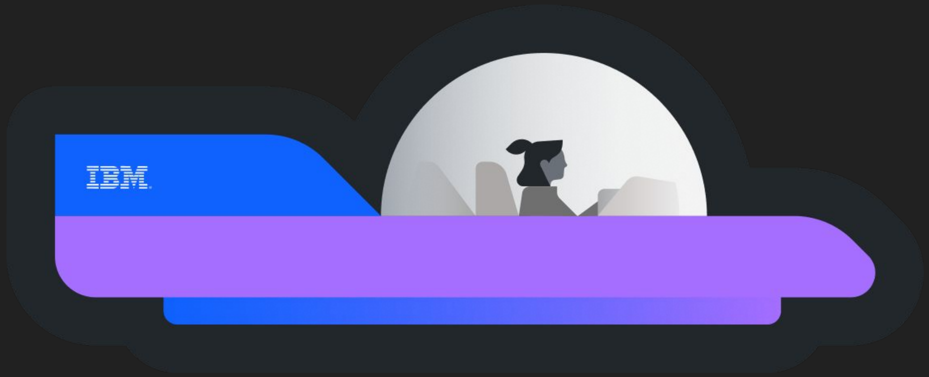
Welcome to the Qiskit Bootcamp!

Python 101

Python

- Qiskit, the Software Development Kit we’ll be using, is built on top of the coding language Python
- If you’ve ever coded before, you know enough to use Qiskit
- We’ll briefly review the basic syntax of Python so you know enough to get going.
- All you need to know is:
    - How to use variables
    - How to print
    - How to do basic operations
    - How to use numpy arrays



Qiskit 101: Circuits

Cbits and Qbits

Classical bits (cbits)

- Cbits have binary, or 1 of 2, states
  - $|0\rangle$ and $|1\rangle$
- At any given time the bit will be in only one of the two states

A classical computer with three bits have 8 = 2^3 possible states:

$|000\rangle$, $|001\rangle$, $|010\rangle$, $|011\rangle$

$|100\rangle$, $|101\rangle$, $|110\rangle$, $|111\rangle$

Classical gates implement binary operations on binary inputs

Quantum bits (qbits)

- Qbits are also binary, but their quantum behavior means they occupy their two states a bit differently:
  - Superposition and entanglement allow qbits to exist in their $|0\rangle$ and $|1\rangle$ states simultaneously

A general Qbit is a linear combination in a vector space of n dimensions:

$$\alpha|0\rangle + \alpha|1\rangle$$

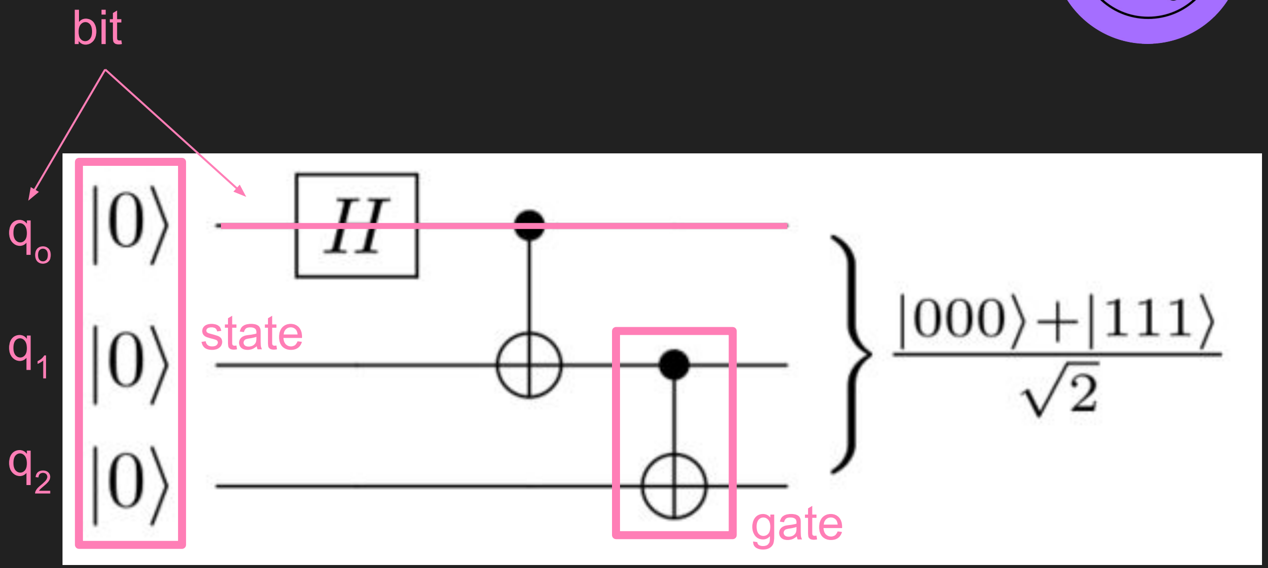
Basic operations can act on all Qbits simultaneously

