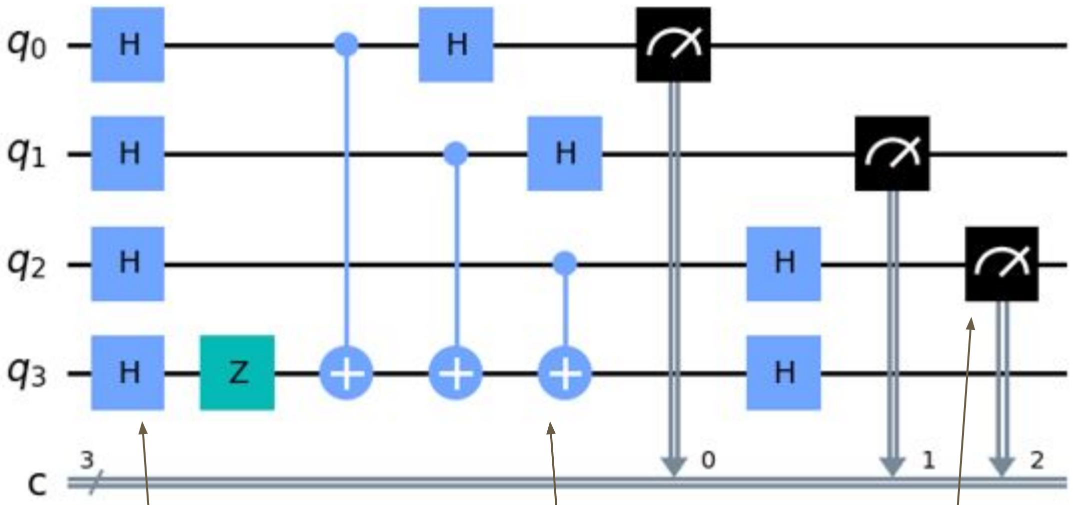
qLearn Week 5: Quantum Coding Crash Course

