1. Basic Statistical Measures using NumPy

```
import numpy as np

data = np.array([12, 45, 67, 23, 89, 10, 45])

mean = np.mean(data)
median = np.median(data)
std_dev = np.std(data)
variance = np.var(data)

print("Array:", data)
print("Mean:", mean)
print("Median:", median)
print("Standard Deviation:", std_dev)
print("Variance:", variance)
```

2. Sorting a NumPy Array by Rows and Columns

```
import numpy as np

array_2d = np.array([[9, 2, 7], [4, 6, 1], [5, 3, 8]])

# Sort by columns (axis=0)
sorted_by_columns = np.sort(array_2d, axis=0)
# Sort by rows (axis=1)
sorted_by_rows = np.sort(array_2d, axis=1)

print("Original Array:\n", array_2d)
print("Sorted by Columns:\n", sorted_by_columns)
print("Sorted by Rows:\n", sorted_by_rows)
```

3. Search and Count a Value in NumPy Array

```
import numpy as np

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

# Search for a value and return indices
indices = np.where(arr == 20)
# Count occurrences of the value
count = np.count_nonzero(arr == 20)

print("Array:", arr)
print("Indices of 20:", indices)
print("Count of 20:", count)
```

4. Difference Between Copy and View in NumPy

```
import numpy as np
```

```
original = np.array([1, 2, 3, 4])
view_array = original.view()
copy_array = original.copy()

view_array[0] = 100
copy_array[1] = 200
original[2] = 300

print("Original:", original)
print("View:", view_array)
print("Copy:", copy_array)
```

5. NumPy Matrix Operations: Addition, Subtraction, Multiplication

```
import numpy as np

A = np.matrix([[1, 2], [3, 4]])
B = np.matrix([[5, 6], [7, 8]])

addition = A + B
subtraction = A - B
multiplication = A * B

print("Matrix A:\n", A)
print("Matrix B:\n", B)
print("Addition:\n", addition)
print("Subtraction:\n", subtraction)
print("Multiplication:\n", multiplication)
```

6. Dot Product of Vectors and Matrix Multiplication

```
import numpy as np

v1 = np.array([1, 2, 3])
v2 = np.array([4, 5, 6])
dot_product = np.dot(v1, v2)

A = np.array([[1, 2], [3, 4]])
B = np.array([[5, 6], [7, 8]])
matrix_product = np.dot(A, B)

print("Dot Product:", dot_product)
print("Matrix Product:\n", matrix_product)
```

7. Solving a System of Linear Equations

```
import numpy as np

A = np.array([[2, 1], [1, 3]])
b = np.array([8, 13])
```

```
x = np.linalg.solve(A, b)

print("Solution:", x)
```

8. Eigenvalues and Eigenvectors

```
import numpy as np

A = np.array([[4, -2], [1, 1]])
eigenvalues, eigenvectors = np.linalg.eig(A)

print("Eigenvalues:", eigenvalues)
print("Eigenvectors:\n", eigenvectors)
```

9. Inverse and Determinant of a Matrix

```
import numpy as np

A = np.array([[4, 7], [2, 6]])
det = np.linalg.det(A)

try:
    inv = np.linalg.inv(A)
    print("Determinant:", det)
    print("Inverse:\n", inv)
except np.linalg.LinAlgError:
    print("Matrix is singular and cannot be inverted.")
```

10. Singular Value Decomposition (SVD) and Reconstruction

```
import numpy as np

A = np.array([[1, 2], [3, 4], [5, 6]])
U, S, VT = np.linalg.svd(A, full_matrices=False)

# Reconstruct matrix using SVD components
S_matrix = np.diag(S)
A_reconstructed = np.dot(U, np.dot(S_matrix, VT))

print("Original Matrix:\n", A)
print("Reconstructed Matrix:\n", A_reconstructed)
```