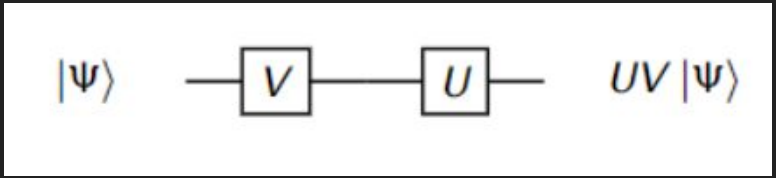
The Circuit

Quantum circuits are drawn with 3 basic elements

- The bits
- The states
- The gates

The next section will cover gates in more depth





Qiskit 101: Gates

The Pauli Gates (XYZ) Act on a Single Qubit at a time

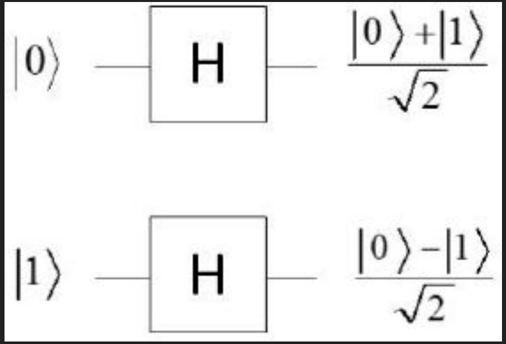
There are 3 “Pauli” or 180-degree rotation gates:

- X (classical Not gate)
  - Flips the state of the qbit from $|0\rangle$ to $|1\rangle$ and vice versa
- Z
  - Flips the phase of a qbit in the $|1\rangle$ state but does not change $|0\rangle$ states
- Y
  - Flips the phase and state of the qbit (like doing X and Z operations at the same time)

$$\sigma_x = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \qquad \sigma_y = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix} \qquad \sigma_z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$$

The Hadamard (H) Also Acts on a Single Qubit

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



- Discuss with your neighbors:
  - What does the H gate do to a qubit?

The H gate transforms a qubit into an equal superposition of $|0\rangle$ and $|1\rangle$

  - How does the H gate affect the measurements in a quantum circuit?

When you measure the qubit, you don't always get a definitive 0 or 1. Instead, the qubit could be in either state, depending on the superposition

  - Why is the H gate important to quantum computing?

Creating superpositions allows qubits to represent multiple possibilities simultaneously, which is the key feature in quantum computing

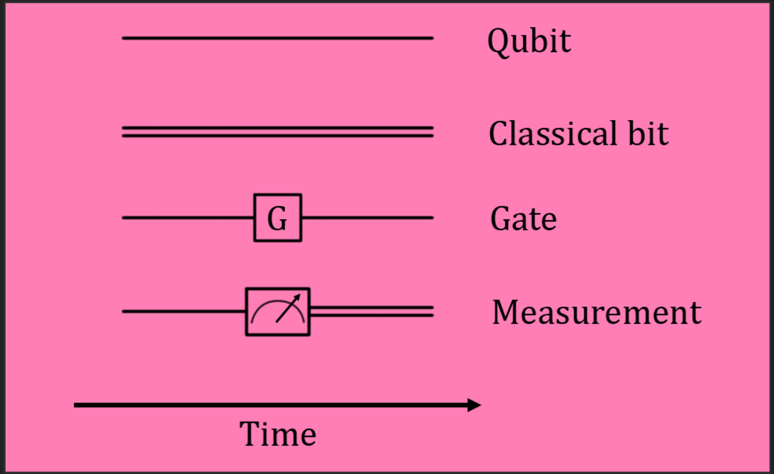
  - What happens when you apply the H gate twice to a qubit?

It returns the qubit to its original state. Mathematically, applying the gate twice is equivalent to the Identity Matrix

Measurement Gates

In quantum computing, the state of our qubits are probabilistic:

- When we take measurements, we collapse our qbit into either $|0\rangle$ or $|1\rangle$
- The probability is changed by the operations (gates) we applied before measuring
- In qiskit, measurements on quantum circuits need to be mapped to classical registers



Qiskit 102: Multi-Qubit Gates

Controlled NOT (CNOT/CX)

