TASK 3: Bell States and Entanglement Entropy

Aim:
To construct Bell states and compute their entanglement entropy.

Algorithm:

· Create entangled Bell states using tensor products.
· Reshape the states for partial trace computation.
· Calculate reduced density matrix.
· Compute von Neumann entropy.

Program :-

```
import numpy as np

print("\n" + "="*50)
print("TASK 3: BELL STATES AND ENTANGLEMENT
ENTROPY")
print("="*50)

def tensor_product(a, b):
    """Compute tensor product of two vectors"""
    return np.kron(a, b)

# Define qubit basis states
qubit_0 = np.array([1, 0])  # |0⟩
qubit_1 = np.array([0, 1])  # |1⟩

# Create Bell states using tensor products
bell_00 = (tensor_product(qubit_0, qubit_0) +
tensor_product(qubit_1, qubit_1)) / np.sqrt(2)  # |Φ⁺⟩
```

```
bell_01 = (tensor_product(qubit_0, qubit_1) +
tensor_product(qubit_1, qubit_0)) / np.sqrt(2)  # |Ψ⁺⟩
bell_10 = (tensor_product(qubit_0, qubit_0) -
tensor_product(qubit_1, qubit_1)) / np.sqrt(2)  # |Φ⁻⟩
bell_11 = (tensor_product(qubit_0, qubit_1) -
tensor_product(qubit_1, qubit_0)) / np.sqrt(2)  # |Ψ⁻⟩

print("Bell state |Φ⁺⟩ =", bell_00)
print("Bell state |Φ⁻⟩ =", bell_10)

def entanglement_entropy(state):
    """Calculate entanglement entropy for 2-qubit pure state"""
    # Reshape into 2x2 matrix
    psi_matrix = state.reshape(2, 2)
    # Reduced density matrix by tracing out qubit B
    rho_a = psi_matrix @ psi_matrix.conj().T
    # Eigenvalues of reduced density matrix
    eigenvals = np.linalg.eigvals(rho_a).real
    eigenvals = eigenvals[eigenvals > 1e-12]  # filter numerical
zeros
    # Von Neumann entropy
    entropy = -np.sum(eigenvals * np.log2(eigenvals))
    return entropy

# Compute entropies
print(f"\nEntanglement entropy of |Φ⁺⟩:
{entanglement_entropy(bell_00):.3f}")
print(f"Entanglement entropy of |Φ⁻⟩:
{entanglement_entropy(bell_10):.3f}")
```

```
# Compare with separable state
separable = tensor_product(qubit_0, qubit_0)
print(f"Entanglement entropy of |00⟩:
{entanglement_entropy(separable):.3f}")
```