**Quick Intro to NumPy**

As we know, **NumPy**, aka (Numerical Python) is a Python-based open-source package that is used for performing scientific computations, and mathematical and statistical operations. Along with this, NumPy uses N-dimensional arrays, so it is a great choice for data science projects.

Now, with further ado, let’s discuss the most common NumPy-related questions you might come across during your interviews.

**Top 70 NumPy Interview Questions and Answers**

In this section, we have compiled a list of frequently asked **NumPy interview questions and answers** for freshers as well as for **experienced** that might appear in your next interview.

**Q.1 What is NumPy?**

The Python package [NumPy](https://www.geeksforgeeks.org/python-numpy), which stands for “Numerical Python,” is the basis for numerical and scientific computing. It offers support for arrays, matrices, and a variety of mathematical operations that can effectively operate on these arrays. In the [Python](https://www.geeksforgeeks.org/python-programming-language) environment, NumPy is a fundamental library for manipulating and analyzing data. It also used in many additional libraries that are used in data science, machine learning areas.

**Q.2 How do I create a NumPy array?**

We can create NumPy arrays using various methods. Here are some common ways to create NumPy arrays:

1. [Using np. array()](https://www.geeksforgeeks.org/basics-of-numpy-arrays)
2. [np.zeros()](https://www.geeksforgeeks.org/basics-of-numpy-arrays)
3. [np.ones()](https://www.geeksforgeeks.org/numpy-ones-python)
4. [np.full()](https://www.geeksforgeeks.org/numpy-full-python)
5. [np.arange()](https://www.geeksforgeeks.org/numpy-arrange-in-python)
6. [np.linspace()](https://www.geeksforgeeks.org/numpy-linspace-python)

**Q.3 What are the main features of Numpy?**

Here are some main features of the NumPy.

1. Arrays
2. Efficiency
3. Mathematical Functions
4. Broadcasting
5. Integration with other libraries
6. Multi-dimensional arrays
7. Indexing and Slicing
8. Memory Management

**Q.4 How do you calculate the dot product of two NumPy arrays?**

Calculating the dot product of two NumPy arrays we used [numpy.dot()](https://www.geeksforgeeks.org/numpy-dot-python) function and we also used the @ operator:

**Using numpy.dot() function:**

numpy.dot(a, b)

a: The first input array (NumPy array).  
b: The second input array (NumPy array).

**Using the @ operator**

a @ b

Both methods will return the dot product of the two arrays as a scalar value.

**Q.5 How do I access elements in a NumPy array?**

In NumPy, we can access elements in an array by indexing and slicing. Here’s we can do it:

**Slicing:** You can also access a range of elements using slicing.[Slicing](https://www.geeksforgeeks.org/python-list-slicing) allows you to extract a subset of the array based on the indices or ranges you provide.

**Boolean indexing: B**oolean indexing is used to access elements based on a condition. it help us when we want to select elements that meet the critiria.

**Q.6 What is the difference between a shallow copy and a deep copy in NumPy?**

In numPy we have two ways to copy an array. shallow copy and deep copy are two most used methods used in numpy to copy an array. Here is the main difference between both of them.

| **Shallow copy** | **Deep copy** |
| --- | --- |
| A shallow copy is a new array that is a view of the original array’s data. | A deep copy is a completely new and independent array that does not share any data with the original array. |
| The new array is referencing the same data as the original array, so there’s actually no duplication happening. | The data is duplicated, creating a separate copy in memory. |
| Changes made in the original array will also be reflected in the shallow copy, and vice versa. | Changes made in the original array do not affect the deep copy, and vice versa. |

**Q.7 How do you reshape a NumPy array?**

We can reshape a NumPy array by using the reshape() method or the [np.reshape()](https://www.geeksforgeeks.org/numpy-reshape-python) function. it help us to change the dimensions of the array and keep all the elements constant.

**Using the reshape() method:**

array1= original\_array.reshape(new\_shape)

**Using the np.reshape() function:**

array1 = np.reshape(original\_array, new\_shape)

In both cases, original\_array is the existing NumPy array you want to reshape, and new\_shape is a tuple specifying the desired shape of the new array.

**Q.8 How to perform element-wise operations on NumPy arrays?**

To perform element-wise operations on NumPy arrays, you can use standard arithmetic operators.NumPy automatically applies these operations element-wise when you use them with arrays of the same shape.

import numpy as np  
  
# Create two NumPy arrays  
array1 = np.array([1, 2, 3, 4, 5])  
array2 = np.array([6, 7, 8, 9, 10])  
  
# Perform element-wise operations  
result\_addition= array1 + array2  
result\_subtract = array1 - array2  
result\_multiply = array1 \* array2  
result\_divide = array1 / array2  
result\_power = np.power(array1, 2)

**Q.9 Define the var function in NumPy.**

In NumPy, the var function is used to compute the variance of elements in an array or along a specified axis. Variance is a measure of the spread or dispersion of data points.

np.var(a, axis=None, dtype=None)

a: The input array for which you want to calculate the variance.  
axis: Axis or axes along which the variance is computed. If not specified, the variance is calculated for the whole array. It can be an integer or a tuple of integers to specify multiple axes.  
dtype: The data type for the returned variance. If not specified, the data type is inferred from the input array.

