

Curso de Qiskit

Instalación e Introducción

Pablo Gonzalez

Qiskit Summer Jam Mexico, Agosto 2021



Tabla de Contenido

① Introducción

② Instalación de Qiskit

③ Introducción a Qiskit

Librerías en Python

Construcción de un circuito cuántico

Compuertas de un solo qbit

Compuertas de dos o mas qbit

Visualización de circuitos cuanticos

Visualización de resultados

④ Herramientas en la nube





Tabla de Contenido

① Introducción

② Instalación de Qiskit

③ Introducción a Qiskit

Librerías en Python

Construcción de un circuito cuántico

Compuertas de un solo qbit

Compuertas de dos o mas qbit

Visualización de circuitos cuanticos

Visualización de resultados



Instalación de Qiskit en Python

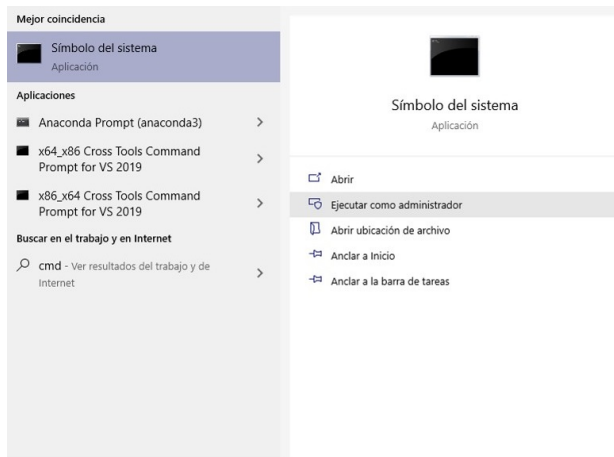
Para realizar una instalación directa de lo que es Qiskit y sus simuladores se deberán seguir los siguientes pasos:

- **Abrimos el Command Prompt(CMD):** Para utilizar esta consola basta con ir a nuestro buscador (variando entre versiones de Windows) y poner ahí la palabra `cmd`".



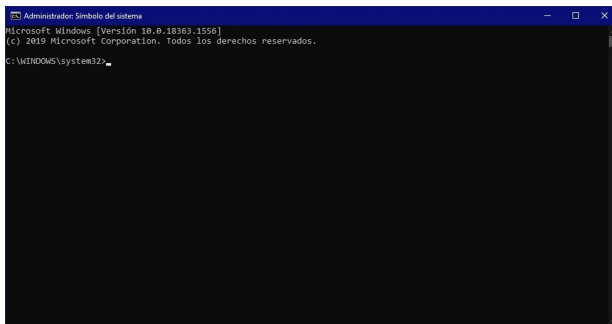
Instalación de Qiskit en Python

Una vez hecho esto, daremos a buscar (o enter) y nos debería aparecer la siguiente ventana del buscador.



Instalación de Qiskit en Python

Le daremos click a la opción resaltada 'Ejecutar como administrador' solo por seguridad, pero puede funcionar incluso sin los permisos de administrador. Cuando termine de cargar, nos debe aparecer la siguiente ventana.



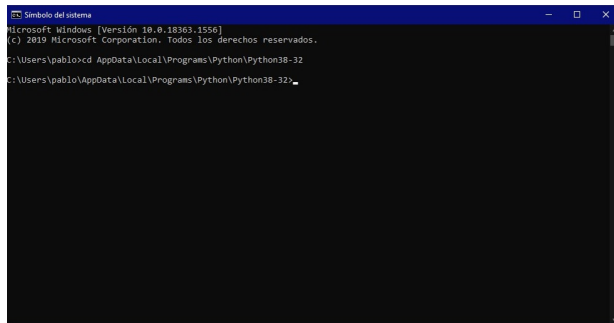
```
Administrador: Símbolo del sistema
Microsoft Windows [Versión 10.0.18363.1556]
(c) 2019 Microsoft Corporation. Todos los derechos reservados.

C:\WINDOWS\system32>
```



