Qiskit in Action: A Practical Deep Dive into Quantum Randomness, Teleportation, and the Quantum Fourier Transform

What is Quantum Artificial Intelligence (QAI)?

QAI is the emerging field that combines the two most revolutionary technologies of our time:

1. **Quantum Computing:** Using the strange rules of quantum mechanics (like superposition and entanglement) to build machines that can process information in totally new ways.
2. **Artificial Intelligence (AI) and Machine Learning (ML):** Teaching computers to learn from data, recognize patterns, and make decisions without being explicitly programmed.

In simple terms: Quantum AI is about using the extreme power of **quantum computers** to supercharge **AI algorithms**, making them faster, smarter, and capable of solving problems that are currently impossible for any classical supercomputer.

? Why do we need QAI? (The Problem Statement)

Classical AI, which runs on the computers we use every day, has two major limitations:

1. **Computation Time:** Training an extremely complex AI model (like a new Large Language Model or a self-driving car AI) can take weeks or months, using massive amounts of energy.
2. **Data Space (Big Data):** Many of the world's most complex problems—like modeling a new drug molecule or optimizing global supply chains—involve so many possible combinations that a classical computer simply cannot store and search the possibilities in a realistic time frame. The computing time required might be longer than the age of the universe!

QAI offers the solution: By using quantum properties, QAI can represent and process that massive data space *exponentially* faster.

1. Superposition: The "Spinning Coin"

What is Superposition?

Superposition is the ability of a quantum particle (like the **qubit** or quantum bit) to exist in **multiple states at once** until it is measured.

- A **classical computer bit** is like a light switch: it can be either **0 (Off)** or **1 (On)**.
- A **qubit in superposition** is like a coin spinning in the air: it is simultaneously **Heads (0) and Tails (1)**.

$|\psi\rangle=\alpha|0\rangle+\beta|1\rangle$

When you stop the coin (i.e., **measure** the qubit), it instantly "collapses" into one definite state, either Heads (0) or Tails (1), based on the probability of the spin.

Easiest Example: Finding a Way Through a Maze

Imagine you need to find the exit of a maze.

Classical Approach	Quantum Approach (Superposition)
You try **one path at a time** (Path A, then Path B, then Path C, etc.).	You put the maze solver in a **superposition** of all possible paths.
This means you check the paths **sequentially**.	The solver explores **all paths simultaneously** and instantly knows which one leads to the exit when you "measure" the result.

Superposition allows quantum computers to perform massive parallel computation.

2. Entanglement: The "Connected Coins"

What is Entanglement?

Entanglement is a bizarre and powerful connection between two or more qubits where they become **linked together**, sharing the same fate, no matter how far apart they are.

- When two qubits are entangled, measuring the state of one instantly tells you the state of the other. The two are highly **correlated**.

Easiest Example: The Teleportation Pair

Imagine two magic, perfectly linked coins, Coin A and Coin B.

1. **Preparation:** Alice takes Coin A, and Bob takes Coin B. They travel to opposite sides of the Earth.
2. **Entanglement:** Because the coins are entangled, if Alice stops her coin (measures it) and it lands on **Heads (0)...**
3. **Instant Correlation:** ...Bob's coin instantly, at that exact moment, will be **Tails (1)** (or vice versa, depending on how they were prepared).

This link is *stronger* than any classical connection and is what makes Quantum Teleportation (like in your Practical 3) and highly efficient quantum algorithms possible.

3. Quantum Gates: The Circuit Tools

Quantum Gates are the basic operations (like NOT, AND, and OR in classical computers) that modify the state of qubits. Instead of being simple ON/OFF switches, quantum gates are often represented as **matrices** that perform **rotations** on the qubit's state.

Types of Quantum Gates

Quantum gates are generally divided into two types based on the number of qubits they affect:

Type	Description
Single-Qubit Gates	Act on only one qubit. They are primarily used to create **superposition** and rotate the qubit's state.
Multi-Qubit Gates	Act on two or more qubits. They are primarily used to create **entanglement** and perform logical operations based on one qubit controlling another.

Export to Sheets

Brief Explanation with Examples

A. Single-Qubit Gates

Gate	Analogy	Brief Explanation	Code Example (Qiskit)
Hadamard Gate (H)	The **"Superposition Starter"**	Takes a qubit from a definite state (|0⟩) and puts it into an **equal superposition** (|0⟩ and |1⟩ simultaneously). This is the gate that creates randomness.	`qc.h(qubit)`
Pauli-X Gate (X)	The **"Classical NOT"**	Flips the state. Turns |0⟩ into |1⟩ and |1⟩ into |0⟩.	`qc.x(qubit)`
Pauli-Z Gate (Z)	The **"Phase Flipper"**	Flips the *sign* (the phase) of the |1⟩ part of the qubit's state. It is important for creating interference in complex circuits.	`qc.z(qubit)`

