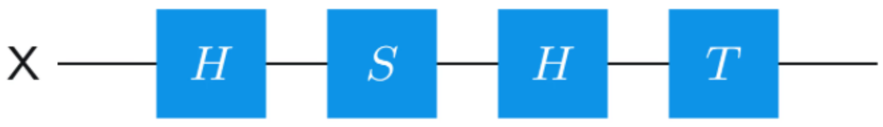
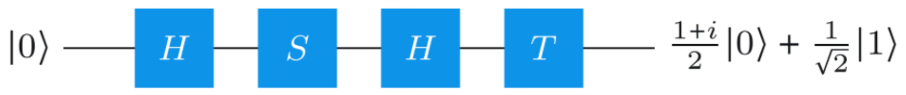
Quantum circuits

In the quantum circuit model, wires represent qubits and gates represent operations acting on these qubits. We'll focus for now on operations we've encountered so far, namely *unitary operations* and *standard basis measurements*. As we learn about other sorts of quantum operations and measurements, we'll enhance our model accordingly.
Here's a simple example of a quantum circuit:

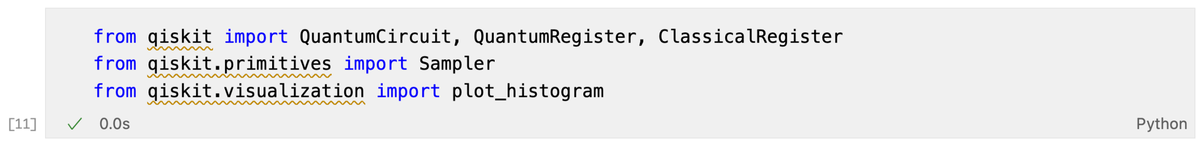


In this circuit, we have a single qubit named X, which is represented by the horizontal line, and a sequence of gates representing unitary operations on this qubit. Just like in the examples above, the flow of information goes from left to right — so the first operation performed is a Hadamard, the second is an S operation, the third is another Hadamard, and the final operation is a T operation. Applying the entire circuit therefore applies the composition of these operations, THSH, to the qubit X.

Sometimes we wish to explicitly indicate the input or output states to a circuit. For example, if we apply the operation $THSH$ to the state $|0\rangle$, we obtain the state $\frac{1+i}{2}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle$. We may indicate this as follows:



Quantum circuits often have all qubits initialized to |0⟩, as we have in this case, but there are also cases where we wish to set the input qubits to different states.
Now let's see how we can specify this circuit in Qiskit, beginning with the imports needed for the current section.



```
from qiskit import QuantumCircuit, QuantumRegister, ClassicalRegister
from qiskit.primitives import Sampler
from qiskit.visualization import plot_histogram
```
[11] ✓ 0.0s Python

To begin, we can create the circuit as follows, by sequentially adding gates from left to right.

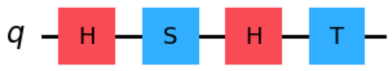
```
circuit = QuantumCircuit(1)

circuit.h(0)
circuit.s(0)
circuit.h(0)
circuit.t(0)

display(circuit.draw(output='mpl'))
```

[4] ✓ 1.6s Python

...



The default names for qubits in Qiskit are q0, q1, q2 etc., and when there is just a single qubit like in our example, the default name is q rather than q0. If we wish to choose our own name we can do this using the Quantum Register class like this:

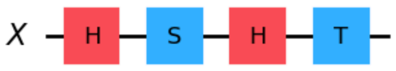
```
X = QuantumRegister(1, "X")
circuit = QuantumCircuit(X)

circuit.h(X)
circuit.s(X)
circuit.h(X)
circuit.t(X)

display(circuit.draw(output='mpl'))
```

[6] ✓ 0.2s Python

...



In this circuit we have a Hadamard gate and a controlled-NOT gate on two qubits X and Y, just like in the previous example. We also have two *classical* bits, AA and B,B, as well as two measurement gates. The measurement gates represent standard basis measurements: the qubits are changed into their post-measurement states, while the measurement outcomes are *overwritten* onto the classical bits to which the arrows point.

