

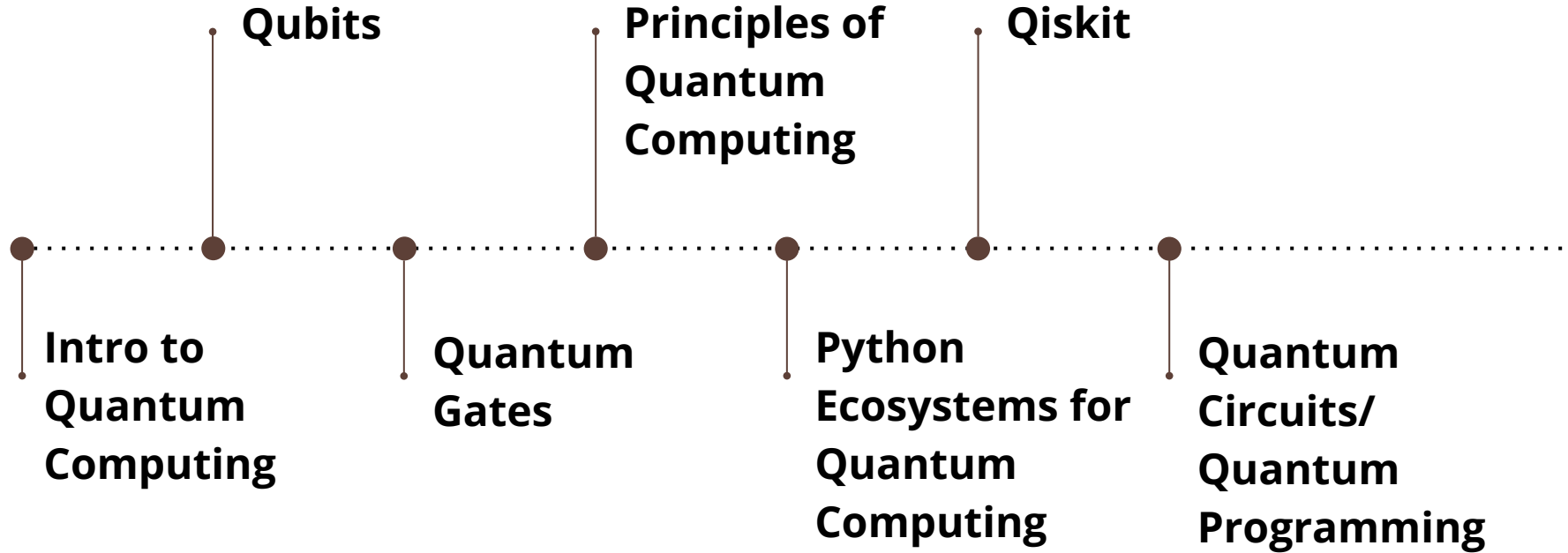
Quantum Computing


Python Deep Dive

**Week 1: Basics of QC
and Qiskit**




Lecture Flow





Introduction to Quantum Computing



Classical Computers

- Uses arrangements of transistors to store bits, which can be one or zero at any point in time.

Quantum Computers

- Uses 'quantum bits', also called qubits, as the building blocks of quantum computers.



Introduction to Qubits



Qubits

- 1 unit of information in quantum computers.
- Exists as a superposition of two values.
- Superposition: a qubit is a linear combination of $|0\rangle$ and $|1\rangle$
- Measuring the qubits 'collapses' the superposition.


- This is how we represent qubits: (symbol of psi)

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$$

$$|0\rangle \equiv \begin{pmatrix} 1 \\ 0 \end{pmatrix}$$

$$|1\rangle \equiv \begin{pmatrix} 0 \\ 1 \end{pmatrix}$$

Qubits

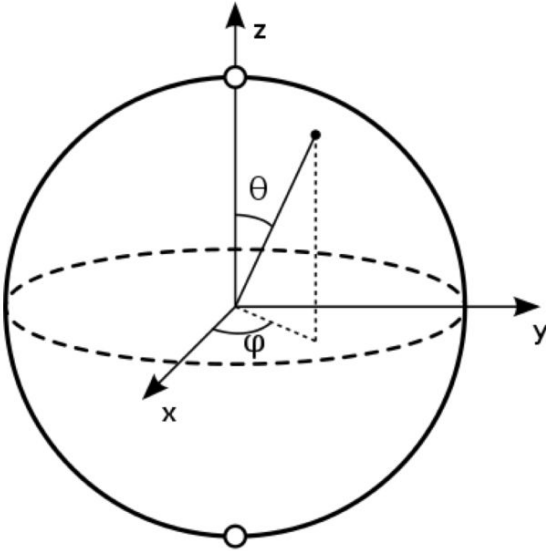
- In the physical world, qubits are quantum mechanical systems at the atomic scale.
 - Qubits are implemented in various ways, the most popular being using the spin of electrons as a qubit.
- 
- Some quantum computers use:
 - Charge of electron
 - Path of photons (optical qc)
 - Superconductors (using the phase of charged particles)
 - Energy levels of atoms.
 - The common property in all these physical realizations is that they are all '2-level systems'.



