NumPy Topics Required for Data Analyst Freshers

1. Introduction to NumPy

What it is:
NumPy (Numerical Python) is a Python library used for **working with arrays**. It provides support for large multi-dimensional arrays and many mathematical functions.

Why it is needed:
As a Data Analyst, you deal with large datasets. NumPy helps you manipulate and process this data efficiently.

Example:

```
import numpy as np

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

print(arr)

# Output: [1 2 3]
```

---

2. Creating Arrays

What it is:
Arrays are like lists, but more powerful. You can create arrays in different ways.

Methods:

- np.array()
- np.zeros()
- np.ones()
- np.arange()
- np.linspace()

Example:

```
a = np.zeros(3)

b = np.ones(3)

c = np.arange(1, 10, 2)

d = np.linspace(1, 10, 4)

print(a)  # [0. 0. 0.]

print(b)  # [1. 1. 1.]

print(c)  # [1 3 5 7 9]

print(d)  # [ 1.  4.  7. 10.]
```

---

3. Array Attributes

What it is:
Learn how to check shape, size, dimensions of an array.

Important attributes:

- .shape
- .size
- .ndim
- .dtype

Example:

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

print(arr.shape)  # (2, 3)

print(arr.size)   # 6

print(arr.ndim)   # 2

print(arr.dtype)  # int64 (depends on system)
```

---

4. Indexing and Slicing

What it is:
You can access and modify parts of the array using indexing and slicing.

Example:

For 1D arrays:

```
arr = np.array([10, 20, 30, 40, 50])

print(arr[1])     # 20

print(arr[1:4])   # [20 30 40]
```

For 2D arrays:

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

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

---

5. Array Operations

What it is:
You can do mathematical operations directly on arrays.

Types:

- Element-wise operations
- Comparison
- Logical operations

Example:

```
a = np.array([1, 2, 3])
b = np.array([4, 5, 6])
print(a + b)  # [5 7 9]
print(a * 2)  # [2 4 6]
print(a > 1)  # [False  True  True]
```

---

6. Array Functions

What it is:
NumPy provides built-in functions for common tasks.

Examples:

- np.sum()
- np.mean()
- np.max(), np.min()
- np.std() – standard deviation

Example:

```
arr = np.array([1, 2, 3, 4])
print(np.sum(arr))   # 10
print(np.mean(arr))  # 2.5
print(np.std(arr))   # 1.118...
```

---

7. Reshaping Arrays

What it is:
Change the shape of the array (without changing its data).

Example:

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

```
reshaped = arr.reshape((2, 3))

print(reshaped)

# [[1 2 3]

#  [4 5 6]]
```

---

8. Stacking and Splitting Arrays

Stacking = Joining arrays
Splitting = Breaking arrays into parts

Example:

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

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

print(np.hstack((a, b)))  # [1 2 3 4]

print(np.vstack((a, b)))  # [[1 2], [3 4]]
```

Splitting:

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

print(np.split(arr, 3))   # [array([1, 2]), array([3, 4]), array([5, 6])]
```

---

9. Broadcasting

What it is:
Allows operations on arrays of different shapes.

Example:

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

b = 2

print(a + b)  # [3 4 5]
```

---

10. Copy vs View

What it is:
Understand the difference between **copy** (new data) and **view** (same data reference).

Example:

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

b = a.copy()
```

```
c = a.view()
a[0] = 10
print(b)  # [1 2 3]
print(c)  # [10 2 3]
```

---

11. Random Module in NumPy

What it is:
Useful for generating random numbers for simulations or data testing.

Examples:

- np.random.rand() – random float numbers
- np.random.randint() – random integers
- np.random.seed() – to get repeatable results

Example:

```
np.random.seed(0)
print(np.random.rand(3))     # [0.5488135  0.71518937 0.60276338]
print(np.random.randint(1, 10, 5))  # [5 1 9 8 9]
```

---

12. Useful NumPy Methods for Data Analysis

- np.unique() – get unique elements
- np.sort() – sort the array
- np.where() – condition-based filtering
- np.argmax(), np.argmin() – index of max/min

Example:

```
arr = np.array([1, 2, 2, 3])
print(np.unique(arr))      # [1 2 3]
print(np.where(arr > 1))    # (array([1, 2, 3]),)
```

---

13. Handling Missing or Invalid Data

Although Pandas handles this better, NumPy also supports:

- np.isnan() – check for NaNs

- np.nan_to_num() – replace NaNs

Example:

```
arr = np.array([1, np.nan, 3])
print(np.isnan(arr))  # [False  True False]
```

Summary Table

Topic	Why It Matters
NumPy Intro	Foundation for data analysis
Creating Arrays	Store and organize data efficiently
Indexing & Slicing	Access and modify parts of data
Array Operations	Perform math easily
Reshaping Arrays	Adapt data format for processing
Random Module	Simulate or create test data
Broadcasting	Work with arrays of different shapes
Useful Functions	Fast summary stats & data manipulation