

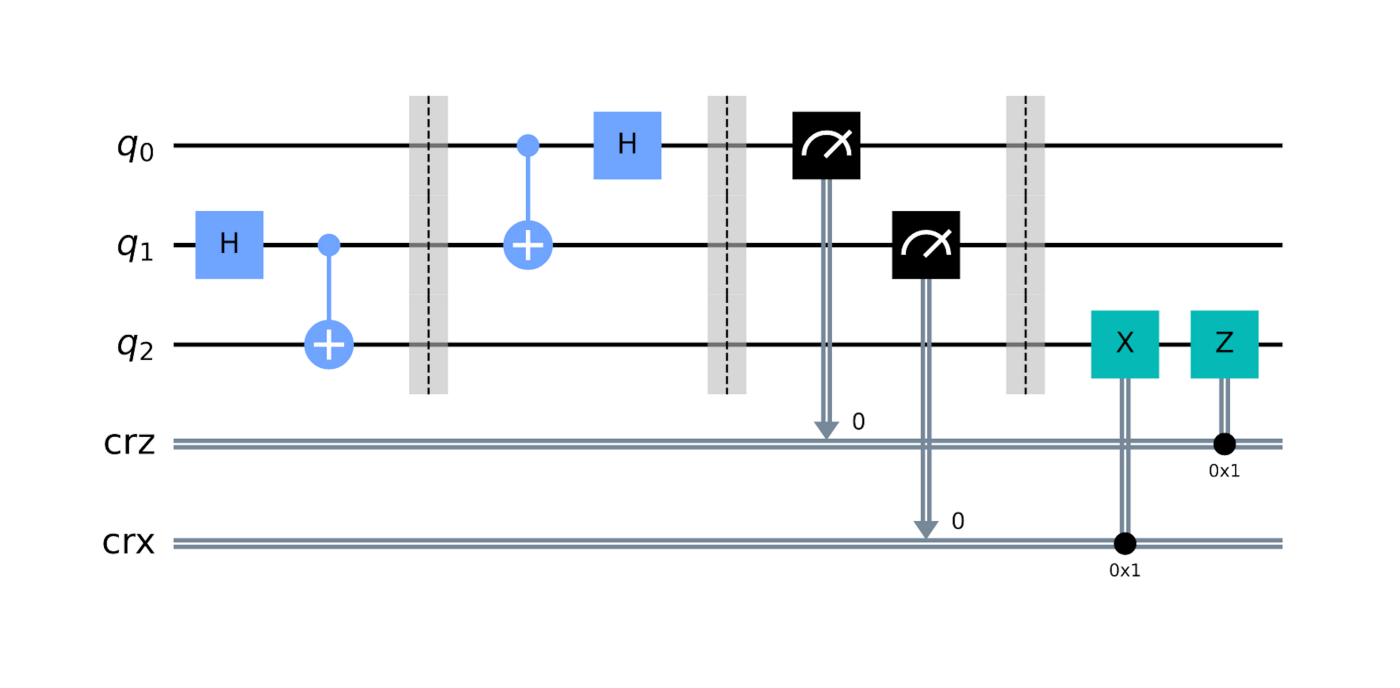
[Qiskit](https://medium.com/@qiskit?source=post_page-----7478d90d125a-----------------------------------)

Jan 6, 2021

·

8 min read

# **Learn Quantum Computing With These Seven Projects**

****

A quantum circuit

*By Ryan F. Mandelbaum, Senior Technical Writer, IBM Quantum and Qiskit*

A new year is a perfect reason to learn something new — like how to program a real quantum computer.

Quantum computers are devices that use the same mathematical rules governing the behavior of subatomic particles to perform calculations. We hope that one day, they’ll be able to solve problems in physics, chemistry, business, and other fields faster or more efficiently than regular computers. The field still has plenty of growth to do, so now is the perfect time to hop on board and help shape its future. Anyone can program a real quantum computer — it just takes a basic understanding of the Python programming language, familiarity with a few high school-level mathematical topics like linear algebra and complex numbers, and a computer connected to the internet.

Don’t know where to start? Well, you’ve already found the Qiskit blog, so you’re well on your way. Qiskit is an open-source software development kit and community dedicated to writing code on real quantum hardware. We’ve released tons of learning resources, from our [Coding with Qiskit](https://www.youtube.com/playlist?list=PLOFEBzvs-Vvp2xg9-POLJhQwtVktlYGbY) video series and our [Learn Quantum Computation using Qiskit](https://qiskit.org/textbook/preface.html) textbook to our free, comprehensive [Introduction to Quantum Computing and Quantum Hardware](https://qiskit.org/learn/intro-qc-qh) course.

