Numpy

1. What is NumPy? Why should we use it?

1. NumPy, short for Numerical Python, is a versatile open-source package designed for efficient array processing.
2. It excels in handling powerful N-dimensional array objects, offering tools for high-performance computing.
3. NumPy is essential for scientific computations, mathematical and logical operations, sorting, I/O functions, and basic statistical and linear algebra tasks.
4. Additionally, it facilitates random simulation and broadcasting functionalities, enhancing its utility.
5. Thanks to its extensive capabilities, NumPy has garnered widespread popularity and is the preferred choice for many developers.

2. Why is NumPy preferred over Matlab, Octave, Idl or Yorick?

NumPy is an open-source, high-performance library for complex mathematical and scientific computations. Utilizes Python for ease and versatility.
Features:

- Powerful functions for fast, complex operations on multi-dimensional arrays.
- Efficient: Up to 50% faster than native lists with loops, crucial for large datasets.
- Simple indexing syntax for easy access to data subsets within arrays.
- Built-in functions for seamless linear algebra and statistical operations.
- Concise: Achieve complex computations in just a few lines of code with NumPy.

3. How are NumPy arrays better than Python's lists?

1. Store data of uniform type, enabling extra functional capabilities for easier operations.
2. **Memory efficiency:** Treated as objects with minimal memory wastage, like Python objects.
3. **Multi-dimensional support:** Easily handle arrays with multiple dimensions.
4. **Powerful functions:** Execute complex computations efficiently with NumPy's functions.
5. **Additional operations:** Access Bitwise, String, Linear Algebraic, and Arithmetic operations not available in Python lists.

4. What are ndarrays in NumPy?

ndarray object is the core of the NumPy package. It consists of n-dimensional arrays storing elements of the same data types and also has many operations that are done in compiled code for optimized performance. These arrays have fixed sizes defined at the time of creation. Following are some of the properties of ndarrays:

- When the size of ndarrays is changed, it results in a new array and the original array is deleted.
- The ndarrays are bound to store homogeneous data.
- They provide functions to perform advanced mathematical operations in an efficient manner.

5. How do you find the data type of the elements stored in the NumPy arrays?
You can find the data type of the elements stored in NumPy arrays using the '**.dtype**' attribute.
NumPy supports the following datatypes:

- i - integer
- S - string
- b - boolean
- f - float
- u - unsigned integer
- c - complex float
- m - timedelta
- M - datetime
- O - object
- U - unicode string
- V - fixed memory chunk for types such as void

6. How can you reverse a NumPy array?
There are two ways of reversing a NumPy array.

- Method 1: Using the **slicing method**: We can make use of **[::-1]** for reversing the array. The following example demonstrates this:
- Method 2: **flipud function**: This function is provided by NumPy to reverse the NumPy array. Let us see the below example about its usage. **np.flipud(arr)**

7. How is np.mean() different from np.average() in NumPy?

- **np.mean()** calculates the arithmetic mean with options for data types and result placement. It allows specifying data types and result placement.
- **'np.average()** computes the weighted average when given a weights parameter. Unlike simple mean, it considers unequal contributions of data points by assigning weights to them.

8. How do you count the frequency of a given positive value appearing in the NumPy array?
Use **'bincount()'** to count occurrences of values in an array. Accepts only positive integers and boolean expressions as arguments. **np.bincount(arr)**

9. How do we check for an empty array (or zero elements array)?
To check if a NumPy array is empty, use the **'size'** attribute. For instance, if 'size' returns zero for an array filled with zeros, it indicates the array is either empty or solely contains zeros. **arr.size**

10. How is arr[:,0] different from arr[:,[0]]
arr[:,0] retrieves the first column of the array 'arr', returning a 1-dimensional array.
arr[:,[0]] retrieves the first column of the array 'arr' as a 2-dimensional array, essentially preserving its column-like structure.

11. How do you multiply 2 NumPy array matrices?
We can make use of the dot() for multiplying matrices represented as NumPy arrays.
A.dot(B)

12. How do you concatenate 2 NumPy arrays?
Concatenating 2 arrays by adding elements to the end can be achieved by making use of the **concatenate()** method of the NumPy package.
Syntax: **np.concatenate((a1, a2, ...), axis=0, out=None)**

13. How do you convert Pandas DataFrame to a NumPy array?
The **to_numpy()** method of the NumPy package can be used to convert Pandas DataFrame, Index and Series objects.
np_arr = df.to_numpy()

14. What do you understand by Vectorization in NumPy?
Vectorization in NumPy refers to the ability to perform operations on entire arrays without the need for explicit looping over individual elements. It leverages optimized, compiled code under the hood to execute operations efficiently across array elements, leading to faster computations compared to traditional looping methods. This approach is a key feature of NumPy, enabling users to write concise and expressive code for numerical and scientific computing tasks.