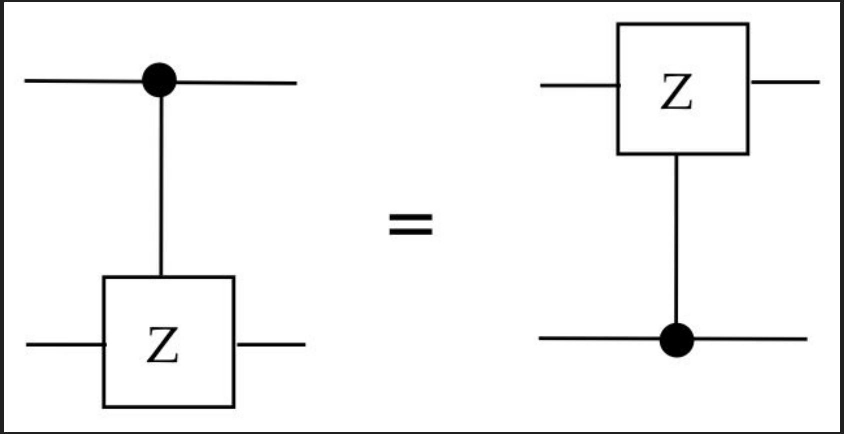
A controlled NOT gate or CNOT works on TWO qubits at the same time.

- The first qbit is the “control” and our operation depends on its state
  - Shown as the black dot on the circuit
- The second qbit is the “target” and this is the bit that the gate acts on
  - Shown as the gate on the circuit
- The NOT gate if only active if the state of the control is |1〉

Controlled-Z (CZ)

Just like the CNOT, the CZ gate operates on TWO qubits at once with a control and target qbit

- Again the Z-gate acts on the target if and only if the control state is $|1\rangle$



Swap Gate

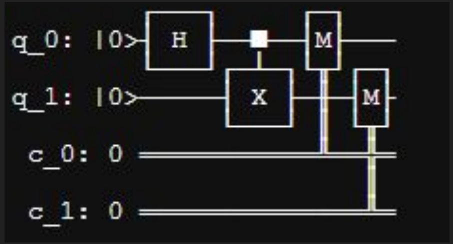
The swap gate does just that – it swaps two qbits

- Say we have two qbits q0 and q1:
  - The swap gate will swap the two bits and all operations done on them up to the swap

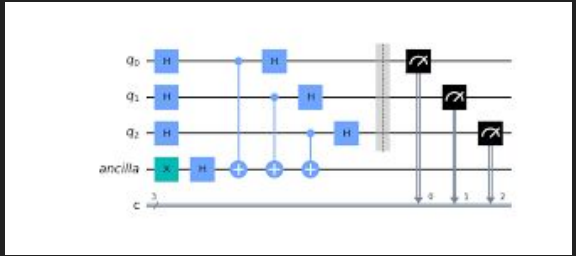
Displaying Circuits in Qiskit

There are many ways to display a quantum circuit in Qiskit.
Make sure you download the respective packages.

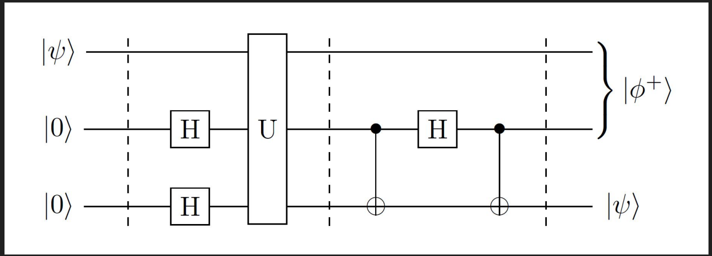
- Text Format or ASCII Art
    - QuantumCircuit.draw()



- Matplotlib Visualization
    - QuantumCircuit.draw(output = ‘mpl’)



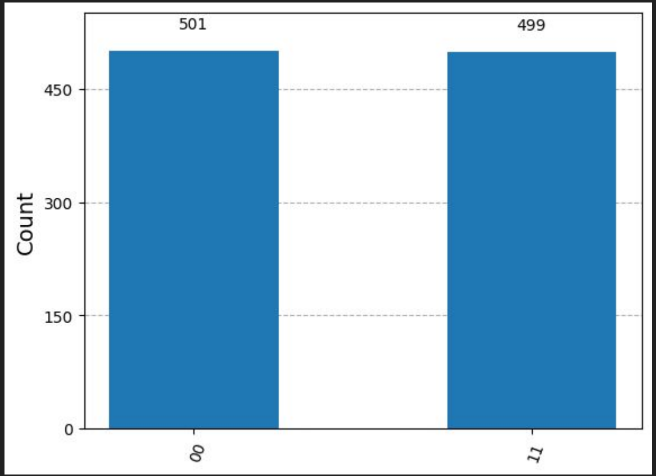
- LaTeX
    - QuantumCircuit.draw(output = ‘latex’)

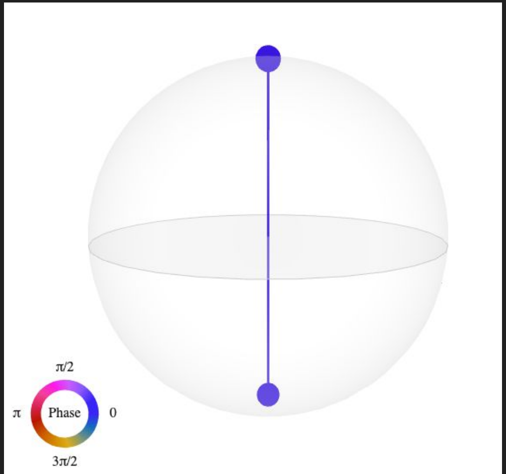


Displaying Measurements in Qiskit

You can display the outcome of the measurements in multiple formats.

