

**Lab 1:**  
**Your first circuit step-by-step with IBM Quantum**

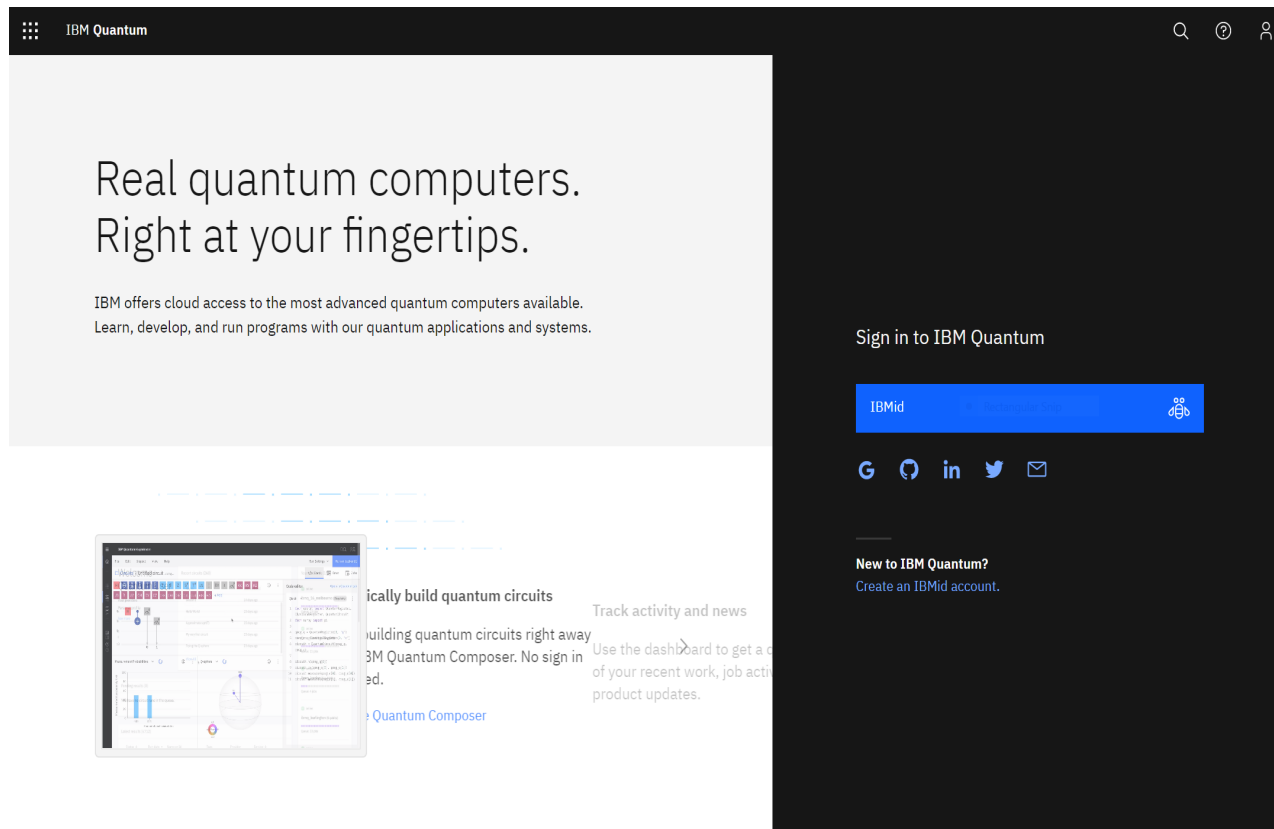
We are going to use throughout this course the environment offered by IBM which is called the IBM Quantum. In this exercise we are going to do together the simplest possible thing that one can do to a qubit: we are going to invert it. That is we are going to move it from quantum state  $|0\rangle$  to quantum state  $|1\rangle$ , i.e. from pointing up to pointing down in the Bloch Sphere.

***TASK:*** To flip a single qubit from  $|0\rangle$  to  $|1\rangle$ .

***(ONLINE) SOLUTION:*** Create a quantum circuit with a single qubit and a X Gate in the online IBM Quantum platform

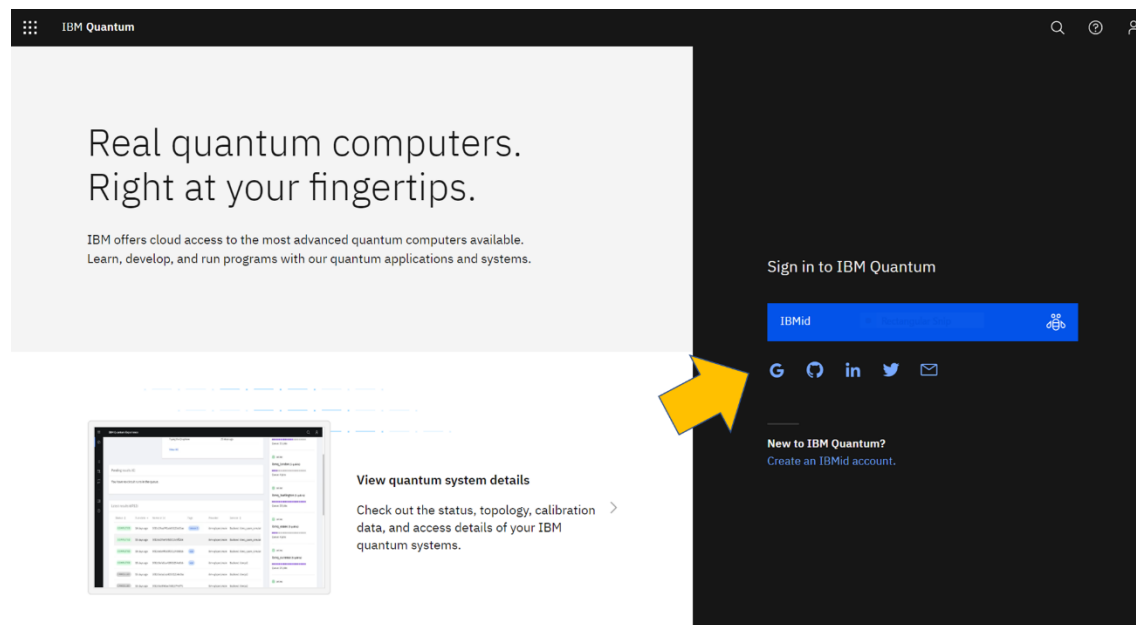
## Step 1: Go to the IBM Quantum website

Visit the IBM Quantum Experience website <https://quantum-computing.ibm.com/login>. As illustrated in the next Figure, here is where you can access the various services offered by IBM for quantum computation.



## Step 2: Register or sign-in into the system

You ought to create a new **IBMid** which will allow you to use the environment. You can do this by filling the short registration form. Alternatively you can register by using your social profile from Google, LinkedIn or Twitter, amongst others.