**Q.10 Define the min and max function in NumPy.**

In NumPy, you can use the min() and max() functions to find the minimum and maximum values in a NumPy array, respectively.

**np.min() Function:**

The np.min() function calculates the minimum value in a NumPy array along a specified axis or for the entire array.

np.min(a, axis=None)

a: The input array in which you want to find the minimum value.  
axis: Axis or axes along which the minimum value is computed. If not specified, the minimum is calculated for the whole array. It can be an integer or a tuple of integers to specify multiple axes.

**np.max() Function:**

The np.max() function calculates the maximum value in a NumPy array along a specified axis or for the entire array.

np.max(a, axis=None)  
  
a: The input array in which you want to find the maximum value.  
axis (optional): Axis or axes along which the maximum value is computed. If not specified, the maximum is calculated for the whole array.

**Q.11 How to generate random numbers with NumPy?**

NumPy provides a wide range of functions for generating random numbers. You can generate random numbers from various probability distributions, set seeds for reproducibility, and more. Here are some common ways to generate random numbers with NumPy:

**Using np.random.rand()**

Generating a Random Float between 0 and 1 using[np.random.rand()](https://www.geeksforgeeks.org/numpy-random-rand-python)

random\_float = np.random.rand()

**Using np.random.randint()**

Generating a Random Integer within a Range using [np.random.randint()](https://www.geeksforgeeks.org/random-sampling-in-numpy-randint-function).

random\_integer = np.random.randint()

**Using np.random.randn()**

random\_float = np.random.rand()

**Using np.random.seed()**

We can set a seed using [np.random.seed()](https://www.geeksforgeeks.org/random-seed-in-python)to ensure that the generated random numbers are reproducible.

np.random.seed(seed\_value)

**Q.12 What is the purpose of NumPy in Python?**

NumPy (Numerical Python) is a fundamental library in Python for scientific computing and data analysis. it is the main purpose for providing support for large and multi-dimensional arrays and matrices.

**Q.13 How can you create a NumPy array from a Python list?**

We can create a NumPy array from a Python list using the np.array() constructor provided by NumPy.

python\_list = [1, 2, 3, 4, 5]  
  
# Convert the Python list to a NumPy array  
numpy\_array = np.array(python\_list)

* Create a list that containing the elements we want to convert into NumPy array.
* Use the np.array() constructor to create a NumPy array from the Python list. Pass python\_list as the argument to the np.array() function.

**Q.14 How can you access elements in a NumPy array based on specific conditions?**

We can access elements in a NumPy array based on specific conditions using boolean indexing. Boolean indexing allows us to create true and false values based on a condition.

arr = np.array([1, 2, 3, 4, 5])  
  
# Define a condition (boolean mask)  
condition = (arr > some\_value)   
  
# Use the condition to access elements  
selected\_elements = arr[condition]

* Create a NumPy array arr or use an existing array.
* Define a condition by creating a boolean mask. Replace some\_value with your specific condition. For each element in the array, a logical expression that results in True or False should be used as the condition.
* Use the boolean mask (condition) to access elements from the array arr. The result, selected\_elements, will be an array containing only the elements that satisfy the condition.

**Q.15 What are some common data types supported by NumPy?**

In NumPy there are so many data types that are used to specify the type of data which stored in array. This data type provide control that how data stored in memory during operations.Some common data types supported by NumPy include:

1. int
2. float
3. complex
4. bool
5. object
6. datetime

**Q.16 How can you concatenate two NumPy arrays vertically?**

We can concatenate two NumPy arrays vertically (along the rows) using the np.vstack() function or the [np.concatenate()](https://www.geeksforgeeks.org/numpy-concatenate-function-python) function with the axis parameter set to 0. Here’s how to do it with both methods:

**Method 1: Using np.vstack()**

array= np.vstack((array1, array2))

**Method 2: Using np.concatenate() with axis**

array= np.concatenate((array1, array2), axis=0)

**Q.17 What is the significance of the random module in NumPy?**

Here’s a brief explanation of the significance of the random module in Python:

1. Random Number Generation
2. Random Sequences
3. Probability Distributions
4. Random Choices

**Q.18 How can you generate random numbers following a normal distribution using NumPy?**

We can generate random numbers following a normal distribution (Gaussian distribution) using NumPy’s random module. NumPy provides several functions for this purpose, with different options for specifying the mean and standard deviation of the distribution. The most commonly used function is numpy.random.normal.