Bloch Spheres



Locating points on a sphere

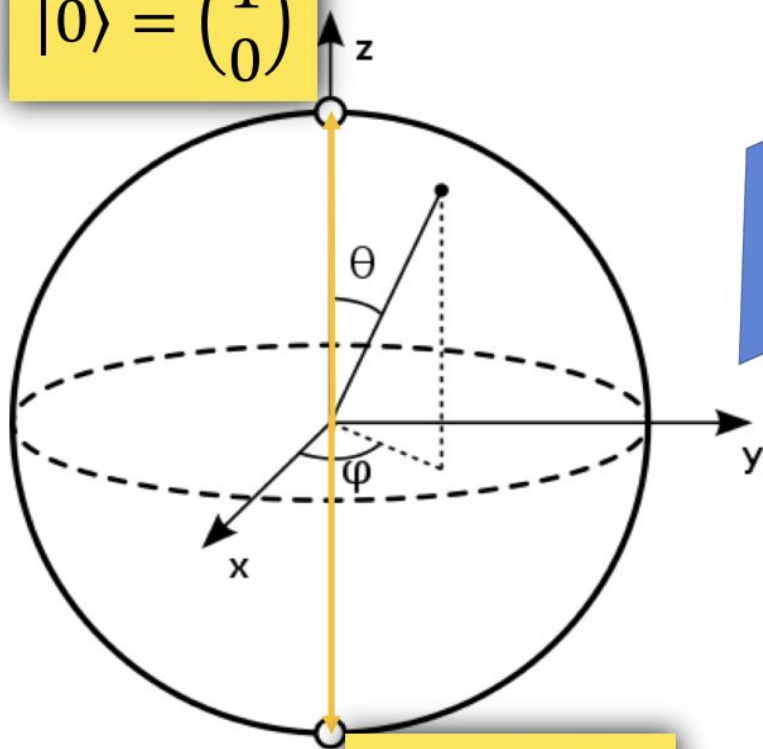


$$\theta \in \{0, \pi\}$$

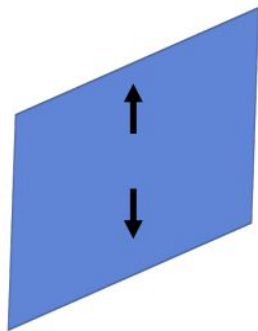
$$\phi \in \{0, 2\pi\}$$

- All single qubit states exist on the 2-D surface of the sphere
- The two angles θ and ϕ give us the phase of the qubit state (its magnitude is always 1)
- The Bloch sphere can also be used to visualize qubit transformations

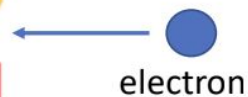
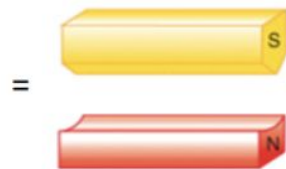
$$|0\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$$

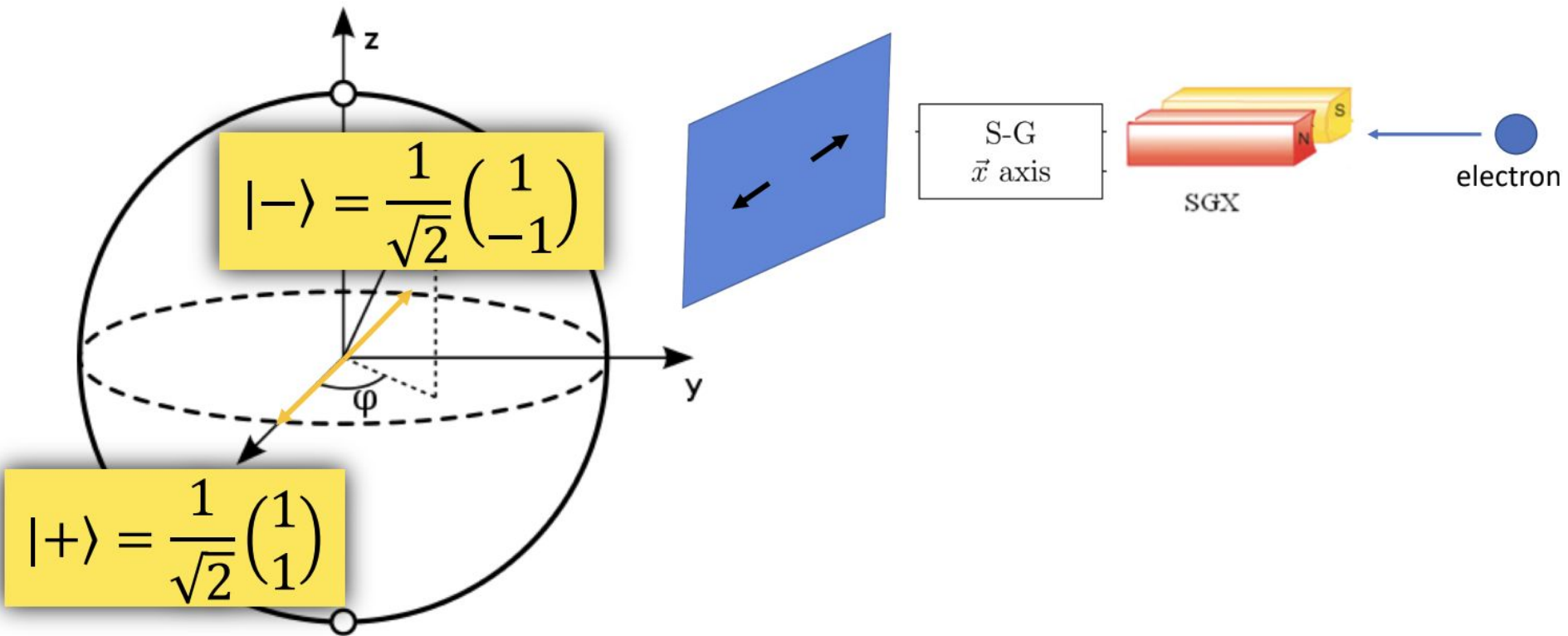


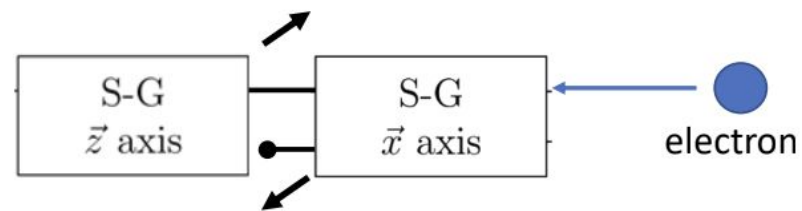
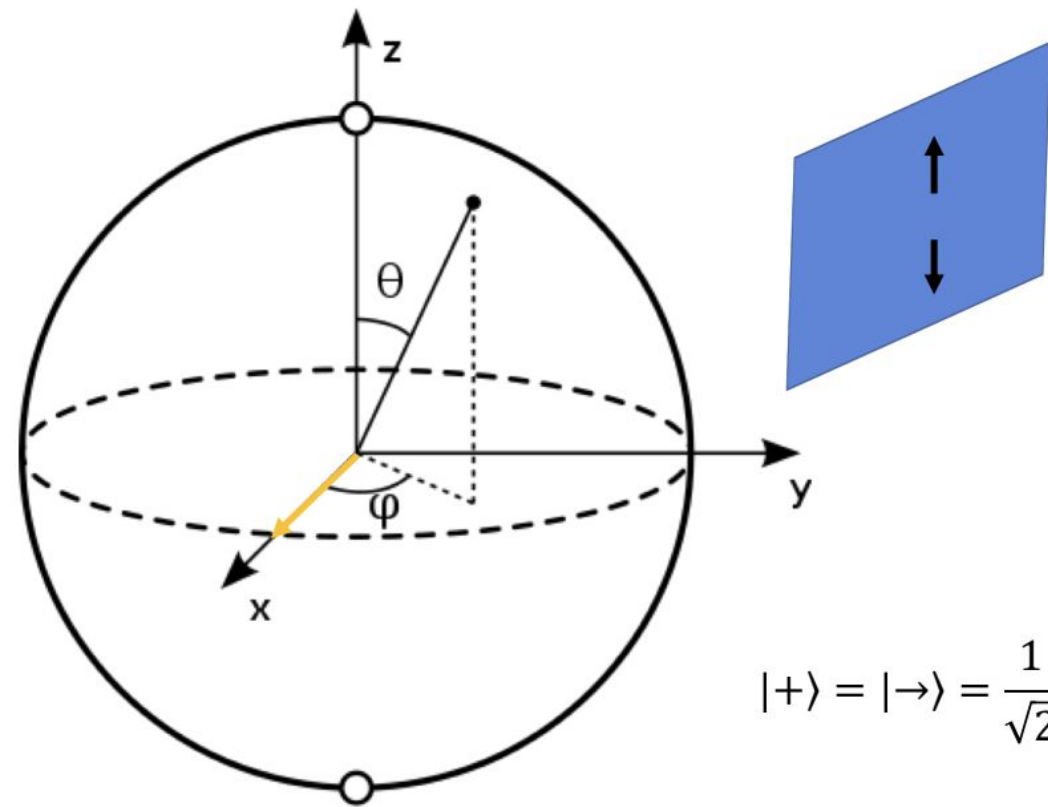
$$|1\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$$