# Welcome, Alonso Pena



Graphically build circuits with  
**IBM Quantum Composer**

[Launch Composer](#)

Develop quantum experiments in  
**IBM Quantum Lab**

[Launch Lab](#)

Jump back in:

[TEMP\\_LOGNORMAL\\_QAE\\_MO...](#)  
[TEMP\\_LOGNORMAL\\_QAE.ipynb](#)  
[TEMP\\_03\\_european\\_call\\_opti...](#)  
[TEMP\\_credit\\_risk\\_analysis.ipy...](#)

API token ⓘ

\*\*\*\*\*

[View account details](#)

Run on quantum systems & simulators:  
**IBM Quantum services**

[View all](#)

8

Your  
systems

5

Your  
simulators

25

Total quantum  
services

**Recent jobs**

[View all](#)

0

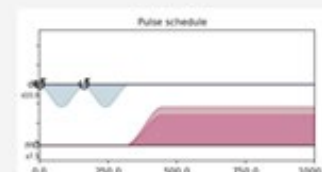
Pending

15

Completed

No pending jobs

## What's new



Use pulse gates to easily scale pulse-level control on OpenQASM circuits. Now available on [Armonk](#), [Sydney](#), [Paris](#), [Toronto](#), [Manhattan](#), [Bogota](#), [Rome](#), [Casablanca](#), and [Guadalupe](#).

[Learn more](#)

QC40: Physics of Computation  
Conference 40th Anniversary

[Submit a talk](#)

How to measure and reset a qubit in  
the middle of a circuit execution

[Read the blog](#)

IBM's roadmap for building an open  
quantum software ecosystem

[Read the blog](#)

Quantum circuits get a dynamic

## Step 3: Go to the Quantum Composer

Go now to *IBM Quantum Composer* in order to construct our first circuit by clicking on the Launch Composer button as indicated in the following Figure.

The screenshot shows the IBM Quantum dashboard for user Alonso Pena. The dashboard includes a header with the IBM Quantum logo and navigation icons. The main content area is divided into several sections:

- Welcome, Alonso Pena**: A greeting message at the top.
- Graphically build circuits with IBM Quantum Composer**: A section with a blue icon and a **Launch Composer** button. A large yellow arrow points to this button.
- Develop quantum experiments in IBM Quantum Lab**: A section with a blue icon and a **Launch Lab** button.
- Jump back in:**: A list of recent experiments with links to their notebooks, such as `TEMP_LOGNORMAL_QAE_MO...` and `TEMP_03_european_call_opti...`.
- API token**: A section showing a masked token and a **View account details** link.
- Run on quantum systems & simulators: IBM Quantum services**: A section with a blue icon and a **View all** link. It displays statistics: 8 Your systems, 5 Your simulators, and 25 Total quantum services.
- Recent jobs**: A section with a blue icon and a **View all** link. It shows 0 Pending jobs and 15 Completed jobs, with a **No pending jobs** message.
- What's new**: A section on the right side with a **Pulse schedule** graph and links to news items like [QC40: Physics of Computation Conference 40th Anniversary](#) and [IBM's roadmap for building an open quantum software ecosystem](#).

IBM Quantum Composer

File Edit Inspect View Share

Setup and run

Untitled circuit

Visualizations seed 5820

H ⊕ ⊗ ⊗ X I T S Z T† S† P RZ • |0⟩ if √X √X† Y RX RY U

RXX RZZ + Add

q0

q1

q2

+

c3

OpenQASM 2.0

Open in Quantum Lab

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

Probabilities

Computational basis states

Probability (%)

0 1 2 3 4 5

Q-sphere

State 000

Probability: 1

Phase angle: 0

State Phase angle

The coloured icons represent different types of quantum gates. The horizontal lines represent the cables to connect the quantum gates to the qubits and classical bits (indicated by  $q$  and  $c$ , respectively). The Bloch sphere is shown on the lower part as well as the statistical graph for the measurements. On the right the equivalent of the circuit is represented in a low-level code called QASM (Quantum Assembler Language).



## Step 4: Create your first circuit

Into the lines indicating the circuit, drag the X gate (circle with cross icon) to q0, then the measurement to q0.

The screenshot displays the IBM Quantum Composer interface. At the top, the title bar reads "IBM Quantum Composer" with search, help, and user icons. Below it is a menu bar with "File", "Edit", "Inspect", "View", and "Share". The main workspace shows an "Untitled circuit" with a toolbar containing various quantum gates: Hadamard (H), CNOT, multi-controlled NOT, multi-controlled T, Identity (I), T, S, Z, T†, S†, P, RZ, a classical control block, |0⟩, a rotation gate, an if statement, multi-controlled X, multi-controlled X†, Y, RX, RY, and U. Below the toolbar, a circuit diagram is visible with four horizontal lines for qubits q0, q1, q2, and a classical register c3. An X gate (a circle with a cross) is placed on the q0 line, and a measurement gate (a circle with a meter) is also on the q0 line. A large yellow arrow points to the X gate. The bottom left shows a "Probabilities" panel with a bar chart where the state 001 has a 100% probability. The bottom right shows a "Q-sphere" visualization with a blue dot at the top pole. On the far right, the OpenQASM 2.0 code is displayed:

```
1 OPENQASM 2.0;
2 include "qelib1.inc";
3
4 qreg q[3];
5 creg c[3];
6
7 x q[0];
8 measure q[0] -> c[0];
```

Now we need to eliminate the unnecessary qubits q1 and q2, that are present by default in the circuit design. By hovering directly above q1 a red rubbish bin icon will appear, click on it to delete:

The screenshot displays the IBM Quantum Composer interface. At the top, the title bar reads "IBM Quantum Composer". Below it is a menu bar with "File", "Edit", "Inspect", "View", and "Share". The main workspace shows an "Untitled circuit" with three qubits: q0, q1, and q2. A yellow arrow points to a red trash bin icon located above q1, indicating the deletion of this qubit. The toolbar at the top of the workspace contains various quantum gates: H, CNOT, T, S, Z, T†, S†, P, RZ, |0⟩, |1⟩, if, √X, X, Y, RX, RY, U, RXX, RZZ, and an "Add" button. On the right side, there is a code editor showing OpenQASM 2.0 code:

```
1 OPENQASM 2.0;
2 include "qelib1.inc";
3
4 qreg q[3];
5 creg c[3];
6
7 x q[0];
8 measure q[0] -> c[0];
```

Below the code editor, there are two visualizations. On the left is a "Probabilities" histogram showing the probability of computational basis states. The y-axis is "Probability (%)" from 0 to 100. The x-axis is "Computational basis states" with values 000, 001, 010, 011, 100, 101, 110, and 111. The bar for 001 is at 100%. On the right is a "Q-sphere" visualization showing a Bloch sphere with a blue dot at the top (representing state 000) and a blue arrow pointing to it. A small circular legend below the sphere shows the phase angle from 0 to 3π/2.

