Introduction to Qiskit

累了嗎 看個短片好嗎?

- https://www.youtube.com/watch?v=JhHMJCUmq28

Installation

- Step 1: Install Anaconda

- Step 2: Open Anaconda Prompt

- Step 3: Run the following comments:

  conda create -n env_name python=3 jupyter

  conda active env_name

https://qiskit.org/documentation/install.html#install

Installation

- Step 4 : Install Qiskit packages

  pip install qiskit

- Step 5: Open jupyter notebook

  jupyter notebook

- Step 6: Get started!

https://qiskit.org/documentation/install.html#install

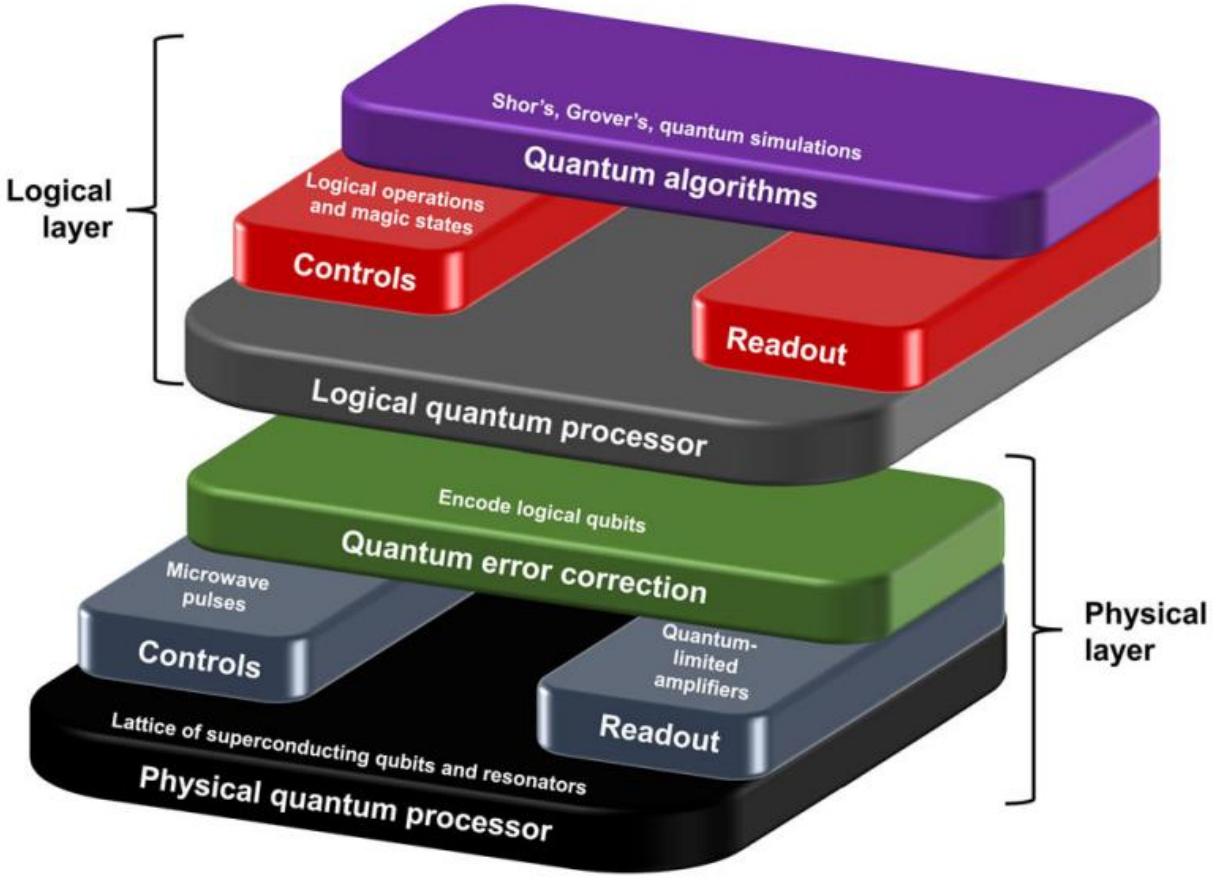
Google Colab

- !pip install qiskit

Only For Summer Course

- 140.112.2.67:2045
- 140.112.2.68:2045 (備用)
- 台大ip下可使用
- 密碼: summerq

Quantum Computing Stack

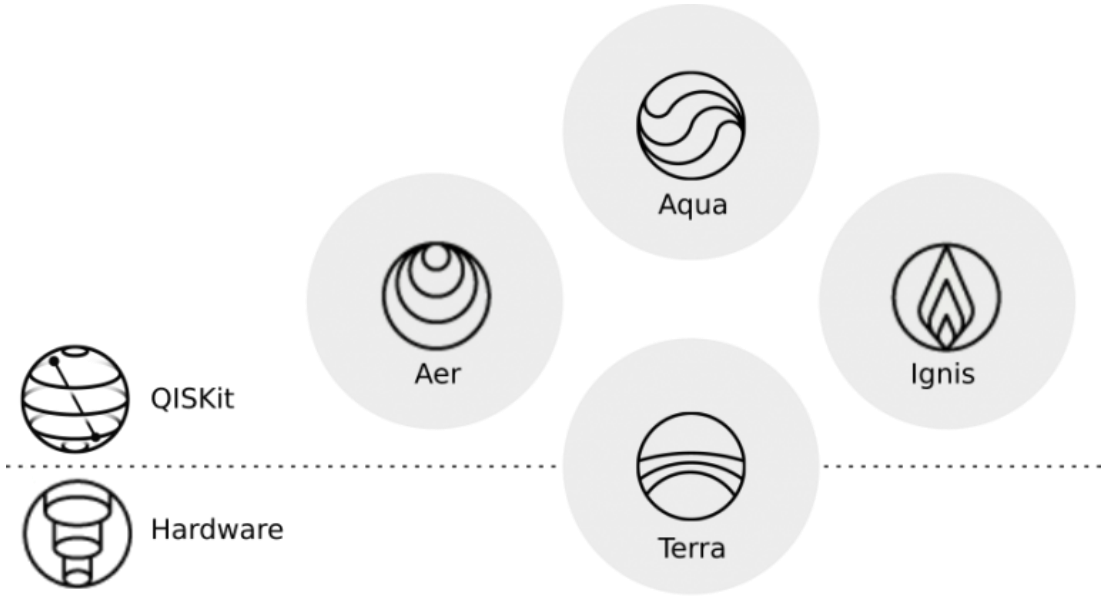


Source: Gambetta, J.M., Chow, J.M. & Steffen, M. Building logical qubits in a superconducting quantum computing system. *npj Quantum Inf* **3,** 2 (2017).

Four Elements of Life

Element Ambigrams from Angels & Demons

The Qiskit Elements



Terra, the ‘earth’ element, is the foundation on which the rest of the software lies.

Aer, the ‘air’ element, permeates all Qiskit elements. For accelerating development via simulators, emulators and debuggers

Aqua, the ‘water’ element, is the element of life. For building algorithms and applications.

Ignis, the ‘fire’ element, is dedicated to fighting noise and errors and to forging a new path

Qiskit Terra

- Terra, the ‘earth’ element, is the foundation on which the rest of Qiskit lies.
- Qiskit Terra is organized in six main modules:
    1. qiskit.circuit
    2. qiskit.pulse
    3. qiskit.transiler
    4. qiskit.providers
    5. qiskit.quantum_info
    6. qiskit.visualization

Qiskit Aer

- Aer, the ‘air’ element, permeates all Qiskit elements.
- Qiskit Aer includes three high performance simulator backends:

- `QasmSimulator`

  Allows ideal and noisy multi-shot execution of qiskit circuits and returns counts or memory. There are multiple methods that can be used that simulate different circuits more efficiently. These include:

  1. ***statevector*** - Uses a dense statevector simulation.
  2. ***stabilizer*** - Uses a Clifford stabilizer state simulator that is only valid for Clifford circuits and noise models.
  3. ***extended_stabilizer*** - Uses an approximate simulator that decomposes circuits into stabilizer state terms, the number of which grows with the number of non-Clifford gates.
  4. ***matrix_product_state*** - Uses a Matrix Product State (MPS) simulator.