Actualizar nuestra versión de pip

Ya una vez en la ventana del CMD lo que necesitamos es acceder a Python desde la carpeta de aplicaciones locales de nuestro ordenador, por ello es necesario escribir la siguiente instrucción:



```
Simbolo del sistema
Microsoft Windows [Versión 10.0.18363.1556]
(c) 2019 Microsoft Corporation. Todos los derechos reservados.

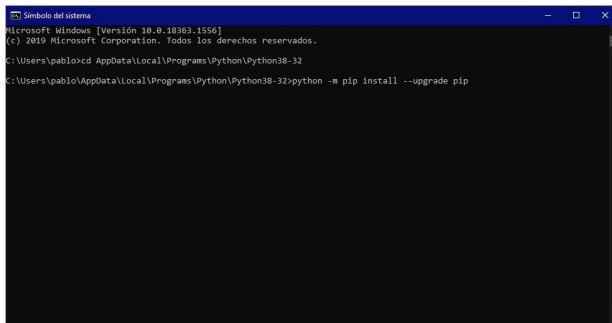
C:\Users\pablo>cd AppData\Local\Programs\Python\Python38-32

C:\Users\pablo\AppData\Local\Programs\Python\Python38-32>_
```



Actualizar nuestra versión de pip

Al hacer esto, hemos llamado a nuestra versión de Python desde donde estaremos actualizando la versión de pip (21.1.2) mediante el comando:



```
Símbolo del sistema
Microsoft Windows [Versión 10.0.18363.1556]
(c) 2019 Microsoft Corporation. Todos los derechos reservados.

C:\Users\pablo>cd AppData\Local\Programs\Python\Python38-32

C:\Users\pablo\AppData\Local\Programs\Python\Python38-32>python -m pip install --upgrade pip
```



Actualizar nuestra versión de pip

Añadido a esto para conocer la versión actual de nuestro pip basta con ejecutar el siguiente comando.

```
Simbolo del sistema
Microsoft Windows [Versión 10.0.18363.1556]
(c) 2019 Microsoft Corporation. Todos los derechos reservados.

C:\Users\pablo>cd AppData\Local\Programs\Python\Python38-32

C:\Users\pablo\AppData\Local\Programs\Python\Python38-32>python -m pip install --upgrade pip
Collecting pip
  Downloading pip-21.1.2-py3-none-any.whl (1.5 MB)
    | 1.5 MB 1.3 MB/s
Installing collected packages: pip
  Attempting uninstall: pip
    Found existing installation: pip 20.2
    Uninstalling pip-20.2:
      Successfully uninstalled pip-20.2
Successfully installed pip-21.1.2

C:\Users\pablo\AppData\Local\Programs\Python\Python38-32>pip --version
pip 21.1.2 from c:\users\pablo\appdata\local\programs\python\python38-32\lib\site-packages\pip (python 3.8)

C:\Users\pablo\AppData\Local\Programs\Python\Python38-32>
```



Instalar Qiskit de manera local

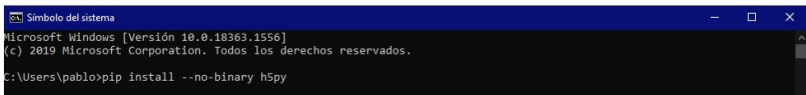
Una vez tengas la versión más actualizada de pip, podrás ingresar el siguiente comando para instalar todas las librerías con las que se trabajaran de qiskit.

```
Simbolo del sistema - pip install qiskit
Microsoft Windows [Versión 10.0.18363.1556]
(c) 2019 Microsoft Corporation. Todos los derechos reservados.
C:\Users\pablo>pip install qiskit
```



Instalar complementos de Qiskit

Puede ocurrir que durante la instalación de qiskit nos den algunos errores, para ello sugerimos seguir los pasos a continuación para darle solución a estos problemas.



```
Símbolo del sistema
Microsoft Windows [Versión 10.0.18363.1556]
(c) 2019 Microsoft Corporation. Todos los derechos reservados.
C:\Users\pablo>pip install --no-binary h5py
```

Esto ya que algunos complementos necesarios para la instalación de qiskit no vienen en nuestros ordenadores al instalar Python y necesitaremos instalarlos de manera manual para resolver estos conflictos.



Prueba de instalación exitosa

Una vez haya terminado la instalación, es conveniente saber que estamos en la versión más actualizada de qiskit, para comprobar este punto te sugerimos introducir el siguiente comando.

```
1 pip list
```

Para actualizar nuestra versión de Qiskit instalada a la más reciente, sugerimos introducir el siguiente comando y revisar que las versiones más actuales (26 de Junio del 2021) sean similares a la imagen más adelante.