**Q.19 What is Matrix Inversion in NumPy?**

Matrix inversion in NumPy refers to the process of finding the inverse of a square matrix. The identity matrix is produced when multiplying the original matrix by the inverse of the matrix. In other words, if A is a square matrix and A^(-1) is its inverse, then A \* A^(-1) = I, where I is the identity matrix.

NumPy provides a convenient function called numpy.linalg.inv() to compute the inverse of a square matrix. Here’s how you can use it:

# Define a square matrix  
A = np.array([[a11, a12, ...],  
 [a21, a22, ...],  
 [..., ..., ...]])  
  
# Calculate the inverse of the matrix  
A\_inverse = np.linalg.inv(A)

**Q.20 Define the mean function in NumPy.**

The arithmetic mean (average) in NumPy can be calculated using numpy.mean(). This method tallies elements in an array, whether it be along a specified axis or the whole array, if no axis is explicitly mentioned. The summation of all elements is then divided by the overall number of elements, which provides the average.

numpy.mean(a, axis=None)

a: The input array for which you want to calculate the mean.

axis : The axis or axes along which the mean is computed. If not specified, the mean is calculated over the entire array.

**Q.20 Convert a multidimensional array to 1D array.**

You can convert a multidimensional array to a 1D array (also known as flattening the array) in NumPy using various methods. Two common methods are using for the Convert a multidimensional array to 1D array.

**Using flatten():**

# Create a multidimensional array  
multidimensional\_array = np.array([[1, 2, 3], [4, 5, 6], [7, 8, 9]])  
  
# Use the flatten() method to convert it to a 1D array  
one\_dimensional\_array = multidimensional\_array.flatten()

**Using ravel():**

# Create a multidimensional array  
multidimensional\_array = np.array([[1, 2, 3], [4, 5, 6], [7, 8, 9]])  
  
# Use the ravel() method to convert it to a 1D array  
one\_dimensional\_array = multidimensional\_array.ravel()

Both of these methods will flatten the multidimensional array into a 1D array. The primary difference between them is that flatten() returns a new copy of the array, while ravel() returns a flattened view of the original array whenever possible. If you modify the elements in the flattened view obtained from ravel().

**Q.21 Write a NumPy code snippet to create an array of zeros.**

With the help of numpy.zeros() function we can create an array of zeros in numpy. it help us to specify the shape and data type of the resulting array.

# Create a 1D array of zeros with a specified length (e.g., 5)  
zeros\_1d = np.zeros(5)

**Q.22 How can you identify outliers in a NumPy array?**

Identifying and removing outliers in a NumPy array involves several steps. Outliers are data points that significantly deviate from the majority of the data and can adversely affect the results of data analysis. Here’s a general approach to identify and remove outliers:

**Identifying Outliers:**

* **Calculate Descriptive Statistics:**Compute basic statistics like the mean and standard deviation of the array to understand the central tendency and spread of the data.
* **Define a Threshold:**
* Decide on a threshold where data points are considered outliers.

**Q.23 How do you remove missing or null values from a NumPy array?**

Removing missing or null values from NumPy array we can use boolean indexing or the [numpy.isnan()](https://www.geeksforgeeks.org/numpy-isnan-python) function to create a mask that identifies the missing values.

mask = np.isnan(my\_array)  
  
# Use the mask to filter out missing values  
filtered\_array = my\_array[mask]

**Q.24 What is the difference between slicing and indexing in NumPy?**

In NumPy, both slicing and indexing are fundamental operations for accessing and manipulating elements in arrays, but there are some main difference are avialable.

| **slicing** | **indexing** |
| --- | --- |
| Slicing is the process of extracting a portion or a subset of elements from a NumPy array by specifying a range or slice of indices. | Indexing refers to the process of accessing individual elements or subsets of elements from a NumPy array using specific indices or index values. |
| We can use slicing to create a new array that contains a contiguous subset of elements from the original array. | You can use indexing to retrieve a single element from an array or to access multiple elements at specific positions. |
| Slicing involves using a colon : inside square brackets to specify a range of indices. | Indexing typically involves using square brackets [] and providing one or more index values separated by commas (for multidimensional arrays). |

In summary, indexing is used to access individual or specific elements from an array, while slicing is used to create a new array that contains a contiguous subset of elements from the original array based on a specified range of indices.

**Q.25 How do you compute the Fourier transform of a signal using NumPy?**

To compute the Fourier transform of a signal using NumPy we will use numpy.fft module that provides functions for various types of Fourier transforms. for computing the Fourier transform of a signal we use the Fast Fourier Transform (FFT) algorithm. Here’s a step-by-step guide on how to do it.

t = np.linspace(0, 1, 1000, endpoint=False) # Time vector  
signal = np.sin(2 \* np.pi \* 5 \* t)   
  
# Compute the FFT of the signal  
fft\_result = np.fft.fft(signal)

**Q.26 How can you create array with same values.**

We can create a NumPy array with the same values using various functions and methods depending on your specific needs. Here are a few common approaches:

**Using numpy.full():**

You can use the [numpy.full()](https://www.geeksforgeeks.org/numpy-full-python" \t "_blank) function to create an array filled with a specific value. This function takes two arguments: the shape of the array and the fill value.