Two helpful visualizations include a histogram and a bloch sphere.





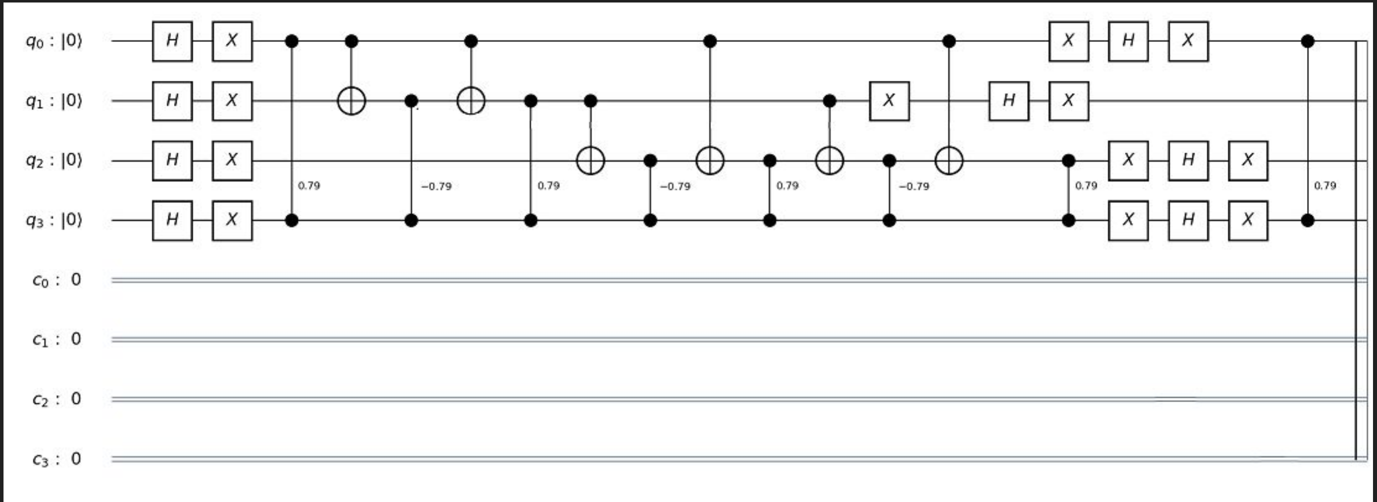
Using Gates

Qiskit 201

Now that we know how to create basic gates, how can we combine them to make things?

Using Multiple Gates

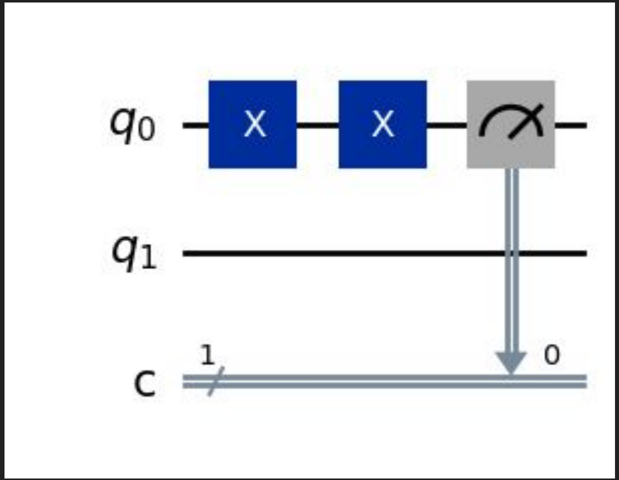
- Applying multiple gates to the same circuit in succession lets us perform more advanced manipulation of our qubits
- Gates can be added to a circuit sequentially using the same syntax as we used before



Fun Fact:
This is the circuit for Grover's Search

Simple Example: 2 NOT Gates

- Let’s create a circuit with two not gates and then a measurement.
- We’ll simulate its execution 100 times.
- What do we expect the measurement to be?