Prueba de instalación exitosa

```
1      pip install qiskit[visualization] -U
2      pip install qiskit-nature -U
3      pip install --upgrade qiskit==0.27.0
```

```
qiskit                0.27.0
qiskit-aer             0.8.2
qiskit-aqua            0.9.2
qiskit-ibmq-provider   0.14.0
qiskit-ignis           0.6.0
qiskit-nature          0.1.3
qiskit-terra           0.17.4
```



Tabla de Contenido

- 1 Introducción
- 2 Instalación de Qiskit
- 3 Introducción a Qiskit

Librerías en Python

Construcción de un circuito cuántico

Compuertas de un solo qbit

Compuertas de dos o mas qbit

Visualización de circuitos cuanticos

Visualización de resultados



Ahora que hemos instalado el paquete Qiskit en nuestro sistema, podemos empezar a escribir un pequeño código para empezar a comprender las configuraciones básicas de los sistemas de simulación de una computadora cuántica.

Pero antes de comenzar tratar código, vamos a darle una pequeña revisión a lo que es Qiskit, su uso y básicamente ¿qué es qiskit? Qiskit [kiss-kit] es un SDK(Software Development Kit) de código abierto para trabajar con computadoras cuánticas a nivel de pulsos, circuitos y módulos de aplicación. Lo que logra acelerar el desarrollo de aplicaciones cuánticas aportando toda una gama de herramientas para muchas aplicaciones distintas pero con la gran ventaja de eliminar por completo el ruido captado por un ordenador cuántico dejando el resultado puro.



Librerías en Python

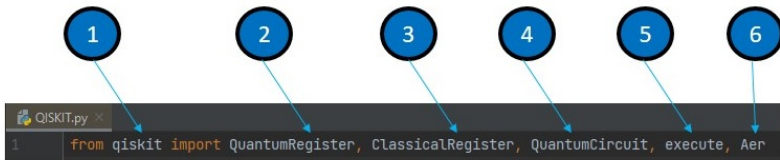
Para lograr dicho cometido, qiskit cuenta con una lista de simuladores que podremos llamar para realizar pruebas a nuestros circuitos, obteniendo con ello diferentes métodos para trabajar, resultados y comportamientos que son parte del funcionamiento de cada simulador. Como ejemplo serían Aer, Aqua, Terra, entre otros que serán abarcados a lo largo de este curso.

Para esta introducción estaremos manejando los simuladores de Aer mediante la IDE de PyCharm, la cual es la recomendada para trabajar esta parte del curso aunque si se desea usar otra IDE es perfectamente válido.



Librerías en Python

Para empezar a trabajar los circuitos cuánticos con Qiskit, deberemos llamar a las librerías con las que realizaremos nuestro primer circuito cuántico, para ello debemos definir lo siguiente en nuestra IDE.



Elementos en Qiskit

- **1.- Modulo Qiskit:** Es el modulo principal donde se alojan todas las herramientas con las que podemos disponer para realizar nuestras simulaciones.
- **2.- Librería de registros cuánticos:** Es la libreria con la que podemos agregar los qbits que necesitemos para nuestro trabajo, a partir de aqui podremos cambiarlos de signo, rotarlos, entre muchas otras acciones hasta cambiar las magnitudes de su estado.
- **3.- Librería de registros clásicos:** Para poder revisar el comportamiento de nuestros qbits necesitaremos una forma de medir su estado, la mejor manera es utilizando bits clásicos (0 o 1) y con esta librería podremos agregar los circuitos cuánticos.



Elementos en Qiskit

- **4.- Librería de circuitos cuánticos:** Es la librería con la que podemos agregar los circuitos cuánticos que necesitemos en nuestro trabajo a partir de los qbits o bits clásicos registrados con anterioridad.
- **5.- Librería ejecuciones:** Con esta librería podremos realizar los trabajos de simulación seleccionando nuestros "disparos." shots, el simulador para conseguir los resultados, entre otras cosas más.
- **6.- Librería del simulador:** Básicamente, son las características con las que ejecutaremos así como otras cosas con las que podemos dotar a nuestro circuito como un tipo de ruido que tenga nuestro sistema.



Elementos en Qiskit

Cabe resaltar, que el estudiante siempre revise los elementos con los que estará trabajando en el circuito que desee diseñar, mediante el uso de la documentación oficial de qiskit (<https://qiskit.org/>). Ahora, ya hemos comenzado a utilizar las librerías con las que crearemos nuestro circuito, ahora utilizaremos estas librerías para utilizar sus elementos.