- `StatevectorSimulator`

  Allows ideal single-shot execution of qiskit circuits and returns the final statevector of the simulator after application.

- `UnitarySimulator`

  Allows ideal single-shot execution of qiskit circuits and returns the final unitary matrix of the circuit itself. Note that the circuit cannot contain measure or reset operations for this backend.

Qiskit Ignis

- Ignis, the ‘fire’ element, is dedicated to fighting noise and errors and to forging a new path.

- `qiskit.ignis.characterization` Characterization experiments are designed to measure parameters in the system such as noise parameters (T1, T2-star, T2), Hamiltonian parameters such as the ZZ interaction rate and control errors in the gates.
- `qiskit.ignis.verification` Verification experiments are designed to verify gate and small circuit performance. Verification includes state and process tomography, quantum volume and randomized benchmarking (RB). These experiments provide the information to determine performance metrics such as the gate fidelity.
- `qiskit.ignis.mitigation` Mitigation experiments run calibration circuits that are analyzed to generate mitigation routines that can be applied to arbitrary sets of results run on the same backend. Ignis code will generate a list of circuits that run calibration measurements. The results of these measurements will be processed by a Fitter and will output a Filter than can be used to apply mitigation to other results.

Qiskit Aqua

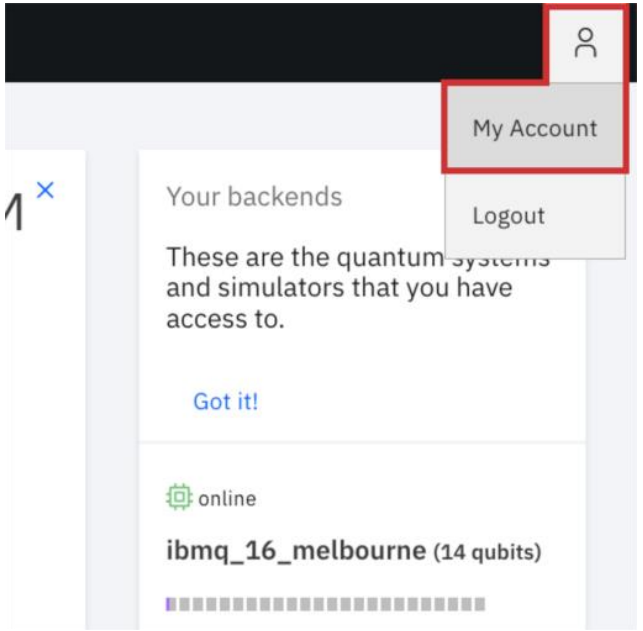
- Aqua, the ‘water’ element, is the element of life.

- Chemistry, Finance, Machine Learning, Optimization.

