NumPy Notes

1. Difference Between Python List and NumPy Array

Feature	List	NumPy Array
Type	Heterogeneous	Homogeneous
Speed	Slow	Fast (vectorized operations)
Memory Efficiency	Low	High
Operations	Manual loops needed	Vectorized operations (automatic)
Data Type Conversion	If mixed types, elements are converted to a common type	Automatically converts to a compatible type (e.g., `[1,2,"hello"]` → `["1","2","hello"]`)

---

2. Array Creation

2.1 Basic Array

```
import numpy as np

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

2.2 Special Arrays

```
np.zeros(6)          # Array of 6 zeros
np.zeros((4,8))      # 4x8 matrix of zeros
np.ones((2,2))       # 2x2 matrix of ones
np.eye(3)            # 3x3 Identity matrix
```

2.3 Arrays with Range

```
np.arange(1, 19)         # 1 to 18
np.arange(1, 10, 2)      # 1 to 9, step of 2
```

```
np.linspace(1,5,3)       # 3 elements evenly spaced from 1 to 5 → [1,3,5]
```

2.4 Random Arrays

```
np.random.rand(5)            # 5 random numbers [0,1)
np.random.randn(5)           # 5 random numbers ~ normal dist. (-3,3 approx)
np.random.randint(1,10,5)    # 5 random integers from 1 to 9
```

---

3. Array Attributes

```
arr = np.array([[1,2,3],[2,3,4]])
arr.shape    # (2,3) → rows, columns
arr.size     # Total elements → 6
arr.dtype    # Data type of elements
arr.ndim     # Number of dimensions → 2
```

---

4. Array Methods

```
arr.min()       # Minimum value
arr.max()       # Maximum value
arr.sum()       # Sum all elements
arr.sum(axis=0) # Column sum
arr.sum(axis=1) # Row sum
arr.mean()      # Mean
arr.std()       # Standard deviation
arr.argmax()    # Index of maximum element
```

---

5. Reshaping and Resizing

```
arr.reshape(4,4)     # Change shape to 4x4
arr.flatten()        # Flatten to 1D array
arr.ravel()          # Similar to flatten (shallow copy)
```

---

6. Indexing and Slicing

6.1 1D Array

```
arr = np.arange(1,19)
arr[6]        # Indexing
arr[2:5]      # Slice from 2 to 4
arr[1:]       # Slice from index 1 to end
arr[:5]       # Slice from start to 4
arr[3::2]     # Slice from 3 to end, step 2
```

6.2 2D Array

```
arr = np.arange(1,17).reshape(4,4)
arr[0,0]          # Access element (row 0, col 0)
arr[0:2,1:3]      # 2x2 slice (rows 0-1, cols 1-2)
arr[3:,3:]        # Last row and column
arr[:,1:2]        # All rows, column 1
```

---

7. Boolean Indexing

```
arr = np.arange(1,11)
bool_index = arr % 2 == 0
arr[bool_index]   # Returns only even numbers
```

---

8. Arithmetic Operations

```
A1 = np.arange(1,10)
A2 = np.arange(11,20)
A1 + A2
A1 * A2
A1 / A2
A1 // A2
```

```
A1 ** A2

# Broadcasting (array + scalar)
arr + 10
arr * 5
```

Broadcasting Rules:

1. If dimensions differ, prepend 1 to the smaller shape.
2. Dimensions are compatible if they are equal or one of them is 1.

---

9. Copying Arrays

```
# Shallow copy (view)
slice = arr[:5]
slice *= 10        # Changes reflected in original array

# Deep copy
copy_arr = arr.copy()
copy_arr *= 10     # Original array unaffected
```

---

10. Matrix Operations

```
A @ B          # Matrix multiplication
np.dot(A,B)    # Matrix multiplication
A.T            # Transpose
```

10.1 Advanced Array Stacking

```
np.hstack((a,b))    # Horizontal stack (side by side)
np.vstack((a,b))    # Vertical stack (top and bottom)

np.hsplit(c,2)      # Split into 2 horizontal parts (columns)
np.vsplit(c,2)      # Split into 2 vertical parts (rows)
```

---

11. Additional Notes / Important Tips

- `arr.dtype` can be used to explicitly convert types: `arr.astype(np.float32)`
- Use `np.where(condition)` to get indices satisfying a condition
- `np.unique(arr)` → Returns unique elements
- `np.sort(arr)` → Sort elements
- `np.concatenate((a,b))` → Generic stacking
- Use `np.random.seed(0)` for reproducible random numbers