15. How is vstack() different from hstack() in NumPy?
Both methods are used for **combining the NumPy arrays**. The main difference is that the hstack method combines arrays **horizontally** whereas the vstack method combines arrays **vertically**.

16. How is Vectorization related to Broadcasting in NumPy?

- **Vectorization** : Vectorization is a way NumPy speeds up Python by handling array operations in optimized C code, making it faster by processing whole arrays at once.
- **Broadcasting** : Broadcasting in NumPy lets you do math with arrays of different sizes or shapes. It makes smaller arrays bigger so they fit together, so you can do the math without errors.
- **Integration of Broadcasting and Vectorization** : Broadcasting precedes vectorization to align array shapes, facilitating vectorized operations across arrays of differing dimensions, optimizing computation efficiency.

17. What happens when the split() method is used for splitting NumPy arrays?
np.split() : Equally splits arrays into multiple sub-arrays. It raises Value Error when the split cannot be equal. **np.split(array, sections, axis=0)**

where,
array - array that needs to be split
sections -
If we give an integer X, X equal sub-arrays are obtained after dividing the array. If the split is not possible, ValueError is raised.

18. What happens when we use the arrays_split() method for splitting the NumPy array?
The **array_split()** method is similar to the **split()** method as it helps in splitting a given array to multiple subarrays. The main difference is that the array_split() allows sections to be an integer which does not result in equal array division. For an array of length L, if we want it to split to N subarrays, then L % N subarrays of size (L//N + 1) and remaining subarrays are of size L//N. **Syntax : np.array_split(array, sections, axis=0)**
For example if there are 5 elements in the array, we want to split the array to 3 subarrays. So L % N = 5%3 = 2 subarrays of size (L//N +1) = (5//3 +1) = 2 are returned and the remaining 1 subarray of size L//N = 1 is returned.

19. Explain convolve:
The **convolve()** method in NumPy performs convolution, a mathematical operation that combines two functions to produce a third function expressing how one function modifies the shape of the other. It slides a kernel over an input signal, multiplying and summing elements to compute convolution values. This operation is essential in signal and image processing for tasks like filtering and feature extraction.

20. How is fliplr different from flipud methods in NumPy?

In NumPy, 'fliplr()' and 'flipud()' are methods used to flip arrays along the horizontal and vertical axes, respectively.

- **'fliplr()':** This method flips an array ***horizontally***, reversing the order of elements along each row while keeping the same order of rows. It flips the array from left to right.
- **'flipud()':** On the other hand, 'flipud()' flips an array ***vertically***, reversing the order of rows while keeping the same order of elements within each row. It flips the array from top to bottom.

In summary, ***'fliplr()' flips an array horizontally, while 'flipud()' flips it vertically.***

Numpy 02

1. **What is NumPy?**
   NumPy is a Python library used for numerical computations, providing support for large, multi-dimensional arrays and matrices, along with a collection of mathematical functions to operate on these arrays efficiently.

2. **How do you import NumPy in Python?**
   NumPy can be imported in Python using the following statement: 'import numpy as np'

3. **What is the primary data structure in NumPy?**
   The primary data structure in NumPy is the **ndarray** (n-dimensional array).

4. **How do you create a NumPy array?**
   You can create a NumPy array using the **'numpy.array()**' function by passing a Python list as an argument.

5. **What is the difference between a Python list and a NumPy array?**
   NumPy arrays are homogeneous and have a fixed size at creation, whereas Python lists can contain elements of different data types and sizes.

6. **How do you create a 2D array in NumPy?**
   You can create a 2D array in NumPy by passing a list of lists to the 'numpy.array()' function.

7. **What is the shape of an array in NumPy?**
   The shape of an array in NumPy is a **tuple** that specifies the size of each dimension.

8. **How do you access elements of a NumPy array?**
   You can access elements of a NumPy array using **indexing** and **slicing**.

9. **How do you perform element-wise addition of two NumPy arrays?**
   You can perform element-wise addition of two NumPy arrays using the '**+**' operator.

10. **What is broadcasting in NumPy?**
    Broadcasting is a mechanism in NumPy that allows **arrays with different shapes** to be combined in **arithmetic operations**.