Do the same for q2 to delete:

IBM Quantum Composer

File Edit Inspect View Share

Untitled circuit Saved

Visualizations seed 5820

Setup and run

OpenQASM 2.0

Open in Quantum Lab

```
1 OPENQASM 2.0;
2 include "qelib1.inc";
3
4 qreg q[2];
5 creg c[2];
6
7 x q[0];
8 measure q[0] -> c[0];
```

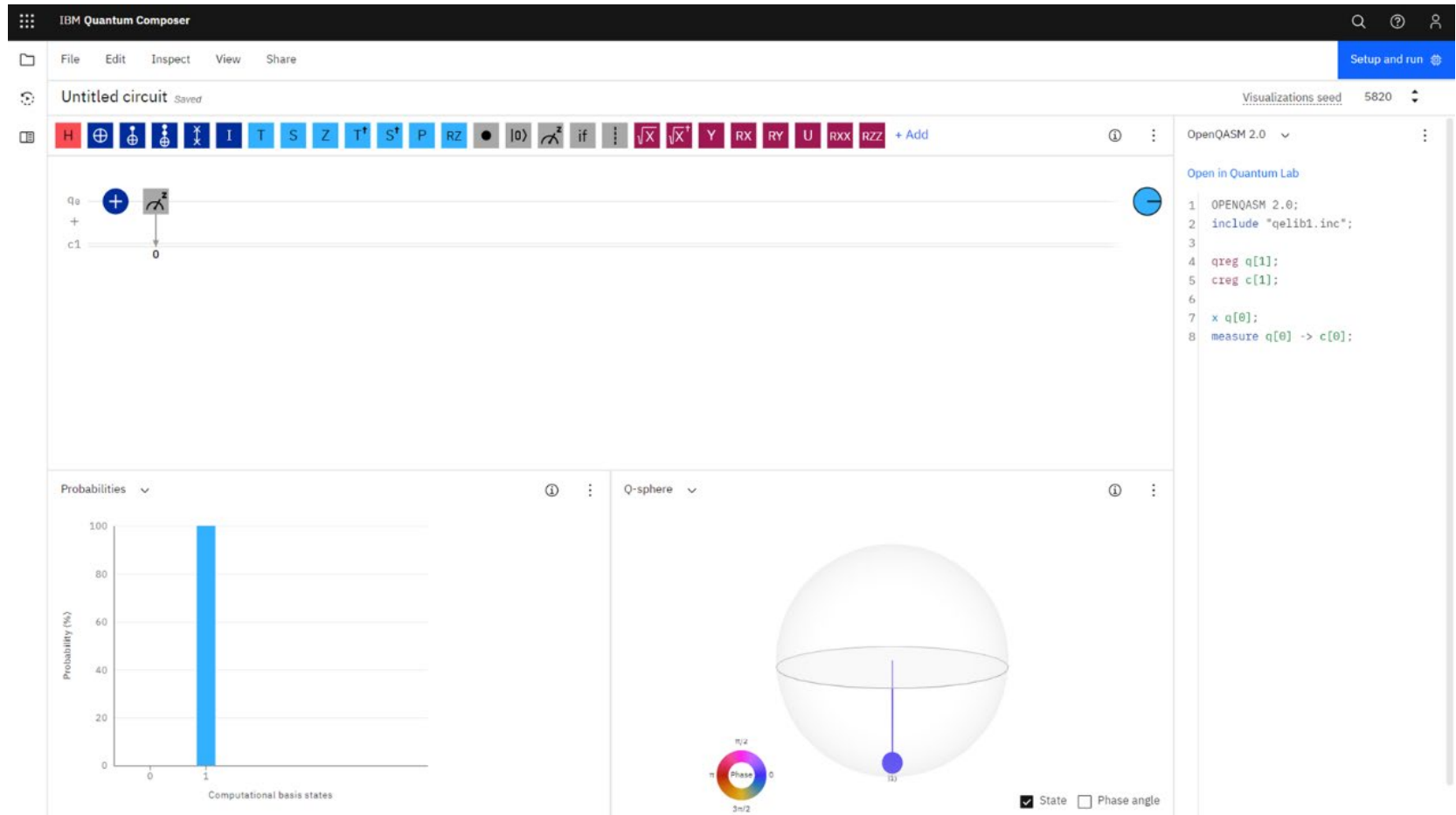
Probabilities

Q-sphere

State ☒ Phase angle ☐

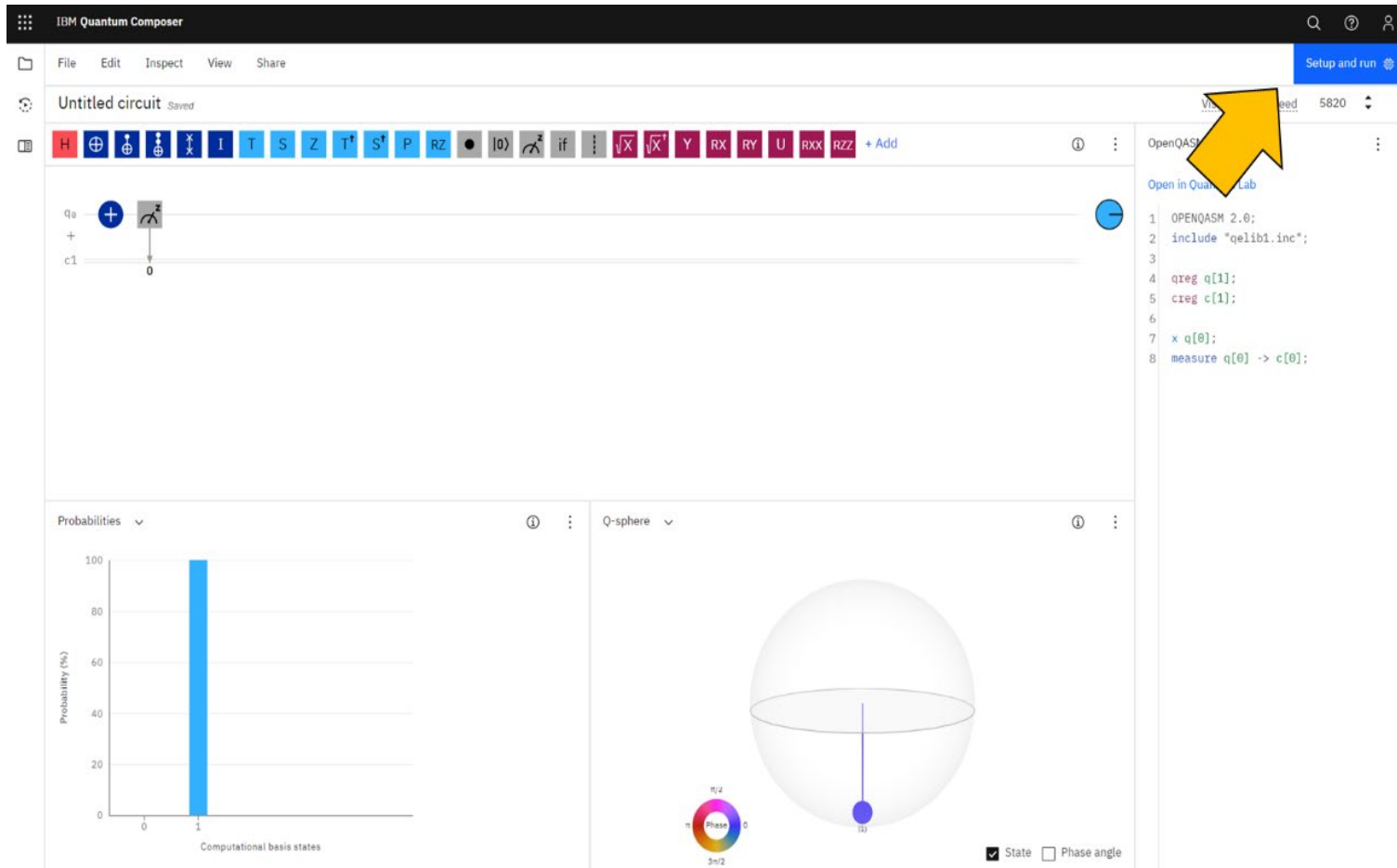
Computational basis states	Probability (%)
00	0
01	100
10	0
11	0

Your circuit is now looking like this



## Step 5 : Setup your Circuit

Select the parameters to run your circuit by clicking on the top right button:



The screenshot displays the IBM Quantum Composer interface. At the top, the title bar reads "IBM Quantum Composer". Below it is a menu bar with "File", "Edit", "Inspect", "View", and "Share". A toolbar contains various quantum gates and operations. The main workspace shows a quantum circuit with two qubits,  $q_0$  and  $c_1$ .  $q_0$  has a Hadamard gate and a measurement gate.  $c_1$  has a measurement gate. The circuit is labeled "Untitled circuit" and "Saved".

