PYTHON INTERVIEW Q&A

EDITION 2025

Developed by Swapnjeet S (Data Tutorials)

1. What is NumPy, and why is it important for data analysis?

- **Answer:**
  NumPy, short for "Numerical Python," is a fundamental Python library for numerical and scientific computing. It provides support for large, multi-dimensional arrays and matrices, along with a collection of mathematical functions to operate on them efficiently.
  Importance for data analysis:
  - Fast array computations
  - Memory efficiency
  - Extensive mathematical functionality
  - Foundation for other libraries like Pandas and Scikit-learn.

---

2. How do you create a NumPy array? Provide examples.

- **Answer:**
  NumPy arrays can be created using the np.array() function:

```
import numpy as np

# From a list

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

print(arr)

# Output: [1 2 3 4]


# From a range

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

print(arr_range)

# Output: [0 2 4 6 8]
```

---

3. What are the differences between Python lists and NumPy arrays?

- **Answer:**
  - **Speed:** NumPy arrays are faster due to optimized C code.

- **Memory:** NumPy uses less memory by storing elements of the same data type.
- **Operations:** NumPy supports vectorized operations, unlike Python lists.
- **Functionality:** NumPy offers advanced operations like broadcasting, slicing, and more.

Intermediate NumPy Questions

4. What is broadcasting in NumPy? Give an example.

- **Answer:**
  Broadcasting allows NumPy to perform arithmetic operations on arrays with different shapes by "stretching" the smaller array to match the shape of the larger array.

```
import numpy as np

arr1 = np.array([1, 2, 3])

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

result = arr1 + arr2

print(result)

# Output:

# [[2 3 4]

#  [3 4 5]

#  [4 5 6]]
```

---

5. How do you handle missing data in NumPy?

- **Answer:**
  NumPy provides np.nan to represent missing data. You can use functions like np.isnan() to detect missing values and np.nan_to_num() to replace them:

```
arr = np.array([1, 2, np.nan, 4])

print(np.isnan(arr))  # Output: [False False  True False]

print(np.nan_to_num(arr))  # Output: [1. 2. 0. 4.]
```

---

6. How do you calculate statistical measures in NumPy?

- **Answer:**
  NumPy provides several built-in functions for statistical analysis:

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

print(np.mean(arr))   # Mean: 3.0
print(np.median(arr)) # Median: 3.0
print(np.std(arr))    # Standard deviation: 1.4142135623730951
print(np.var(arr))    # Variance: 2.0
```

Advanced NumPy Questions

7. How do you reshape and flatten arrays in NumPy?

- **Answer:**
  - **Reshape:** Change the shape of an array without altering its data.

```
arr = np.arange(6).reshape(2, 3)
print(arr)
# Output:
# [[0 1 2]
#  [3 4 5]]
```

  - **Flatten:** Convert a multi-dimensional array into a 1D array:

```
flat = arr.flatten()
print(flat)
# Output: [0 1 2 3 4 5]
```

---

8. Explain slicing in NumPy with an example.

- **Answer:**
  Slicing allows you to extract portions of an array:

```
arr = np.array([10, 20, 30, 40, 50])
print(arr[1:4])  # Output: [20 30 40]
print(arr[::2])  # Output: [10 30 50]
```

---

9. What is the difference between np.copy() and assignment (=)?

- **Answer:**
  - Assignment (=) creates a reference to the original array. Changes to the new array affect the original.
  - np.copy() creates a new array. Changes to the new array do not affect the original.

```
arr = np.array([1, 2, 3])
arr_copy = arr.copy()
```

```
arr_copy[0] = 10

print(arr)       # Original: [1 2 3]

print(arr_copy)   # Copy: [10 2 3]
```

---

10. How can you optimize large matrix operations in NumPy?

- **Answer:**
  - Use vectorized operations instead of loops.
  - Leverage built-in functions like np.dot() for matrix multiplication.
  - Use np.linalg for linear algebra operations:

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

B = np.array([[5, 6], [7, 8]])

result = np.dot(A, B)

print(result)

# Output:

# [[19 22]

#  [43 50]]
```

Scenario-Based Questions

11. How would you create a NumPy array of random numbers between 0 and 1?

- **Answer:**

```
arr = np.random.rand(3, 3)

print(arr)

# Example Output:

# [[0.5488135  0.71518937 0.60276338]

#  [0.54488318 0.4236548  0.64589411]

#  [0.43758721 0.891773   0.96366276]]
```

---

12. You have a 2D array representing sales data. How would you calculate the total sales for each row?

- **Answer:**

```
sales = np.array([[100, 200, 300], [400, 500, 600]])
```

```
row_totals = sales.sum(axis=1)

print(row_totals)

# Output: [ 600 1500 ]
```

---

13. Explain the difference between np.zeros() and np.empty() with examples.

- **Answer:**
    - np.zeros() creates an array filled with zeros:

```
arr = np.zeros((2, 3))

print(arr)

# Output:

# [[0. 0. 0.]

#  [0. 0. 0.]]
```

