Quantum Computing Workshop

Foundations of Quantum Computing

CREO Research Center

Presenter: Chibuike Okekeogbu

Today’s Mission:

Explore the basics of quantum computing, build your first quantum circuit and understand why quantum computing will revolutionize cybersecurity

📱 Have ready: Laptop with Python installed

Classical Mechanics vs Quantum Mechanics

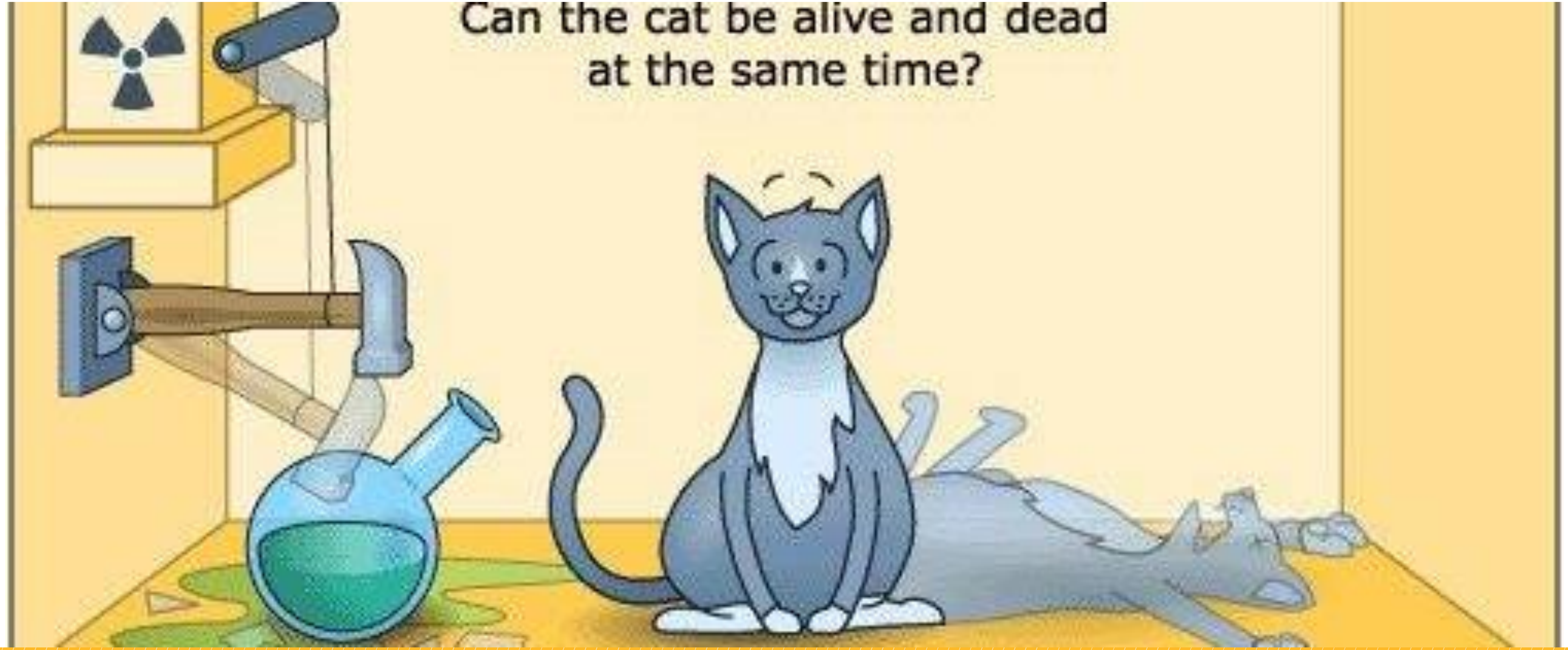
Properties of Quantum Mechanics

- Superposition: describes the behavior of particles or systems when they exist in multiple states

- Entanglement: This is a quantum phenomenon where the properties of two or more particles become correlated in such a way that the state of one particle is dependent on the state of another.

- Interference: This refers to the interaction or overlapping of waves, signals or other phenomena that results in a combined effect that is different from the individual effects of each component.