On the right side, there is a "Setup and run" button, which is highlighted by a yellow arrow. Below this button, there is a section for "OpenQASM" code, showing the following code:

```
1 OPENQASM 2.0;  
2 include "qelib1.inc";  
3  
4 qreg q[1];  
5 creg c[1];  
6  
7 x q[0];  
8 measure q[0] -> c[0];
```

At the bottom left, there is a "Probabilities" section showing a bar chart for the computational basis states. The y-axis is labeled "Probability (%)" and ranges from 0 to 100. The x-axis is labeled "Computational basis states" and has values 0 and 1. The bar for state 1 is at 100%.

At the bottom right, there is a "Q-sphere" section showing a Bloch sphere visualization. The sphere is centered at the origin, and a point is plotted on the surface. A legend below the sphere indicates "State" (checked) and "Phase angle" (unchecked).

You should now see

The screenshot displays the IBM Quantum Composer interface. On the left, a quantum circuit is shown with two qubits,  $q_0$  and  $c1$ .  $q_0$  has a Hadamard gate and a CNOT gate controlled by  $c1$ .  $c1$  has a Z gate. The circuit is titled "Untitled circuit" and is saved. Below the circuit, a bar chart shows the probabilities of the computational basis states. The y-axis is labeled "Probability (%)" and ranges from 0 to 100. The x-axis is labeled "Computational basis states" and has two bars: one at state 0 with a probability of 100%, and one at state 1 with a probability of 0%. The chart is titled "Probabilities" and has a "Q-sphere" dropdown menu. On the right, a panel titled "Set up and run your circuit" is open. It contains two steps: "Step 1: Choose a system or simulator" and "Step 2: Choose your settings". In Step 1, a list of systems is shown, with  $ibmq\_santiago$  selected. The list includes  $ibmq\_santiago$ ,  $ibmq\_athens$ ,  $ibmq\_belem$ ,  $ibmq\_quito$ , and  $ibmq\_16\_melbourne$ . Each system entry shows its status (Online), total pending jobs, and qubit count. In Step 2, the provider is set to  $ibm-q/open/main$ , the number of shots is 1000, and the job limit is 5 remaining. The job name is "e.g. Untitled circuit job". At the bottom right, there is a blue button labeled "Run on ibmq\_santiago" and a "Close" button.

IBM Quantum Composer

File Edit Inspect View Share

Untitled circuit Saved

q<sub>0</sub> +  
+  
c1 0

Probabilities

100  
80  
60  
40  
20  
0

0 1

Computational basis states

Q-sphere

Set up and run your circuit

Step 1  
Choose a system or simulator

Search by system or simulator name

☒  $ibmq\_santiago$  See details  
System status Online  
Total pending jobs 5944  
5 Qubits 32 Quantum volume

☐  $ibmq\_athens$  See details  
System status Online  
Total pending jobs 5771  
5 Qubits 32 Quantum volume

☐  $ibmq\_belem$  See details  
System status Online  
Total pending jobs 15  
5 Qubits 16 Quantum volume

☐  $ibmq\_quito$  See details  
System status Online  
Total pending jobs 457  
5 Qubits 16 Quantum volume

☐  $ibmq\_16\_melbourne$  See details

Step 2  
Choose your settings

Provider  
 $ibm-q/open/main$

Shots  
1000

Job limit: 5 remaining

Optional  
Name your job

Job name  
e.g. Untitled circuit job

Tags  
Add tags

Close Run on  $ibmq\_santiago$

Complete the information requested. In *Step 1: Choose a system or simulator*, select the **ibmq\_qasm\_simulator**.

The screenshot displays the IBM Quantum Composer interface. On the left, a quantum circuit diagram is visible, showing a single qubit with a Hadamard gate and a measurement. Below the circuit, a probability bar chart shows a 100% probability for the state '1'. The main panel on the right is titled 'Set up and run your circuit' and contains two steps:

- Step 1: Choose a system or simulator**
  - Search by system or simulator name
  - 100 Qubits
    - ☐ simulator\_extended\_stabilizer [See details](#)
      - Simulator status: Online
      - Total pending jobs: 2
    - 63 Qubits
      - ☐ simulator\_statevector [See details](#)
        - Simulator status: Online
        - Total pending jobs: 2
      - 32 Qubits
        - ☒ **ibmq\_qasm\_simulator** [See details](#)
          - Simulator status: Online
          - Total pending jobs: 3
        - 32 Qubits
          - ☐ ibmq\_5\_yorktown [See details](#)
            - System status: Maintenance
            - Total pending jobs: 2070
          - 5 Qubits 8 Quantum volume
  - Step 2: Choose your settings**
    - Provider: ibmq-q/open/main
    - Shots: 1000
    - Job limit: 5 remaining
    - Optional: Name your job (Job name: e.g., Untitled circuit job)
    - Tags: Add tags

At the bottom of the dialog, there are two buttons: 'Close' and 'Run on ibmq\_qasm\_simulator'.

Then in *Step 2: Choose your settings*, type 100 to use 100 simulations or steps and name the circuit TEST\_1.

The screenshot displays the IBM Quantum Composer interface. On the left, a quantum circuit is shown with a qubit  $q_0$  and a classical bit  $c_1$ . The circuit includes a Hadamard gate on  $q_0$ , followed by a CNOT gate with  $q_0$  as control and  $c_1$  as target. The  $c_1$  line then branches into two paths, each ending with a measurement gate. The bottom left shows a probability plot with a single bar at 100% for the state  $|1\rangle$ .

On the right, the 'Set up and run your circuit' panel is open, showing two steps:

- Step 1: Choose a system or simulator**
  - Search by system or simulator name
  - 100 Qubits
  - ☐ simulator\_extended\_stabilizer (Online, 2 pending jobs)
  - ☐ simulator\_statevector (Online, 2 pending jobs)
  - ☒ ibmq\_qasm\_simulator (Online, 3 pending jobs)
  - ☐ ibmq\_5\_yorktown (Maintenance, 2070 pending jobs)
- Step 2: Choose your settings**
  - Provider: ibm-q/open/main
  - Shots: 100
  - Job limit: 5 remaining
  - Optional: Name your job
  - Job name: TEST\_1
  - Tags: Add tags

At the bottom right, there is a blue button labeled 'Run on ibmq\_qasm\_simulator'.



## Step 6 Run your Circuit

Click on the bottom right button to run the ibmq-qasm-simulator as below:

The screenshot displays the IBM Quantum Composer interface. On the left, a quantum circuit is shown with a qubit labeled 'q0' and a classical bit labeled 'c1'. The circuit includes a Hadamard gate on 'q0', followed by a CNOT gate with 'q0' as the control and 'c1' as the target. Below the circuit, a bar chart titled 'Probabilities' shows the probability of computational basis states. The x-axis is labeled 'Computational basis states' with values 0 and 1. The y-axis is labeled 'Probability (%)' with values from 0 to 100. A single bar at state 1 reaches 100% probability.