An important component of learning is doing. Qiskit community members are constantly devising cool new projects both on their own and during [hackathons](https://medium.com/qiskit/these-amazing-quantum-computing-ideas-came-out-of-our-landmark-qiskit-hackathon-global-905a0063f8e5) that can serve as inspiration; I’ve [linked](https://medium.com/qiskit/rothko-inspired-generative-quantum-art-6f34ca9d17cb) [just](https://medium.com/qiskit/introducing-procedural-generation-using-quantum-computation-956e67603d95) a [few](https://medium.com/qiskit/these-amazing-quantum-computing-ideas-came-out-of-our-landmark-qiskit-hackathon-global-905a0063f8e5) of them [here](https://medium.com/qiskit/rasqberry-quantum-computing-is-the-coolest-project-for-raspberry-pi-3f64bec5a133?source=collection_archive---------9-----------------------). We’ve also generated a host of Jupyter notebooks for relevant projects that will help you understand these devices, their applications, and why we’re excited about this field. If you need help installing Qiskit and running Jupyter notebooks, you can read the [Qiskit documentation here](https://qiskit.org/documentation/install.html) or watch the dedicated episode of [Coding with Qiskit here](https://www.youtube.com/watch?v=M4EkW4VwhcI).

## **[Qiskit](http://qisk.it/2O7IcGt)**

### [Qiskit is an open source software development kit (SDK) for working with OpenQASM and the IBM Q quantum processors…](http://qisk.it/2O7IcGt)

[qisk.it](http://qisk.it/2O7IcGt)

## **Build a Quantum Random Number Generator**

Randomness sits at the very heart of quantum computing. One of the first counterintuitive principles behind quantum mechanics is that there are systems you can know everything about (such as the wavefunction that describes the position of an electron around an atom) and yet when you actually observe the system, the outcome (which orbital you measure the electron in) seems to be randomly determined. We can use this concept, called superposition, to our advantage by creating a random number generator.

At their core, quantum random number generators set up a superposition on one or more qubits using the Hadamard gate. This gate moves qubits from the 0 state to an equal superposition of the 0 and 1 state, meaning that you’re equally likely to measure a qubit value of 0 or 1 at the end of the quantum circuit. If you treat a string of qubits as a binary value (1111 = 15, for example), then applying a Hadamard gate to every qubit and then making a measurement will generate a random bitstring, and therefore a random number. You can use this random number wherever randomness is required; for example, by creating a bot that tells random [Quantum Dad Jokes.](https://github.com/rochisha0/quantum-ugly-duckling)

You can find various tutorials and implementations of random number generators using Qiskit [here](https://github.com/qiskit-community/qiskit-community-tutorials/tree/master/Cryptography) and [here](https://medium.com/@justinwritescrap/development-of-quantum-programs-for-use-on-mobile-devices-177ca3729963).

