Introduction To Numpy

In [ ]:
```
import numpy as np
```

1 Creating a numpy array

In [ ]:
```
# import numpy as np
array = np.array([1, 2, 3])
print(array)
# Output: [1 2 3]
```

```
[1 2 3]
```

2 Working with Arrays

Creating and manipulating arrays

In [ ]:
```
# import numpy as np
array1 = np.array([1, 2, 3])
array2 = np.array([[1, 2], [3, 4]])

# Accessing elements
print(array1[0])  # Output: 1

# Array operations
print(array1 + 1)  # Output: [2 3 4]
```

```
1
[2 3 4]
```

3 Array Manipulation

Reshaping and concatenation

In [ ]:
```
array = np.array([[1, 2], [3, 4]])
reshaped = array.reshape(4)
print(reshaped)  # Output: [1 2 3 4]
print(reshaped.shape)

array1 = np.array([1, 2])
array2 = np.array([3, 4])
concatenated = np.concatenate((array1, array2))
print(concatenated)  # Output: [1 2 3 4]
```

```
[1 2 3 4]
(4,)
[1 2 3 4]
```

4 Numerical Operations

Statistical functions

In [ ]:
```
array = np.array([1, 2, 3, 4])
mean = np.mean(array)
print(mean)  # Output: 2.5

random_numbers = np.random.rand(3)
print(random_numbers)  # Output: Random array of 3 numbers
```

```
2.5
[0.35717328 0.39906476 0.05518842]
```

5 Broadcasting

In [ ]:
```
a = np.array([1, 2, 3])
b = np.array([[1], [2], [3]])
result = a + b
print(result)
# Output:
# [[2 3 4]
#  [3 4 5]
#  [4 5 6]]
```

```
[[2 3 4]
 [3 4 5]
 [4 5 6]]
```

6. Universal Functions (ufuncs)

In [ ]:
```
array = np.array([0, np.pi/2, np.pi])
print(array)
sin_array = np.sin(array)
print(sin_array)
# Output: [0. 1. 0.]
```

```
[0.         1.57079633 3.14159265]
[0.0000000e+00 1.0000000e+00 1.2246468e-16]
```

7. Advanced Array Manipulation

Boolean Masking

In [ ]:
```
array = np.array([1, 2, 3, 4, 5])
mask = array > 3
print(mask)
```

```
print(array[mask])
# Output: [4 5]
```

```
[False False False  True  True]
[4 5]
```

Sorting

In [ ]:
```
array = np.array([3, 1, 4, 2])
sorted_array = np.sort(array)
print(sorted_array)  # Output: [1 2 3 4]
```

```
[1 2 3 4]
```

8. Linear Algebra

Matrix Manipulation

In [ ]:
```
matrix1 = np.array([[1, 2], [3, 4]])
matrix2 = np.array([[5, 6], [7, 8]])
result = np.dot(matrix1, matrix2)
print(result)
# Output:
# [[19 22]
#  [43 50]]
```

```
[[19 22]
 [43 50]]
```

9. NumPy with Real-World Data

Loading CSV Data

In [ ]:
```
# data = np.loadtxt('data.csv', delimiter=',')
# print(data)
# # Output: 2D NumPy array with data from CSV file
```

10. Performance Optimization

Vectorization

In [ ]:
```
array = np.array([1, 2, 3, 4])
result = array * 2  # Vectorized operation
print(result)  # Output: [2 4 6 8]
```

```
[2 4 6 8]
```

11. Fourier Transforms and Signal Processing

Fourier Transformation

In [ ]:
```
array = np.array([0, 1, 2, 3])
fft_result = np.fft.fft(array)
print(fft_result)
# Output: Fourier transform of the array
```

```
[ 6.+0.j -2.+2.j -2.+0.j -2.-2.j]
```

12. Multidimensional Data

Working with 3D Array

In [ ]:
```
array_3d = np.array([[[1, 2], [3, 4]], [[5, 6], [7, 8]]])
print(array_3d.shape)  # Output: (2, 2, 2)
```

```
(2, 2, 2)
```

13. Parallel Processing with NumPy

Parallel computation (using multiprocessing package)

In [ ]:
```
from multiprocessing import Pool

def square(x):
    return x * x

with Pool(4) as p:
    result = p.map(square, np.array([1, 2, 3, 4]))
    print(result)  # Output: [1, 4, 9, 16]
```