Quantum Random Number Generator (QRNG)

A Hackathon 2025 Project: Leveraging quantum mechanics for truly random outcomes, built with Qiskit.

IBM Quantum

The Quest for True Randomness

Classical Limitations

Traditional random number generators are pseudo-random, relying on deterministic algorithms that can be predicted.

Quantum Superiority

Quantum systems harness superposition and measurement to achieve truly unpredictable, random results.

Essential for Security

True randomness is crucial for cryptography, secure communications, and robust simulations.

Hackathon Challenge: Build Your QRNG

Your mission: Create a quantum circuit that generates truly random numbers.

Define Your Quantum System

Decide the number of qubits and the total possible outcomes (n) for your random numbers.

Prepare Superposition

Utilize Hadamard (H) gates to create a superposition state, ensuring 'n' equally likely outcomes.

Map & Measure

Map the measured quantum states to numerical values, enabling your program to print a unique random number.

Getting Started: Your First Steps

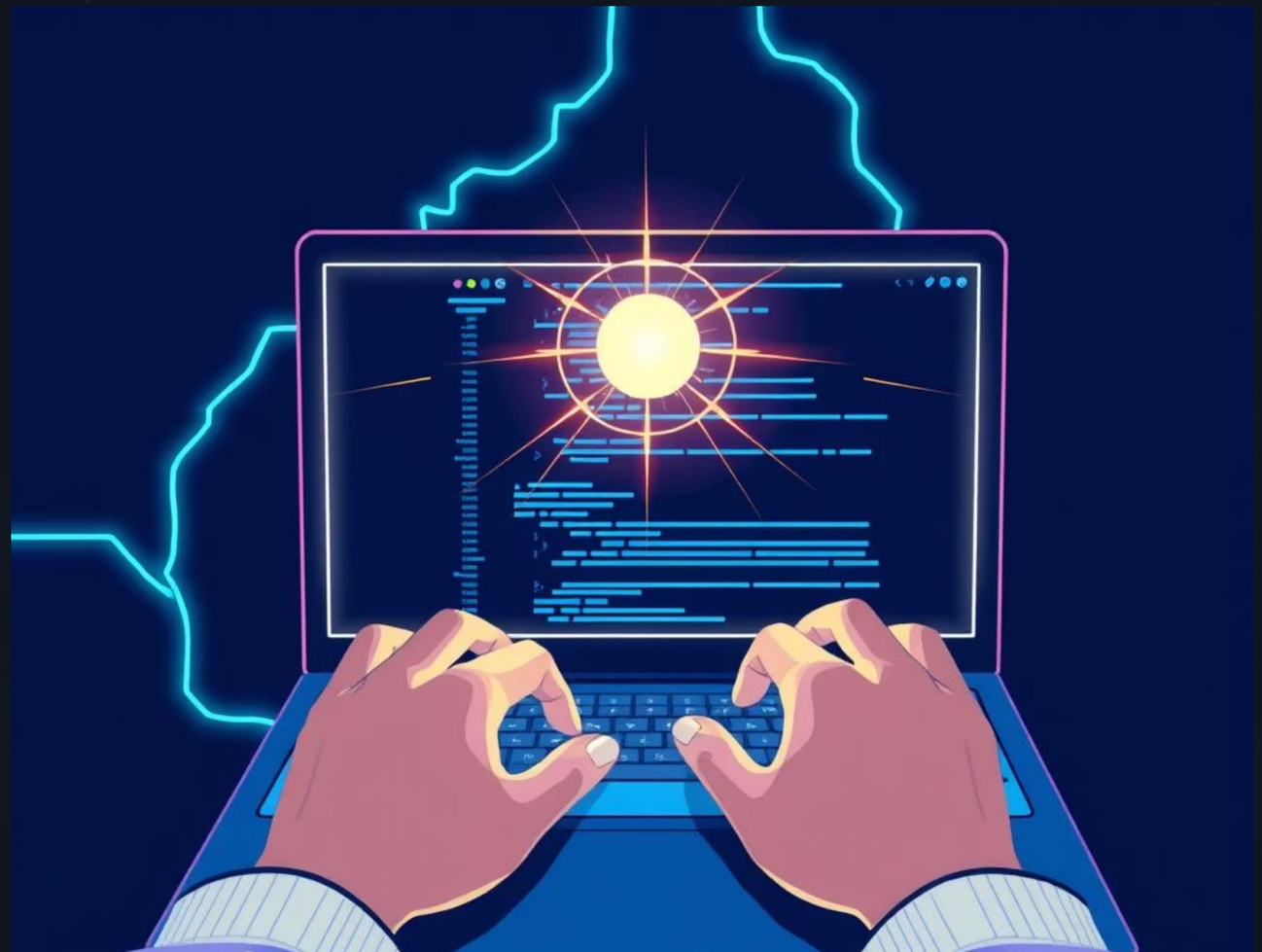
Clone the Repository

Begin by cloning the provided GitHub repository to access the starter code for your QRNG project.

```
git clone https://github.com/Dhakal-Unique/quantum-random-number-generator.git
```

Navigate into your project directory:

```
cd quantum-random-number-generator
```



Seamless Setup

The repository contains all the necessary files to kickstart your quantum random number generator development.

- README.md: Project overview
- LICENSE.md: Licensing details
- requirements.txt: Dependencies
- qrng/circuit.py: Quantum circuit logic
- qrng/runner.py: Execution script