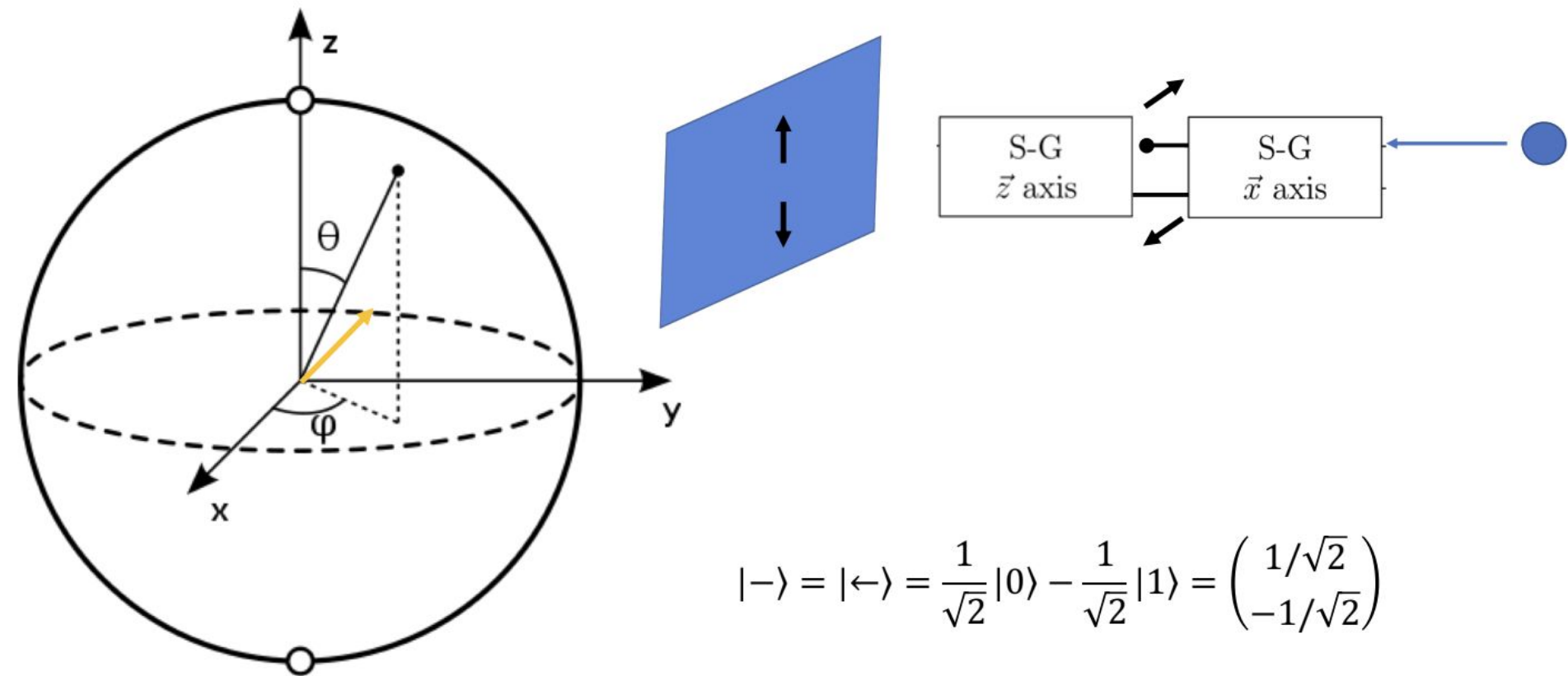
S-G
 \vec{z} axis



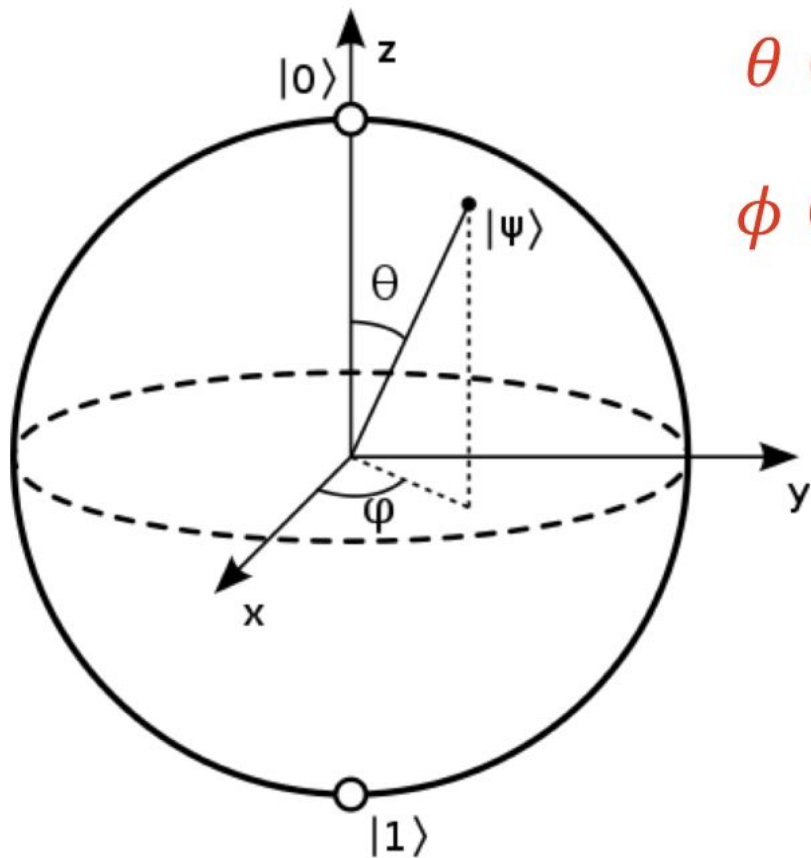




$$|+\rangle = |\rightarrow\rangle = \frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle = \frac{1}{\sqrt{2}}\begin{pmatrix} 1 \\ 0 \end{pmatrix} + \frac{1}{\sqrt{2}}\begin{pmatrix} 0 \\ 1 \end{pmatrix} = \begin{pmatrix} 1/\sqrt{2} \\ 1/\sqrt{2} \end{pmatrix}$$



$$|-\rangle = |\leftarrow\rangle = \frac{1}{\sqrt{2}}|0\rangle - \frac{1}{\sqrt{2}}|1\rangle = \begin{pmatrix} 1/\sqrt{2} \\ -1/\sqrt{2} \end{pmatrix}$$



$$\theta \in \{0, \pi\}$$

$$\phi \in \{0, 2\pi\}$$

The Bloch Sphere

$$\psi(\theta, \phi) = \begin{pmatrix} \cos\left(\frac{\theta}{2}\right) \\ e^{i\phi} \sin\left(\frac{\theta}{2}\right) \end{pmatrix}$$



Quantum Gates



Quantum Gates

- Maps one the quantum state to another quantum state
- Moves a point on the Bloch Sphere to another point
- We can represent them with matrices

Some basic Quantum gate matrices

Pauli gates:

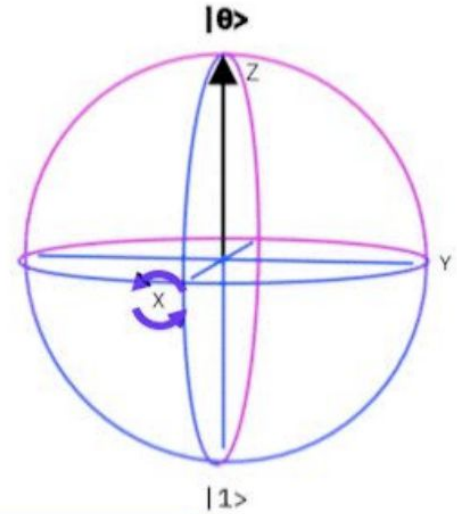
$$\sigma_x = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \quad \text{Pauli-X operator (X)}$$

$$\sigma_y = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix} \quad \text{Pauli-Y operator (Y)}$$

$$\sigma_z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} \quad \text{Pauli-Z operator (Z)}$$

