr)
print(replaced_arr)
```

31. Find the median of an array.

Scenario:
You have an array. Calculate its median.

Logic:

- Use np.median().

Program:

```
import numpy as np
arr = np.array([1, 2, 3, 4, 5])
median = np.median(arr)
print(median)
```

32. Calculate the standard deviation of an array.

Scenario:
You have an array. Calculate its standard deviation.

Logic:

- Use np.std().

Program:

```
import numpy as np
arr = np.array([1, 2, 3, 4, 5])
std_dev = np.std(arr)
print(std_dev)
```

33. Calculate the variance of an array.

Scenario:
You have an array. Calculate its variance.

Logic:

- Use np.var().

Program:

```
import numpy as np
arr = np.array([1, 2, 3, 4, 5])
variance = np.var(arr)
print(variance)
```

34. Find the percentile of an array.

Scenario:
You have an array. Find the 75th percentile.

Logic:

- Use np.percentile().

Program:

```
import numpy as np
arr = np.array([1, 2, 3, 4, 5])
percentile_75 = np.percentile(arr, 75)
print(percentile_75)
```

35. Generate a random array and shuffle it.

Scenario:
You need to generate a random array and shuffle its elements.

Logic:

- Use np.random.rand() and np.random.shuffle().

Program:

```
import numpy as np
arr = np.random.rand(5)
np.random.shuffle(arr)
print(arr)
```

36. Generate a random array with a specific shape.

Scenario:
You need to generate a random array of shape (3, 3).

Logic:

- Use np.random.rand().

Program:

```
import numpy as np
arr = np.random.rand(3, 3)
print(arr)
```

37. Generate a random integer array within a range.

Scenario:
You need to generate a random integer array with values between 1 and 10.

Logic:

- Use np.random.randint().

Program:

```
import numpy as np
arr = np.random.randint(1, 10, size=(3, 3))
print(arr)
```

38. Create an identity matrix.

Scenario:
You need to create a 3x3 identity matrix.

Logic:

- Use np.eye().

Program:

```
import numpy as np
identity_matrix = np.eye(3)
print(identity_matrix)
```

39. Create a diagonal matrix from a 1D array.

Scenario:
You have a 1D array. Create a diagonal matrix from it.

Logic:

- Use np.diag().

Program:

```
import numpy as np
arr = np.array([1, 2, 3])
diagonal_matrix = np.diag(arr)
print(diagonal_matrix)
```

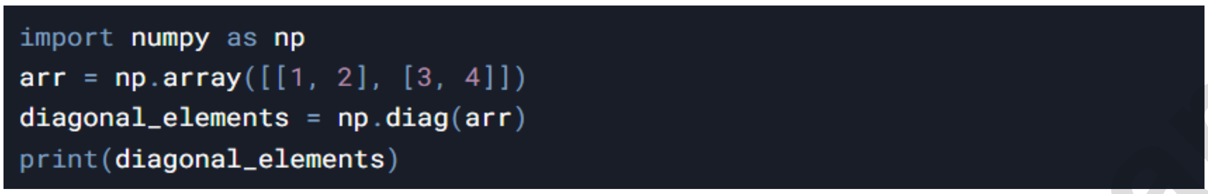
40. Extract the diagonal elements of a 2D array.

Scenario:
You have a 2D array. Extract its diagonal elements.

Logic:

- Use np.diag().

Program:



```
import numpy as np
arr = np.array([[1, 2], [3, 4]])
diagonal_elements = np.diag(arr)
print(diagonal_elements)
```