📘 Module-3: Linear Algebra using NumPy and SciPy

🔹 Introduction

Linear algebra is a foundational component in scientific computing, data science, and engineering. Python libraries like **NumPy** and **SciPy** provide powerful tools to perform linear algebra operations efficiently.

In this module, learners will:

- Solve systems of linear equations
- Perform matrix operations like **inverse**, **determinant**, and **least square solutions**
- Use **LU and Cholesky decomposition** for efficient solving of equations
- Compute **eigenvalues and eigenvectors** using `numpy.linalg` and `scipy.linalg`

💻 Python Programmes

🔹 1. Solve Simultaneous Equations using NumPy

```
import numpy as np
A = np.array([[3, 1], [1, 2]])
b = np.array([9, 8])
x = np.linalg.solve(A, b)
print("Solution x:", x)
```

🔹 2. Solve Simultaneous Equations using SciPy

```
from scipy.linalg import solve
A = [[3, 1], [1, 2]]
b = [9, 8]
x = solve(A, b)
print("Solution x:", x)
```

🔹 3. Find Matrix Inverse using NumPy

```
import numpy as np
A = np.array([[2, 1], [5, 3]])
A_inv = np.linalg.inv(A)
print("Inverse of A:\n", A_inv)
```

🔹 4. Find Matrix Inverse using SciPy

```
from scipy.linalg import inv
A = [[2, 1], [5, 3]]
A_inv = inv(A)
```

```
print("Inverse of A:\n", A_inv)
```

🔹 5. Calculate Determinant using NumPy

```
import numpy as np
A = np.array([[2, 3], [1, 4]])
det = np.linalg.det(A)
print("Determinant:", det)
```

🔹 6. Calculate Determinant using SciPy

```
from scipy.linalg import det
A = [[2, 3], [1, 4]]
d = det(A)
print("Determinant:", d)
```

🔹 7. Least Squares Solution using NumPy

```
import numpy as np
A = np.array([[1, 1], [1, 2], [1, 3]])
b = np.array([1, 2, 2])
x, residuals, rank, s = np.linalg.lstsq(A, b, rcond=None)
print("Least Squares Solution x:", x)
```

🔹 8. Least Squares Solution using SciPy

```
from scipy.linalg import lstsq
A = [[1, 1], [1, 2], [1, 3]]
b = [1, 2, 2]
x, residuals, rank, s = lstsq(A, b)
print("Least Squares Solution x:", x)
```

🔹 9. LU Decomposition using SciPy

```
from scipy.linalg import lu
import numpy as np
A = np.array([[2, 3], [5, 4]])
P, L, U = lu(A)
print("P:\n", P)
print("L:\n", L)
print("U:\n", U)
```

🔹 10. Cholesky Decomposition using NumPy

```
import numpy as np
A = np.array([[4, 2], [2, 3]])
L = np.linalg.cholesky(A)
```

```
print("Cholesky Decomposition (L):\n", L)
```

🔹 11. Cholesky Decomposition using SciPy

```
from scipy.linalg import cholesky
A = [[4, 2], [2, 3]]
L = cholesky(A, lower=True)
print("Cholesky Decomposition (L):\n", L)
```

🔹 12. Find Eigenvalues and Eigenvectors using NumPy

```
import numpy as np
A = np.array([[2, 0], [0, 3]])
eigvals, eigvecs = np.linalg.eig(A)
print("Eigenvalues:", eigvals)
print("Eigenvectors:\n", eigvecs)
```

🔹 13. Find Only Eigenvalues using NumPy

```
import numpy as np
A = np.array([[2, -1], [-1, 2]])
eigvals = np.linalg.eigvals(A)
print("Eigenvalues:", eigvals)
```

🔹 14. Find Eigenvalues and Eigenvectors using SciPy

```
from scipy.linalg import eig
A = [[4, 2], [1, 3]]
eigvals, eigvecs = eig(A)
print("Eigenvalues:", eigvals)
print("Eigenvectors:\n", eigvecs)
```

🔹 15. Solve General Eigenvalue Problem using SciPy

```
from scipy.linalg import eigvals
A = [[2, 0], [0, 3]]
ev = eigvals(A)
print("Eigenvalues:", ev)
```