Compuertas cuanticas

Para construir un circuito cuántico tenemos diferentes elementos y funciones que nos ayudaran con esta tarea, para ello vamos a conocer cada uno junto a su comportamiento correspondiente. **Las Compuertas de Pauli:** Debe estar familiarizado con las matrices de Pauli de la sección de álgebra lineal. Todas estas matrices o compuertas serán las utilizadas para manejar los qbits en nuestro circuito.



Compuerta X

Referida comúnmente como NOT-gate por su comportamiento análogo, la compuerta X da una rotación de π radianes en la esfera de Bloch, lo que sería igual decir que cambia el signo de nuestro qbit.

Su visualización matemática es la siguiente:

$$X = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \quad (1)$$

Mientras que su declaración en código es la siguiente:

```
1 Myqc = QuantumCircuit(1)
2 Myqc.x(0)
3 Myqc.draw()
```



Compuerta Y Z

Al igual que la compuerta X, su comportamiento es el mismo dando una rotación en su eje respectivo. Su visualización matemática es la siguiente:

$$Y = \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix} \quad (2)$$

$$Z = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \quad (3)$$

Mientras que su declaración en código es la siguiente:

```
1 Myqc = QuantumCircuit(1)
2 Myqc.y(0)
3 Myqc.z(0)
4 Myqc.draw()
```



Compuerta Hadamard(H)

Una de las compuertas más usadas, su comportamiento nos ayuda a realizar dos rotaciones de manera simultánea, lo que genera un estado de superposición de $|0\rangle$ y $|1\rangle$. La rotación es equivalente una de π sobre el eje \hat{z} seguido de una rotación de $\pi/2$ sobre el eje \hat{y} .

Su visualización matemática es la siguiente:

$$H = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \quad (4)$$

Mientras que su declaración en código es la siguiente:

```
1 Myqc = QuantumCircuit(1)
2 Myqc.h(0)
3 Myqc.draw()
```



Compuerta Rotación(R)

Esta compuerta nos es muy útil para cuando de trabajar con rotaciones se trata, el principio es darle al qbit una rotación parametrizada para desplazarlo por toda la esfera de Bloch. Necesita un ángulo ϕ para decirle exactamente a donde ir.

Su visualización matemática es la siguiente:

$$R_{\phi} = \begin{bmatrix} 1 & 0 \\ 0 & e^{i\phi} \end{bmatrix} \quad (5)$$



Compuerta Rotación(R)

Mientras que su declaración en código es la siguiente:

```
1  from math import pi, sqrt
2  Myqc = QuantumCircuit(1)
3  Myqc.rx(pi/4,0) #angle , target qbit
4  Myqc.draw()
```



Compuerta Tolerant(T)

Muchas aplicaciones cuánticas crearan compuertas por error debido al ruido, campos magnéticos, etc, para evitar esto se trata de generar un codificador que haga más robusta la transmisión de información. La compuerta Tolerante (Fault-Tolerant) nos ayuda con este problema, aunque si bien es efectiva siempre existe un rango de error aproximadamente a: $\phi + 0.00000001$. Siempre tendremos un límite en las aplicaciones que realicemos o las veces que usemos esta compuerta.

Su rotación es en el eje Z al por $\phi = \pi/4$.



Compuerta Tolerant(T)

Su declaración en código es la siguiente:

```
1      Myqc = QuantumCircuit(1)
2      Myqc.t(0)
3      Myqc.draw()
```



Compuerta Identidad(I)

Aunque parezca muy sencilla de entender, la compuerta identidad es la compuerta 1 a 1 con el qbit que es inicializado, básicamente no modifica nada en cuanto a datos se refiere. Aunque, es útil conocerla puesto que esta se conforma de dos o más compuertas, una de las más vitales es la compuerta X, la compuerta identidad será igual al producto de dos compuertas X.



Compuerta Identidad(I)

Su visualización matemática es la siguiente:

$$I = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \quad (6)$$

Al ser una compuerta cuyo resultado es siempre la compuerta original, queda a disposición del lector deducir como obtener dicho resultado con las compuertas hasta ahora vistas.