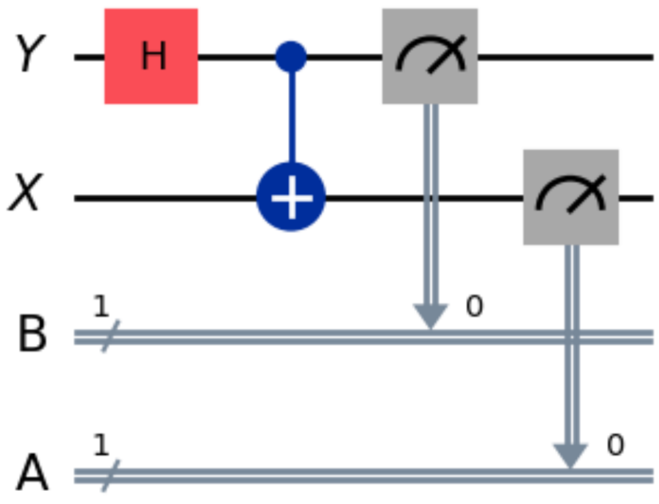
Here is an implementation of this circuit using Qiskit:

```
X = QuantumRegister(1, "X")
Y = QuantumRegister(1, "Y")
A = ClassicalRegister(1, "A")
B = ClassicalRegister(1, "B")

circuit = QuantumCircuit(Y, X, B, A)
circuit.h(Y)
circuit.cx(Y, X)
circuit.measure(Y, B)
circuit.measure(X, A)

display(circuit.draw(output='mpl'))
```

[7] ✓ 0.3s Python

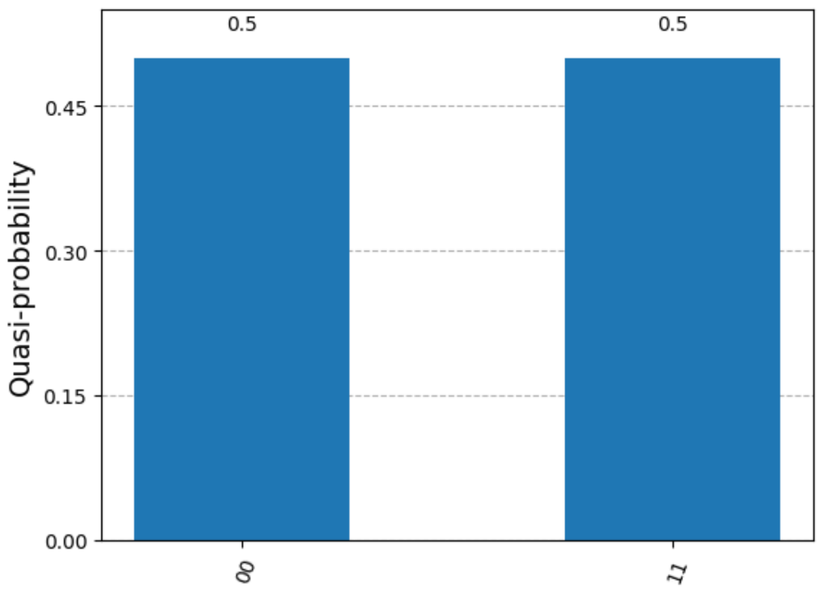


The circuit can be simulated using the Sampler primitive.

```
results = Sampler().run(circuit).result()
statistics = results.quasi_dists[0].binary_probabilities()
display(plot_histogram(statistics))
```

[8] ✓ 0.1s Python

```
... /var/folders/l4/w_211f190ws9m8xgcfy_f_ph0000gn/T/ipykernel_7538/1530918655.py:1: DeprecationWarning: The
  results = Sampler().run(circuit).result()
```



0.5
0.5
Quasi-probability
0.45
0.30
0.15
0.00
00
11