Access IBM Quantum Systems

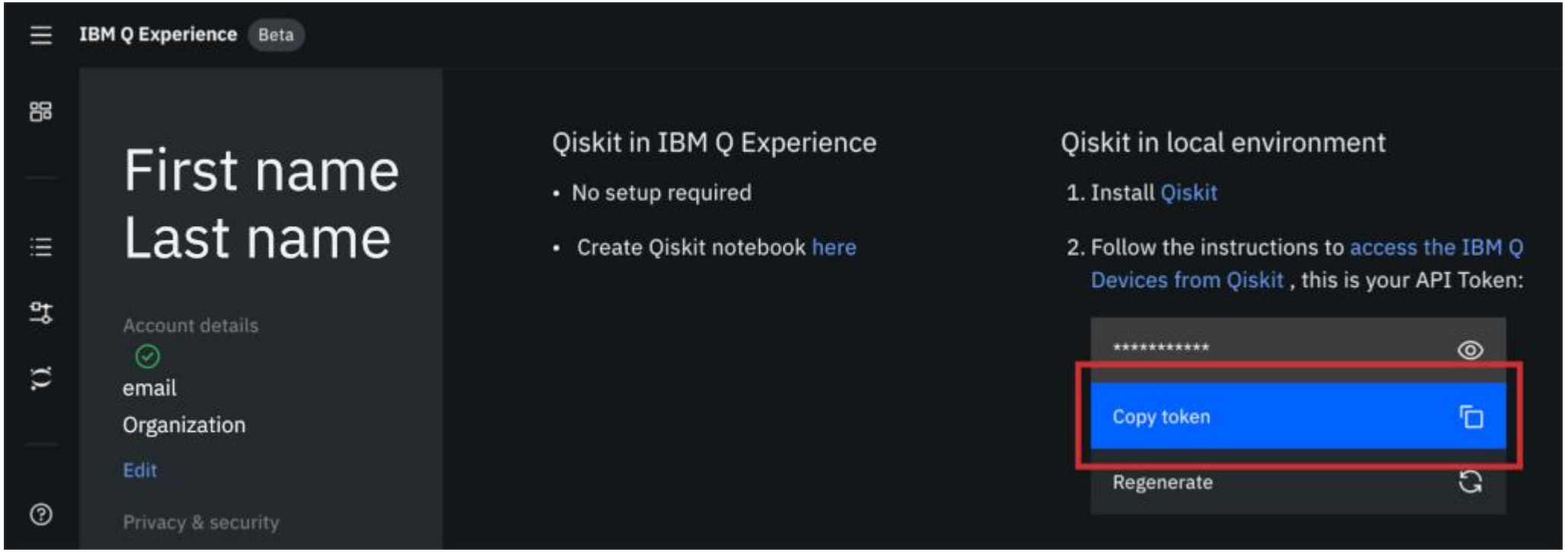
- To configure your account, you create a local configuration file which includes an API key

1. Create a free IBM Quantum Experience account.
2. Navigate to **My Account** to view your account settings.



Access IBM Quantum Systems

3. Click on **Copy token** to copy the token to your clipboard. Temporarily paste this API token into your favorite text editor so you can use it later to create an account configuration file.



Access IBM Quantum Systems

4. Run the following commands to access IBM Quantum Device. Replace the MY_API_TOKEN with the API token value that you stored in your text editor.

```
from qiskit import IBMQ
apitoken='MY_API_TOKEN'
IBMQ.enable_account(token=apitoken)
```

Qubits, Operators and Measurement

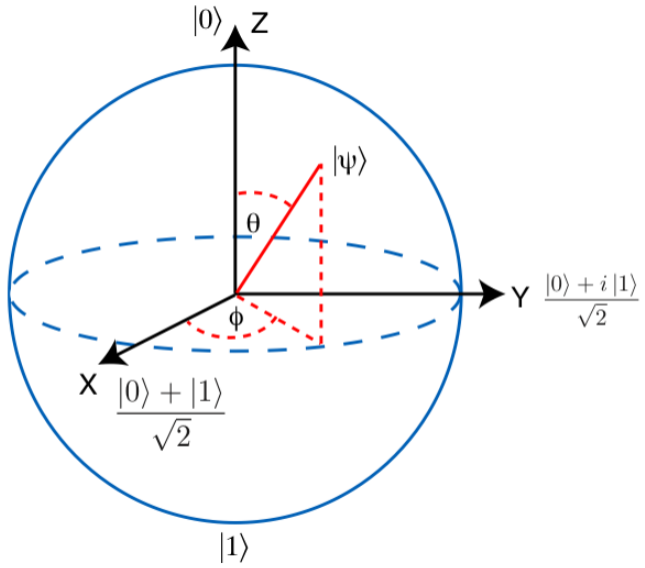
- A quantum computation is a collection of three elements
  ① A quantum register or a set of quantum register.
  ② A unitary matrix, which is used to execute a given quantum algorithm.
  ③ Measurement to extract information we need.