Michael Silver, ECE2T6

The Quantum Circuit Model of Quantum Algorithms

Qubit registers



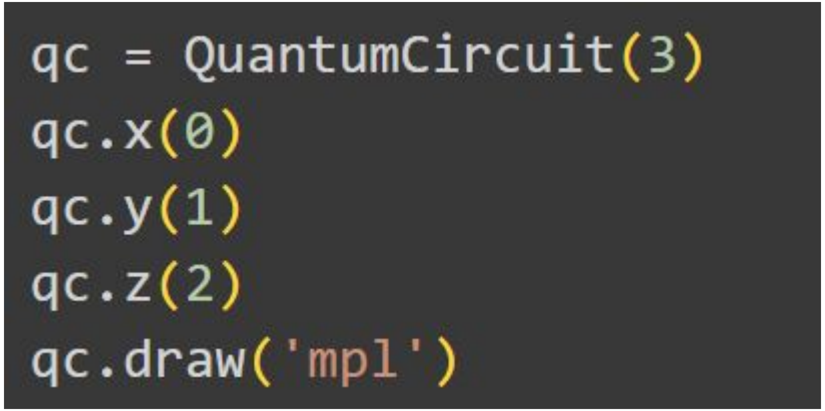
Classical channels

Single qubit gates

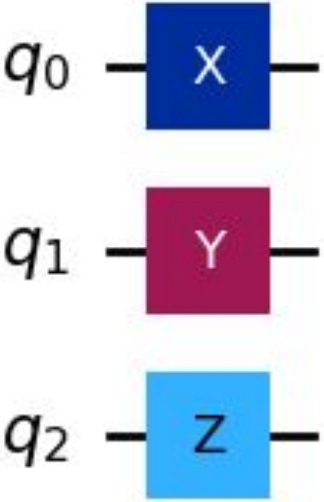
Multi qubit gates

Qubit Measurements

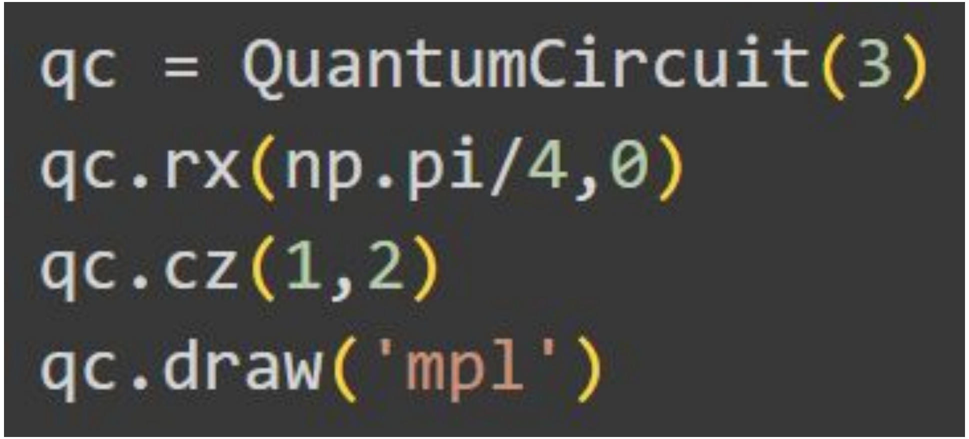
Creating a Quantum Circuit



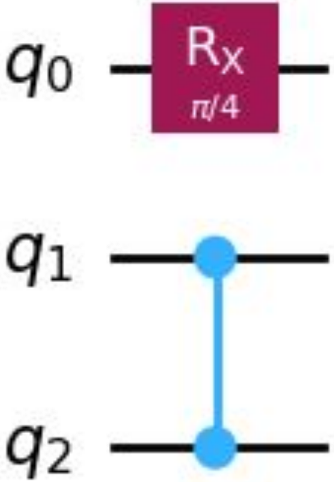
```
qc = QuantumCircuit(3)
qc.x(0)
qc.y(1)
qc.z(2)
qc.draw('mpl')
```



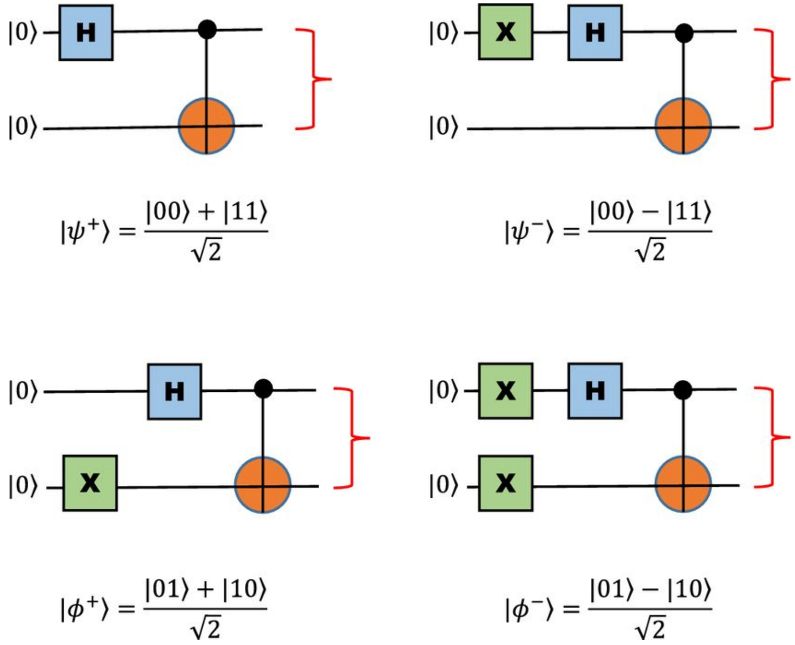
More Quantum Operations



```
qc = QuantumCircuit(3)
qc.rx(np.pi/4,0)
qc.cz(1,2)
qc.draw('mpl')
```