# Create a 1D array with 5 elements, all set to 7  
arr = np.full(5, 7)

**Using Broadcasting:**

If you want to create an array of the same value repeated multiple times, you can use broadcasting with NumPy.

# Create a 1D array with 5 elements, all set to 7  
arr = 7 \* np.ones(5)  
  
# Create a 2D array with dimensions 3x4, all elements set to 2.0  
arr\_2d = 2.0 \* np.ones((3, 4))

You can also create an array with the same values using a list comprehension and then converting it to a NumPy array.

# Create a 1D array with 5 elements, all set to 7  
arr = np.array([7] \* 5)  
  
# Create a 2D array with dimensions 3x4, all elements set to 2.0  
arr\_2d = np.array([[2.0] \* 4] \* 3)

**Q.27 How can you modify the data type of a NumPy array?**

We can modify the data type of a NumPy array using the astype() method or by directly assigning a new data type to the dtype attribute. Here’s how you can do it:

**Using the astype() method:**

You can use the astype() method to create a new array with the desired data type while leaving the original array unchanged.

new\_array = original\_array.astype(float)

**Directly assigning a new data type to the dtype attribute:**

You can also modify the data type of a NumPy array in place by assigning a new data type directly to the dtype attribute.

# Create a NumPy array with the original data type (e.g., int)  
original\_array = np.array([1, 2, 3, 4, 5])  
  
# Change the data type of the original array to float  
original\_array.dtype = float

**Q.28 What is a masked array in NumPy.**

A masked array in NumPy is a special type of array that includes an additional Boolean mask, which marks certain elements as invalid or masked. This allows you to work with data that has missing or invalid values without having to modify the original data. Masked arrays are particularly useful when dealing with real-world datasets that may have missing or unreliable data points.

**Q.29 What are some of the limitations of NumPy.**

NumPy is a powerful library for numerical and scientific computing in Python but it have some limitations depending on some requirements. Here are some of the limitations of NumPy.

1. Homogeneous Data Types
2. Memory Usage
3. Single-threaded
4. Limited Support for Missing Data
5. Limited Support for Labeling Data
6. Limited Support for Advanced Statistics
7. Performance Overheads for Small Arrays
8. Limited Support for GPU Acceleration
9. Complex Installation for Some Platforms
10. Limited Support for Distributed Computing

**Q.30 How do you sort a NumPy array in ascending or descending order?**

To arrange a NumPy array in both ascending and descending order we use numpy.sort() to create an ascending one and numpy.argsort() for a descending one. Here’s how to do it:

**Ascending Order:**

You can use the sort function that numpy offers to sort your array in ascending order. The function will return a new sorted array, while still leaving the original array unchanged.

# Create a NumPy array  
my\_array = np.array([3, 1, 2, 4, 5])  
  
# Sort the array in ascending order  
sorted\_array = np.sort(my\_array)

**Sorting in Descending Order:**

To sort a NumPy array in descending order, you can use the [numpy.argsort()](https://www.geeksforgeeks.org/numpy-argsort-in-python)function to obtain the indices that would sort the array in ascending order and then reverse those indices to sort in descending order.

# Create a NumPy array  
my\_array = np.array([3, 1, 2, 4, 5])  
  
# Get the indices to sort the array in ascending order  
ascending\_indices = np.argsort(my\_array)

**Q.31 How to use NumPy with Matplotlib?**

NumPy provides powerful array manipulation capabilities, and Matplotlib is a popular library for creating various types of plots and charts. Here’s how you can use NumPy with Matplotlib:

x = np.linspace(0, 2 \* np.pi, 100)   
y = np.sin(x)   
# Create a basic line plot  
plt.plot(x, y)

**Q.32 What is the use of diag() square matrix ?**

The diag() function in NumPy is used to extract or manipulate the diagonal elements of a square matrix (a matrix with the same number of rows and columns). It can serve various purposes in matrix operations and linear algebra. Here are some common uses of the diag() function:

**Q.33 How are NumPy Arrays better than Lists in Python?**

NumPy arrays offer several advantages over Python lists when it comes to numerical and scientific computing. Here are some key reasons why NumPy arrays are often preferred:

1. Performance
2. Vectorization
3. Broadcasting
4. Multidimensional Arrays
5. Memory Management
6. Standardization

**Q.34 what is negative indexing in NumPy arrays?**

Negative­ indexing in NumPy arrays allows individuals to access ele­ments from the end of an array by using ne­gative integers as indice­s. This feature offers a conve­nient way to retrieve­ elements re­lative to the array’s end, without the­ need for precise­ knowledge of its length. In NumPy, -1 corre­sponds to the last eleme­nt, -2 refers to the se­cond-to-last element, and so forth.