More Interesting Example: Generating Bell States

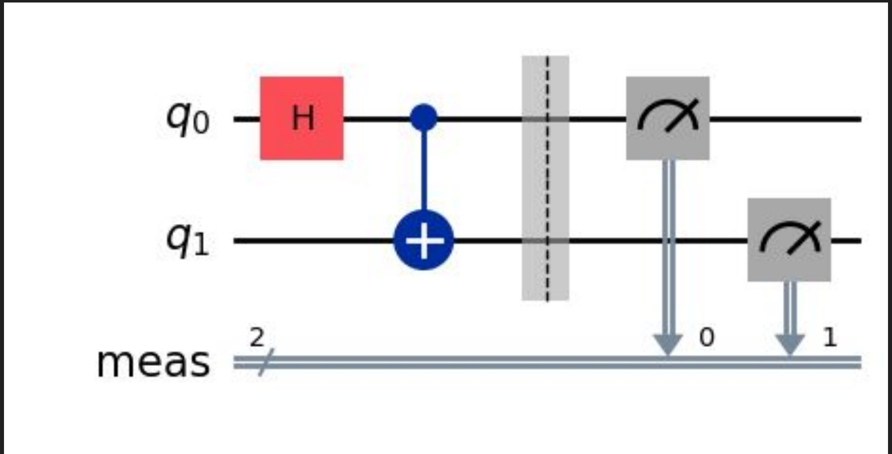
- Bell States are some of the simplest examples of quantum entanglement
- Used to put two qubits in an EQUAL superposition. Thus, there is an equal probability of measuring either state
- We call the two qubits “maximally entangled” in the Bell States
- Use Cases:
    - Quantum Teleportation
    - Quantum Computing
    - Pretty much every quantum algorithm

Bell State Circuit

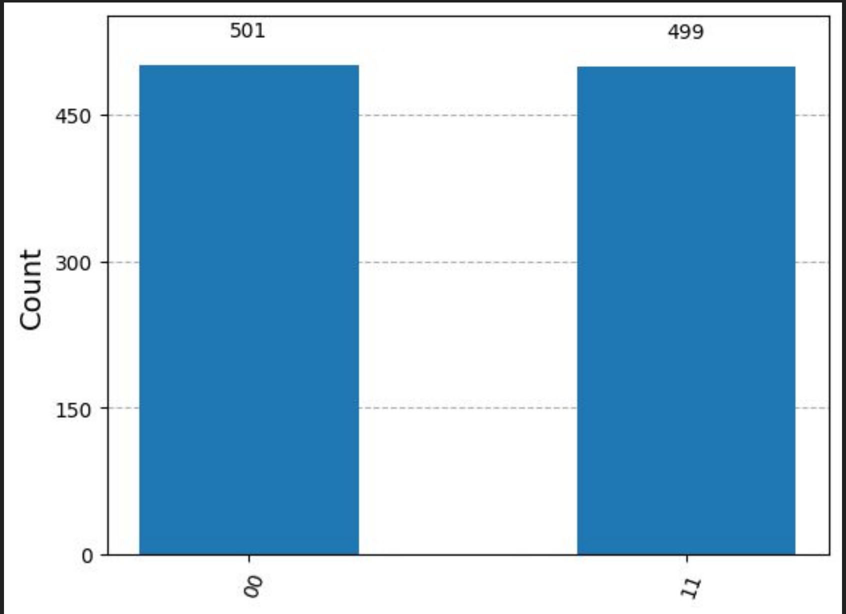
Here's the circuit used to generate a bell state.

Notice what gates are being used.

Let's try and make it ourselves!



Bell State Result



The two qubits have the same state!

What if we wanted to entangle more than 2 qubits?

Extending the Bell State to Multiple Qubits

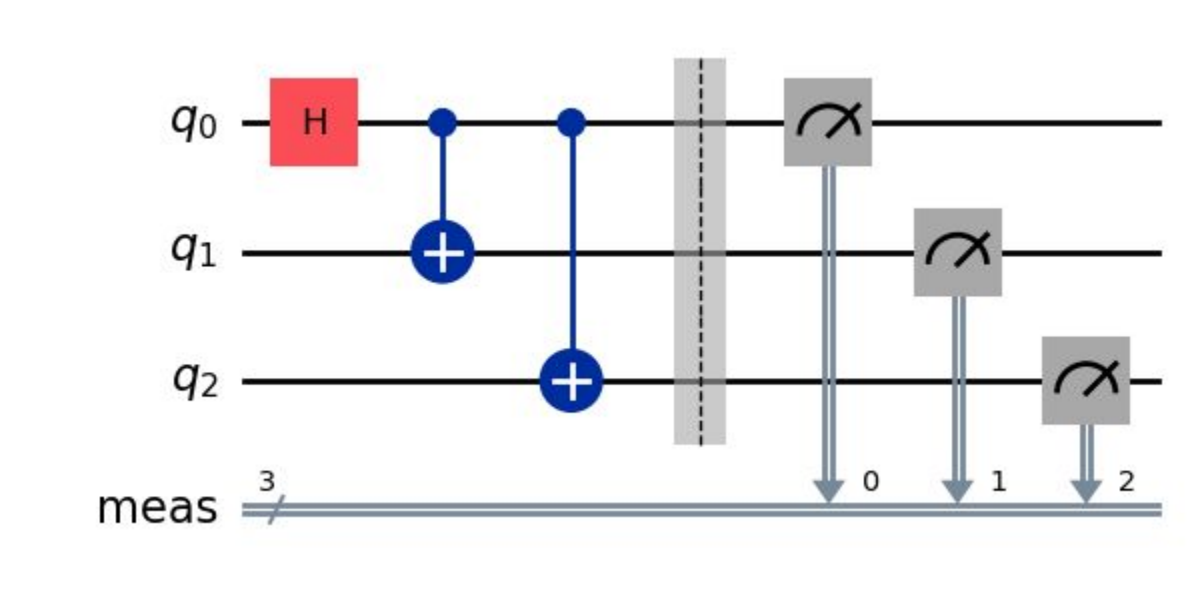
Any ideas of how we would entangle 3 qubits to have the same value?