- np.empty() creates an array without initializing values (faster but may contain garbage values):

```
arr = np.empty((2, 3))

print(arr)

# Example Output:

# [[2.12199579e-314 0.00000000e+000 6.95335581e-310]

#  [6.95335581e-310 0.00000000e+000 0.00000000e+000]]
```

---

14. How do you concatenate two NumPy arrays? Provide an example.

- **Answer:**

```
arr1 = np.array([1, 2, 3])

arr2 = np.array([4, 5, 6])

concatenated = np.concatenate((arr1, arr2))

print(concatenated)

# Output: [1 2 3 4 5 6]
```

15. How do you calculate the cumulative sum and product of a NumPy array?

- **Answer:**
  Use np.cumsum() for cumulative sum and np.cumprod() for cumulative product:

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

print(np.cumsum(arr))  # Output: [ 1  3  6 10]
```

print(np.cumprod(arr)) # Output: [1 2 6 24]

---

16. How can you filter data in NumPy based on a condition?

- **Answer:**
  Use boolean indexing to filter elements that satisfy a condition:

arr = np.array([10, 20, 30, 40, 50])

filtered = arr[arr > 30]

print(filtered) # Output: [40 50]

---

17. Explain how to normalize a dataset using NumPy.

- **Answer:**
  Normalize data to have values between 0 and 1:

data = np.array([10, 20, 30, 40])

normalized = (data - data.min()) / (data.max() - data.min())

print(normalized) # Output: [0.   0.33 0.67 1.]

---

18. How can you find unique values and their counts in a NumPy array?

- **Answer:**
  Use np.unique() with the return_counts parameter:

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

unique_values, counts = np.unique(arr, return_counts=True)

print(unique_values) # Output: [1 2 3]

print(counts) # Output: [1 2 3]

---

19. Given a 2D array, how would you find the maximum value in each column and each row?

- **Answer:**
  Use axis in np.max():

arr = np.array([[10, 20, 30], [40, 50, 60]])

print(np.max(arr, axis=0)) # Max in each column: [40 50 60]

print(np.max(arr, axis=1)) # Max in each row: [30 60]

---

20. What is the difference between np.linalg.inv() and np.linalg.pinv()? When would you use each?

- **Answer:**
    - np.linalg.inv() computes the inverse of a square matrix.
    - np.linalg.pinv() computes the Moore-Penrose pseudoinverse for non-square or singular matrices.

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

print(np.linalg.inv(A))  # Inverse of A

print(np.linalg.pinv(A))  # Pseudoinverse of A
```

---

21. Explain how you would handle a missing value in a NumPy array.

- **Answer:**
  Replace missing values (np.nan) using np.nan_to_num() or calculate statistical measures ignoring np.nan using functions like np.nanmean().

```
arr = np.array([1, 2, np.nan, 4])

print(np.nanmean(arr))  # Output: 2.33

clean_arr = np.nan_to_num(arr)

print(clean_arr)  # Output: [1. 2. 0. 4.]
```

---

22. How would you generate a random 2D array and set a specific seed for reproducibility?

- **Answer:**
  Use np.random.seed() before generating the array:

```
np.random.seed(42)

arr = np.random.rand(3, 3)

print(arr)

# Example Output:

# [[0.37454012 0.95071431 0.73199394]

#  [0.59865848 0.15601864 0.15599452]

#  [0.05808361 0.86617615 0.60111501]]
```

---

23. What is the purpose of np.meshgrid()? Provide an example.

- **Answer:**
  np.meshgrid() creates coordinate grids for vectorized operations:

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

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

X, Y = np.meshgrid(x, y)

print(X)  # X-coordinates

print(Y)  # Y-coordinates
```

---

24. How can you stack arrays vertically and horizontally in NumPy?

- **Answer:**
  Use np.vstack() for vertical stacking and np.hstack() for horizontal stacking:

```
arr1 = np.array([1, 2, 3])

arr2 = np.array([4, 5, 6])

print(np.vstack((arr1, arr2)))

# Output:

# [[1 2 3]

#  [4 5 6]]

print(np.hstack((arr1, arr2)))

# Output: [1 2 3 4 5 6]
```

---

25. How would you create a rolling mean using NumPy?

- **Answer:**
  Calculate a rolling mean using slicing:

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

window_size = 3

rolling_mean = np.convolve(arr, np.ones(window_size)/window_size, mode='valid')

print(rolling_mean)  # Output: [2. 3. 4.]
```

---

26. Scenario: You have a dataset with outliers. How can you identify and remove them using NumPy?

- **Answer:**
  Outliers can be identified using z-scores or IQR (Interquartile Range):

```
data = np.array([10, 12, 14, 100, 16, 18])

Q1, Q3 = np.percentile(data, [25, 75])

IQR = Q3 - Q1
```

```
lower_bound = Q1 - 1.5 * IQR

upper_bound = Q3 + 1.5 * IQR

filtered_data = data[(data >= lower_bound) & (data <= upper_bound)]

print(filtered_data)  # Output: [10 12 14 16 18]
```

---

27. How can you perform matrix multiplication using NumPy?

- **Answer:**
  Use np.dot() or @ for matrix multiplication:

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

B = np.array([[5, 6], [7, 8]])

print(np.dot(A, B))

# Output:

# [[19 22]

#  [43 50]]
```