**Q.35 Can you create a plot in NumPy?**

Using NumPy and Matplotlib together, you can create a simple plot. NumPy is primarily a library for numerical computations with arrays, while Matplotlib is a popular Python library for creating various plots and charts. To create a plot, first import NumPy and Matplotlib, then use the functions from both libraries.

# Create any sample data using NumPy  
x = np.linspace(0, 2 \* np.pi, 100) #generate 100 points between 0 and 2\*pi  
y = np.sin(x) #compute the sine of each point  
plt.plot(x,y,label = 'Sine Wave') #plotting the sine wave   
plt.xlabel('X-axis')  
plt.ylabel('Y-axis')  
plt.legend()  
plt.grid(True)  
#show the plot   
plt.show()

**Q.36 Discuss uses of vstack() and hstack() functions?**

The vstack() and hstack() functions in NumPy are used to stack or concatenate arrays vertically and horizontally, respectively. These functions are essential for combining arrays in different dimensions and are widely used in various data processing and manipulation tasks. Here’s a discussion of their uses:

**vstack() (Vertical Stack):**

np.vstack() is used to vertically stack or concatenate arrays along the vertical axis (axis 0). This means that it stacks arrays on top of each other.

It is commonly used when you want to combine arrays with the same number of columns (i.e., the same number of features or variables) but different rows (i.e., different data points).

**hstack() (Horizontal Stack):**

np.hstack() is used to horizontally stack or concatenate arrays along the horizontal axis (axis 1). This means that it stacks arrays side by side.

It is commonly used when we want to combine arrays with the same number of rows.

**Q.37 How does NumPy handle numerical exceptions?**

NumPy handles numerical exceptions, such as overflow, underflow, and invalid mathematical operations, in a way that ensures stability and well-defined behavior in numerical computations.

NumPy’s handling of numerical exceptions is designed to provide predictable behavior in numerical computations while adhering to standard conventions and IEEE 754 rules. It allows users to work with exceptional cases and maintain control over error handling when needed.

**Q.38 What is the significance of the random module in NumPy?**

The random module in NumPy is a powerful and essential component for generating random numbers and performing various random processes in scientific computing, simulations, statistics, and machine learning. Here are some significant aspects of the random module in NumPy.

**Q.39 How to Get the eigen values of a matrix.**

With the help of [np.eigvals()](https://www.geeksforgeeks.org/python-numpy-np-eigvals-method)method, we can get the eigen values of a matrix by using np.eigvals() method.

np.eigvals(matrix)

**Q.40 How to Calculate the determinant of a matrix using NumPy?**

The Determinant of a square matrix is a unique number that can be derived from a square matrix. Using the numpy.linalg.det() method, NumPy gives us the ability to determine the determinant of a square matrix.

numpy.linalg.det(array)

**Q.41 Find a matrix or vector norm using NumPy.**

We employ the numpy.linalg.norm() method of the NumPy Python library to determine a matrix or vector norm. Depending on the value of its parameters, this function returns either one of the seven matrix norms or one of the infinite vector norms.

numpy.linalg.norm(x, ord=None, axis=None)

**Q.42 How to compare two NumPy arrays?**

Here, we’ll concentrate on the array comparison performed with NumPy. When two NumPy arrays are compared, the presence of the same element at each corresponding index indicates whether the arrays are comparable.

Method 1: We generally use the == operator to compare two NumPy arrays to generate a new array object. Call ndarray.all() with the new array object as ndarray to return True if the two NumPy arrays are equivalent.

**Method 2: Using array\_equal()**

This array\_equal() function checks if two arrays have the same elements and same shape.

numpy.array\_equal(arr1, arr2)

**Q.43 Calculate the QR decomposition of a given matrix using NumPy.**

A matrix’s decomposition into the form “A=QR,” where Q is an orthogonal matrix and R is an upper-triangular matrix, is known as QR factorization. With the aid of numpy, we can determine the QR decomposition of a given [matrix.linalg.qr()](https://www.geeksforgeeks.org/get-the-qr-factorization-of-a-given-numpy-array).

numpy.linalg.qr(a, mode=’reduced’)

a : matrix(M,N) which needs to be factored.

mode : it is optional.

**Q.44 How to filter out integers from float NumPy array.**

The aim is to remove integers from an array that contains float and integers using a numpy array. Let’s look at a couple approaches to completing a task.

* Using astype(int)
* Using np.equal() and np.mod()
* Using np.isclose()
* Using round()

**Q.45 Define a polynomial function.**

Using NumPy’s numpy.poly1d class, you can define polynomial functions. Using the coefficients of a polynomial, we can create a new function using this class. In NumPy, define a polynomial function as follows:

numpy.poly1d(arr, root, var)

**Q.46 What are ndarrays in NumPy?**

An ndarray (short for “N-dimensional array”) is a fundamental data structure used in NumPy (Numerical Python) for effectively storing and manipulating data, particularly numerical data. It offers a multidimensional, homogeneous array with a variety of capabilities for mathematical and array-oriented operations and is the fundamental building block of the NumPy library.