Let’s call back to how we did this for 2 qubits:

1. Use a Hadamard gate to put a single qubit in a superposition of $|0\rangle$& $|1\rangle$
2. Use a CNOT gate to tie that state to a second qubit

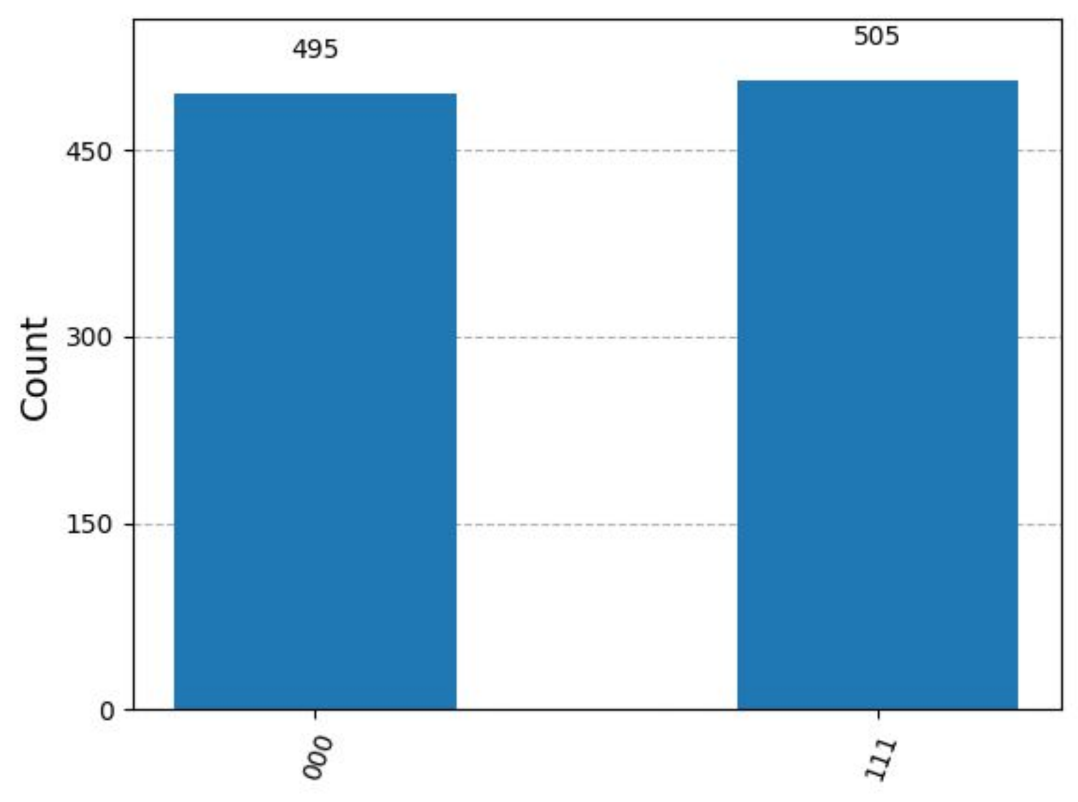
Can we extend this to 3? Give it a try

The Correct Circuit



Why does this work?