- Quantum circuit model
  - Universal quantum computer

Quantum Gates

- Quantum Operators
  - In gate-based quantum computers, these operator used to evolve the state.
    - Unitary and reversable.
    - Single qubit gate: rotation in block sphere.

- There exist universal quantum gates set.

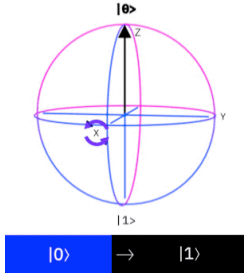


Group Theory Here.

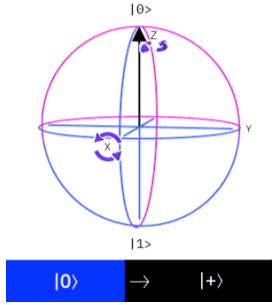
Reference: https://quantum-computing.ibm.com/docs/circ-comp/q-gates

Single Qubit Gates

$$X \equiv \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}; \quad Y \equiv \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix}; \quad Z \equiv \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$$



$$H = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}; \quad S = \begin{bmatrix} 1 & 0 \\ 0 & i \end{bmatrix}; \quad T = \begin{bmatrix} 1 & 0 \\ 0 & \exp(i\pi/4) \end{bmatrix}$$

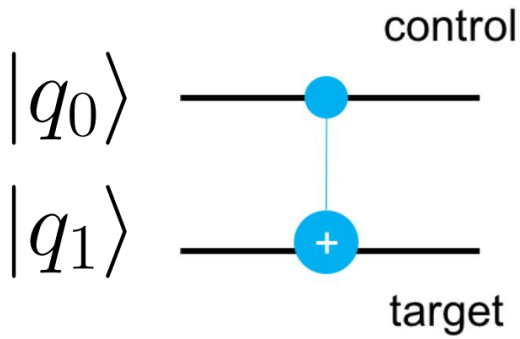


$$u3(\theta, \phi, \lambda) = \begin{pmatrix} \cos(\theta/2) & -e^{i\lambda}\sin(\theta/2) \\ e^{i\phi}\sin(\theta/2) & e^{i\lambda + i\phi}\cos(\theta/2) \end{pmatrix}. \qquad u2(\phi, \lambda) = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & -e^{i\lambda} \\ e^{i\phi} & e^{i(\phi+\lambda)} \end{pmatrix}. \qquad u1(\lambda) = \begin{pmatrix} 1 & 0 \\ 0 & e^{i\lambda} \end{pmatrix}.$$

Quantum Gates in Qiskit

- Very Important!
  - The LSB (Least Significant Bit) is from right to left in qiskit.

$$|q_{n-1}, \cdots, q_1, q_0\rangle = |q_{n-1}\rangle \otimes |q_{n-2}\rangle \otimes \cdots |q_1\rangle \otimes |q_0\rangle$$