## **Implement Grover’s Search Algorithm**

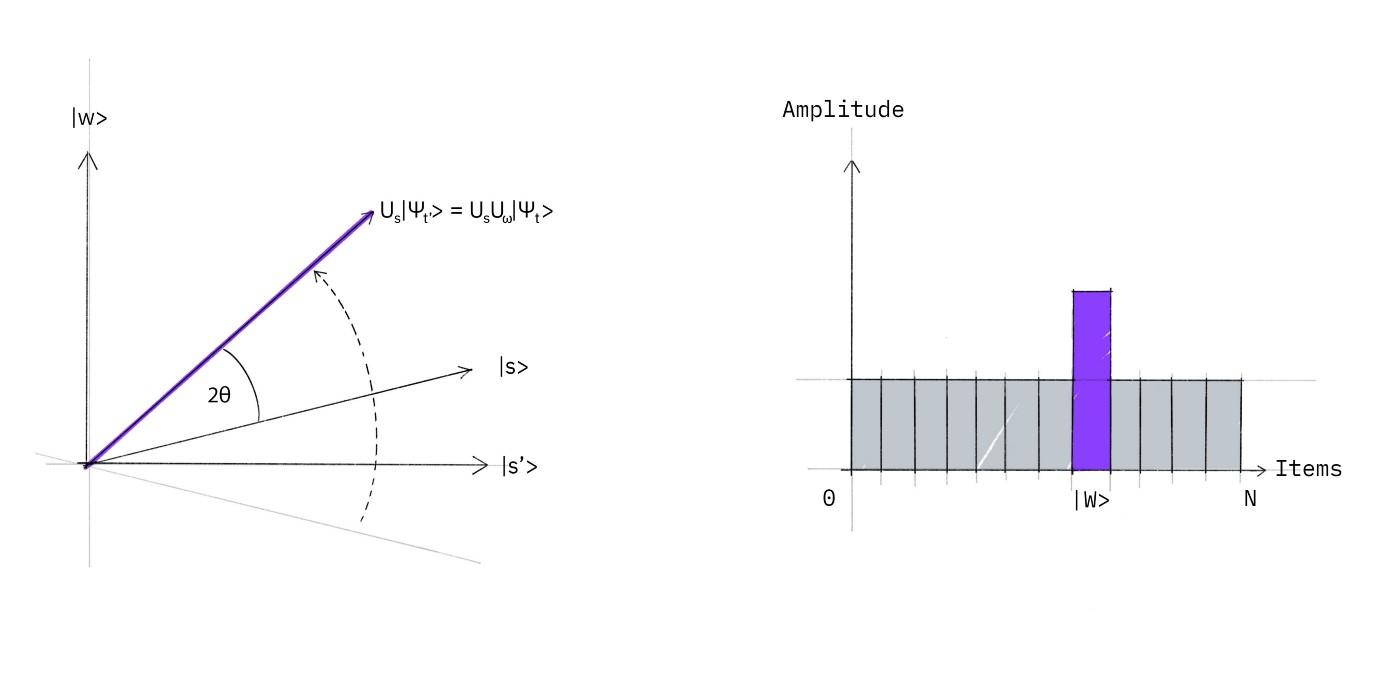
****

Image: Qiskit

One of the most talked-about applications of future quantum computers is the ability to perform an unstructured search through a database faster and more efficiently than a classical computer with the help of Grover’s algorithm. Grover’s algorithm could offer a quadratic speedup in these unstructured searches, meaning that, while a classical search through a list of size N could take between N/2 and N tries to find an item, Grover’s algorithm would only take √N steps. However, Grover’s algorithm is more than just a searching tool; it uses a routine called amplitude amplification that might provide quadratic speedups for other algorithms as well.

Let’s say you wanted to search through a list of numbers to see if a given number is there. Grover’s algorithm begins with an oracle function which marks any values that fulfill the given criterion. But merely marking the item isn’t enough for us to find it. We then apply another function called the diffuser function which makes the algorithm slightly more likely to return the location of the marked item than any other item. Roughly √N repetitions of this process, where N is the size of the list, amplifies the amplitude value associated with the location of the marked item enough to confidently return that location.

You can find a Jupyter Notebook walking through how to implement Grover’s algorithm [linked here](https://github.com/qiskit-community/intro-to-quantum-computing-and-quantum-hardware/blob/master/labs/introqcqh-lab-2.zip?raw=true), as part of the Introduction to Quantum Computing and Quantum Hardware course, or take a look at [the Qiskit textbook.](https://qiskit.org/textbook/ch-algorithms/grover.html)