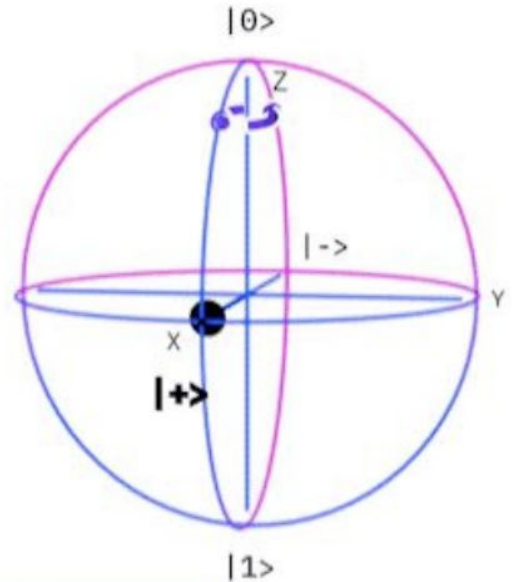
$$\sigma_x = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$$

Pauli-X operator (X)



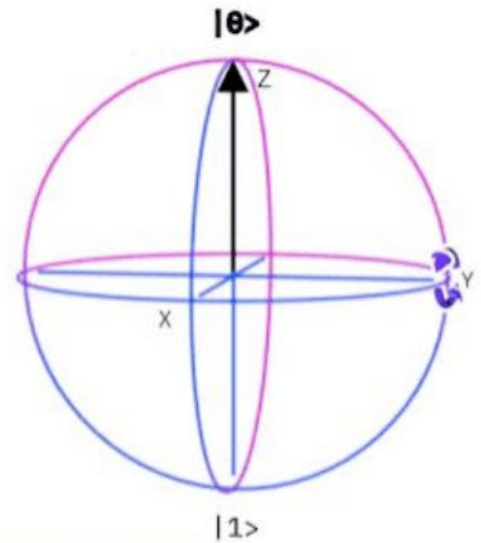
Rotates the state around the **X** axis by 180 degrees

$$\sigma_z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} \quad \text{Pauli-Z operator (Z)}$$



Rotates the state around the **Z** axis by 180 degrees

$$\sigma_y = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix} \quad \text{Pauli-Y operator (Y)}$$



Rotates the state around the **y** axis by 180 degrees

Other important gates

Hadamard gate

The Hadamard gate is a single-qubit operation that maps the basis state $|0\rangle$ to $\frac{|0\rangle+|1\rangle}{\sqrt{2}}$ and $|1\rangle$ to $\frac{|0\rangle-|1\rangle}{\sqrt{2}}$, thus creating an equal superposition of the two basis states.

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

CNOT Gate

The CNOT gate is two-qubit operation, where the first qubit is usually referred to as the control qubit and the second qubit as the target qubit. Expressed in basis states, the CNOT gate:

- leaves the control qubit unchanged and performs a Pauli-X gate on the target qubit when the control qubit is in state $|1\rangle$;
- leaves the target qubit unchanged when the control qubit is in state $|0\rangle$.