Some Useful Quantum States



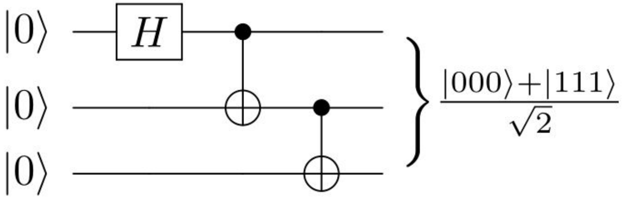
$$|\psi^+\rangle = \frac{|00\rangle + |11\rangle}{\sqrt{2}}$$

$$|\psi^-\rangle = \frac{|00\rangle - |11\rangle}{\sqrt{2}}$$

$$|\phi^+\rangle = \frac{|01\rangle + |10\rangle}{\sqrt{2}}$$

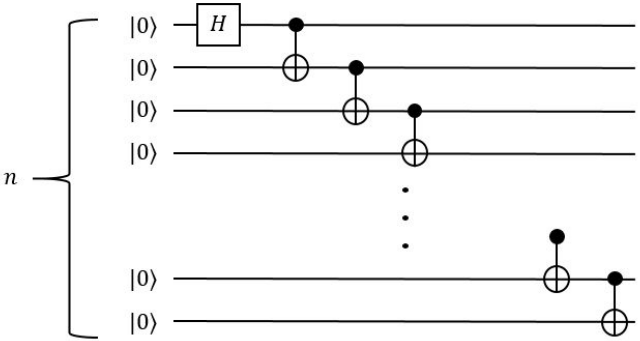
$$|\phi^-\rangle = \frac{|01\rangle - |10\rangle}{\sqrt{2}}$$

'Bell Pair' States + Circuits



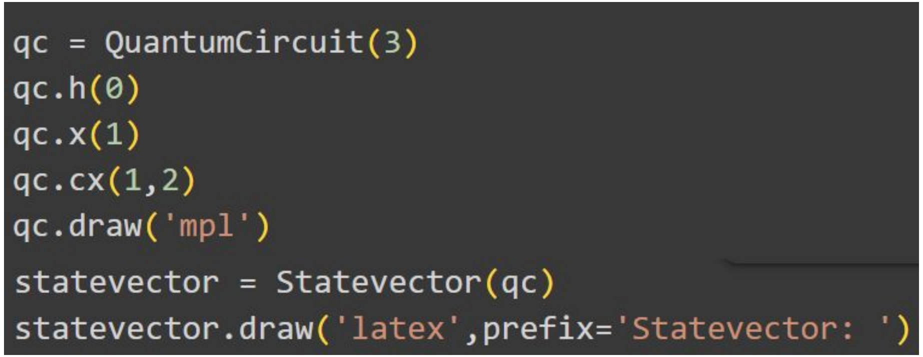
$$\frac{|000\rangle + |111\rangle}{\sqrt{2}}$$

3-qubit GHZ state

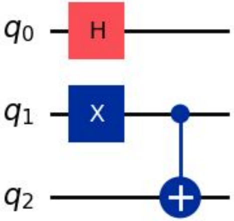


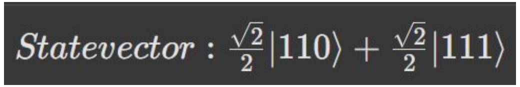
N-qubit GHZ state

Types of Circuit Outputs

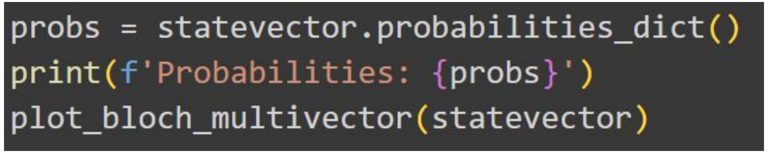


```
qc = QuantumCircuit(3)
qc.h(0)
qc.x(1)
qc.cx(1,2)
qc.draw('mpl')
statevector = Statevector(qc)
statevector.draw('latex',prefix='Statevector: ')
```

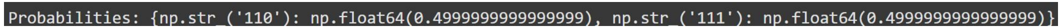




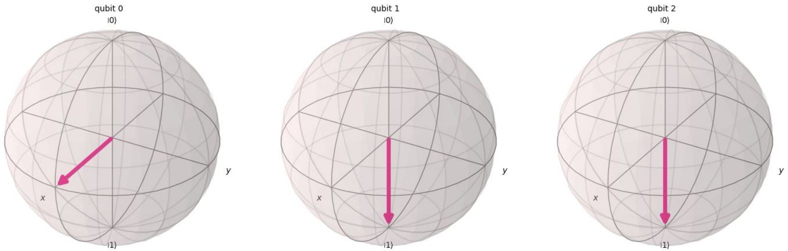
$$Statevector: \frac{\sqrt{2}}{2}|110\rangle + \frac{\sqrt{2}}{2}|111\rangle$$