Starting state		Ending State
|00>	→	|00>
|10>	→	|10>
|01>	→	|11>
|11>	→	|01>

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}$$

$$|0\rangle\langle 0| \otimes I + |1\rangle\langle 1| \otimes X$$

$$q_0 \otimes q_1$$

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix}$$

$$X \otimes |1\rangle\langle 1| + I \otimes |0\rangle\langle 0|$$

$$q_1 \otimes q_0$$

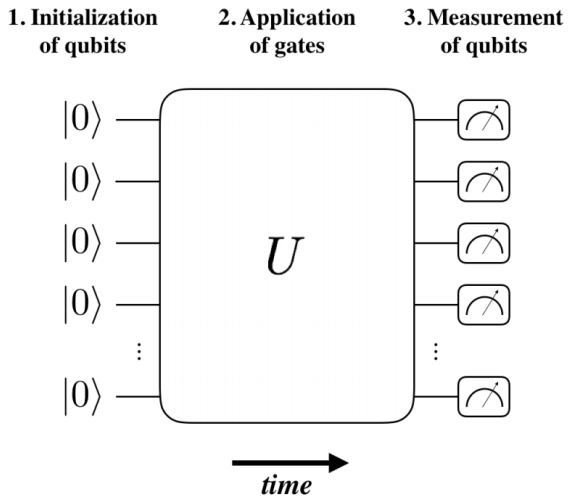
→ Also, this show the difference between entangle and separate state

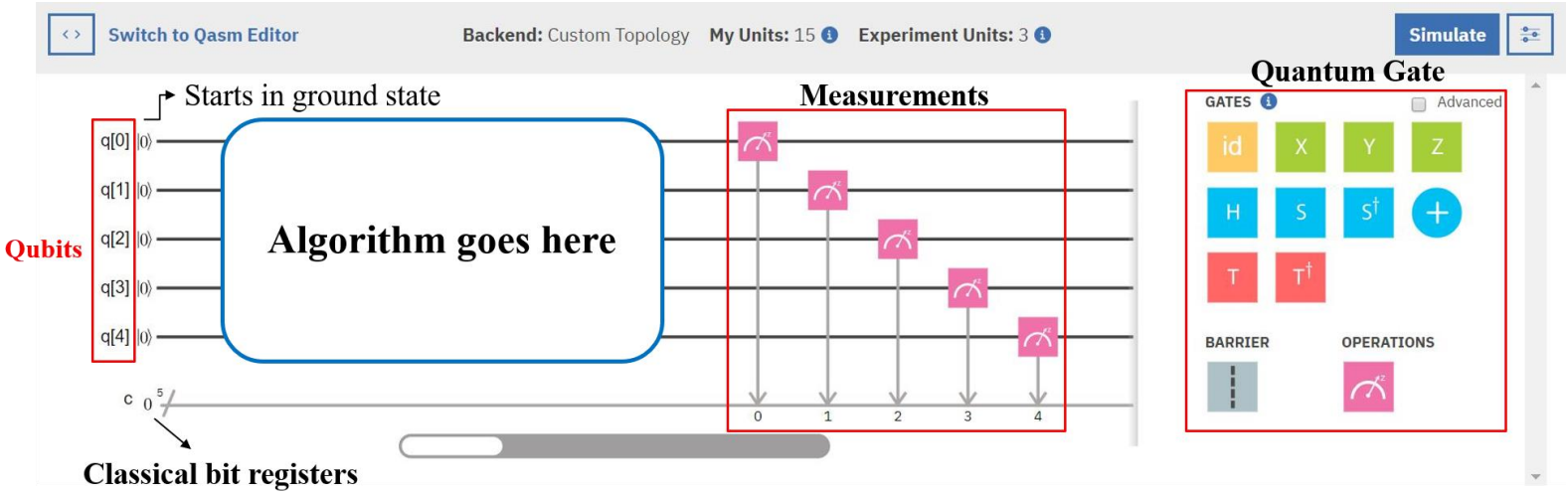
Reference: https://qiskit.org/documentation/stubs/qiskit.circuit.QuantumCircuit.html#qiskit.circuit.QuantumCircuit

Getting Started with Qiskit

- The workflow of using Qiskit consists of three high-level steps:

- **Build**: design a quantum circuit that represents the problem you are considering.
- **Execute**: run experiments on different backends (***which include both systems and simulators***).
- **Analyze**: calculate summary statistics and visualize the results of experiments.





Resource

https://www.mustythoughts.com/resources.html

https://www.youtube.com/c/qiskit/playlists

https://qiskit.org/textbook/preface.html

https://github.com/microsoft/QuantumKatas