**Q.47 What are the main features that make NumPy unique.**

Due to a number of distinguishing characteristics, NumPy (Numerical Python) is a distinctive and crucial library in the Python ecosystem for numerical and scientific computing.

The ndarray (N-dimensional array), a very effective, homogeneous, and multi-dimensional data structure, is introduced by NumPy. An ndarray’s same data types enable for memory-efficient storage and mathematical operations that are optimised.

**Q.48 What is the difference between shape and size attributes of NumPy array.**

Shape and size attributes of an array give information about the dimension of the array but there are some differences.

| **shape** | **size** |
| --- | --- |
| The array’s dimensions are represented by a tuple that the shape attribute returns. The size of the array along each axis or dimension is represented by an individual member of the tuple. | The attribute ‘size’ gives the amount of elements in the array, regardless of its shape. It provides a single integer value representing the size of the flattened array. |
| It gives details about the array’s dimensions, including the number of rows, columns, and other dimensions. | It is the result of all the shape tuple’s components. |
| A single element serving as the array’s length makes up the form tuple for a 1D array. | It provides you with the array’s overall element count. |

**Q.49 What are some important differences between the standard Python sequences and NumPy arrays?**

The key distinctions between NumPy arrays and common Python sequences (such as lists and tuples) make NumPy arrays the best option for numerical and scientific computing. Here are some significant variations:

While NumPy arrays are homogeneous and require that every member have the same data type, Python sequences can hold items of various data types. In NumPy, this homogeneity enables effective storage and optimised operations.

**Q.50 What are Universal functions in NumPy?**

The core building block of [NumPy](https://www.geeksforgeeks.org/numpy-tutorial) (Numerical Python), universal functions, or “ufuncs,” allow for effective element-wise operations on ndarrays (N-dimensional arrays). Ufuncs in NumPy are functions that work on ndarray items, allowing you to efficiently and vectorizely carry out operations like exponentiation, trigonometry, and more.

**Q51 What are the difference between ndarray and array in NumPy.**

The terms “ndarray” and “array” in NumPy essentially refer to the same object: a multi-dimensional array object that makes it possible to effectively conduct mathematical operations on data. While “array” is a more casual term that is frequently used in everyday speech and documentation to refer to NumPy arrays, “ndarray” is the official term used within NumPy. Both terms refer to the same data structure that NumPy uses for effective numerical computations.

**ndarray**

* The official NumPy word for a multi-dimensional array is “ndarray”.
* In particular, it alludes to the NumPy array object.
* You generate “ndarray” objects when you use NumPy functions or constructors to create arrays, such as np.array(), np.zeros(), or np.ones().

**array**

* While it is not the official nomenclature, the word “array” is frequently used informally to refer to NumPy arrays.
* In reality, when people use the word “array” in relation to NumPy, they often mean “ndarray.”

**Q52 How would you convert a pandas dataframe into NumPy array.**

You can use the DataFrame’s.values attribute to convert a Pandas DataFrame into a NumPy array.

import pandas as pd  
import numpy as np  
  
# Create a Pandas DataFrame (replace this with your actual DataFrame)  
data = {'A': [1, 2, 3], 'B': [4, 5, 6]}  
df = pd.DataFrame(data)  
  
# Convert the DataFrame to a NumPy array  
numpy\_array = df.values

**Q53 Explain vectorization in NumPy.**

In NumPy, the term “[vectorization](https://www.geeksforgeeks.org/vectorization-in-python)” refers to the capacity to carry out element-wise operations on whole arrays or sizable sections of arrays without the use of explicit Python loops. You can perform a single action on the entire array or a sizable section of it all at once rather than iterating through each element in an array one at a time. This idea is comparable to how operations on vectors or matrices operate in linear algebra.

**Q54 How would you reverse a numpy array?**

We can reverse a NumPy array using the [::-1] slicing technique.

import numpy as np  
  
# Create a NumPy array (replace this with your array)  
original\_array = np.array([1, 2, 3, 4, 5])  
  
# Reverse the array  
reversed\_array = original\_array[::-1]  
  
print(reversed\_array)

**Q55 How do you remove missing or null values from a NumPy array?**

Unlike Pandas DataFrames, NumPy arrays do not by default support missing or null values. However, you may achieve a similar feature by filtering out missing or null data using a masked array or a boolean mask.