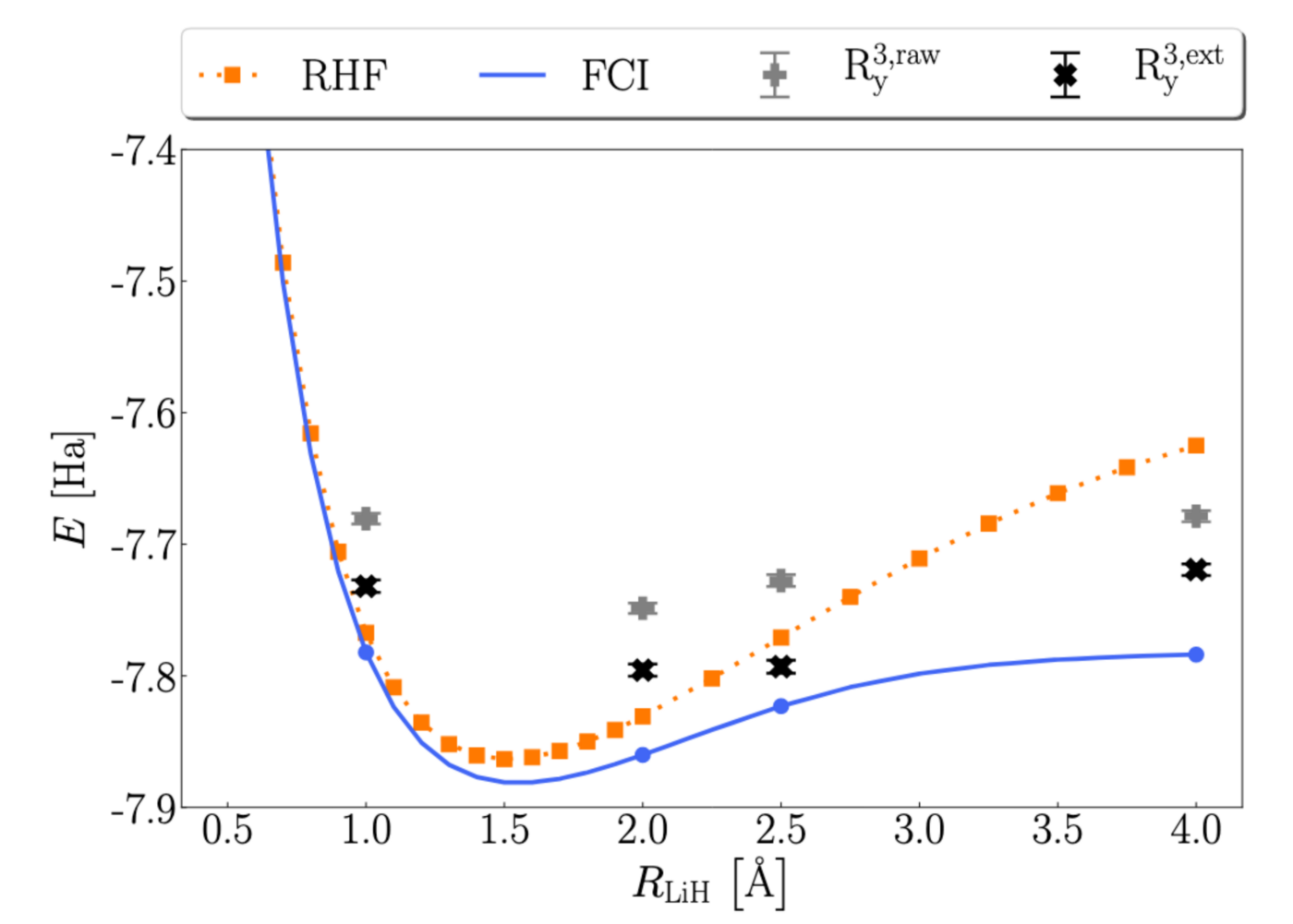
## **Use Shor’s Algorithm to Factor a Number**

Few quantum algorithms are as famous as the Shor’s factoring algorithm, thanks to its potential cybersecurity implications. Today, much of our security online uses RSA (Rivest–Shamir–Adleman) encryption, relying on the fact that large numbers are much harder to factor than they are to multiply together; multiplying is basically trivial, but the difficulty of factoring increases exponentially with the size of the input. However, given a powerful enough quantum computer, the difficulty to factor the number instead increases polynomially with the size of the input. It could be many years before a quantum computer running Shor’s algorithm becomes a real cybersecurity concern, but you can already use the algorithm to factor small numbers.

The magic behind Peter Shor’s namesake algorithm is that it treats factoring as a period finding problem; that is, a problem that finds how long it takes for a repeating function to, well, repeat. If you format the problem correctly, then the algorithm can return the factors of a given number N with high probability. This algorithm is a worthwhile project not only for its fame, but also because you’ll learn how to employ a few important quantum computing topics along the way, including the [Quantum Fourier Transform](https://qiskit.org/textbook/ch-algorithms/quantum-fourier-transform.html) and the [Quantum Phase Estimation algorithm](https://qiskit.org/textbook/ch-algorithms/quantum-phase-estimation.html).

You can find multiple walkthroughs of Shor’s algorithm including this [link here](https://github.com/qiskit-community/intro-to-quantum-computing-and-quantum-hardware/blob/master/labs/introqcqh-lab-2.zip?raw=true), the [Qiskit textbook](https://qiskit.org/textbook/ch-algorithms/shor.html#Example:-Factoring-15), and this [Qiskit Medium blog.](https://medium.com/qiskit/applying-shors-algorithm-bbdfd6f05f7d)

## **Find the Ground State Energy of a Lithium Hydride Molecule**

****

A few years ago, the IBM Quantum team [made headlines](https://www.ibm.com/blogs/research/2017/09/quantum-molecule/) by simulating the ground state energy of the lithium hydride molecule, a tantalizing hint toward quantum computers’ potential usefulness for hard and relevant problems. We hope that one day, quantum computing will [flip the script](https://medium.com/qiskit/ive-simulated-a-molecule-with-a-quantum-computer-now-what-aa9e2dfd92c5) in the field of chemistry, allowing chemists to predict what might happen in a chemical reaction before reaching for the beaker. We’re not quite there yet, but you can recreate this experiment on your own.

