Q1. What are the benefits of the built-in array package, if any?

Ans1- The built-in array package in Python provides a basic way to create and manipulate arrays, which are collections of elements of the same data type. Benefits of using the array package include efficient memory usage and improved performance compared to lists, especially when dealing with large datasets.

Q2. What are some of the array package's limitations?

Ans2- he array package has some limitations. It only supports homogeneous data types, meaning all elements in an array must be of the same type. Additionally, its functionality is relatively limited compared to more advanced libraries like NumPy, which offer a broader range of mathematical and array manipulation capabilities.

Q3. Describe the main differences between the array and numpy packages.

Ans3- The main differences between the array package and the NumPy package in Python include:

Data Types: Arrays in the array package are limited to a few basic data types, whereas NumPy supports a wide range of numerical data types.

Functionality: NumPy provides extensive functionality for mathematical and array operations, including broadcasting, linear algebra, and statistical functions, which the array package lacks.

Performance: NumPy is highly optimized and typically faster for array operations, making it the preferred choice for numerical computing tasks.

Q4. Explain the distinctions between the empty, ones, and zeros functions.

Ans4- In NumPy, the functions empty, ones, and zeros are used to create arrays:

np.empty(shape): Creates an uninitialized array with the specified shape, filled with arbitrary values.

np.ones(shape): Generates an array filled with ones of the specified shape.

np.zeros(shape): Creates an array filled with zeros of the specified shape

Q5. In the fromfunction function, which is used to construct new arrays, what is the role of the callable argument?

Ans5- In the np.fromfunction(function, shape) function, the function argument should be a callable (like a Python function or a lambda function) that defines the values for each element of the resulting array based on its indices. It is used to construct new arrays where the value of each element is determined by the provided function and the element's position in the array.

Q6. What happens when a numpy array is combined with a single-value operand (a scalar, such as an int or a floating-point value) through addition, as in the expression A + n?

Ans6- When a NumPy array A is combined with a single-value operand n (a scalar) through addition (A + n), NumPy performs element-wise addition, adding the scalar n to each element of the array A.

Q7. Can array-to-scalar operations use combined operation-assign operators (such as += or \*=)? What is the outcome?

Ans7- Array-to-scalar operations can use combined operation-assign operators like += or \*=. These operations modify the original array in place, applying the operation to each element. For example, A += n adds n to each element of array A.

Q8. Does a numpy array contain fixed-length strings? What happens if you allocate a longer string to one of these arrays?

Ans8- NumPy arrays can contain fixed-length strings. If you attempt to assign a longer string to an element in such an array, it will be truncated to fit the specified length. If the assigned string is shorter, it may be padded with null bytes to maintain the fixed length.

Q9. What happens when you combine two numpy arrays using an operation like addition (+) or multiplication (\*)? What are the conditions for combining two numpy arrays?

Ans9- When you combine two NumPy arrays using operations like addition (+) or multiplication (\*), NumPy performs element-wise operations if the arrays have compatible shapes. Compatible shapes typically means the arrays have the same dimensions or can be broadcasted to the same shape. If the shapes are not compatible, NumPy will raise a ValueError.

Q10. What is the best way to use a Boolean array to mask another array?

Ans10- You can use a Boolean array to mask another array by applying the Boolean array as an index. For example, if you have a Boolean array mask and an array data, you can retrieve the elements of data where mask is True using data[mask].

Q11. What are three different ways to get the standard deviation of a wide collection of data using both standard Python and its packages? Sort the three of them by how quickly they execute.

Ans11- Three different ways to calculate the standard deviation of data in Python and their approximate execution speed from fastest to slowest are:

NumPy: Using np.std(data), which is the fastest for large datasets.

Standard Python (math module): Using a custom function with a for loop.

Standard Python (statistics module): Using statistics.stdev(data), which is the slowest for large datasets.

12. What is the dimensionality of a Boolean mask-generated array?

Ans12- The dimensionality of a Boolean mask-generated array matches the dimensionality of the original array. If you apply a Boolean mask to a 1D array, you get another 1D array with the same length, and the same applies to 2D, 3D, and higher-dimensional arrays. The Boolean mask determines which elements are selected from the original array, but the resulting array maintains the same dimensionality.