# Programming Assignment #2

MAT 399A, Fall 2025

*Due Friday, September 26*

## Qiskit quantum circuit package

A quantum circuit is created using the `QuantumCircuit` class. For a pure quantum circuit, which has no measurement:

```
<circuit> = qiskit.QuantumCircuit(<n>)
```

where `<n>` is the number of qubit wires to use. A practical quantum circuit also has associated classical bits, which will be the result of measuring the output of the circuit. We will address this in the next assignment. In this assignment we will only consider unmeasured circuits.

### Basic quantum operations

There are many quantum circuit gates (operations). However, the circuits we will use only use a small collection of these.

*Pauli and Hadamard gates.* The Pauli matrices $X, Y, Z$ and the Hadamard matrix $H$ can be applied to a single qubit wire. To add a single qubit gate to a quantum circuit:

```
<circuit>.<x|y|z|h>(<i>)
```

where `<i>` is the qubit wire number to apply the gate to. A gate can also be applied to multiple qubits:

```
<circuit>.<x|y|z|h>(<list>)
```

where `<list>` is a list of qubit wire numbers.

*Controlled X gates.* An $X$ gate on one wire can be controlled by another wire:

```
<circuit>.cx(<i>,<j>)
```

where `<i>` is the controlling wire, and `<j>` is the wire with the target $X$ gate. More generally, a multi–controlled $X$ gate gate (also called a multi–controlled Toffoli gate, or simply a Toffoli gate) uses a list `<list>` of controlling wire numbers:

```
<circuit>.mcx(<list>,<j>)
```

*Phase shift gates.* A phase shift gate $P(\theta)$ applies a phase shift of $\theta$ to a target qubit wire:

```
<circuit>.p(<theta>,<i>)
```

where `<theta>` is the phase shift angle (measured in radians) and `<i>` is the qubit wire number to apply the phase shift. There is also a controlled version of a phase shift gate that is used in the quantum Fourier transform.

*Barriers.* Sections of a quantum circuit can be separated by using a barrier, which affects all wires:

```
<circuit>.barrier()
```

Note that this is not the same as a barrier in the sense of parallel computation. It is simply a visual device (although it will also affect circuit optimization).

## Qiskit conventions

There are two conventions used by Qiskit to be aware of.

*Wire order.* Ordering is from top to bottom. For a circuit with $n$ qubits, qubit 0 is on the top, and qubit $(n-1)$ is on the bottom.

*Basis order.* Computational basis vectors use big–endian order. That is, for integer $I$ with binary representation

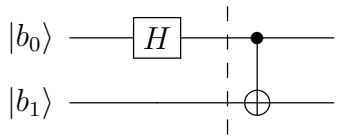
$$I = b_{n-1} \cdots b_1 b_0 \quad (= 2^{n-1} b_{n-1} + \cdots 2^1 b_1 + 2^0 b_0)$$

the corresponding computational basis vector is

$$|I\rangle = |b_{n-1}\rangle \cdots |b_1\rangle \, |b_0\rangle \, .$$

## An example

```
import qiskit
qc = qiskit.QuantumCircuit(2)
qc.h(0)
qc.barrier()
qc.cx(0,1)
```




The `qc.draw()` call can be used to draw the quantum circuit, but the result will not be quite as pretty as the circuit depiction on the right. Note that the $\oplus$ symbol on wire number 1 is the same as a Pauli $X$ gate. Although the $\oplus$ symbol, which also is used to denote addition modulo 2, is perhaps more common in the literature.

The net unitary transformation that the circuit realizes is the composition of unitaries

$$U = CNOT_{0,1} \circ I \otimes H. \tag{1}$$

($CNOT_{0,1}$ is the controlled $X$ gate with wire 0 the control, and wire 1 the target). The factor on the right is the transformation

$$I \otimes H : |b_1\rangle \otimes |b_0\rangle \mapsto |b_1\rangle \otimes H\,|b_0\rangle\,, \quad H = \tfrac{1}{\sqrt{2}} \left(\begin{smallmatrix} 1 & 1 \\ 1 & -1 \end{smallmatrix}\right).$$

To obtain the $2 \times 2$ matrix for $I \otimes H$, we compute its effect on each element of the computational basis:

$$\begin{aligned}
|00\rangle &= |0\rangle \otimes |0\rangle \mapsto |0\rangle \otimes \tfrac{1}{\sqrt{2}}(|0\rangle + |1\rangle) = \tfrac{1}{\sqrt{2}}(|0\rangle \otimes |0\rangle + |0\rangle \otimes |1\rangle) = \tfrac{1}{\sqrt{2}}(|00\rangle + |01\rangle) \\
|01\rangle &= |0\rangle \otimes |1\rangle \mapsto |0\rangle \otimes \tfrac{1}{\sqrt{2}}(|0\rangle - |1\rangle) = \tfrac{1}{\sqrt{2}}(|0\rangle \otimes |0\rangle - |0\rangle \otimes |1\rangle) = \tfrac{1}{\sqrt{2}}(|00\rangle - |01\rangle) \\
|10\rangle &= |1\rangle \otimes |0\rangle \mapsto |1\rangle \otimes \tfrac{1}{\sqrt{2}}(|0\rangle + |1\rangle) = \tfrac{1}{\sqrt{2}}(|1\rangle \otimes |0\rangle + |1\rangle \otimes |1\rangle) = \tfrac{1}{\sqrt{2}}(|10\rangle + |11\rangle) \\
|11\rangle &= |1\rangle \otimes |1\rangle \mapsto |1\rangle \otimes \tfrac{1}{\sqrt{2}}(|0\rangle - |1\rangle) = \tfrac{1}{\sqrt{2}}(|1\rangle \otimes |0\rangle - |1\rangle \otimes |1\rangle) = \tfrac{1}{\sqrt{2}}(|10\rangle - |11\rangle).
\end{aligned}$$