Using Masked array:

import numpy as np  
  
# Create a NumPy array with missing values (NaN)  
arr = np.array([1.0, 2.0, np.nan, 4.0, 5.0])  
  
# Create a masked array where missing values are masked  
masked\_arr = np.ma.masked\_invalid(arr)  
  
# Access only non-missing values  
clean\_data = masked\_arr.compressed()  
  
print(clean\_data)

**Q56 What is the difference between slicing and indexing in NumPy?**

**Indexing**

* In[NumPy,](https://www.geeksforgeeks.org/introduction-to-numpy) the term “indexing” refers to the process of choosing particular elements or groups of elements from an array according to their precise places, generally by using integers or boolean masks.
* You can use it to access and change individual items or particular elements at predetermined locations in the array.
* When you wish to locate and change or remove particular values in the array based on their positions, indexing might be handy for focused data cleaning operations.
* For instance, indexing can be used to set specific items to NaN (not-a-number) for missing data or to replace all negative values with zeros.

**Slicing**

* Using a range of indices or a boolean mask, slicing entails extracting a continuous portion of an array.
* It enables you to build an array from a portion of the original data, which is helpful for extracting particular rows or columns.
* When you want to filter and extract rows or columns that fit specific criteria or when you only want to work with a section of the data, slicing is useful for cleaning the data.
* Slicing, for instance, can be used to choose rows that meet particular criteria, such as all rows where values in a particular column are greater than a predetermined threshold.

**Q57 How do you create a masked array in NumPy, and what is its purpose?**

The[np.ma.masked\_array()](https://www.geeksforgeeks.org/how-to-mask-an-array-using-another-array-in-python) function in NumPy is used to build a masked array. It is useful when dealing with data when some numbers are regarded as “masked” or “invalid” and ought to be left out of calculations. When dealing with missing data or poor data quality, masked arrays are very helpful since they let you conduct operations while disregarding the masked values.

import numpy as np  
  
# Create a NumPy array with some missing or invalid values (e.g., -999)  
data = np.array([1, 2, -999, 4, -999, 6])  
  
# Create a masked array where -999 is considered masked (invalid)  
masked\_data = np.ma.masked\_array(data, mask=(data == -999))

In conclusion, a masked array in NumPy is a useful tool for working with data that contains blank or incorrect values. It enables you to perform operations on legitimate data while excluding masked numbers, guaranteeing that the data used in your calculations and analysis is accurate.

**Q58 What are some common techniques for normalizing data in a NumPy array?**

Data analysis and machine learning frequently utilize normalization, a preprocessing method, to scale data to a standard range, usually between 0 and 1, or with a mean of 0 and a standard deviation of 1. Many machine learning algorithms can perform better with the aid of normalized data, which also ensures that features with various scales contribute equally to the study.

Some common technique are:

* Min-Max Scaling
* Z-Score Normalization (Standardization)
* Log Transformation
* Box-Cox Transformation
* Robust Scaling

**Q59 How do you remove missing or null values from a NumPy array?**

Unlike Pandas DataFrames, NumPy arrays do not by default support missing or null values. However, you may deal with missing values in NumPy arrays by either filtering them out or by replacing them with suitable values.

Replacing Missing values:

import numpy as np  
  
# Create a NumPy array with missing values   
arr = np.array([1, 2, -999, 4, -999, 6])  
  
# Replace -999 with a specific value  
arr[arr == -999] = 0

**Q60 Create a two 2-D array. Plot it using matplotlib.**

import NumPy  
print(“Printing Original array”)  
sampleArray = NumPy.array([[34,43,73],[82,22,12],[53,94,66]])   
print (sampleArray)  
print(“Array after deleting column 2 on axis 1”)  
sampleArray = NumPy.delete(sampleArray , 1, axis = 1)   
print (sampleArray)  
arr = NumPy.array([[10,10,10]])  
print(“Array after inserting column 2 on axis 1”)  
sampleArray = NumPy.insert(sampleArray , 1, arr, axis = 1)   
print (sampleArray)

**Q61 What is difference between NumPy and Pandas?**

The main goal of NumPy is to enable huge, multi-dimensional arrays and matrices while also offering a range of mathematical operations that may be used effectively to these arrays. It serves as the basis for Python’s numerical computations.

On the other hand, Pandas is based on NumPy and intended for data analysis and manipulation. Series and DataFrame, two fundamental data structures that are effective for managing structured data types like spreadsheets and SQL tables, are introduced. With its high-level features, Pandas makes data cleaning, transformation, and analysis simple. It is also frequently used for data exploration and manipulation jobs.

In conclusion, NumPy and Pandas are both crucial Python tools for data analysis, but NumPy concentrates on numerical computing using arrays, while Pandas is an expert in data manipulation and analysis with

**Q62 Why NumPy is faster than list?**

The following justifies why Python Lists are slower than NumPy Arrays:

Due to its uniform data type storage, which enables best memory consumption and effective vectorized operations, NumPy is quicker than lists. Compared to Python’s interpreted code, NumPy offers faster execution because it is implemented in C. It makes use of parallel processing capabilities and well optimized numerical algorithms to speed up numerical computations. Its memory management guarantees improved cache locality and utilization. The combination of NumPy with compiled languages also improves performance for computationally demanding jobs.