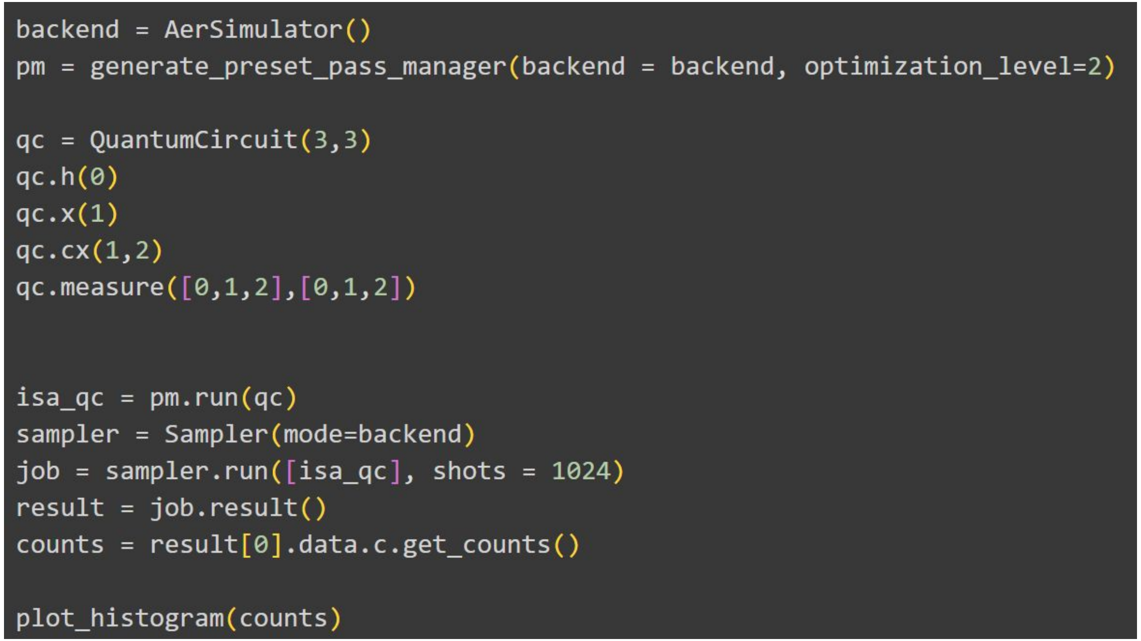
```
probs = statevector.probabilities_dict()
print(f'Probabilities: {probs}')
plot_bloch_multivector(statevector)
```



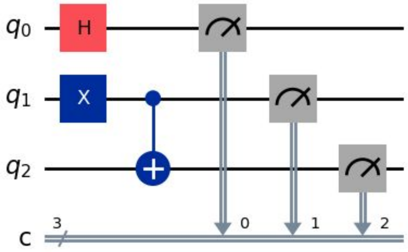
```
Probabilities: {np.str_('110'): np.float64(0.4999999999999999), np.str_('111'): np.float64(0.4999999999999999)}
```