Compuerta Control (CNOT)

Esta compuerta es muy utilizada en varias aplicaciones, esta compuerta basa su comportamiento cambiando de signo a un qbit por medio de otro qbit de control, es por ello su nombre. Dependiendo de cuál sea el eje que queramos manejar será el tipo de control (x,y o z) que debemos utilizar.

Su visualización matemática es la siguiente:

$$CNOT = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} \quad (7)$$



Compuerta Control (CNOT)

Mientras que su declaración en código es la siguiente:

```
1      Myqc = QuantumCircuit(2)
2      Myqc.cx(0,1)
3      Myqc.cy(0,1)
4      Myqc.cz(0,1)
5      Myqc.draw()
```



Compuerta Toffoli (CCNOT)

Funcionando casi igual a la anterior, funcionando con tres qbits, dos de control y uno a controlar o target.

Su visualización matemática es la siguiente:

$$CCNOT = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix} \quad (8)$$



Compuerta Toffoli (CCNOT)

Mientras que su declaración en código es la siguiente:

```
1      Myqc = QuantumCircuit(3)
2      Myqc.ccx(0,1,2)
3      Myqc.draw()
```



Compuerta Swap

Algunas veces necesitaremos realizar rotaciones sobre los qbits para realizar cambios de estado o transportar información de un carril a otro, por ello esta condición nos ayuda a realizar dicha tarea y es mucho más efectivo que igualar estados mediante otras compuertas.

Su visualización matemática es la siguiente:

$$SWAP = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (9)$$



Compuerta Swap

Mientras que su declaración en código es la siguiente:

```
1      Myqc = QuantumCircuit(2)
2      Myqc.swap(1,2)
3      Myqc.draw()
```



Visualización de circuitos cuánticos

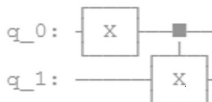
Ahora ya que tenemos construido nuestro circuito, naturalmente tendremos la curiosidad de observar si lo que hemos declarado en nuestra sintaxis ha sido justamente lo que queríamos, para ello tendremos diferentes maneras de ver nuestros circuitos cuánticos mediante varias herramientas.



Construcción por ASCII

Para mostrar un circuito de forma sencilla basta con utilizar la siguiente instrucción:

```
1 Myqc = QuantumCircuit(2)
2 Myqc.x(0)
3 Myqc.cx(0,1)
4 print(Myqc)
```



Mediante el comando print, se crea un circuito de forma ASCII que es bastante sencillo de interpretar.



Construcción por Matplotlib

Para mostrar un circuito con una construcción visualmente elaborada se utiliza la siguiente instrucción:

```
1 Myqc = QuantumCircuit(2)
2 Myqc.x(0)
3 Myqc.cx(0,1)
4 Myqc.draw(output = 'mpl')
```



Mediante el comando draw el circuito pasa por un tratado en matplotlib cuando le indicamos este tipo de salida (`output = 'mpl'`).



Resultado por QASM

Es de los simuladores más utilizados para obtener los resultados de un circuito cuántico, lo que nos arroja son todos los resultados calculados para cada una de las posibles combinaciones de los qbtis en nuestro circuito. Incluye modelos de ruido altamente configurables e incluso se puede cargar con modelos de ruido aproximados generados automáticamente en función de los parámetros de calibración de los dispositivos de hardware reales. Para llamarlo, se necesita la siguiente instrucción:

```
1  sim = Aer.get_backend('qasm_simulator')
```



Resultado por state vector

Simula la ejecución ideal de un circuito cuántico y devuelve el vector de estado cuántico final del dispositivo al final de la simulación. Esto es útil para la educación, así como para el estudio teórico y la depuración de algoritmos. Para llamarlo, se necesita la siguiente instrucción:

```
1  sim = Aer.get_backend('statevector_simulator')
```



Resultado por esfera de Bloch

Más que visualizar un circuito cuántico, estaremos viendo el resultado de los qbits por medio de la esfera de Bloch para ver su posición final pero es una excelente manera de interpretar un circuito cuántico final. Para ello se necesitan seguir los siguientes pasos:

Primero necesitamos llamar a las siguientes librerías y módulos:

```
1 from qiskit import QuantumCircuit, assemble, Aer
2 from qiskit.visualization import
  plot_bloch_multivector, plot_histogram
3 sim = Aer.get_backend('aer_simulator')
```

Estas contienen los recursos necesarios para graficar dicha esfera, por lo que requieren un primer simulador que nos ayude a obtener el resultado del circuito.



