Statistical Analysis

1. **Task Description**

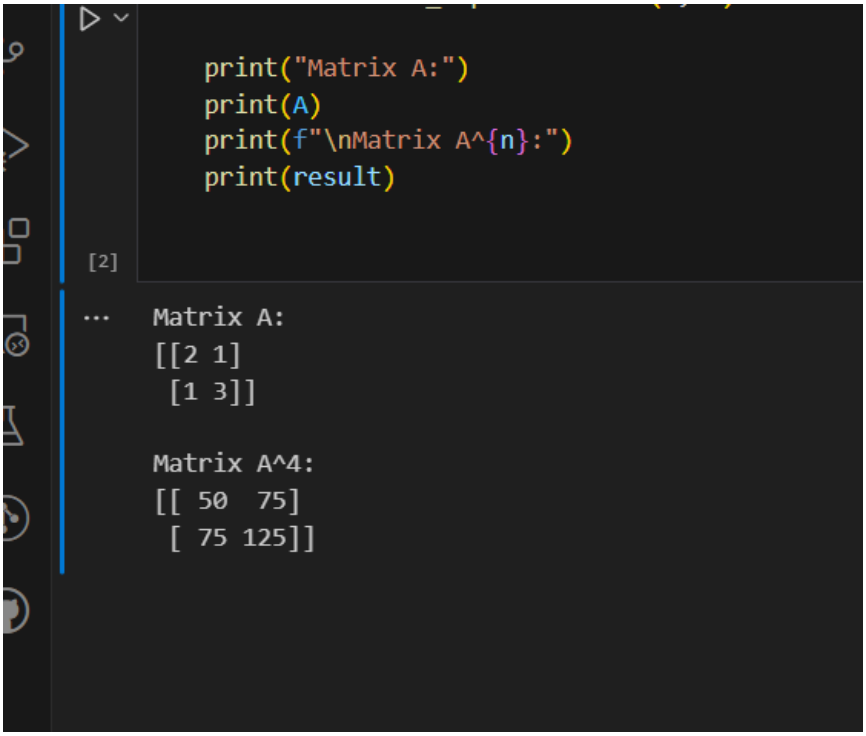
Implement matrix exponentiation for a square matrix using numpy.

Matrix exponentiation refers to raising a square matrix AAA to a non-negative integer power nnn, which is computed as An=A·A·…·AA^n = A \cdot A \cdot \ldots \cdot AAn=A·A·…·A (multiplied nnn times). In Python, you can efficiently implement this using **NumPy**.

Explanation

1. **Input Validation**:
   - Ensure the input matrix is square (rows = columns).
   - Verify that the exponent is a non-negative integer.
2. **Matrix Exponentiation**:
   - Use `np.linalg.matrix_power` from NumPy, which is optimized for matrix multiplication. It uses fast exponentiation (repeated squaring) for better performance when nnn is large.

2.Task Output



- **CODE:**

```python
import numpy as np

def matrix_exponentiation(matrix, power):
    """
    Computes the exponentiation of a square matrix using numpy.

    Parameters:
        matrix (ndarray): A square numpy array.
        power (int): The power to which the matrix is to be raised.

    Returns:
        ndarray: The result of matrix raised to the given power.
    """
    if not isinstance(matrix, np.ndarray):
        raise ValueError("Input must be a NumPy array.")
    if matrix.shape[0] != matrix.shape[1]:
        raise ValueError("Matrix must be square.")
    if not isinstance(power, int):
        raise ValueError("Power must be an integer.")

    return np.linalg.matrix_power(matrix, power)
```

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

n = 4
result = matrix_exponentiation(A, n)

print("Matrix A:")
print(A)
print(f"\nMatrix A^{n}:")
print(result)
```