Export to Sheets

B. Multi-Qubit Gates

Gate	Analogy	Brief Explanation	Code Example (Qiskit)
Controlled-NOT (CNOT or CX)	The **"Entanglement Maker"**	It has a **Control** qubit and a **Target** qubit. It only flips the Target qubit if the Control qubit is |1⟩. This is the primary gate used to create **entanglement**.	`qc.cx(control_q, target_q)`
Toffoli Gate (CCNOT)	The **"Double Control"**	A three-qubit gate. It flips the Target qubit *only if* **both** of the two Control qubits are |1⟩. It's a universal gate for classical computation.	`qc.ccx(q1, q2, q3)`

Practical 1: QAI_PRACT_1.ipynb

Title: Implementation of 16 Qubit Random Number Generator

- This notebook implements a basic **Quantum Random Number Generator (QRNG)** using a 16-qubit quantum circuit. A **Classical Bit** is a light switch (ON or OFF).
- A **Qubit** is a spinning coin (both HEADS and TAILS at once). This is called **Superposition**.
- **Quantum Gates** are like special operations (spins, flips, or rotations) that control the qubit's state.
- **Entanglement** is a special quantum link: if you have two entangled coins, flipping one instantly tells you the state of the other, no matter how far apart they are.

🔬 Practical 1: Quantum Random Number Generator (QRNG)

Title: Implementation of 16 Qubit Random Number Generator

Problem Statement (What and Why)

Every computer needs **random numbers** (for things like creating secure passwords or simulating complex weather patterns). Classical computers can only create *pseudo-random* numbers—they look random, but they are generated by a math formula, meaning they are predictable if you know the formula.

The **Quantum Random Number Generator (QRNG)** uses the true, natural uncertainty of **superposition** to create numbers that are **perfectly random** and absolutely unpredictable. We use 16 qubits to generate a 16-digit binary random number.

Practical 2: QAI_PRACT_2.ipynb

Title: Tackle Noise with Error Correction

*Note: Although the title mentions "Error Correction," the code is an exact implementation of the **16-qubit Quantum Random Number Generator** from Practical 1. The code does not contain any explicit quantum error correcting codes (QECC).* Problem Statement (What and Why)

Note: The code for this practical is identical to Practical 1 (the QRNG).

However, the title introduces an important concept: Noise and Error Correction.

Real quantum computers are extremely sensitive to noise (like small temperature changes or vibrations) that can randomly flip a qubit's state from $8$0$ to $9$1$.[10] **Quantum Error Correction (QEC)** is a crucial technique where we use *extra, entangled qubits* to check for and fix errors on the important ones, like having a backup system or a team of "error checkers."

The Code: Since the code is exactly the same as Practical 1, it implements a 16-qubit QRNG, not an explicit error correction code.

Practical 3: QAI_PRACT_3.ipynb

Title: Implement Quantum Teleportation Algorithm

This notebook implements the **Quantum Teleportation** protocol to transfer the quantum state of one qubit (Alice's) to another (Bob's). Problem Statement (What and Why)

Quantum Teleportation is not like in science fiction! It does **not** teleport matter. It transfers the *quantum information* (the state of a qubit) from a sender (Alice) to a receiver (Bob) instantly, using **entanglement** and classical communication.

Example: Alice prepares a spinning coin (qubit q_0) with a secret state. She wants Bob to have that *exact* state. They already share a pair of **entangled** qubits (q_1 and q_2). Alice performs a special measurement on her two qubits (q_0 and q_1). This measurement gives her two classical bits (0s or 1s), which she sends to Bob over a regular phone line. Bob uses this classical information to apply the final correction gate to his qubit (q_2), which instantly transforms his qubit into Alice's *original* secret state!

Practical 4: QAI_Pract_4.ipynb

Title: The Randomized Benchmarking Protocol.

This notebook uses the **Cirq** framework to demonstrate a simplified **Randomized Benchmarking (RB)** protocol, which is a method to characterize the average performance (fidelity) of quantum gates. Problem Statement (What and Why)

Since quantum computers are prone to noise, how can we tell if the gates are working well? **Randomized Benchmarking (RB)** is the industry standard to measure the *average error rate* or **fidelity** (reliability) of a quantum computer's gates.[11]

Example: Imagine a perfect machine that, when you press "UNDO," it completely reverses the last 100 things it did. RB is like this:

1. **Random Sequence:** Apply a long, random sequence of gates (like a complicated dance).
2. **Inverse Sequence:** Calculate and apply the *perfect opposite* (the "un-do" dance) of that sequence.
3. **Check:** If the quantum computer's gates are perfect, the qubit should end up exactly back where it started (State $\vert 0 \rangle$). If it ends up as $\vert 1 \rangle$, it means an error occurred during the sequence. By doing this many times, we find the average error rate.