11. **How do you perform matrix multiplication in NumPy?**
    Matrix multiplication in NumPy can be performed using the '**numpy.dot()**' function or the '**@**' operator.

12. **What is the difference between 'numpy.dot()' and 'numpy.matmul()'?**
    'numpy.dot()' performs matrix multiplication for 2D arrays and dot product for 1D arrays, while **'numpy.matmul()'** is strictly used for matrix multiplication.

13. **How do you transpose a NumPy array?**
    You can transpose a NumPy array using the '**numpy.transpose()**' function or the '**.T**' attribute.

14. **What is the difference between shallow copy and deep copy in NumPy?**
    A shallow copy creates a new array object that still references the original data, while a deep copy creates a completely new array with its own data.

15. **How do you concatenate two NumPy arrays?**
    You can concatenate two NumPy arrays using the '**numpy.concatenate()**' function.

16. **What is the purpose of the 'numpy.reshape()' function?**
    The 'numpy.reshape()' function is used to change the shape of an array without changing its data.

17. **How do you find the maximum and minimum values of a NumPy array?**
    You can use the '**numpy.max()**' and '**numpy.min()**' functions to find the maximum and minimum values of a NumPy array, respectively.

18. **What is the purpose of the 'numpy.mean()' function?**

The 'numpy.mean()' function calculates the **arithmetic mean** of the elements in an array.

19. **How do you find the sum of all elements in a NumPy array?**
    You can use the '**numpy.sum()**' function to find the sum of all elements in a NumPy array.

20. **What is the purpose of the 'numpy.argsort()' function?**
    The '**numpy.argsort()**' function returns the indices that would sort an array.

21. **How do you find the unique elements in a NumPy array?**
    You can use the '**numpy.unique()**' function to find the unique elements in a NumPy array.

22. **What is the purpose of the 'numpy.where()' function?**
    The 'numpy.where()' function returns the **indices** of elements in an array where a specified condition is true.

23. **How do you calculate the dot product of two arrays in NumPy?**
    You can calculate the dot product of two arrays using the **'numpy.dot()**' function or the **'numpy.matmul()**' function.

24. **What is the purpose of the 'numpy.linalg' module?**
    The 'numpy.linalg' module provides linear algebra functions, such as **matrix inversion**, **eigenvalue decomposition**, and **matrix norms.**

25. **How do you calculate the eigenvalues and eigenvectors of a matrix in NumPy?**
    You can use the **'numpy.linalg.eig()**' function to calculate the eigenvalues and eigenvectors of a matrix.

26. **What is the purpose of the 'numpy.random' module?**
    The 'numpy.random' module provides functions for generating random numbers and random arrays.

27. **How do you generate random integers in a specified range in NumPy?**
    You can use the **'numpy.random.randint()**' function to generate random integers in a specified range.

28. **What is the purpose of the 'numpy.save()' and 'numpy.load()' functions?**

The 'numpy.save()' function is used to save arrays to disk in NumPy '.npy' format, while the 'numpy.load()' function is used to load arrays from '.npy' files.

29. **How do you perform element-wise multiplication of two arrays in NumPy?**
    You can perform element-wise multiplication of two arrays using the '*' operator.

30. **What is the purpose of the 'numpy.meshgrid()' function?**
    The 'numpy.meshgrid()' function is used to create a grid of coordinates based on the input arrays. It is often used for generating 2D plots and surfaces.

Numpy 03

1. **Explain the concept of array broadcasting in NumPy.**
   Array broadcasting in NumPy allows arrays with different shapes to be combined in arithmetic operations. It involves automatically aligning the dimensions of arrays so that they are compatible for element-wise operations.

2. **How do you create a diagonal matrix in NumPy?**
   You can create a diagonal matrix in NumPy using the '**numpy.diag()**' function by passing a 1D array as an argument, which will be placed along the diagonal of the matrix.

3. **What are the advantages of using NumPy arrays over Python lists for numerical computations?**
   NumPy arrays offer advantages such as faster computation speed, less memory consumption, and optimized mathematical operations compared to Python lists.

4. **How do you perform element-wise comparison between two NumPy arrays?**
   You can perform element-wise comparison between two NumPy arrays using comparison operators like **'=='**, **'!='**, **'<'**, **'>'**, **'<='**, and **'>='**.

5. **Explain the difference between 'numpy.zeros()' and 'numpy.empty()' functions.**
   'numpy.zeros()' creates an array filled with zeros, while 'numpy.empty()' creates an array with uninitialized (random) values, which may vary between executions.