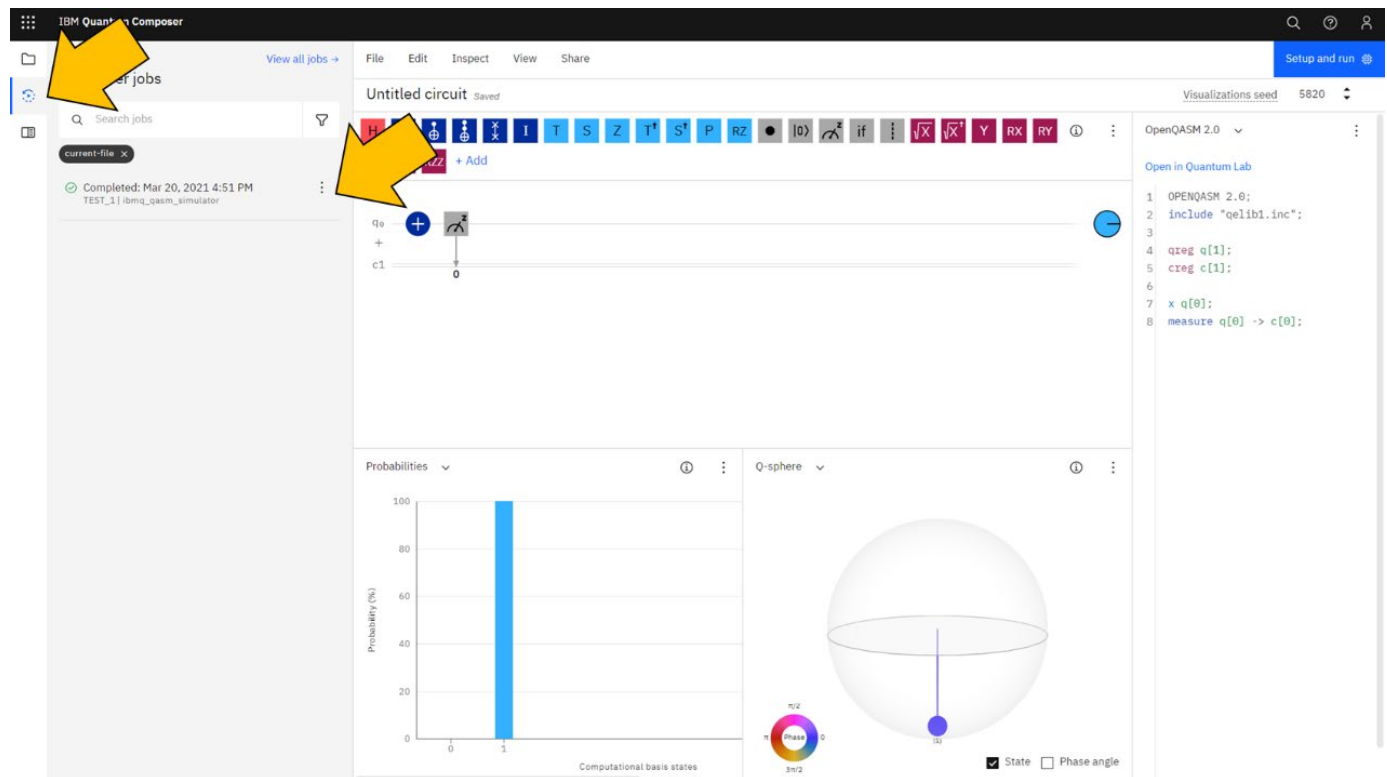
On the right, a panel titled 'Set up and run your circuit' is open. It contains two steps:

- Step 1: Choose a system or simulator**
  - Search by system or simulator name
  - 100 Qubits
    - ☐ simulator\_extended\_stabilizer [See details](#)
      - Simulator status: Online
      - Total pending jobs: 2
    - 63 Qubits
      - ☐ simulator\_statevector [See details](#)
        - Simulator status: Online
        - Total pending jobs: 2
      - 32 Qubits
        - ☒ ibmq\_qasm\_simulator [See details](#)
          - Simulator status: Online
          - Total pending jobs: 3
        - 32 Qubits
          - ☐ ibmq\_5\_yorktown [See details](#)
            - System status: Maintenance
            - Total pending jobs: 2070
          - 5 Qubits 8 Quantum volume- Step 2: Choose your settings**
  - Provider: ibmq-q/open/main
  - Shots: 100
  - Job limit: 5 remaining
  - Optional: Name your job
    - Job name: TEST\_1
  - Tags: Add tags

A large yellow arrow points from the 'ibmq\_qasm\_simulator' option in Step 1 to the 'Run on ibmq\_qasm\_simulator' button at the bottom right of the panel.

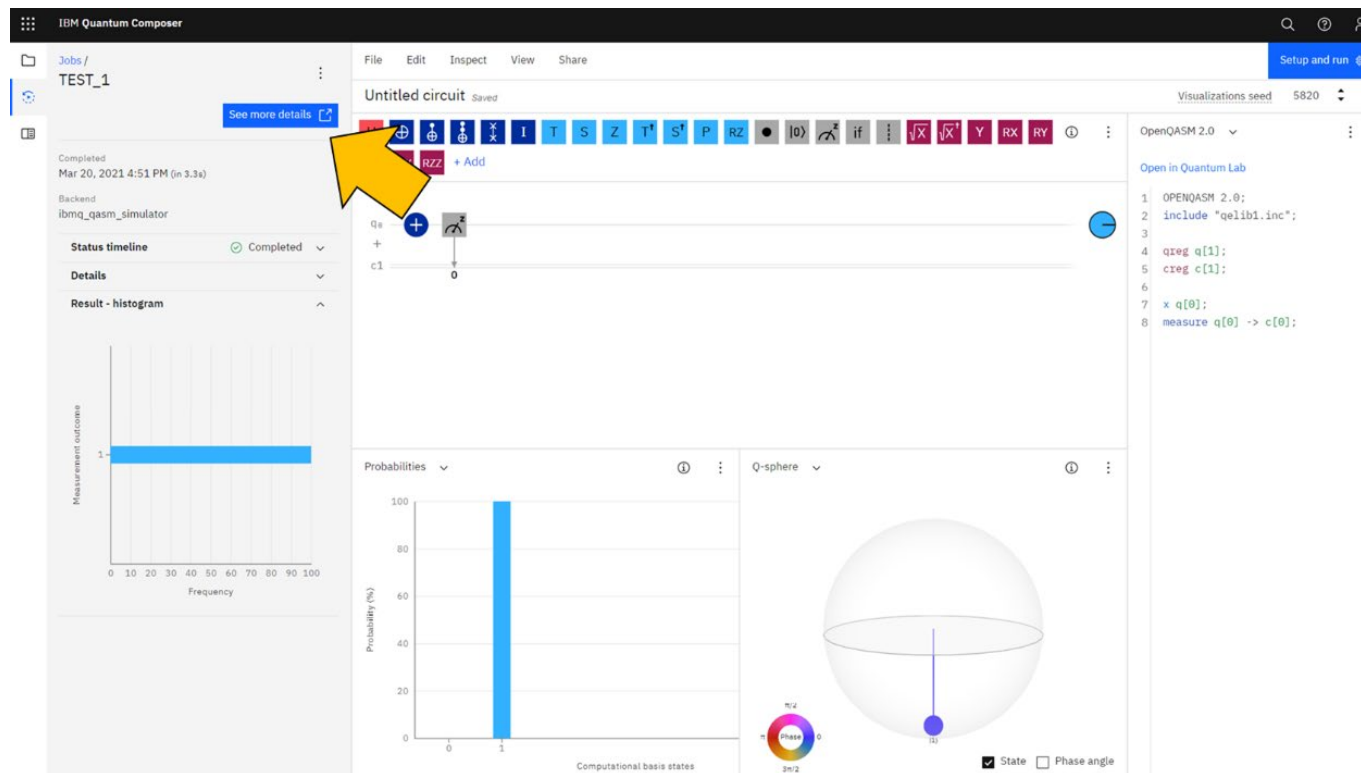
## Step 7 Review the progress of your job

After a few seconds, click on the left button **Composer Jobs** to review the status of your job. In our case we have done a very simple circuit which will be calculated very quickly. For more complex jobs, more time will be required.

