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

- Answer: The built-in array package provides a memory-efficient way to store and manipulate homogeneous data elements in a fixed-size array, similar to lists but with a more compact memory representation.

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

- Answer: The array package is limited to one-dimensional arrays, lacks many of the functionalities offered by more advanced data structures like NumPy arrays, and doesn't support complex data types or operations.

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

- Answer: NumPy provides a more powerful and flexible array implementation compared to the built-in array package. NumPy arrays can have multiple dimensions, support a wide range of data types, and offer a rich set of mathematical and manipulation functions.

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

- Answer:

- `empty(shape)`: Creates an uninitialized array with the specified shape.

- `ones(shape)`: Creates an array filled with ones with the specified shape.

- `zeros(shape)`: Creates an array filled with zeros with the specified shape.

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

- Answer: The callable argument in `fromfunction` is a function that defines the values of the array based on the coordinates of each element.

Q6. What happens when a NumPy array is combined with a single-value operand (a scalar) through addition, as in the expression A + n?

- Answer: The scalar is broadcasted to the shape of the array, and element-wise addition is performed between the array and the scalar.

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

- Answer: Yes, NumPy allows combined operation-assign operators with scalars. For example, `A += n` will add the scalar `n` to all elements in array `A`.

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

- Answer: NumPy arrays can contain fixed-length strings, and they enforce the specified string length. If you allocate a longer string, it may be truncated or result in an error, depending on the context.

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?

- Answer: NumPy combines arrays element-wise when using operations like addition or multiplication. The arrays must have compatible shapes (usually the same shape or broadcastable shapes) for these operations.

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

- Answer: You can use Boolean indexing to mask another array. For example, if you have a Boolean array `mask` and an array `data`, you can obtain 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.

- Answer: Three ways include:

1. Using NumPy's `numpy.std` function for efficient array-based computation.

2. Using Python's `statistics.stdev` for standard Python lists.

3. Manually calculating the standard deviation using the mathematical formula. Sorting by execution speed, NumPy is typically the fastest, followed by the Python statistics module, and manual calculation is usually the slowest.

Q12. What is the dimensionality of a Boolean mask-generated array?

- Answer: The dimensionality of a Boolean mask-generated array depends on the shape of the mask. It typically has the same dimensionality as the mask, resulting in a subset of the original array with elements selected based on the True/False values in the mask.