Measurement Output



Another Application: Quantum Teleportation

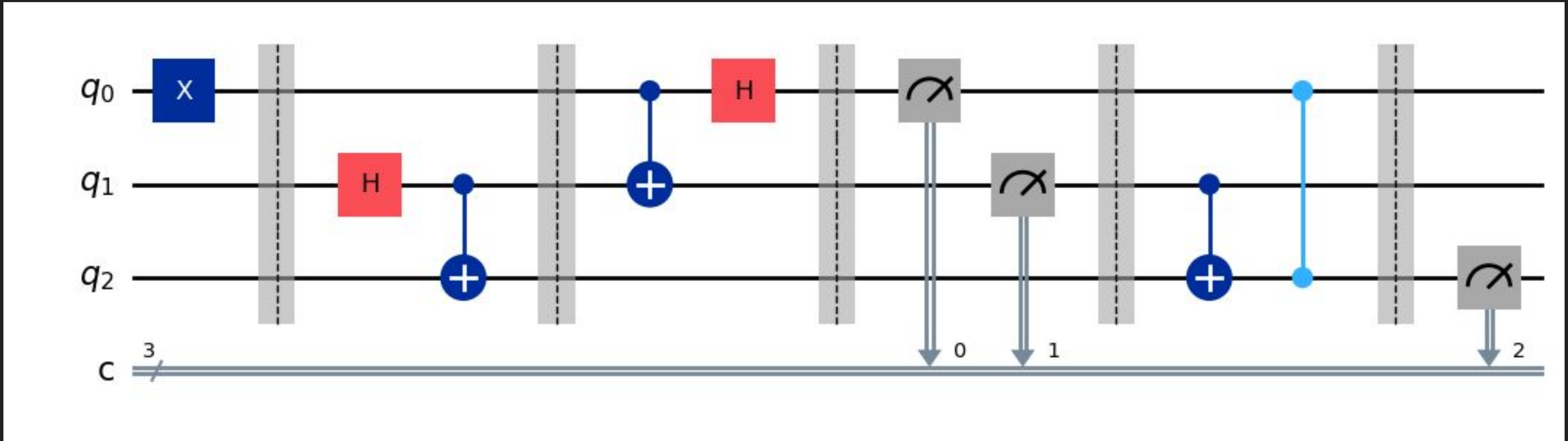
- What if one user wanted to send the state of a qubit to some other user?
- **No Cloning Theorem**: It's impossible to create an independent copy of a quantum state
    - Why not? "Moving" the state would require disrupting it, meaning it loses its quantumness
- What do we do instead? Teleport!
- **Quantum Teleportation**: Destroy the original quantum state and recreate it elsewhere
- What does this mean for Qiskit?
    - Send a quantum state from one register to another

Quantum Teleportation in Qiskit

Let's try it together!

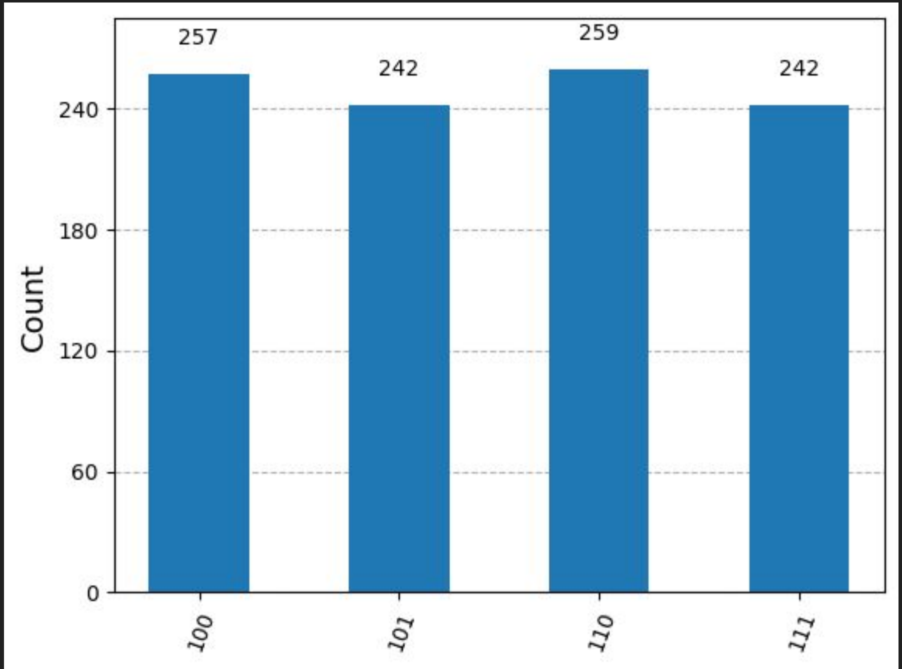
- Create a state on the first quantum register to teleport
- Create two more registers to entangle and create a destination for teleportation
- Measure the circuits
    - If everything went right, the **third** register will always show the state we set up on the **first** register.

The Teleportation Circuit



Try to create it with a partner!

