CHSH Inequality Experiment - Simple Guide

Core Concepts & Implementation

1. Understanding the CHSH Inequality

Conceptual Background:

- CHSH inequality demonstrates quantum vs classical correlations
- Tests predictions of quantum mechanics against local hidden variable theories
- Involves measuring correlations between entangled particles

Implementation Approach:

```
%pip install qiskit[visualization]

%pip install qiskit_aer

%pip install qiskit_ibm_runtime
```

Programming Decisions:

- Using `%pip` instead of `!pip` for better Jupyter integration
- Installing visualization components upfront
- Including all dependencies to avoid runtime errors

2. Creating the Quantum Circuit

Quantum Concepts:

- Circuit creates and manipulates entangled qubit pairs
- Parameterized rotations allow measurement basis changes
- CNOT gate creates entanglement between qubits

Implementation Example:

```
theta = Parameter("\\theta")

chsh_circuit = QuantumCircuit(2)

chsh_circuit.h(0)         # Hadamard gate
```

```
chsh_circuit.cx(0, 1)     # CNOT gate

chsh_circuit.ry(theta, 0) # Rotation
```

Programming Decisions:

- Using `QuantumCircuit` as foundation class
- Parameterizing rotation angles for flexibility
- Sequential gate application for clarity

3. Defining Observables

Quantum Concepts:

- Observables represent measurement operators
- Combinations of Pauli operators needed for CHSH
- Different measurement bases reveal quantum correlations

Implementation:

```
observable1 = SparsePauliOp.from_list([

    ("ZZ", 1),

    ("ZX", -1),

    ("XZ", 1),

    ("XX", 1)

])
```

Programming Decisions:

- Using `SparsePauliOp` for efficient representation
- Clear operator combinations for CHSH quantities
- Structured observable definitions

4. Running the Experiment

Quantum Concepts:

- Multiple measurement angles needed
- Circuit optimization crucial for accuracy
- Backend selection affects results

Implementation:

```
number_of_phases = 21

phases = np.linspace(0, 2 * np.pi, number_of_phases)

estimator = Estimator(backend=backend)
```

Programming Decisions:

- Linear phase spacing for comprehensive analysis
- Using Estimator primitive for efficient computation
- Backend optimization for improved results

5. Result Analysis

Quantum Concepts:

- CHSH values exceed classical bounds
- Maximum quantum violation is $2\sqrt{2}$
- Results demonstrate quantum non-locality

Implementation:

```
fig, ax = plt.subplots(figsize=(10, 6))

ax.plot(phases / np.pi, chsh1_est, "o-", label="CHSH1")

ax.plot(phases / np.pi, chsh2_est, "o-", label="CHSH2")
```

Programming Decisions:

- Clear visualization of results
- Including classical and quantum bounds
- Comprehensive plot labeling

Troubleshooting and Best Practices

Common Issues:

1. Environment Problems:
    - Solution: Restart kernel and reinstall packages
    - Reason: Package conflicts or incomplete installations
2. Execution Errors:
    - Solution: Run cells in order, verify parameters
    - Reason: Quantum objects need proper initialization

Best Practices:

1. Always verify package versions
2. Run calibration circuits first
3. Use clear naming conventions
4. Include error checking
5. Document expected results

References:

- Qiskit Documentation
- IBM Quantum Services
- Bell's Theorem and CHSH Inequality