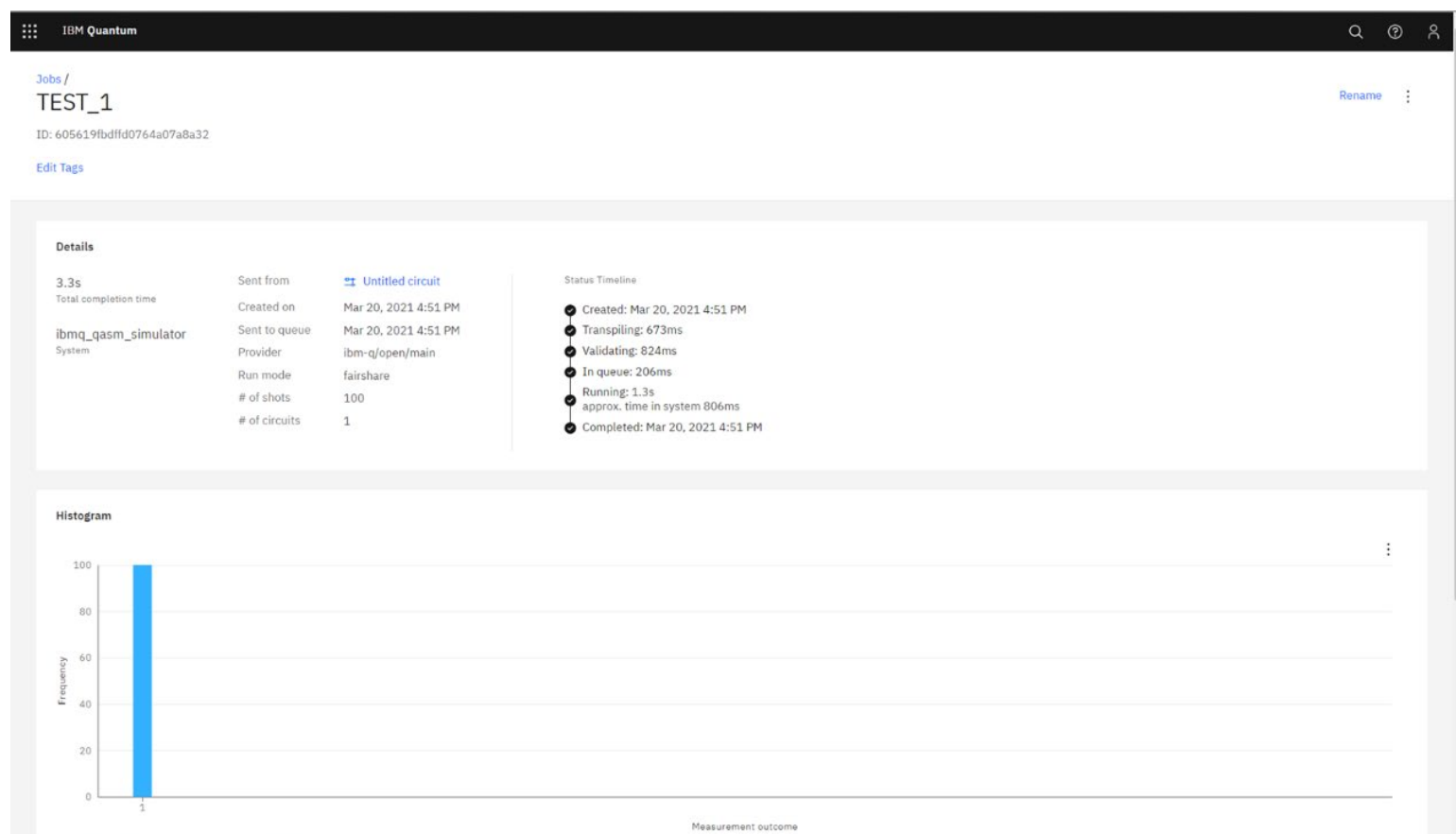


## Step 8 : Get your simulation results

Clicking on the name of the job will open the results of the simulation in a new tab in your browser.



Clicking on the blue button “See More Details” will show more results in a new tab in the browser:



Congratulations! You have now created and executed your first quantum circuit. As you can see in the Histogram section, the probabilities of measuring a 1 in the qubit are 100%, just as we expected. In the next section, we will explore a complementary tool called Qiskit, which will allow us to do the same thing within Python.



**Qiskit**

# Qiskit: an open-source framework for quantum computing

Qiskit is an open-source software development kit (SDK) for working with quantum computers. It is developed and maintained by IBM. It allows users to easily develop quantum programs as part of a Python environment. It is extremely flexible, allowing direct and detailed manipulation of circuits, pulses, and algorithms. These can then be run locally (in your own computer) or remotely (in one of the real quantum computers of IBM). Qiskit can be found in <https://qiskit.org/>.

# **Lab 2:**

## **Your first circuit step-by-step with QISKIT**



## Lab 2: Your first circuit step-by-step with QISKIT

As we said, Qiskit then can be understood as module or library for Python, which allow us to run quantum programs. Note that it can be used for both: running the quantum program in a simulator or running the program in a real quantum computer. Here, we are going to repeat the exercise we did before with the IBM Quantum inLab 1, i.e. flipping a single qubit. But instead of running it online, we are going to run it in our own computer using our local Python environment.

**TASK:** To flip a single qubit from  $|0\rangle$  to  $|1\rangle$ .

**(LOCAL) SOLUTION:** Create a quantum circuit with a single qubit and a X Gate in Python using the module Qiskit.


## Step 1: Install Python in your computer

[Products ▾](#)[Pricing](#)[Solutions ▾](#)[Resources ▾](#)[Blog](#)[Company ▾](#)