Schrödinger's Cat



Quantum System

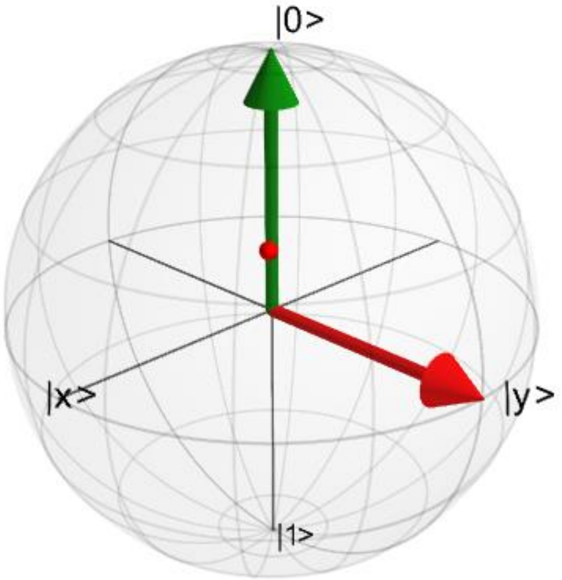
- In quantum world, a quantum bit denoted as qubit can be 0 and 1
- Encode classical bit 0 as $|0> = \begin{bmatrix}1\\0\end{bmatrix} \epsilon\ \mathbb{C}^2$
- Classical bit 1 $\rightarrow$ $|1> = \begin{bmatrix}0\\1\end{bmatrix} \epsilon\ \mathbb{C}^2$
- Superposition $\rightarrow$ $|0> + |1>$
- $|\psi> = \alpha\,|0> + \beta\,|1>\ \alpha, \beta \epsilon\ \mathbb{C}$
- $|\,\alpha\,|^2 + |\beta|^2 = 1$

Quantum System

- In a quantum system the |±> is an equal superposition of 0 and 1

- $|+\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) = \frac{1}{\sqrt{2}}\begin{pmatrix}1\\1\end{pmatrix} \in \mathbb{C}^2$

- $|-\rangle = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle) = \frac{1}{\sqrt{2}}\begin{pmatrix}1\\-1\end{pmatrix} \in \mathbb{C}^2$



Quantum System

- Tensor Product: This is a special operator that takes in two vectors and stitches them together to get one big vector.

- $\otimes : \mathbb{C}^{d_1} \text{ x } \mathbb{C}^{d_2} \rightarrow \mathbb{C}^{d_1 d_2} \rightarrow$ n qubits dimension, 2^n

- $(|\psi> \otimes |\phi>) \rightarrow |\psi>|\phi> (i,j) = \psi_i \phi_j$

- $|0> \otimes |1> = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}$

Quantum Gates

Pauli Matrices:

- X Gate (NOT gate): $X = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$ → X|0> = $\begin{bmatrix} 0 \\ 1 \end{bmatrix}$ = |1>
- Y Gate: $Y = \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix}$
- Z Gate: The Pauli Z gate allows us to inject a relative phase into our quantum state. $Z = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$ → Z|1> = -|1>
- Z|+> = |-> → Z is a NOT gate in |+>, |-> bases

Quantum Gates

- Hadamard Gate: Creates superposition. It takes us from the classical world to the quantum world and vice-versa. The Hadamard gate is used to create and destroy superposition.

- $\mathrm{H} = \frac{1}{\sqrt{2}}\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$

- $\mathrm{H}|0> = \frac{1}{\sqrt{2}}\begin{pmatrix} 1 \\ 1 \end{pmatrix} = |+> \rightarrow \mathrm{H}|+> = |0>$

- CNOT Gate: The Controlled NOT (CNOT) gate applies a NOT on the target qubit only if the control is set to 1. If the control is zero, nothing will happen

Why Quantum Computer?

The Crisis

- **2030:** Quantum computers will break RSA encryption
- **Today:** Hackers are stealing encrypted data to decrypt later
- **Impact:** $10.5 trillion annual cybercrime cost by 2025

The Quantum Advantage

Shor's Algorithm (1994):
Factor large numbers exponentially faster

- Classical: 300,000 years
- Quantum: 8 hours

Classical Limits

- Moore's Law is ending
- Exponential problems remain unsolvable
- Current encryption will be obsolete

Real Application Today

- Drug discovery (COVID vaccines)
- Financial modeling (Goldman Sachs)
- Cybersecurity (IBM, Google)
- AI/ML enhancement (quantum neural networks)

Quick Setup

```
#  Install Qiskit
pip install qiskit qiskit-aer matplotlib

# Create IBM Quantum account
# Visit: quantum-computing.ibm.com
# Get your API token

# 3. Connect to real quantum computer
from qiskit_ibm_runtime import QiskitRuntimeService
from qiskit import QuantumCircuit
import warnings
warnings.filterwarnings('ignore')


print("✅ Libraries imported successfully!")
print("📝 Next: Replace 'YOUR_API_TOKEN' below with your actual token")

# SSave your credentials (REPLACE 'YOUR_API_TOKEN' with your actual token!)
# ⚠️ IMPORTANT: Only run this ONCE per computer/environment


# REPLACE THIS with your actual token from quantum.ibm.com
YOUR_API_TOKEN = "YOUR_API_TOKEN_HERE" # ← Put your token between the quotes


try:
# Save account credentials (one-time setup)
service = QiskitRuntimeService.save_account(
channel="ibm_quantum_platform",
token=YOUR_API_TOKEN,
overwrite=True
)
print("✅ Account saved successfully!")
print("🎉 You can now access IBM Quantum computers!")
except Exception as e:
if "YOUR_API_TOKEN" in YOUR_API_TOKEN:
print("❌ ERROR: You need to replace 'YOUR_API_TOKEN_HERE' with your actual token!")
print("🔗 Get your token from: https://quantum.ibm.com")
else:
print(f"❌ Error saving account: {e}")

print("💡 Make sure your token is correct and  try again")
```