Construcción por esfera de Bloch

Una vez hecho, construimos nuestro circuito de la misma manera que hemos hecho antes.

```
1      Myqc = QuantumCircuit (2)
2      Myqc.x(0)
3      Myqc.cx(1,0)
```

En teoría, nuestro circuito nos debe arrojar un qbit en estado bajo y otro en alto. Para lograr ver esto necesitamos llamar a las siguientes instrucciones.



Construcción por esfera de Bloch

Para finalizar, debemos almacenar el estado en el que se encuentran nuestros qbits, después de guardarlos lo que hacemos es pasarlos por el simulador para que obtengamos los resultados de dicho experimento y por último, mostramos la esfera con los vectores en las posiciones que el simulador cálculo.

```
1 Myqc.save_statevector()  
2 qobj = assemble(Myqc)  
3 state = sim.run(qobj).result().get_statevector()  
4 plot_bloch_multivector(state)
```



Construcción por esfera de Bloch

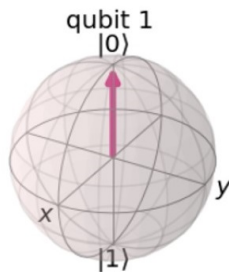
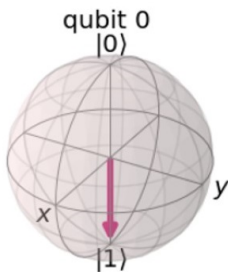


Tabla de Contenido

① Introducción

② Instalación de Qiskit

③ Introducción a Qiskit

Librerías en Python

Construcción de un circuito cuántico

Compuertas de un solo qbit

Compuertas de dos o mas qbit

Visualización de circuitos cuanticos

Visualización de resultados

④ Herramientas en la nube



IBM Quantum Experience

Incluso con todo lo anterior, si necesitamos más herramientas para trabajar en computación cuántica o requerimos aplicaciones donde la computación cuántica es una herramienta fundamental, tenemos en la nube diferentes opciones para trabajar de manera gratuita con la computación cuántica. Nuevamente, son simuladores de alta precisión debido al ruido que estas máquinas llegan a capturar.

Ejemplo de esto es la IBM Quantum Experience donde podremos trabajar en tiempo real con una de las múltiples computadoras cuánticas que nos ofrece la compañía. Aprendamos un poco como seleccionar un buen modelo para empezar a trabajar y conocer el ambiente.

Para acceder a la herramienta, hay que dirigirnos a esta página:

<https://quantum-computing.ibm.com/>



IBM Quantum Experience

Entrando al link anterior, nos encontraremos esta página de bienvenida donde podemos interactuar par conocer un poco más sobre la forma de trabajar de la mano de IBM. Para registrarnos, debemos hacer click en el botón azul que dice IBMid.

Real quantum computers.
Right at your fingertips.

IBM offers cloud access to the most advanced quantum computers available. Learn, develop, and run programs with our quantum applications and systems.

Learn more about IBM Quantum Experience




Graphically build quantum circuits

Start building quantum circuits right away with IBM Quantum Composer. No sign in required.

[Explore Quantum Composer](#)

Sign in to IBM Quantum

IBMID 

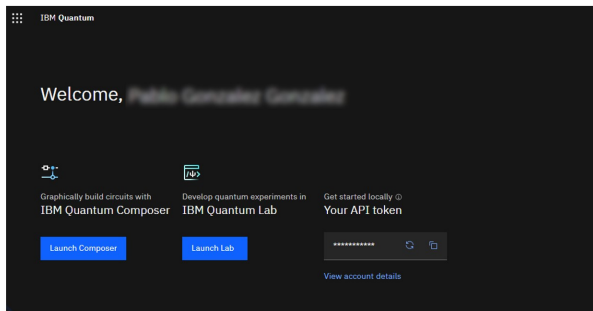
[G](#) [in](#) [in](#) [in](#) [in](#) [in](#) [in](#)

New to IBM Quantum?
[Create an IBMID account](#)