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

Rotation Gates

We're not just limited by 180 degree rotations

For example we can rotate around the x axis by 90 degrees

A diagram showing the notation $R_x(\pi/2)$ for a rotation gate. Three blue arrows point from the labels 'rotation', 'axis', and 'angle' to the components of the notation: 'rotation' points to R , 'axis' points to x , and 'angle' points to $(\pi/2)$.

$$R_x(\pi/2)$$

rotation axis angle

We can rotate by an angle θ around x or y or z axis:

$$R_x(\theta) = e^{-i \frac{\theta}{2} \sigma_x}$$

$$R_y(\theta) = e^{-i \frac{\theta}{2} \sigma_y}$$

$$R_z(\theta) = e^{-i \frac{\theta}{2} \sigma_z}$$

Let's take rotation by an angle θ around the x axis as an example:

$$R_x(\theta) = e^{-i\frac{\theta}{2}\sigma_x} = \cos\frac{\theta}{2} I - i \sin\frac{\theta}{2} \sigma_x$$

Matrix representation?

$$\begin{aligned} R_x(\theta) &= \cos(\theta/2) \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} - i \sin(\theta/2) \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \\ &= \begin{pmatrix} \cos(\theta/2) & 0 \\ 0 & \cos(\theta/2) \end{pmatrix} + \begin{pmatrix} 0 & -i\sin(\theta/2) \\ -i\sin(\theta/2) & 0 \end{pmatrix} = \begin{pmatrix} \cos(\theta/2) & -i\sin(\theta/2) \\ -i\sin(\theta/2) & \cos(\theta/2) \end{pmatrix} \end{aligned}$$

Let's take rotation by an angle θ around the x axis as an example:

$$R_z(\theta) = e^{-i\frac{\theta}{2}\sigma_z} = \cos\frac{\theta}{2} I - i \sin\frac{\theta}{2} \sigma_z$$

Matrix representation?

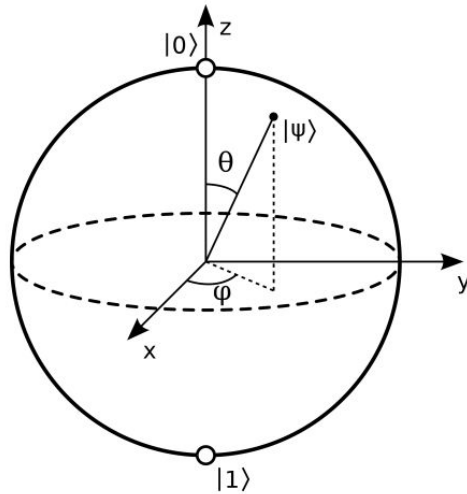
$$\begin{aligned} R_z(\theta) &= \cos(\theta/2) \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} - i \sin(\theta/2) \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} \\ &= \begin{pmatrix} \cos(\theta/2) - i \sin(\theta/2) & 0 \\ 0 & \cos(\theta/2) + i \sin(\theta/2) \end{pmatrix} \end{aligned}$$

Let's try to condense all this information into code through these examples...

Examples:

$$R_X(\pi) |0\rangle$$

$$R_X(\pi/2) |0\rangle$$



Examples:

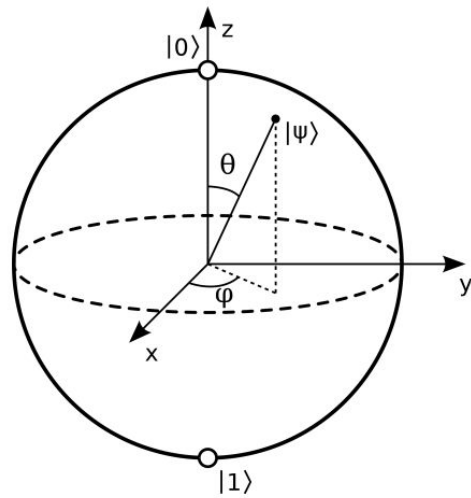
$$\begin{aligned} R_x(\pi) |0\rangle &= (\cos(\pi/2) \mathbb{I} - i \sin(\pi/2) \sigma_x) |0\rangle \\ \checkmark \\ \times \pi\text{-pulse} \quad &= \cos(\pi/2) \cdot \mathbb{I} |0\rangle - i \sin(\pi/2) \sigma_x |0\rangle \\ &= -i \sigma_x |0\rangle = \cancel{-i |1\rangle} \end{aligned}$$

$$\begin{aligned} R_x(\pi/2) |0\rangle &= (\cos(\pi/4) \cdot \mathbb{I} - i \sin(\pi/4) \sigma_x) |0\rangle \\ &= \cos(\pi/4) \cdot \mathbb{I} |0\rangle - i \sin(\pi/4) \sigma_x |0\rangle \\ &= \frac{1}{\sqrt{2}} |0\rangle - i \frac{1}{\sqrt{2}} |1\rangle = \frac{|0\rangle - i |1\rangle}{\sqrt{2}} \end{aligned}$$

Examples:

$$R_Y(\pi/2) |0\rangle$$

$$R_Z(\pi/2) |+\rangle$$

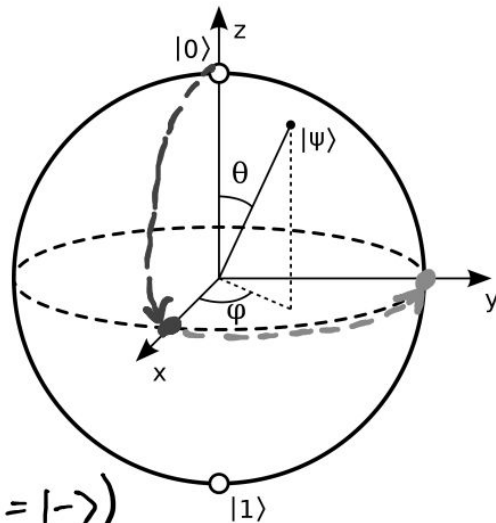


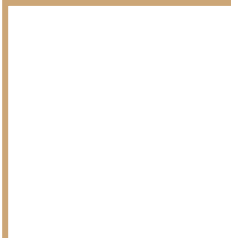
Examples:

$$\begin{aligned} R_y(\pi/2) |0\rangle &= \cos(\pi/4) \cdot I |0\rangle - i \sin(\pi/4) \sigma_y |0\rangle \\ &= \frac{1}{\sqrt{2}} |0\rangle + \frac{1}{\sqrt{2}} |1\rangle = \frac{|0\rangle + |1\rangle}{\sqrt{2}} \equiv |+\rangle \end{aligned}$$


$$\begin{aligned} R_z(\pi/2) |+\rangle &= \cos(\pi/4) \cdot I |+\rangle - i \sin(\pi/4) \sigma_z |+\rangle \\ &= \frac{1}{\sqrt{2}} |+\rangle - \frac{i}{\sqrt{2}} |-\rangle \end{aligned} \quad (\sigma_z |+\rangle = |-\rangle)$$

$$= \frac{1}{\sqrt{2}} \frac{1}{\sqrt{2}} [|0\rangle + |1\rangle - i|0\rangle + i|1\rangle] = \frac{|0\rangle - i|1\rangle}{\sqrt{2}}$$





