NumPy Complete Guide – Part 1 (Basics)

1. Introduction to NumPy

NumPy (Numerical Python) is an essential library for scientific computing in Python. It provides:
- A powerful N-dimensional array object.
- Broadcasting functions.
- Tools for integrating C/C++ code.
- Useful linear algebra, Fourier transform, and random number tools.

Example 1:

```
import numpy as np

# Creating a 1D NumPy array
arr = np.array([1, 2, 3, 4, 5])
print("1D Array:", arr)
```

Example 2:

```
import numpy as np

# Checking version
print("NumPy Version:", np.__version__)
```

Example 3:

```
import numpy as np

# Creating a 2D array
arr2d = np.array([[1, 2], [3, 4]])
print("2D Array:\n", arr2d)
```

Example 4:

```
import numpy as np

# Type of the array
arr = np.array([10, 20, 30])
print("Type:", type(arr))
```

Example 5:

```
import numpy as np

# Array with mixed types
arr = np.array([1, 2.5, 3])
print("Array with mixed types:", arr)
```

2. Array Creation

NumPy offers several functions to create arrays, such as:
- np.array()
- np.zeros()
- np.ones()
- np.empty()
- np.arange()
- np.linspace()

Example 1:

```
import numpy as np
arr = np.zeros((2, 3))
print('Zeros array:\n', arr)
```

Example 2:

```
import numpy as np
arr = np.ones((2, 2))
print('Ones array:\n', arr)
```

Example 3:

```
import numpy as np
arr = np.empty((3, 3))
print('Empty array:\n', arr)
```

Example 4:

```
import numpy as np
arr = np.arange(0, 10, 2)
print('Array with arange:', arr)
```

Example 5:

```
import numpy as np
arr = np.linspace(0, 1, 5)
print('Array with linspace:', arr)
```

3. Array Attributes

Each NumPy array has attributes like:
- shape: Dimensions of the array
- ndim: Number of dimensions

- size: Total elements
- dtype: Data type of elements

Example 1:

```
import numpy as np
arr = np.array([[1, 2, 3], [4, 5, 6]])
print('Shape:', arr.shape)
```

Example 2:

```
import numpy as np
arr = np.array([1, 2, 3])
print('Dimensions:', arr.ndim)
```

Example 3:

```
import numpy as np
arr = np.array([[10, 20], [30, 40]])
print('Size:', arr.size)
```

Example 4:

```
import numpy as np
arr = np.array([1.0, 2.0])
print('Data Type:', arr.dtype)
```

Example 5:

```
import numpy as np
arr = np.array([1, 2, 3])
print('All Attributes:', arr.shape, arr.ndim, arr.size, arr.dtype)
```

4. Indexing and Slicing

Indexing and slicing help access specific elements or subarrays.

Example 1:

```
import numpy as np
arr = np.array([10, 20, 30, 40, 50])
print('Element at index 2:', arr[2])
```

Example 2:

```
import numpy as np
arr = np.array([1, 2, 3, 4, 5])
print('Slice [1:4]:', arr[1:4])
```

Example 3:

```
import numpy as np
arr = np.array([[1, 2, 3], [4, 5, 6]])
print('Row 0:', arr[0])
```

Example 4:

```
import numpy as np
arr = np.array([[10, 20], [30, 40]])
print('Element [1,1]:', arr[1,1])
```

Example 5:

```
import numpy as np
arr = np.array([1,2,3,4,5,6])
print('Reverse:', arr[::-1])
```

5. Array Operations

Operations can be performed on arrays element-wise.

Example 1:

```
import numpy as np
a = np.array([1, 2, 3])
b = np.array([4, 5, 6])
print('Addition:', a + b)
```

Example 2:

```
import numpy as np
a = np.array([10, 20, 30])
print('Multiplication by 2:', a * 2)
```

Example 3:

```
import numpy as np
a = np.array([1, 2, 3])
print('Square:', a ** 2)
```

Example 4:

```
import numpy as np
a = np.array([4, 9, 16])
print('Square Root:', np.sqrt(a))
```

Example 5:

```
import numpy as np
a = np.array([1, 2, 3])
b = np.array([3, 2, 1])
print('Comparison:', a > b)
```