Practical 5: QAI_PRACT_5.ipynb

Title: Implementing a 5 qubit Quantum Fourier transform

This notebook implements the **Quantum Fourier Transform (QFT)** on a 5-qubit quantum state. That's a fantastic request! Understanding the code and the concepts behind it is the best way to learn quantum computing.

For a 9th-grade level explanation, we'll use simple analogies:

- A **Classical Bit** is a light switch (ON or OFF).[1]
- A **Qubit** is a spinning coin (both HEADS and TAILS at once).[2] This is called **Superposition**.
- **Quantum Gates** are like special operations (spins, flips, or rotations) that control the qubit's state.[3]
- **Entanglement** is a special quantum link: if you have two entangled coins, flipping one instantly tells you the state of the other, no matter how far apart they are.[4]

🔬 Practical 1: Quantum Random Number Generator (QRNG)

Title: Implementation of 16 Qubit Random Number Generator

Problem Statement (What and Why)

Every computer needs **random numbers** (for things like creating secure passwords or simulating complex weather patterns).[5] Classical computers can only create *pseudo-random* numbers—they look random, but they are generated by a math formula, meaning they are predictable if you know the formula.[6]

The **Quantum Random Number Generator (QRNG)** uses the true, natural uncertainty of **superposition** to create numbers that are **perfectly random** and absolutely unpredictable.[7] We use 16 qubits to generate a 16-digit binary random number.

🛡️ Practical 2: Tackle Noise with Error Correction

Title: Tackle Noise with Error Correction

Problem Statement (What and Why)

Note: The code for this practical is identical to Practical 1 (the QRNG).

However, the title introduces an important concept: Noise and Error Correction.

Real quantum computers are extremely sensitive to noise (like small temperature changes or vibrations) that can randomly flip a qubit's state from [8]0 to [9]1.[10] **Quantum Error Correction (QEC)** is a crucial technique where we use *extra, entangled qubits* to check for and fix errors on the important ones, like having a backup system or a team of "error checkers."

The Code: Since the code is exactly the same as Practical 1, it implements a 16-qubit QRNG, not an explicit error correction code.

🛰️ Practical 3: Quantum Teleportation

Title: Implement Quantum Teleportation Algorithm

Problem Statement (What and Why)

Quantum Teleportation is not like in science fiction! It does **not** teleport matter. It transfers the *quantum information* (the state of a qubit) from a sender (Alice) to a receiver (Bob) instantly, using **entanglement** and classical communication.

Example: Alice prepares a spinning coin (qubit q_0) with a secret state. She wants Bob to have that *exact* state. They already share a pair of **entangled** qubits (q_1 and q_2). Alice performs a special measurement on her two qubits (q_0 and q_1). This measurement gives her two classical bits (0s or 1s), which she sends to Bob over a regular phone line. Bob uses this classical information to apply the final correction gate to his qubit (q_2), which instantly transforms his qubit into Alice's *original* secret state!

Practical 4: Randomized Benchmarking (RB)

Title: The Randomized Benchmarking Protocol.

Problem Statement (What and Why)

Since quantum computers are prone to noise, how can we tell if the gates are working well? **Randomized Benchmarking (RB)** is the industry standard to measure the *average error rate* or **fidelity** (reliability) of a quantum computer's gates.[11]

Example: Imagine a perfect machine that, when you press "UNDO," it completely reverses the last 100 things it did. RB is like this:

1. **Random Sequence:** Apply a long, random sequence of gates (like a complicated dance).
2. **Inverse Sequence:** Calculate and apply the *perfect opposite* (the "un-do" dance) of that sequence.
3. **Check:** If the quantum computer's gates are perfect, the qubit should end up exactly back where it started (State $\vert 0 \rangle$). If it ends up as $\vert 1 \rangle$, it means an error occurred during the sequence. By doing this many times, we find the average error rate.

Practical 5: Quantum Fourier Transform (QFT)

Title: Implementing a 5 qubit Quantum Fourier transform

Problem Statement (What and Why)

The **Fourier Transform** is a math tool that takes a complex signal (like a sound wave) and breaks it down into its simple components (like figuring out the individual notes in a chord).[12]

The **Quantum Fourier Transform (QFT)** is the quantum version of this tool.[13] It's one of the most important components in quantum computing because it can perform this decomposition exponentially faster than classical computers, which is the key to powerful quantum algorithms (like Shor's algorithm for breaking encryption).

We implement the QFT on a **5-qubit state**.

LINKS YOU CAN CHECK

https://medium.com/@arti.singh280/what-is-quantum-1ff4a2d2ea2c

https://medium.com/@arti.singh280/an-overview-of-quantum-computation-7686b66e418a

https://medium.com/@arti.singh280/quantum-computing-qubits-are-the-fundamental-units-of-information-d7a8a8ea8e75

https://medium.com/@arti.singh280