Basic Principles of QC



1. Superposition Principle

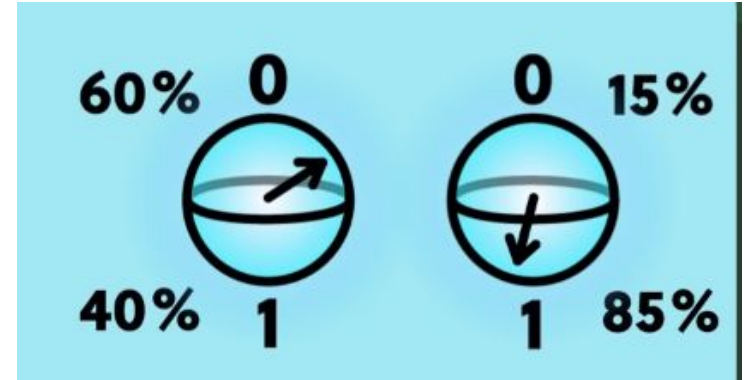
(Qubits exist as superpositions of two states.)

2. Quantum Entanglement

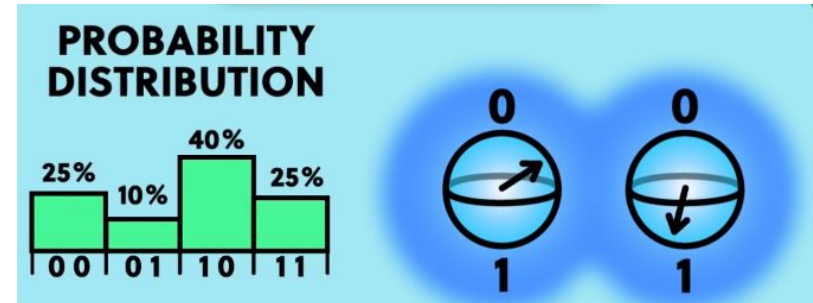
3. Interference

Quantum Entanglement

- In classical computers, each bit is independent of other bits.
- Qubits can be 'entangled', which means they can be made dependent on each other.
- We can no longer think of entangled qubits as two individual qubits, but as a system of qubits.



2 qubits



2 entangled qubits

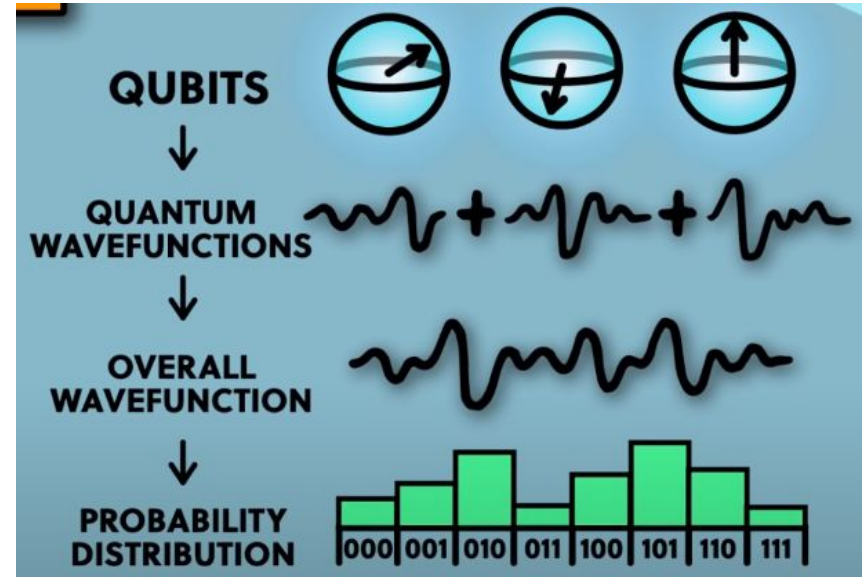
Quantum Entanglement

- This is what differentiates classical computers from quantum computers.
- Quantum computers can simultaneously compute an exponential number of states.

NUMBER OF QUBITS	NUMBER OF STATES
1	2
2	4
3	8
4	16
5	32
⋮	⋮
N	2^N

Interference

- Particles at the atomic level are represented by wavefunctions.
- When these particles interact, their wavefunctions are added. This is called interference.
- In quantum computers, interference is used to implement gates and measure qubits, at the hardware level.

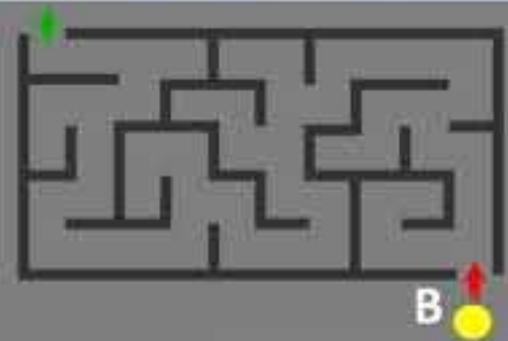
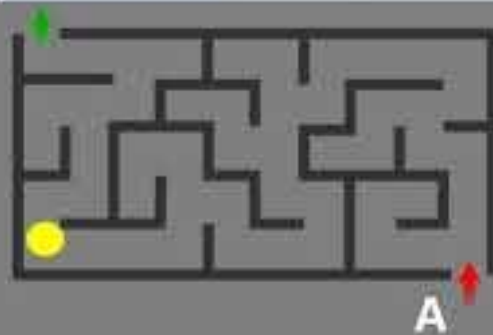


We need to develop quantum algorithms that exploit all these basic principles, such that the wave-function collapses to the desirable output.

Let's have a look at classical maze solver vs quantum maze solver

Quantum Maze

Quantum computers take advantage of quantum phenomena, such as superposition and entanglement; they can process many inputs simultaneously instead of having to go through them one by one like a conventional machine.