A chemistry project is the perfect introduction to a popular near-term quantum algorithm called the Variational Quantum Eigensolver (VQE). VQE starts with a quantum state prepared by a parametrized quantum circuit, called an ansatz. The algorithm passes the state’s corresponding wavefunction back and forth between a classical and a quantum computer, optimizing it by varying the quantum circuit’s parameters. At the end of the run, it should return a set of parameters used to calculate the ground state of the molecular system. You can use this algorithm’s results to create plots of how a molecule’s ground state energy varies with the distances between its atoms.

There are plenty of resources out there for you to simulate simple molecules like molecular hydrogen or lithium hydride yourself; there are resources [here](https://github.com/qiskit-community/intro-to-quantum-computing-and-quantum-hardware/blob/master/labs/introqcqh-lab-8.zip?raw=true) and [here](https://qiskit.org/textbook/ch-applications/vqe-molecules.html), or you can try out the new [Qiskit chemistry module.](https://medium.com/qiskit/introducing-the-new-qiskit-chemistry-module-and-gradients-framework-for-next-level-quantum-ebaf2be4c1a)

## **Tackle Noise With Error Correction**

Among the biggest challenges quantum computing engineers face today is handling noise caused by qubits interacting with the outside world. We still hope to find useful applications even on near-term noisy devices, but a general-purpose quantum computer that can offer many robust computational speedups needs to be protected against this noise. It needs to be error-corrected.

Error correction strategies exist at both the hardware and software level. One such error correction technique you can try on your own is the repetition code, which combines the values of multiple qubits in order to encode a single logical bit (rather than a logical qubit). You probably won’t be using the repetition code in your future algorithms, but it provides a basic introduction of the kinds of things that quantum programmers are thinking about when trying to protect these devices against noise.

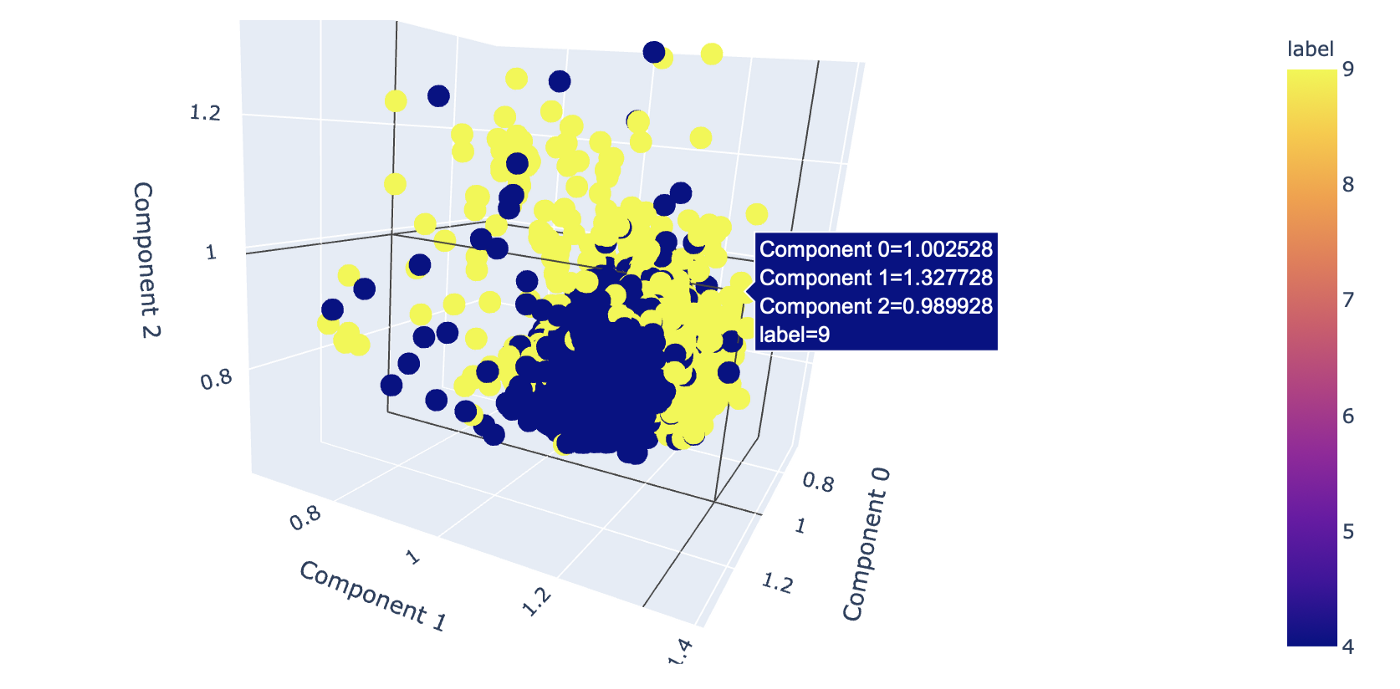
You can find walkthroughs of the repetition code in the [Qiskit Textbook here](https://qiskit.org/textbook/ch-quantum-hardware/error-correction-repetition-code.html) and in [this Jupyter notebook.](https://github.com/qiskit-community/intro-to-quantum-computing-and-quantum-hardware/blob/master/labs/introqcqh-lab-5.zip?raw=true)