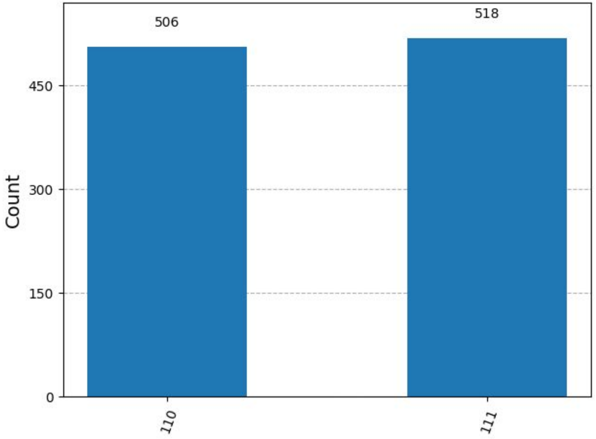


Simulating a Quantum Circuit in Qiskit

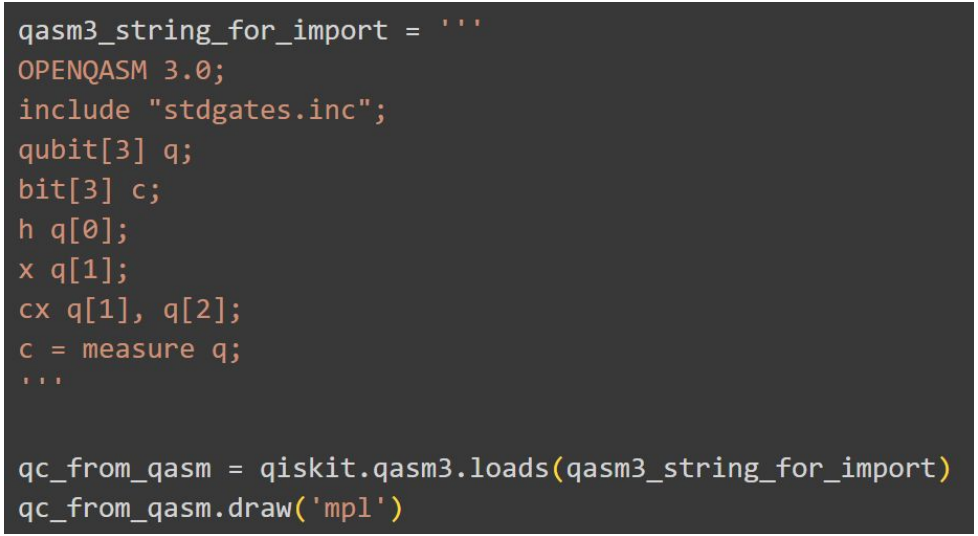


```
backend = AerSimulator()
pm = generate_preset_pass_manager(backend = backend, optimization_level=2)

qc = QuantumCircuit(3,3)
qc.h(0)
qc.x(1)
qc.cx(1,2)
qc.measure([0,1,2],[0,1,2])


isa_qc = pm.run(qc)
sampler = Sampler(mode=backend)
job = sampler.run([isa_qc], shots = 1024)
result = job.result()
counts = result[0].data.c.get_counts()

plot_histogram(counts)
```