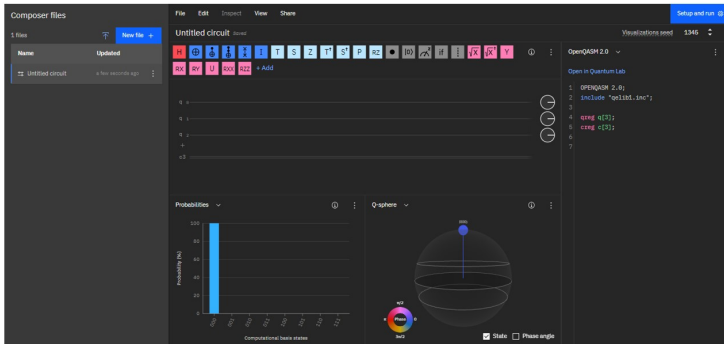
IBM Quantum Experience

Avanzaremos hasta lo que es nuestra página de inicio de sesión de usuario, en esta se encuentran diferentes opciones como el laboratorio de IBM para escribir código, el simulador de una computadora cuántica y varias opciones más, lo que hoy nos llama es entrar a la parte del Composer, el cual es el botón azul que dice "Launch Composer".



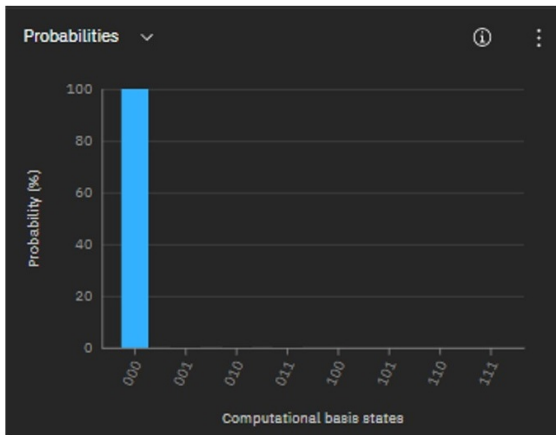
IBM Quantum Experience

En esta parte podemos disfrutar de una herramienta muy completa para desarrollar circuitos cuánticos, conozcamos un poco más nuestro ambiente.



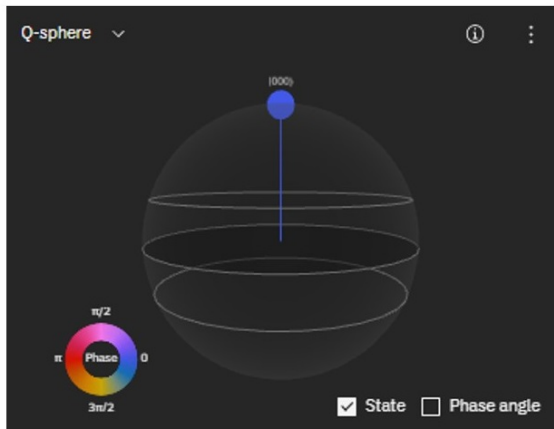
IBM Quantum Experience

Segundo, cuando terminemos de diseñar nuestro circuito y queramos ver los resultados en forma gráfica y de porcentajes de que ocurran, tenemos esta útil grafica que nos facilita el trabajo.



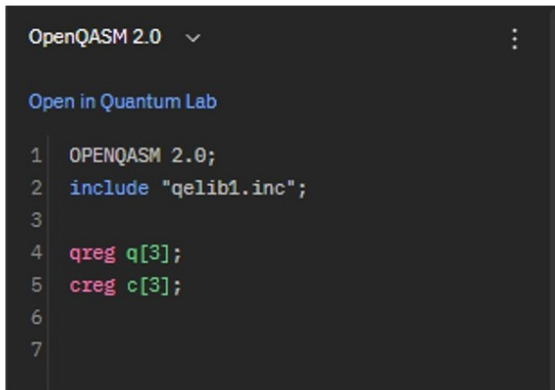
IBM Quantum Experience

Tercero, nuestra esfera de Bloch móvil donde podemos ver la dirección de nuestros vectores en tiempo real, logrando intercalar entre estado y ángulo de fase.



IBM Quantum Experience

Cuarto y último, si preferimos probar la entrada por código tenemos una sencilla interfaz para el trabajo, con disponibilidad para usar el lenguaje OpenQASM 2.0 o una versión de Python con Qiskit.



The screenshot shows a code editor window titled "OpenQASM 2.0" with a dropdown arrow. Below the title is a link "Open in Quantum Lab". The editor contains the following code:

```
1 OPENQASM 2.0;  
2 include "qelib1.inc";  
3  
4 qreg q[3];  
5 creg c[3];  
6  
7
```