These form the columns of $I \otimes H$ in the stated order. I.e.,

$$I \otimes H = \tfrac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 & 0 & 0 \\ 1 & -1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & -1 \end{pmatrix}.$$

The left factor in equation (1) is the transformation

$$CNOT_{0,1} : |b_1\rangle \otimes |b_0\rangle \mapsto X^{b_0}\,|b_1\rangle \otimes |b_0\rangle = |b_0 \oplus b_1\rangle \otimes |b_0\rangle\,, \quad X = \left(\begin{smallmatrix} 0 & 1 \\ 1 & 0 \end{smallmatrix}\right).$$

And again we compute its effect on the computational basis elements.

$$\left.\begin{aligned}
|00\rangle &= |0\rangle \otimes |0\rangle \mapsto |0 \oplus 0\rangle \otimes |0\rangle = |0\rangle \otimes |0\rangle = |00\rangle \\
|01\rangle &= |0\rangle \otimes |1\rangle \mapsto |0 \oplus 1\rangle \otimes |1\rangle = |1\rangle \otimes |1\rangle = |11\rangle \\
|10\rangle &= |1\rangle \otimes |0\rangle \mapsto |1 \oplus 0\rangle \otimes |0\rangle = |1\rangle \otimes |0\rangle = |10\rangle \\
|11\rangle &= |1\rangle \otimes |1\rangle \mapsto |1 \oplus 1\rangle \otimes |1\rangle = |0\rangle \otimes |1\rangle = |01\rangle
\end{aligned}\right\} \Rightarrow CNOT_{0,1} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix}.$$

The unitary matrix that the quantum circuit realizes is then

$$U = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix} \tfrac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 & 0 & 0 \\ 1 & -1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & -1 \end{pmatrix} = \tfrac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & -1 \\ 0 & 0 & 1 & 1 \\ 1 & -1 & 0 & 0 \end{pmatrix}.$$

# Programming tasks

1. Construct a quantum circuit with 2 qubits (and no classical bits) that realizes the following composition of unitary transformations

$$U = H \otimes I \circ CNOT_{1,0} \circ I \otimes Y \circ H \otimes H. \tag{2}$$

Where $CNOT_{1,0}$ is the controlled $X$ gate with wire 1 the control wire and wire 0 the target. I.e.,

$$CNOT_{1,0} : |b_1\rangle \otimes |b_0\rangle \mapsto |b_1\rangle \otimes |b_0 \oplus b_1\rangle .$$

The function to implement has the form

```
def Q1():
  """
  returns:
    qiskit quantum circuit on 2 qubits that realizes U
  """
```

2. Compute the $4 \times 4$ unitary matrix for $U$ in the qiskit computational basis.

```
def Q2():
  """
  returns:
    numpy matrix for U
  """
```

You will need to hard–code at least some of the matrices involved. While you might be able to massage the `numpy` library functions to compute the tensor products in equation (2), it is probably easier to compute the matrices on paper and hard–code the results. Alternatively, you can write a function to compute the tensor product of two (or more) matrices. In any case, you should make use of matrix multiplication in `numpy` to compute the compositions in equation (2). While you may hard–code the *individual* matrices in the composition (2), you may **not** hard–code the entire matrix for $U$.

3. Compute the computational basis probabilities (these will be the classical register value probabilities) for the output of the quantum circuit for $U$ for an input state

$$|\psi\rangle = A_{00} |00\rangle + A_{01} |01\rangle + A_{10} |10\rangle + A_{11} |11\rangle$$

where it is assumed that $\psi$ is normalized (you do not need to check this):

$$|A_{00}|^2 + |A_{01}|^2 + |A_{10}|^2 + |A_{11}|^2 = 1.$$

```
def Q3(A00,A01,A10,A11):
  """
  where:
    A00,A01,A10,A11 : complex amplitudes of initial qubit register state
  returns:
    list [prob00,prob01,prob10,prob11] where probXY is the probability
    that the final state is |XY> (X,Y=0,1)
  """
```

## What to turn in

The code for all of the above functions should be put in a single file named `exercise2.py`. You may only import the packages `numpy`, `qiskit`, `math`, and `cmath`. However, you may **not** use any of the functionality in `qiskit.quantum_info` class!

## Test driver

A simple test driver `exercise2_test.py` has been uploaded for convenience in testing. If you run

```
python exercise2_test.py
```

the output should match the contents of the file `exercise2_test.output.txt` to within reasonable numerical precision.