6. **How do you create a NumPy array with evenly spaced values over a specified interval?**

You can use the '**numpy.linspace()**' function to create an array with evenly spaced values over a specified interval.

7. **Explain the concept of array slicing in NumPy.**
   Array slicing in NumPy allows you to extract a portion of an array by specifying a range of indices along each dimension.

8. **How do you concatenate two 1D arrays along a specified axis in NumPy?**
   You can use the '**numpy.concatenate()**' function with the '**axis'** parameter to concatenate two 1D arrays along a specified axis.

9. **What is the purpose of the 'numpy.newaxis' attribute?**
   The 'numpy.newaxis' attribute is used to increase the dimension of an array by one, effectively adding a new axis at the specified position.

10. **How do you perform element-wise multiplication of two arrays with different shapes in NumPy?**
    NumPy automatically broadcasts arrays with compatible shapes during arithmetic operations, so you can simply use the **'*'** operator for element-wise multiplication.

11. **Explain the difference between 'numpy.mean()' and 'numpy.average()' functions.**
    'numpy.mean()' calculates the **arithmetic mean of an array**, while 'numpy.average()' allows you to **specify weights for the elements**.

12. **How do you find the indices of non-zero elements in a NumPy array?**
    You can use the '**numpy.nonzero()**' function to find the indices of non-zero elements in a NumPy array.

13. **Explain the purpose of the 'numpy.ma' module.**
    The 'numpy.ma' module provides support for **masked arrays**, allowing you to ***handle data with missing or invalid values***.

14. **How do you perform matrix multiplication of two 2D arrays in NumPy?**
    You can use the '**numpy.matmul()**' function or the '**@**' operator for matrix multiplication of two 2D arrays.

15. **What is the purpose of the 'numpy.fromfunction()' function?**
    The 'numpy.fromfunction()' function allows you to create an array based on a specified function that operates on array indices.

16. **How do you perform element-wise exponentiation of an array in NumPy?**
You can use the 'numpy.exp()' function to perform element-wise exponentiation of an array.

17. **Explain the concept of array masking in NumPy.**
Array masking in NumPy involves creating a boolean mask array based on a specified condition, which can then be used to select elements from another array.

18. **How do you calculate the inverse of a matrix in NumPy?**
You can use the '**numpy.linalg.inv()**' function to calculate the inverse of a matrix.

19. **Explain the purpose of the 'numpy.savez()' function.**
The **'numpy.savez()'** function is used to save multiple NumPy arrays into a single compressed **'.npz'** archive file.

20. **How do you perform element-wise logarithm of an array in NumPy?**
You can use the '**numpy.log()**' function to perform element-wise natural logarithm of an array.

21. **Explain the purpose of the 'numpy.fft' module.**
The 'numpy.fft' module provides functions for computing the **Discrete Fourier Transform (DFT)** and its **inverse**, along with other related operations.

22. **How do you find the indices of maximum and minimum elements in a NumPy array?**
You can use the **'numpy.argmax()**' and **'numpy.argmin()'** functions to find the indices of maximum and minimum elements, respectively, in a NumPy array.

23. **What is the purpose of the 'numpy.ma.masked_where()' function?**
The 'numpy.ma.masked_where()' function **creates a masked array where elements satisfy a specified condition**.

24. **How do you calculate the cross product of two arrays in NumPy?**
You can use the 'numpy.cross()' function to calculate the cross product of two arrays.

25. **Explain the concept of universal functions (ufuncs) in NumPy.**

Universal functions (ufuncs) in NumPy are functions that operate element-wise on arrays, allowing for efficient computation across entire arrays.

26. **How do you calculate the Kronecker product of two arrays in NumPy?**
    You can use the **'numpy.kron()**' function to calculate the Kronecker product of two arrays.

27. **What is the purpose of the 'numpy.ma.masked_array()' function?**
    The 'numpy.ma.masked_array()' function creates **a masked array from an existing array**, where elements can be masked based on a specified condition.

28. **How do you calculate the percentile of elements in a NumPy array?**
    You can use the '**numpy.percentile()**' function to calculate the percentile of elements in a NumPy array.

29. **Explain the difference between 'numpy.apply_along_axis()' and 'numpy.vectorize()' functions.**
    '**numpy.apply_along_axis()**' applies a function along a specified axis of an array, while '**numpy.vectorize()**' converts a scalar function into a vectorized function that can operate on arrays element-wise.

30. **How do you calculate the singular value decomposition (SVD) of a matrix in NumPy?**
    You can use the 'numpy.linalg.svd()' function to calculate the singular value decomposition (SVD) of a matrix.