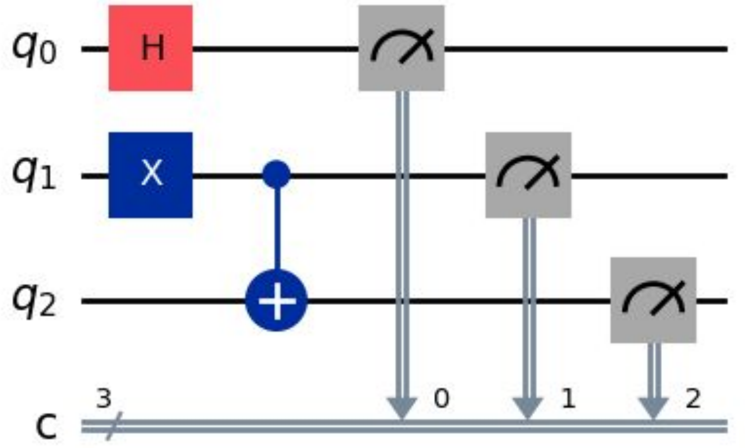




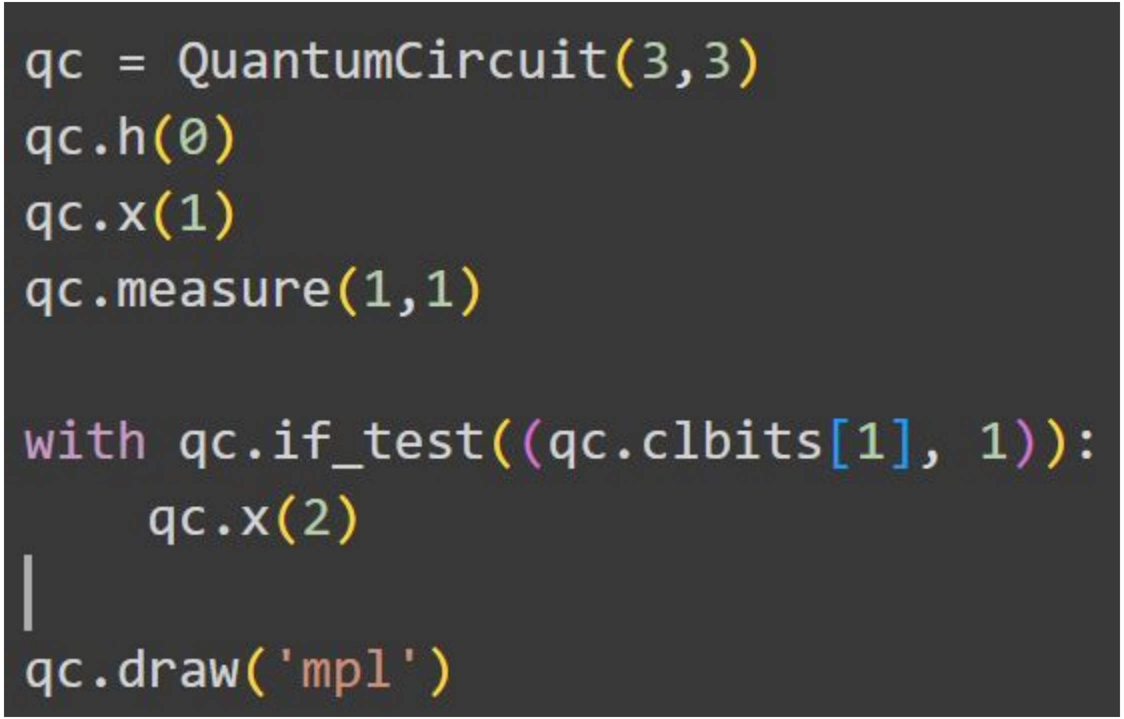
Low-Level Quantum Code: QASM 3



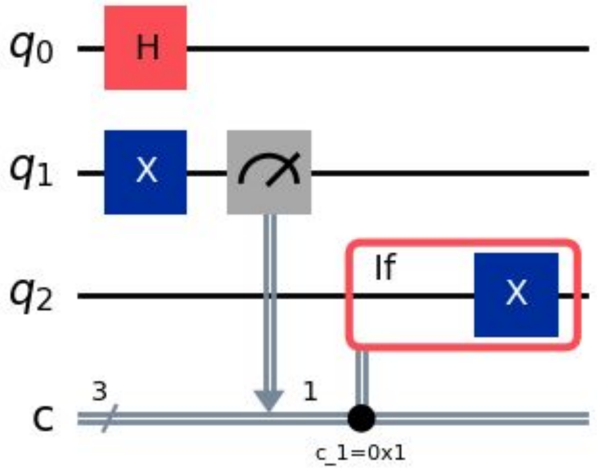
```
qasm3_string_for_import = '''
OPENQASM 3.0;
include "stdgates.inc";
qubit[3] q;
bit[3] c;
h q[0];
x q[1];
cx q[1], q[2];
c = measure q;
'''

qc_from_qasm = qiskit.qasm3.loads(qasm3_string_for_import)
qc_from_qasm.draw('mpl')
```



