ASSIGNMENT - 22

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

Ans: The built-in array package in Python offers arrays that are more memory-efficient than lists when dealing with large homogeneous numeric data sets, as they store a single data type.

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

Ans: Arrays from the built-in package have limited functionality compared to numpy arrays, lacking many of the advanced mathematical operations and functionalities available in numpy.

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

Ans: Differences between array and numpy packages:

* array: Basic support for arrays of a single data type, limited functionalities.
* numpy: Provides multidimensional arrays, extensive mathematical operations, broadcasting, linear algebra, and other powerful array operations.

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

Ans: Distinctions between empty, ones, and zeros functions:

* empty: Creates an array without initializing its values, which can contain random or garbage values.
* ones: Generates an array filled with ones.
* zeros: Produces an array filled with zeros.

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

Ans: The callable argument in numpy.fromfunction() generates array values based on the coordinates of the elements in the output array. It defines a function that takes the coordinates as input and returns the value for that specific element.

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: When a numpy array is combined with a single-value operand through addition (or other arithmetic operations), the operation is performed element-wise between the array and the scalar.

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

Ans: Yes, numpy arrays support combined operation-assign operators like += or \*=. These operations modify the original array in place by applying the operation to each element of the array.

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

Ans: Yes, numpy arrays can contain fixed-length strings. If you allocate a longer string to an array that expects fixed-length strings, the longer string will be truncated to fit the defined 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?

Ans: When combining two numpy arrays using arithmetic operations like addition or multiplication (+, \*), the arrays must have compatible shapes (broadcastable) for the operation to succeed. The operation is performed element-wise.

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

Ans: A Boolean array can be used as a mask to select elements from another array by indexing the array with the Boolean mask. For example, masked\_array = array[mask] selects elements where the mask is True.

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: Three ways to get standard deviation:

* Using standard Python: statistics.stdev(data)
* Using NumPy: numpy.std(data)
* Using Pandas: pandas.Series(data).std()
* Execution speed: numpy.std(data) in NumPy is usually the fastest for larger datasets due to its optimized implementations.

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

Ans: The dimensionality of a Boolean mask-generated array is the same as the dimensionality of the original array. It will have the same number of dimensions but only contain the elements where the mask is True.