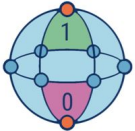
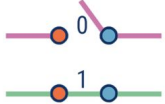

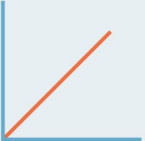
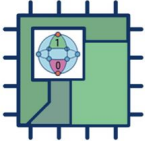
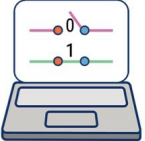


SOTA in Quantum Computing



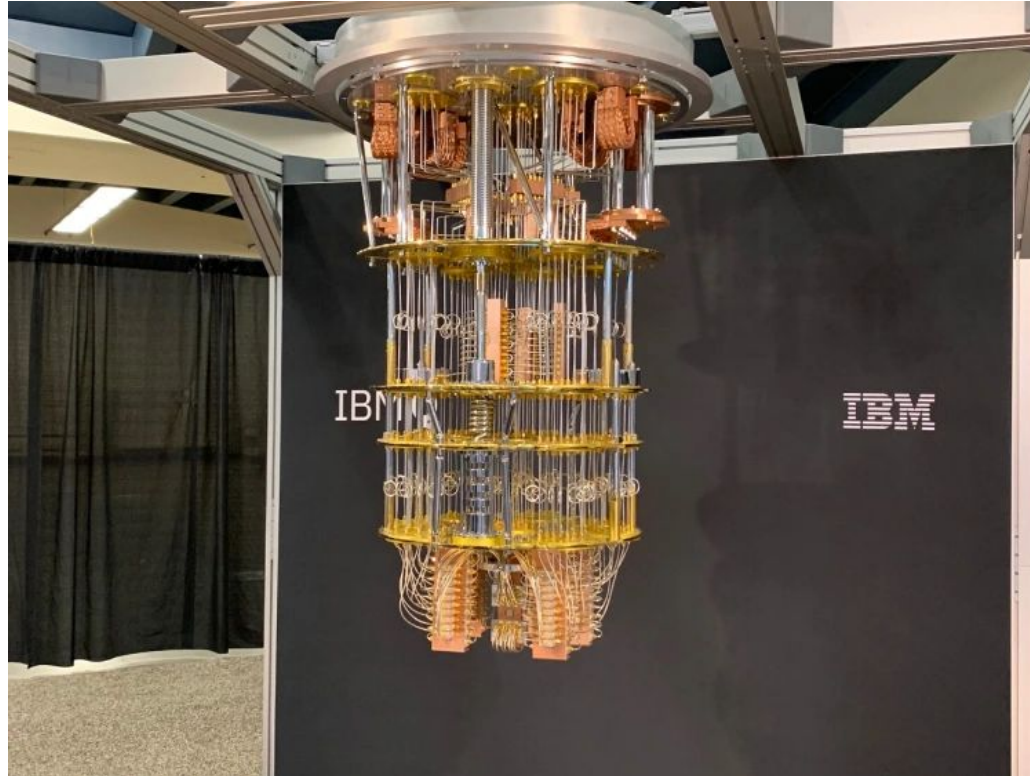
In QC we use the weirdness of quantum mechanics to provide

- high computational power (exponential increase in some cases)
- less energy consumption


This is done by controlling the behavior of small physical objects i.e. microscopic particles like atoms, electrons, photons, etc.

Quantum Computing	Vs.	Classical Computing
 <p>Calculates with qubits, which can represent 0 and 1 at the same time</p>		 <p>Calculates with transistors, which can represent either 0 or 1</p>
 <p>Power increases exponentially in proportion to the number of qubits</p>		 <p>Power increases in a 1:1 relationship with the number of transistors</p>
 <p>Quantum computers have high error rates and need to be kept ultracold</p>		 <p>Classical computers have low error rates and can operate at room temp</p>
 <p>Well suited for tasks like optimization problems, data analysis, and simulations</p>		 <p>Most everyday processing is best handled by classical computers</p>


IBMs 100 qubit quantum computer




Applications of Quantum Computers


- Solving encryption-like problems (Shor's Algorithm). Cybersecurity.
 - Solving optimization problems.
 - Solving problems with exponential search space.
 - Simulating particles at the atomic scale.
- 
- Financial modelling.
 - Weather simulation and climate change.
 - Machine Learning and AI. (Speeding up processes such as gradient descent)

Limitations with Quantum Computers

- Expensive!
 - Operational at -272 degrees celsius, which is almost absolute zero.
 - Interference and noise from outside sources, which increases exponentially with the number of qubits
- 
- Currently we have 100 qubit quantum computers, need thousands of qubits to solve real world problems.
 - Quantum algorithms are very hard to develop



Python Ecosystems for Quantum Computing



IBM Quantum



Qiskit

Qiskit is an SDK for writing, running, and simulating circuits on IBM Quantum's Platform



Cirq

Cirq is a Python framework for writing, running, and analyzing the results of quantum computer programs