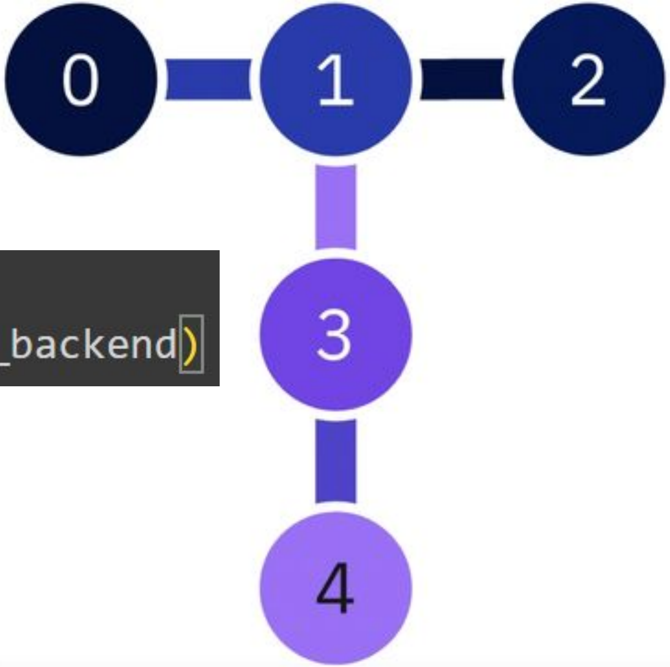
Dynamic Circuits - Classically Controlled Quantum



```
qc = QuantumCircuit(3,3)
qc.h(0)
qc.x(1)
qc.measure(1,1)

with qc.if_test((qc.clbits[1], 1)):
    qc.x(2)

qc.draw('mpl')
```

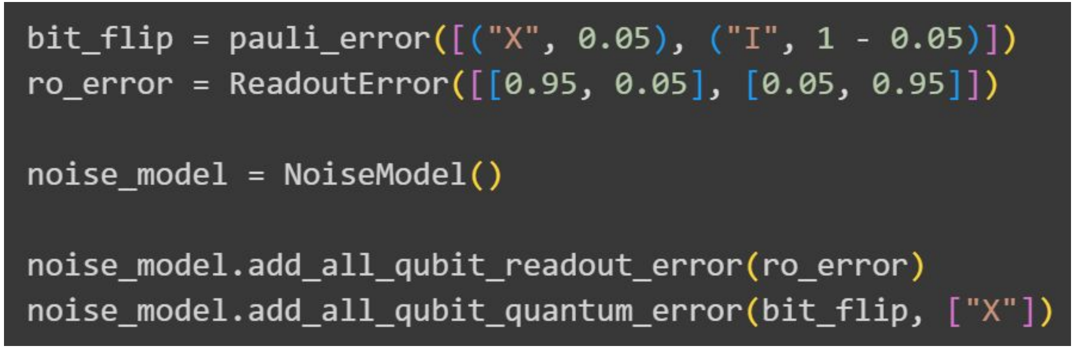


Noisy Simulation

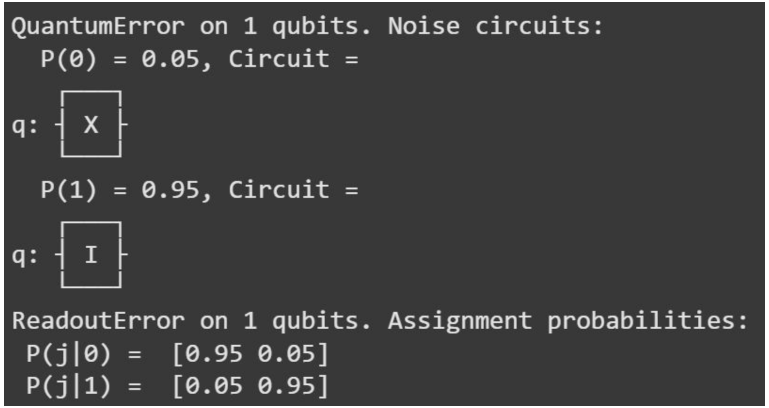


```
fake_backend = FakeVigoV2()
noisy_backend = AerSimulator.from_backend(fake_backend)
```

Noisy Simulation



```python
bit_flip = pauli_error([("X", 0.05), ("I", 1 - 0.05)])
ro_error = ReadoutError([[0.95, 0.05], [0.05, 0.95]])

noise_model = NoiseModel()

noise_model.add_all_qubit_readout_error(ro_error)
noise_model.add_all_qubit_quantum_error(bit_flip, ["X"])
```



```
QuantumError on 1 qubits. Noise circuits:
  P(0) = 0.05, Circuit =
   ┌───┐
q: ┤ X ├
   └───┘
  P(1) = 0.95, Circuit =
   ┌───┐
q: ┤ I ├
   └───┘
ReadoutError on 1 qubits. Assignment probabilities:
 P(j|0) =  [0.95 0.05]
 P(j|1) =  [0.05 0.95]
```