Individual Edition

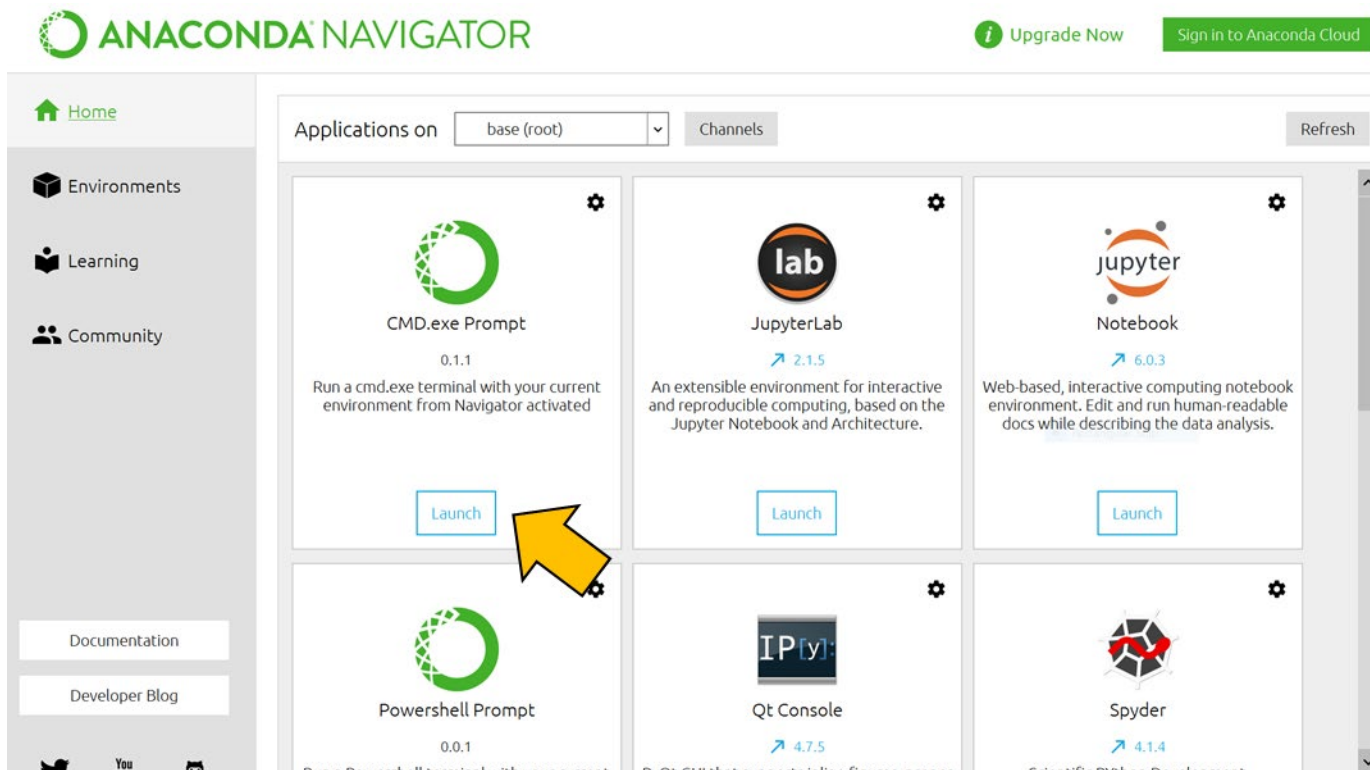
# Your data science toolkit

With over 20 million users worldwide, the open-source Individual Edition (Distribution) is the easiest way to perform Python/R data science and machine learning on a single machine. Developed for solo practitioners, it is the toolkit that equips you to work with thousands of open-source packages and libraries.

[Download](#)

## Step 2: Install the Qiskit module into your Python

Open the Anaconda Navigator and launch the CMD.exe Prompt.



## Step 3: Install the Qiskit tool into your local Python

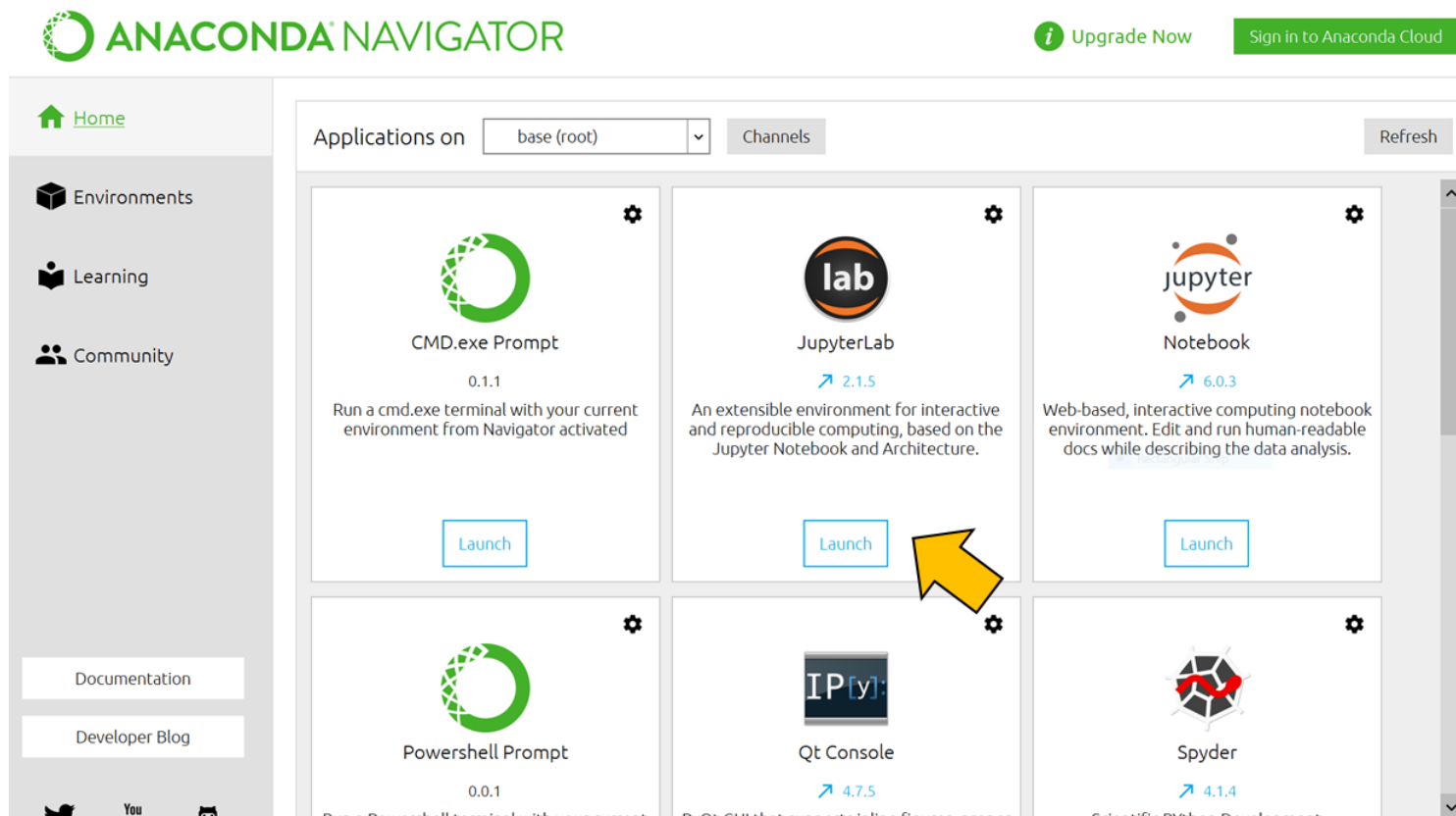
Type in the command line: `pip install qiskit`

```
C:\WINDOWS\system32\cmd.exe - pip install qiskit
Microsoft Windows [Version 10.0.19041.450]
(c) 2020 Microsoft Corporation. All rights reserved.

(base) C:\Users\apena>pip install qiskit
Collecting qiskit
  Downloading qiskit-0.23.1.tar.gz (4.1 kB)
Collecting qiskit-terra==0.16.1
  Downloading qiskit_terra-0.16.1-cp38-cp38-win_amd64.whl (7.8 MB)
    | 7.8 MB 819 kB/s
Collecting qiskit-aer==0.7.1
  Downloading qiskit_aer-0.7.1-cp38-cp38-win_amd64.whl (24.0 MB)
    | 24.0 MB 1.3 MB/s
Collecting qiskit-ibmq-provider==0.11.1
  Downloading qiskit_ibmq_provider-0.11.1-py3-none-any.whl (195 kB)
    | 195 kB 1.3 MB/s
Collecting qiskit-ignis==0.5.1
  Downloading qiskit_ignis-0.5.1-py3-none-any.whl (204 kB)
    | 204 kB 1.6 MB/s
Collecting qiskit-aqua==0.8.1
  Downloading qiskit_aqua-0.8.1-py3-none-any.whl (2.1 MB)
    | 2.1 MB 1.1 MB/s
Collecting networkx>=0.5.0
  Downloading networkx-0.7.1-cp38-cp38-win_amd64.whl (527 kB)
    | 527 kB 819 kB/s
Collecting dill>=0.3
  Downloading dill-0.3.3-py2.py3-none-any.whl (81 kB)
    | 81 kB 862 kB/s
Requirement already satisfied: ply>=3.10 in c:\users\apena\anaconda3\lib\site-packages (from qiskit-terra==0.16.1->qiskit) (3.11)
Requirement already satisfied: scipy>=1.4 in c:\users\apena\anaconda3\lib\site-packages (from qiskit-terra==0.16.1->qiskit) (1.4.1)
```