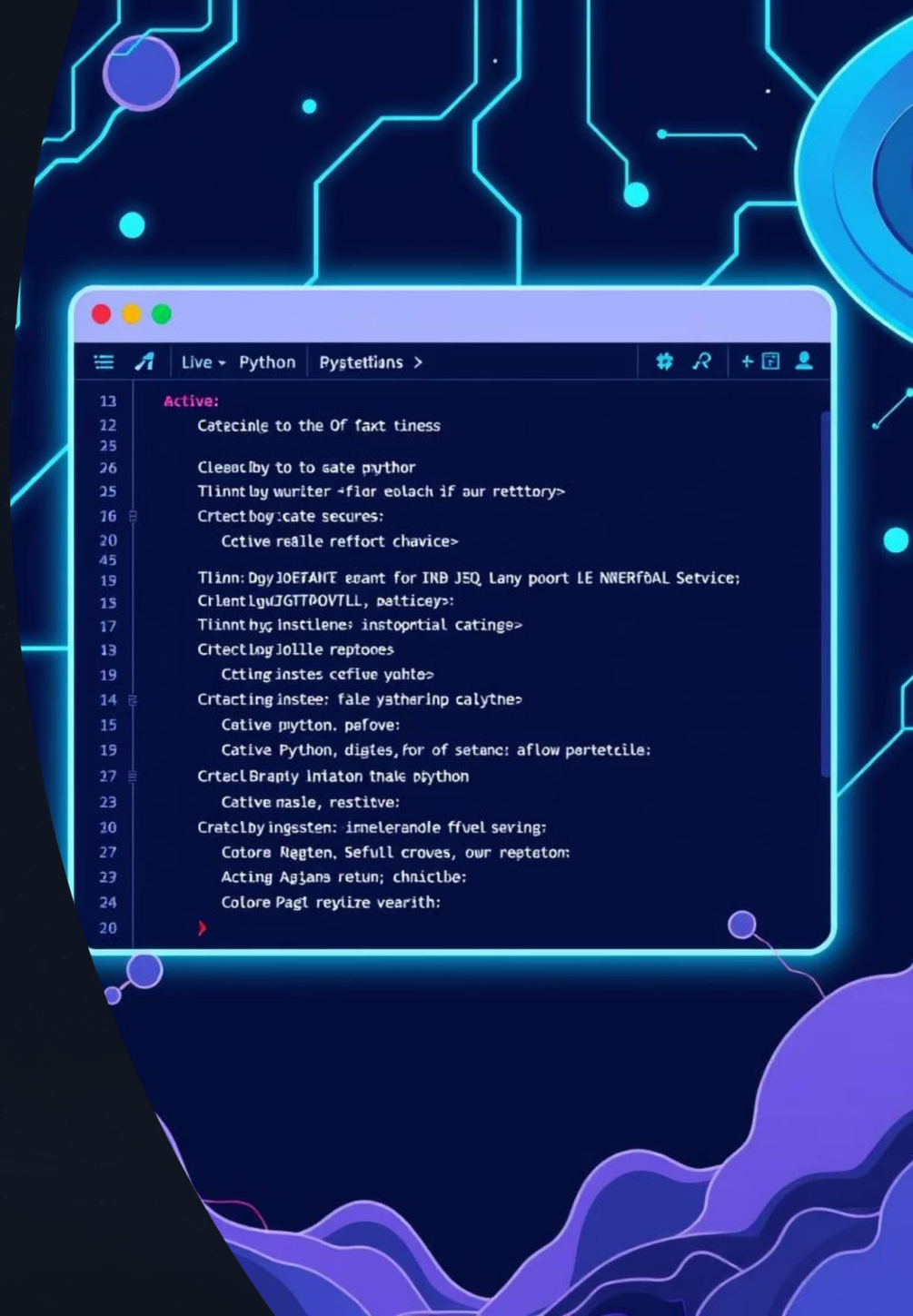
IBM Quantum

Usage Example: Running Your QRNG

Execute your quantum random number generator with a simple Python script.

```
from qrng.runner import run_qrng# Generate a random number with 3 qubits and 100 shotsresult = run_qrng(num_outcomes=8, shots=100)print("Random number:", result)
```

This snippet demonstrates how to call the `run_qrng` function, specifying the number of outcomes and measurement shots, then prints the generated quantum random number.



Tips for Success

Single Shot Execution

Run your QRNG with a single shot to generate one truly random number.

Multiple Shots for Distribution

Execute multiple shots to observe the fairness of your random number distribution and gain insights into quantum behavior.

Noise and Error Mitigation

Be aware that quantum noise can bias results. Explore various error mitigation techniques to enhance the fairness and accuracy of your QRNG.

IBM Quantum

Deeper Questions & Exploration

Distribution Anomalies

Investigate why certain values may appear more frequently in your QRNG outputs than others.

Optimizing Fairness

Identify and implement error mitigation techniques that significantly improve the fairness and true randomness of your results.

Scalability Challenges

Consider the challenges and potential solutions for scaling your QRNG to a larger number of qubits and outcomes.

Judging Criteria: How You'll Be Evaluated

Technical Aspects (30 pts)

Algorithm complexity, optimization, scalability, and effective use of Qiskit.

Originality (25 pts)

Novelty, creativity, and the difficulty of the attempted solution.

Usefulness (25 pts)

Practicality, design quality, and real-world applicability of your QRNG.

Presentation (20 pts)

Clarity of explanation, compelling storytelling, and effective team collaboration.

IBM **Quantum**

Suggested Resources & Further Reading

1

Basics of Quantum Information

Fundamental concepts for understanding quantum mechanics in computing.

2

Quantum Magic Eight Ball

A practical Qiskit example demonstrating quantum randomness.

3

Qiskit Fall Fest 2024 Notebook 1

A valuable resource for hands-on Qiskit learning and examples.