D:wave



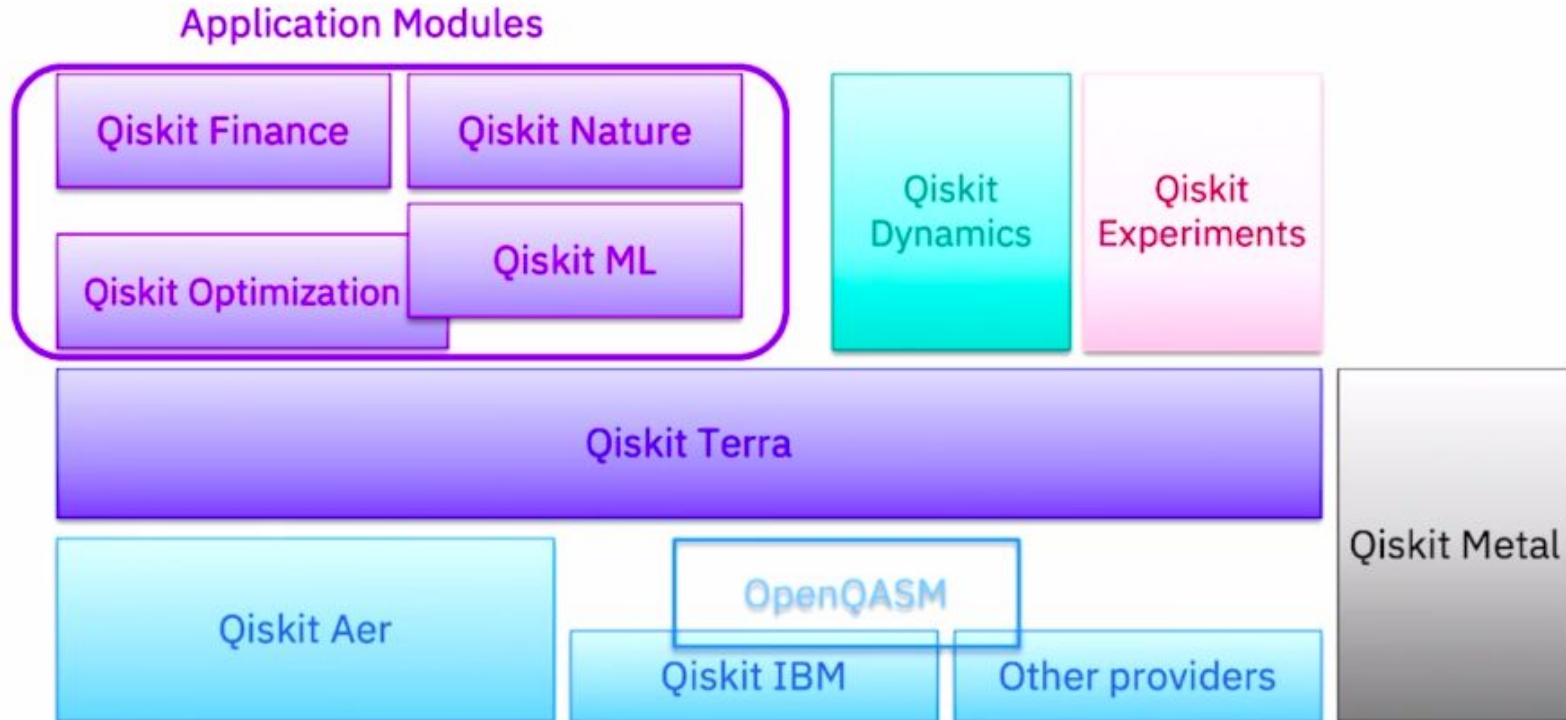
Ocean™ software is a suite of open source Python tools accessible via the Ocean software development kit (SDK) on both the D-Wave GitHub repository and within the Leap quantum cloud service.



Introduction to Qiskit



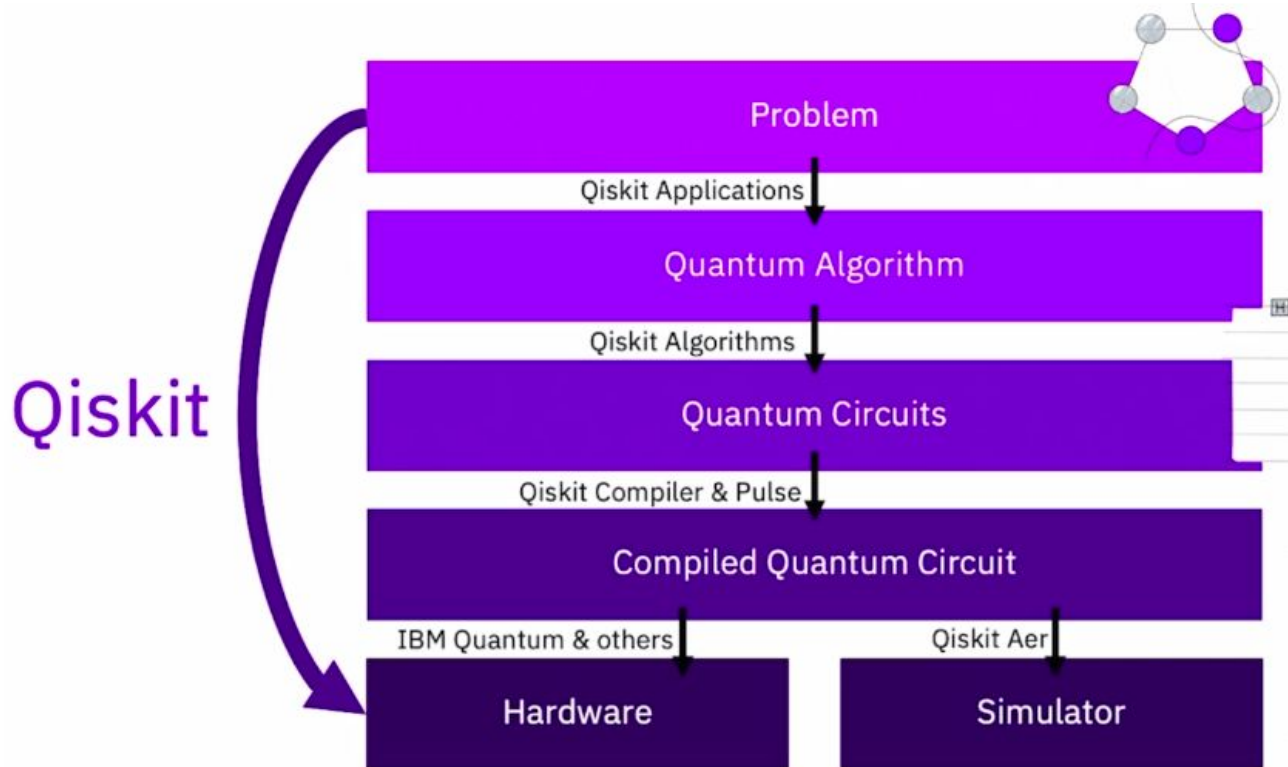
Structure of Qiskit



Qiskit

- Qiskit is an open-source Python library that allows us to create and run quantum circuits and algorithms
- Qiskit Terra provides tools to build and manipulate quantum circuits.
- Qiskit Aer provides tools to simulate quantum circuits.
- **People designing applications don't need to be experts in algorithms and/or hardware.**
- **Qiskit is hardware agnostic. Same code can be run on either simulators or real quantum computers.**

Solving a problem using Qiskit



Making a basic circuit with Qiskit Aer

```
ojus@dunder:~$ pip install numpy matplotlib
```

```
ojus@dunder:~$ pip install qiskit qiskit-ibmq-provider
```

Create an account on <https://quantum-computing.ibm.com/>

Get the token in your profile and copy it. This is important to get access to simulators and real quantum computers.

Making a basic circuit with qiskit

```
import qiskit as q

circuit = q.QuantumCircuit(2,2)
circuit.x(0)
circuit.cx(0, 1)
circuit.measure([0,1], [0,1])

circuit.draw()
# circuit.draw('mpl')
```

Some Resources

- The Qiskit textbook
- List of qiskit resources
- Quantum computing languages landscape
- Qiskit tutorial

References:

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

TCS QubitxQubit