Which Way to a Fault-Tolerant Universal Quantum Computer?

Today's quantum researchers are following one of two development paths. We believe only one will lead to highly generalizable quantum computers capable of solving great problems. The other path is very limited.

KEY
COMPUTATIONAL POWER
GENERALITY
APPLICATIONS

PATH 1 **VERY LIMITED**

Quantum Annealing Computer

DIFFICULTY LEVEL ●●○○○○○○○○

NO PATH TO UNIVERSAL

COMPUTATIONAL POWER
Low (equivalent to a classical computer)

GENERALITY
Restrictive

APPLICATIONS
Optimization problems

PATH 2 **THE WAY FORWARD**

Approximate Universal Quantum Computer

DIFFICULTY LEVEL ●●●●○○○○○○

COMPUTATIONAL POWER
High

GENERALITY
Partial

APPLICATIONS
Optimization problems
Quantum chemistry
Material science
Sampling
Quantum dynamics

THE GOAL

Fault-Tolerant Universal Quantum Computer

DIFFICULTY LEVEL ●●●●●●●●●●

COMPUTATIONAL POWER
Very high

GENERALITY
Complete

APPLICATIONS
Secure computing
Machine learning
Cryptography
Searching
Quantum chemistry
Quantum dynamics
Optimization problems
Sampling

IBM Quantum Computing

IBM Q

>58,000 users

All 7 continents

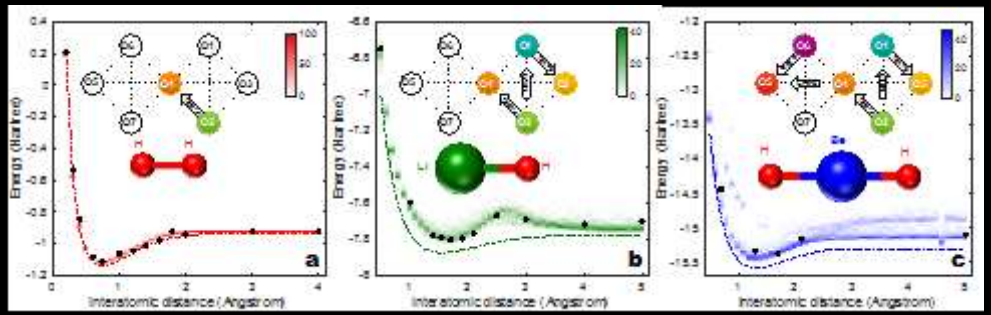
>150 colleges and universities

Over 1.5M experiments

35+ external papers

5Q and 16Q systems

Composer
Community
Tutorials & Resources



Kandala et al, *Nature,* 549, 7671 (2017)

https://www.qiskit.org

Search

Follow @qiskit on twitter!

QISKit

Quantum Information Software Kit

Last version v0.3.6

The Quantum Information Software Kit (QISKit for short) is a software development kit (SDK) for working with OpenQASM and the IBM Q experience (QX).

GitHub

Road map

Learn

Use QISKit to create quantum computing programs, compile them, and execute them on one of several backends (online Real quantum processors, and simulators).

Tutorials

Documentation

Run a quantum program

```
[python3] $ pip install qiskit
```

```
from qiskit import QuantumProgram
qp = QuantumProgram()
qr = qp.create_quantum_register('qr', 2)
cr = qp.create_classical_register('cr', 2)
qc = qp.create_circuit('Bell', [qr], [cr])
qc.h(qr[0])
qc.cx(qr[0], qr[1])
qc.measure(qr[0], cr[0])
qc.measure(qr[1], cr[1])
result = qp.execute('Bell')
print(result.get_counts('Bell'))
```