## **Explore Quantum Hardware With Qiskit Pulse**

Quantum computing is such a new field that many programmers often need a working knowledge of the quantum hardware underlying their programs for their experiments. When you code with Qiskit, you’re actually sending fine-tuned microwave pulses to oscillating superconducting circuits. Qiskit uses a default set of pulses to represent the operations in your quantum circuit, but you can also tweak these pulses to perform real experiments on quantum hardware or try to create custom gates of your own.

The first step engineers take before trying to program their quantum devices is a calibration step, ensuring that they’re sending the correct pulses. You can find a walkthrough of the various experiments we perform in order to calibrate and characterize our hardware in the [Qiskit textbook here,](https://qiskit.org/textbook/ch-quantum-hardware/calibrating-qubits-pulse.html) and use these experiments to get to know quantum hardware in a whole new way. Once you’ve gotten the hang of Qiskit Pulse, you can try making your own [custom quantum gates](https://medium.com/qiskit/qiskit-pulse-can-make-custom-gates-ten-times-better-than-the-default-ones-3461d09683dc), or take a crack at our [IBM Quantum Awards: Open Science Prize.](https://www.ibmquantumawards.com/) You can even try hooking Qiskit up to [your own homemade quantum computer.](https://medium.com/qiskit/how-do-i-build-a-quantum-computer-in-my-house-1c7e9dc0c242)

## **Sort Images Using a Quantum Machine Learning Algorithm**

****

Machine learning is one of the hottest fields of study today, so naturally researchers are also interested in whether quantum computers might offer potential advantages in this space. However, before you go and invest in a quantum machine learning (QML) startup, you should be able to figure out for yourself whether quantum computers can offer any advantages in machine learning, and be able to tell what’s real and what’s hype.

One such quantum machine learning algorithm currently under study is a variational algorithm called the Variational Quantum Classifier (VQC). After a training round on labeled data, it sorts new data into categories. VQC served as the basis of our [Qiskit Challenge India](https://medium.com/qiskit/the-qiskit-challenge-india-proved-the-best-way-to-learn-is-by-doing-fe5926010a23), where participants worked to build an algorithm that sorted a set of handwritten numbers into 4 and 9. That challenge is over; however, the documentation is [still available](https://github.com/Qiskit-Challenge-India/2020) should you want to take a crack at a quantum machine learning project of your own.

Be forewarned; it’s still unclear just how much of a speedup quantum machine learning might offer. Instead treat this is an opportunity to get your hands dirty [exploring a frontier](https://medium.com/qiskit/introducing-the-qiskit-india-challenge-a-taste-of-quantum-machine-learning-for-qiskitters-in-india-4780ddbb03ab) that may or may not bear fruit in the future.

—

These are just a few of the many projects you can check out with a quantum computer; we’ve got a whole repository of other ideas on [GitHub here](https://github.com/qiskit-community/qiskit-community-tutorials), and Qiskitters are constantly coming up with cool new ideas for projects that are pushing the field forward. Want to get started? Begin your path toward learning Quantum Computation using Qiskit by [clicking here.](http://qisk.it/2O7IcGt)