Build a Gate

```
# Gate playground
qc = QuantumCircuit(3, 3)

# Try different gates
qc.h(0)      # Superposition
qc.x(1)      # NOT gate
qc.y(2)      # Y rotation
qc.cx(0, 1)  # CNOT
qc.cz(1, 2)  # Controlled-Z

# Add measurements
qc.measure_all()

# Draw the circuit
print(qc.draw())
```

The Quantum Difference

Interactive Demo: Coin Flip Comparison

Classical coin

```
import random

# Classical bit: 0 OR 1
coin = random.choice([0, 1])
print(f"Result: {coin}")

# Always exactly 0 or 1
# 1 bit = 1 value
```

Quantum Coin

```
from qiskit import QuantumCircuit

# Quantum bit: 0 AND 1
qc = QuantumCircuit(1, 1)
qc.h(0)  # Superposition!
qc.measure(0, 0)

# Before measurement: BOTH
# After measurement: 0 or 1
```

Superposition

Live Coding: Create Your First Superposition

```
# Step 1: Install packages (if needed)
# pip install qiskit qiskit-aer matplotlib

# Step 2: Import libraries
from qiskit import QuantumCircuit
from qiskit_aer import AerSimulator
from qiskit.visualization import plot_histogram
import matplotlib.pyplot as plt

# Configure matplotlib for inline plotting in notebooks
%matplotlib inline

# Step 3: Create a quantum circuit
qc = QuantumCircuit(1, 1)  # 1 qubit, 1 classical bit

# Step 4: Apply Hadamard gate for superposition
qc.h(0)  # Put qubit 0 in superposition

# Step 5: Measure the qubit
qc.measure(0, 0)

print("Circuit diagram:")
print(qc.draw())

# Step 6: Run the circuit 1000 times
simulator = AerSimulator()
job = simulator.run(qc, shots=1000)
result = job.result()
counts = result.get_counts()

print(f"\nResults: {counts}")

print("\nCreating histogram...")
# Visualize the results
fig = plot_histogram(counts, title='Quantum Superposition Results (1000 shots)')
plt.tight_layout()
plt.show()
```

Entanglement

Build a Bell State

```
# Create entangled qubits (Bell State) - Modern Qiskit syntax
from qiskit import QuantumCircuit
from qiskit_aer import AerSimulator
from qiskit.visualization import plot_histogram
import matplotlib.pyplot as plt

# Create circuit with 2 qubits
qc = QuantumCircuit(2, 2)

# Step 1: Create superposition on qubit 0
qc.h(0)

# Step 2: Entangle qubit 0 with qubit 1
qc.cx(0, 1)  # CNOT gate creates entanglement!

# Step 3: Measure both qubits
qc.measure([0, 1], [0, 1])

# Visualize the circuit
print("🔗 Bell State Circuit (Entangled Qubits):")
print(qc.draw())

# Run the entangled circuit
simulator = AerSimulator()
job = simulator.run(qc, shots=1000)
result = job.result()
counts = result.get_counts()

print(f"\n📊 Bell State Results (1000 shots):")
for outcome, count in counts.items():
    print(f"  |{outcome}⟩: {count} times ({count/1000:.1%})")

print("\n🎯 What makes this special?")
print("   - You get ONLY |00⟩ and |11⟩ outcomes")
print("   - You NEVER get |01⟩ or |10⟩!")
print("   - The qubits are perfectly correlated (entangled)")
print("   - If one is 0, the other is always 0")
print("   - If one is 1, the other is always 1")

# Show histogram
plot_histogram(counts, title='Bell State: Entangled Qubits Results')
plt.tight_layout()
plt.show()
```

Challenges

- Noisy Qubits: Errors affecting accuracy and reliability.
- Limited Scalability: Struggles with large datasets and complex problems.
- Coherence Time: Maintaining quantum properties for longer durations.
- Designing QML Algorithms: Adapting classical algorithms effectively.
- Hybrid Quantum-Classical Computing: Integrating both technologies effectively.

https://learning.quantum.ibm.com/course/fundamentals-of-quantum-algorithms

https://www.ibm.com/quantum/qiskit

https://quantum.ibm.com

https://learning.edx.org/course/course-v1:PurdueX+ECE_69501.3+1T2024/home

https://qiskit-community.github.io/qiskit-machine-learning/tutorials/02a_training_a_quantum_model_on_a_real_dataset.html#

https://github.com/Qiskit/textbook/blob/main/notebooks/quantum-machine-learning/supervised.ipynb


Backspace
Screen
Page Up
Home
Thank You
Page Down
Insert
Shift

Questions