array vs NumPy, Creation, Broadcasting, Masking & Stats — Q&A; (Set 22)

Q1. Benefits of the built■in array package

array.array provides compact, memory■efficient, C■contiguous storage for a single primitive type (ints, floats, etc.), supports fast binary I/O (frombytes/tobytes), and avoids per■element Python object overhead.

Q2. Limitations of the array package

Single fixed typecode; mostly 1■D; few numerical operations (no ufuncs/broadcasting); no views/strides; much smaller API than NumPy.

Q3. Main differences: array vs numpy

array: stdlib, 1■D homogeneous buffers, minimal math.
NumPy: external package, N■D arrays, rich dtypes (float, int, bool, datetime, complex, fixed■len strings, object), vectorized ufuncs, broadcasting, views/strides, BLAS/LAPACK integration.

Q4. Distinctions between empty, ones, zeros (NumPy)

np.empty(shape): allocates uninitialized memory (contents arbitrary, fastest).
np.ones(shape): allocates and fills with 1s.
np.zeros(shape): allocates and fills with 0s. Use empty when you will overwrite every element.

Q5. Role of the callable in np.fromfunction

The callable f is invoked with index grids (arrays of coordinates for each axis); its return value populates the array. Example: np.fromfunction(lambda i,j: i+j, (3,3), dtype=int).

Q6. Array + scalar (A + n)

NumPy broadcasts the scalar across all elements and returns an elementwise sum array; dtype promotion rules apply.

*Q7. Combined op■assign with scalars (+=, *=)*

Yes. A += n or A *= n performs in■place broadcasting where possible (mutates A). If n cannot be cast safely to A.dtype with the current casting rule, NumPy may raise an error.

Q8. Fixed■length strings in NumPy

Yes (dtype='Sk' for bytes or 'Uk' for Unicode, fixed length k). Assigning a longer string is truncated to fit. Object dtype ('O') stores arbitrary Python strings without truncation.

*Q9. Combining two NumPy arrays with + or *; conditions*

Operations are elementwise with broadcasting. Shapes must be equal or broadcast■compatible: for each trailing dimension, sizes must match or one of them must be 1.

Q10. Best way to use a Boolean array to mask another array

Use boolean indexing: B = A[mask] to select elements; A[mask] = value to assign. mask must be same shape as A (or match along the indexed axis).

Q11. Three ways to get standard deviation; fastest → slowest

1) NumPy: np.std(arr) (C■level vectorized)
2) statistics module: statistics.pstdev(arr) / statistics.stdev(arr) (pure Python but optimized with math.fsum)
3) Manual Python loop or two■pass with sum/math.fsum.
Pandas Series.std() typically wraps NumPy and is comparable to #1.

Q12. Dimensionality of a Boolean mask■generated array

Using A[mask] where mask has the same shape returns a 1■D array of the selected elements. If you mask along an axis (e.g., A[mask, :]), that axis is reduced; remaining axes are preserved.