1. Why is NumPy Faster than Python Lists?

Answer:

NumPy is faster because:

1. **C-Based Implementation**
   NumPy is written in C, while Python lists are interpreted.
2. **Homogeneous Data**
   NumPy stores same-type elements → efficient memory usage.
   Lists can store mixed data types.
3. **Continuous Memory Allocation**
   NumPy stores data in blocks → faster access.
4. **Vectorization**
   Operations run at once (no loops).

Example:

```
import numpy as np

a = np.array([1,2,3,4])
print(a + 5)
```

Much faster than loop in lists.

2. What is Broadcasting?

Answer:

Broadcasting allows NumPy to perform operations on arrays of different shapes.

Rules:

1. Dimensions must match OR
2. One dimension must be 1

Example:

```
import numpy as np

a = np.array([1,2,3])
b = 10

print(a + b)
```

Output:

```
[11 12 13]
```

3. Features of NumPy

Answer:

- Multidimensional arrays
- Fast computation
- Broadcasting

- Vectorization
- Linear algebra support
- Random number generation
- Memory efficient

4. Advantages of NumPy

Answer:

- High speed
- Less memory
- Easy syntax
- Scientific computing
- ML/AI support
- Works with Pandas, TensorFlow

5. What is Vectorization? Why Important?

Answer:

Vectorization = Performing operations without loops.

Example:

```
a = np.array([1,2,3])
b = np.array([4,5,6])

print(a + b)
```

Importance:

- Faster
- Cleaner code
- Less errors

6. NumPy in Machine Learning

Answer:

NumPy helps ML by:

- Storing datasets
- Matrix operations
- Gradient calculations
- Feature scaling
- Model training

Libraries like:

- Scikit-learn
- TensorFlow
- PyTorch

are built on NumPy.

7. Advantages in Industry

Answer:

- Big data handling
- Financial analysis
- Image processing
- AI models
- Engineering simulation

Used in:
Google, Amazon, Tesla, NASA

8. Creating 1D and Multi-Dimensional Arrays

1D Array:

```
a = np.array([1,2,3])
```

2D Array:

```
b = np.array([[1,2],[3,4]])
```

3D Array:

```
c = np.array([[[1,2],[3,4]]])
```

9. Properties of NumPy

Example:

```
a = np.array([[1,2,3],[4,5,6]])
```

Property	Meaning
ndim	Number of dimensions
shape	Rows, Columns
size	Total elements
dtype	Data type
ndmin	Minimum dimension

Code:

```
print(a.ndim)
print(a.shape)
print(a.size)
print(a.dtype)
```

10. Difference Between ndim and ndmin

ndim	ndmin
Shows dimension	Sets dimension
Read-only	Used while creating

Example:

```
a = np.array([1,2,3], ndmin=3)
print(a.ndim)
```

11. Indexing and Slicing

1D Indexing:

```
a = np.array([10,20,30,40])
print(a[1])   # 20
```

1D Slicing:

```
print(a[1:3])
```

2D Indexing:

```
b = np.array([[1,2,3],[4,5,6]])

print(b[0,1])   # 2
```

2D Slicing:

```
print(b[:,1])
```

12. Linear Algebra Without NumPy

Answer:

Without NumPy:

- Matrix multiplication = slow loops
- High error chances
- Memory waste
- Slow ML training

With NumPy:

```
np.dot(A,B)
np.linalg.inv(A)
```

Makes AI/ML possible.

13. Statistical & Aggregate Functions

Examples:

```
a = np.array([1,2,3,4,5])

np.mean(a)
np.median(a)
np.sum(a)
np.min(a)
np.max(a)
np.std(a)
np.var(a)
```

14. What is type() Function?

Answer:

It checks data type.

```
a = np.array([1,2,3])
print(type(a))
```

Output:

```
<class 'numpy.ndarray'>
```

15. Type Casting in NumPy

Yes, using `astype()`.

```
a = np.array([1,2,3])
b = a.astype(float)
```

16. Supported Data Types

Type	Meaning
int32	Integer
float64	Decimal
complex	Complex
bool	Boolean
str	String

Example:

```
np.array([1,2,3], dtype='float')
```

17. Memory Management in NumPy

Answer:

- Uses contiguous memory
- Fixed-size elements
- Low overhead
- Uses views instead of copies

Example:

```
b = a[1:3]   # View, no new memory
```

18. Importance of NumPy (Use Cases)

Use Cases:

- Face recognition

- Stock prediction
- Weather forecasting
- Medical imaging
- Recommendation systems

19. What is Random in NumPy?

Answer:

It generates random numbers.

```
np.random.rand()
```

20. Types of Random in NumPy

Function	Use
rand()	0 to 1
randint()	Integers
randn()	Normal distribution
choice()	Random selection
shuffle()	Shuffle

Examples:

```
np.random.rand(3)
np.random.randint(1,10)
np.random.choice([1,2,3,4])
```