4

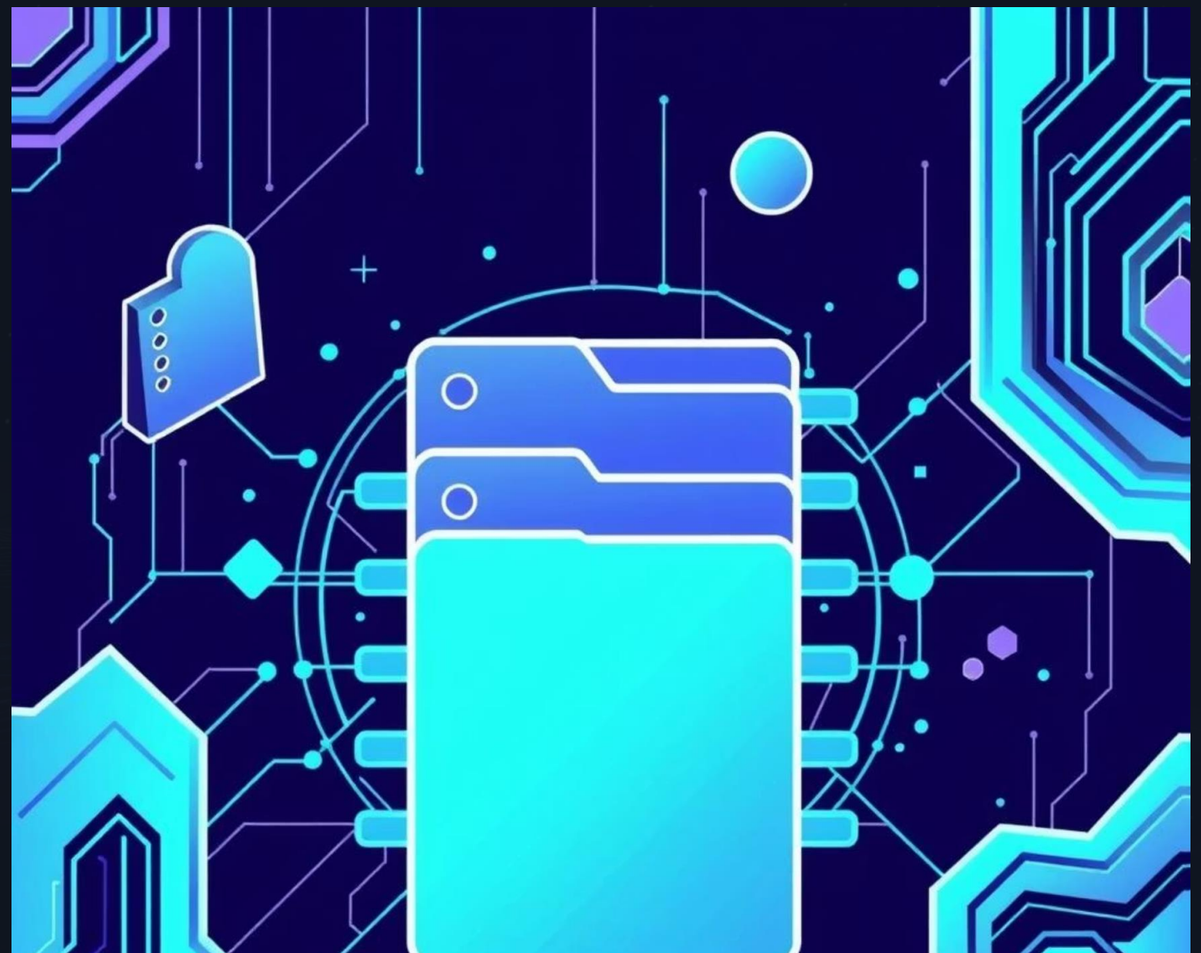
Error Mitigation & Suppression Techniques Documentation

In-depth guides on making your quantum computations more robust.

Project Structure & Licensing

QRNG Project Files

- `README.md`: Comprehensive project overview.
- `LICENSE.md`: MIT License details for open-source use.
- `requirements.txt`: Lists all necessary Python dependencies.
- `qrng/`: Core QRNG module.
- `qrng/circuit.py`: Defines the quantum circuit.
- `qrng/runner.py`: Script to execute the QRNG.



License Information

This project is proudly open-source, distributed under the MIT License.

For full details, please refer to the LICENSE.md file within the project repository.