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

**Ans:**

-In an array, accessing an element is very easy by using the index number.

-The search process can be applied to an array easily.

-2D Array is used to represent matrices.

-For any reason a user wishes to store multiple values of similar type then the Array can be used and utilized efficiently.

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

**Ans:**

-Array size is fixed.

-Array is homogeneous.

-Array is contiguous blocks of memory

-Insertion and deletion is not easy in array

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

**Ans:** If creating arrays of simple data types and doing I/O, the array module will do just fine.

If, on the other hand, you want to do any kind of numerical calculations, the array module doesn't provide any help with that. NumPy (and SciPy) give you a wide variety of operations between arrays and special functions that are useful not only for scientific work but for things like advanced image manipulation or in general anything where you need to perform efficient calculations with large amounts of data.

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

**Ans:**

**np.zeros**

Return a new array setting values to zero.

>>> np.zeros((2, 2))

array([[0., 0.],

[0., 0.]])

**np.empty**

Return a new uninitialized array.

>>> np.empty((2, 2))

array([[1.35807735e-312, 1.35807731e-312],

[1.99637364e-310, 8.69169476e-311]])

**np.ones**

Return a new array setting values to ones.

>>> np.empty((2, 2))

array([[1., 1.],

[1., 1.]])

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

**Ans:** Callable argument means, the function is called with N parameters, where N is the rank of shape. Each parameter represents the coordinates of the array varying along a specific axis. For example, if shape were (2, 2), then the parameters would be array([[0, 0], [1, 1]]) and array([[0, 1], [0, 1]])

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?

**Ans:** It will add n to every element of array.

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

**Ans:** it will throw the syntax error.

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

**Ans:** Numpy requires string arrays to have a fixed maximum length. When you create an empty array with dtype=str, it sets this maximum length to 1 by default. You can see if you do my\_array.dtype; it will show "|S1", meaning "one-character string". Subsequent assignments into the array are truncated to fit this structure.

The "S10" will create an array of length-10 strings. You have to decide how big will be big enough to hold all the data you want to hold.

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?

**Ans:** Shape of both of the arrays should be same.

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

**Ans:** We can understand this with an example:

x = np.array([1,3,-1, 5, 7, -1])

mask = (x < 0)

mask

array([False, False, True, False, False, True], dtype=bool)

here mask becomes a Boolean array with the condition mentioned. When we apply this Boolean array to another array, it will return those elements only which passes this criteria.

x [mask]

x

array([1, 3, 0, 5, 7, 0])

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.

**Ans:** These are three ways already in sorted order, to get standard deviation in python and its packages:

1. import the NumPy library with import numpy as np and use the np.std(list) function.
2. Import the statistics library with import statistics and call statistics.stdev(list) to obtain a slightly different result because it’s normalized with (n-1) rather than n for n list elements – this is called Bessel’s correction.
3. Without External Dependency: Calculate the average as sum(list)/len(list) and then calculate the variance in a list comprehension statement, and then calculate standard deviation.

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

**Ans:** one dimension.