Example Output

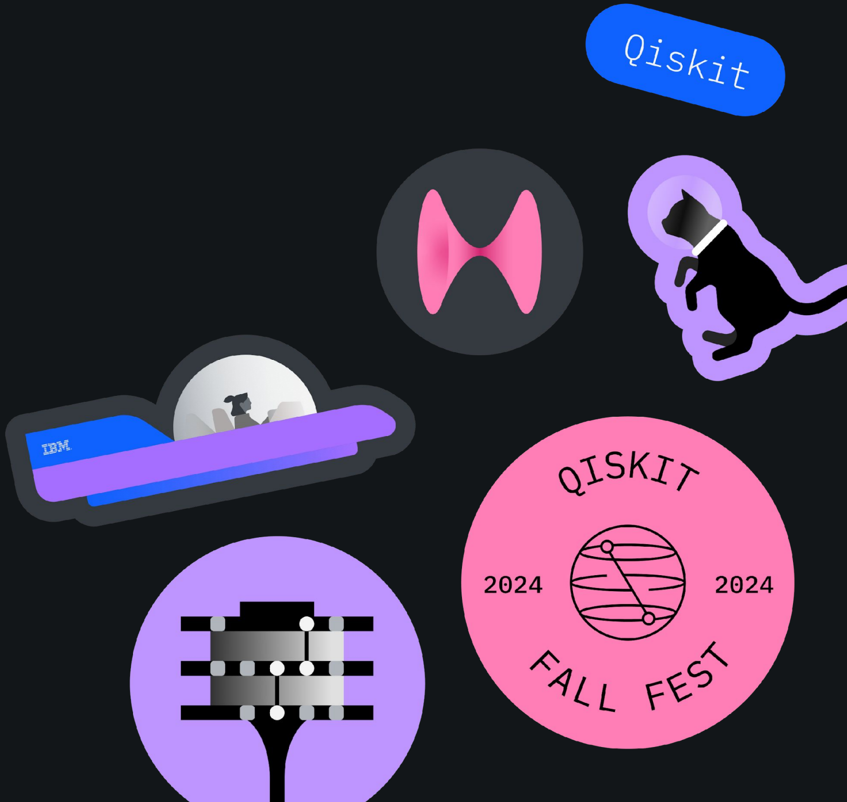


The third register always has the state we initialized in the first register!

For more tutorials and practice:

https://learning.quantum.ibm.com/course/basics-of-quantum-information/single-systems

Thank you!



Qiskit: Single Systems

Qiskit is an open-source quantum computing platform developed by IBM, which uses the Python programming language. Packages can be imported from both software kits.

```
from numpy import array
```

In Python, matrix and vector computations can be performed using the array class from the NumPy library.

```
from qiskit.quantum_info import Statevector
```

Qiskit's Statevector class provides functionality for defining and manipulating quantum state vectors.

Qiskit: Single Systems

Creating a Statevector:

```
v = Statevector([(1 + 2.0j) / 3, -2 / 3])
v.draw("latex")
```

Output: $\left(\frac{1}{3} + \frac{2i}{3}\right)|0\rangle - \frac{2}{3}|1\rangle$

Next, running `v.measure()` simulates a standard basis measurement. It returns the result of that measurement, plus the new quantum state of the system after that measurement.

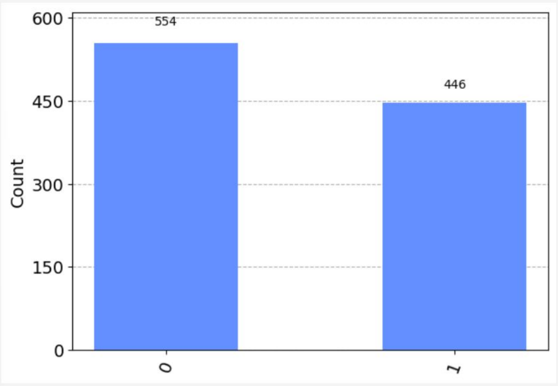
Output:

```
('1', Statevector([ 0.+0.j, -1.+0.j], dims=(2,)))
```

Qiskit: Single Systems

Statevector also comes with a *sample_counts* method that allows for the simulation of any number of measurements on the system. For example, the following cells shows the outcome of measuring the vector *v* 1000 times. You can also visualize the results with *plot_histogram*.

```
{'0': 554, '1': 446}
```




```python
from qiskit.visualization import plot_histogram

statistics = v.sample_counts(1000)
display(statistics)
plot_histogram(statistics)
```

Qiskit: Single Systems

Performing operations with *Operator* **and** *Statevector*

Unitary operations can be defined and performed on state vector in Qiskit.

```
from qiskit.quantum_info import Operator

X = Operator([[0, 1], [1, 0]])
Y = Operator([[0, -1.0j], [1.0j, 0]])
Z = Operator([[1, 0], [0, -1]])
H = Operator([[1 / sqrt(2), 1 / sqrt(2)], [1 / sqrt(2), -1 / sqrt(2)]])
```

Qiskit: Multiple Systems

https://learning-api.quantum.ibm.com/assets/0d1a10eb-4b31-49fe-82cf-a7069db76d4a

website or slides have info

Qiskit: Quantum Circuits

Digital Stickers

These digital stickers can be used to add small visual elements to presentation content. The 2024 badge is the primary mark if a small element is needed to denote Fall Fest content.

Summer School 2024 Badge

Qiskit
2024

Additional Digital Stickers

Qiskit Fall Fest

Qiskit Fall Fest

9

2024 2024

Quantum Computing

Quantum Computing