**Q63 How do you check for an empty (zero Element) array?**

If the variable is an array, you can use the size attribute to determine whether the array is empty. The variable might, however, be of the list or sequence type in which case len() can be used. The size attribute is the best technique to look for a zero element.

>>> a = NumPy.zeros((1,0))  
>>> a.size  
0  
whereas  
>>> len(a)  
1

**Q64 What is the procedure to count the number of times a given value appears in an array of integers?**

The bincount() function can be used to count the instances of a given value. It should be noted that the bincount() function takes boolean expressions or positive integers as arguments. Integers that are negative cannot be used.

Implement NumPy.bincount().

arr = NumPy.array([0, 5, 4, 0, 4, 4, 3, 0, 0, 5, 2, 1, 1, 9])  
NumPy.bincount(arr)

**Q65 How can you sort an array in NumPy?**

The sort function in NumPy can be used to sort an array. This function arranges the members of an array in ascending order while altering the array already in use. As a result, it sorts the array itself rather than returning a new sorted array.

import numpy as np   
# Create an unsorted array   
arr = np.array([3, 2, 1])   
# Sort the array   
arr.sort()   
# Print the sorted array   
print(arr)

**Q66 How can you find the maximum or minimum value of an array in NumPy?**

Using the max and min functions, you may determine an array’s maximum or minimum value in NumPy. These operations accept an array as an input and output the array’s maximum or minimum value.

import numpy as np   
# Create an array   
arr = np.array([3, 2, 1])   
# Find the maximum value of the array   
max\_value = np.max(arr)   
# Find the minimum value of the array   
min\_value = np.min(arr)   
# Print the maximum and minimum values   
print(max\_value)   
print(min\_value)

**Q67 how slicing and indexing can be used for data cleaning?**

In this example, negative values are located and replaced with zeros using indexing, and a new array with more than two members is created using slicing. Both indexing and slicing are useful methods for cleaning data because they let you modify or filter data based on particular criteria or target particular data points for modification.

import numpy as np  
  
# Sample NumPy array  
data = np.array([1, 2, -1, 4, 5, -2, 7])  
  
# Indexing: Replace negative values with zeros  
data[data < 0] = 0  
  
# Slicing: Extract elements greater than 2  
subset = data[data > 2]

**Q68 What is the difference between using the shape and size attributes of a NumPy array?**

A tuple that describes the size of the array along each dimension is the shape attribute of a NumPy array. For instance, an array of the shape (3, 4) has 3 rows and 4 columns. The array’s dimensions can be found by using the form attribute, or the array can be reshaped by altering the sizes of the individual dimensions.

import numpy as np   
# Create a NumPy array   
arr = np.array([[1, 2, 3, 4], [5, 6, 7, 8], [9, 10, 11, 12]])   
# Get the shape of the array   
shape = arr.shape   
print(shape) # Output: (3, 4)   
# Access the individual dimensions   
num\_rows = shape[0]   
num\_cols = shape[1]   
print(num\_rows)   
print(num\_cols)

A NumPy array’s size attribute gives the total number of elements in the array. This is just the sum of the array’s various dimensions’ sizes. For instance, an array with the shape (3, 4) has 3 \* 4 = 12 elements overall.

import numpy as np   
# Create a NumPy array   
arr = np.array([[1, 2, 3, 4], [5, 6, 7, 8], [9, 10, 11, 12]])   
# Get the size of the array   
size = arr.size   
print(size) # Output: 12   
# Calculate the size manually   
num\_rows = arr.shape[0]   
num\_cols = arr.shape[1]   
size = num\_rows \* num\_cols   
print(size)

**Q69 What is a NumPy array and how is it different from a NumPy matrix?**

In a contiguous block of memory, NumPy arrays are data structures that hold values of the same data type. Similar to Python lists, they may hold values of any data type and are more effective for some tasks. Shape, size, and dtype are a few useful properties of umPy arrays. The array’s size along each dimension is specified by a tuple that the shape attribute returns. The array’s total number of elements is returned by the size attribute. The data type of each element in the array is returned by the dtype property.

* A additional data type called a matrix, a subclass of the array data type, is also included in NumPy. Similar to a NumPy array, a NumPy matrix includes a few more properties that make it more practical for linear algebra operations.
* Arrays employ the element-wise \* operator, whereas matrices contain a distinct \* operator for matrix multiplication.
* Additionally, matrices provide attributes for inverse and transpose, respectively.

**Q70 How can you find the unique elements in an array in NumPy?**

Apply the unique function from the NumPy module to identify the unique elements in an array in NumPy. This function returns the array’s unique elements in sorted order, along with a count of how many times each element occurred.

import numpy as np   
array = np.array([1, 2, 3, 1, 2, 3, 3, 4, 5, 6, 7, 5])   
unique, counts = np.unique(array, return\_counts=True)   
print(unique)   
print(counts)

The unique items of the input array array are contained in this example’s output arrays unique and counts, along with a count of each element.