## Step 4: Launch Jupyter Lab

From the Anaconda Explorer launch Jupyter Lab



## Step 5: Type the code below into the first line of a Jupyter Lab Notebook and Run

```
import qiskit

qr = qiskit.QuantumRegister(1) # create qubit
cr = qiskit.ClassicalRegister(1) # create classical bit

program = qiskit.QuantumCircuit(qr, cr) # create quantum circuit with one qubit
and one classical bit

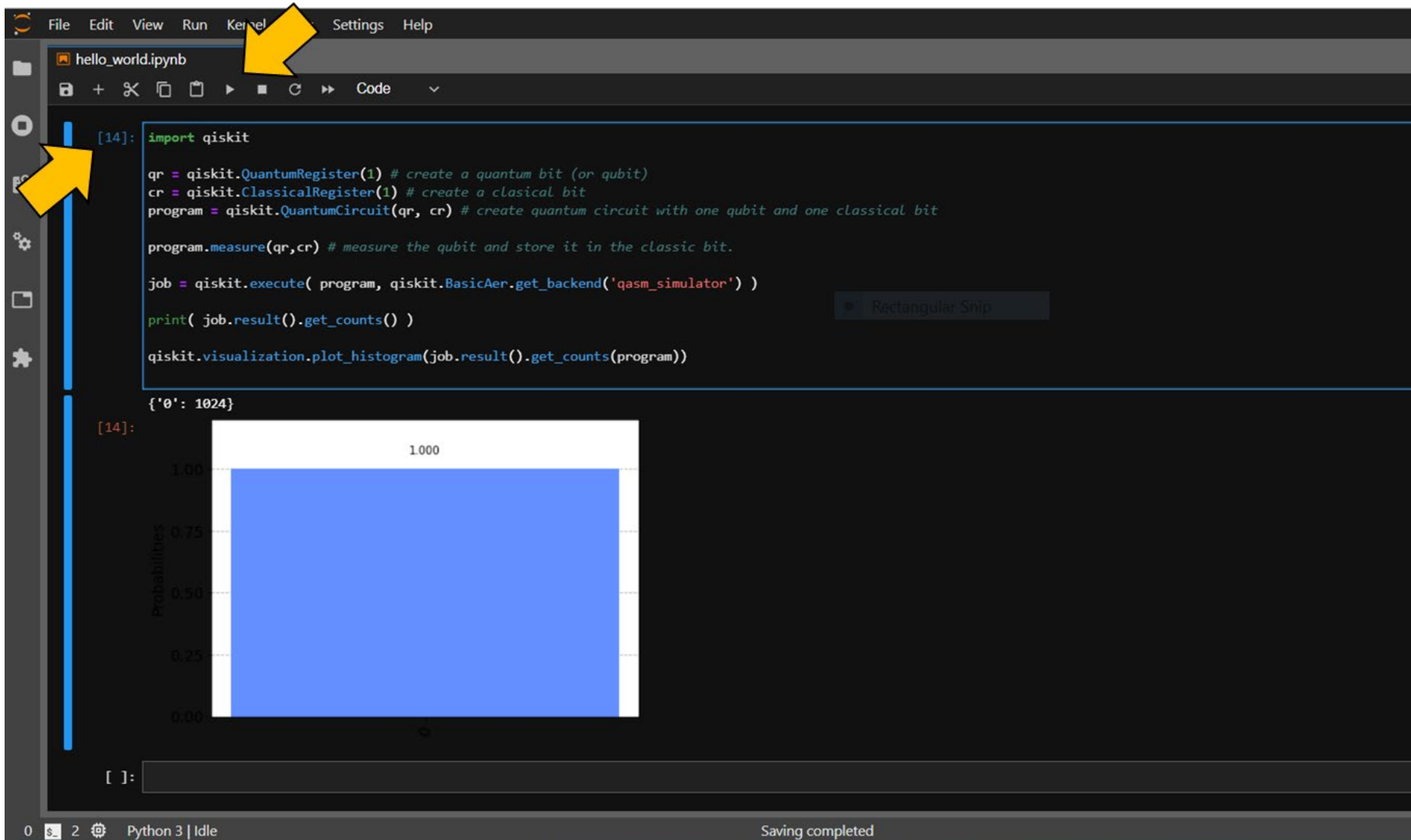
program.x(qr[0]) # apply the X Gate to invert qubit

program.measure(qr,cr) # measure the qubit and store it in the classic bit


job = qiskit.execute( program, qiskit.BasicAer.get_backend('qasm_simulator') ) #
run job

print(job.result().get_counts()) # print numerical results

qiskit.visualization.plot_histogram(job.result().get_counts(program)) # draw
histogram
```



Well done! Your result indicates that the simulation has been called 1024 times (the so-called shots in the IBM Q Experience parlance) and that in all cases the result is that the qubit has been flipped from 0 to 1.





## References

*Main reference book for quantum computing:*

Michael A. Nielsen, Isaac L. Chuang. Quantum Computation and Quantum Information: 10th Anniversary Edition. Cambridge University Press, 2010.

*On quantum computing in general:*

Robert S. Sutor. Dancing with Qubits: How quantum computing works and how it can change the world. Packt Publishing, 2019

Noson S. Yanofsky, Mirco A. Mannucci. Quantum Computing for Computer Scientists. Cambridge University Press, 2008.

Wolfgang Scherer. Mathematics of Quantum Computing: An Introduction. Springer-Nature, 2019.

Rajendra K. Bera. The Amazing World of Quantum Computing. Springer Nature, 2020.

### *On using IBM solutions:*

Robert Lored. Learn Quantum Computing with Python and IBM Quantum Experience. Packt Publishing, 2020.

Hassi Norlén. Quantum Computing in Practice with Qiskit and IBM Quantum Experience. Packt Publishing, 2020.

### *Online Courses:*

Quantum Computing Fundamentals, MIT xPRO

<https://learn-xpro.mit.edu/quantum-computing>

Understanding Quantum Computers

<https://www.futurelearn.com/courses/intro-to-quantum-computing>

## *Research Papers:*

Woerner, S., Egger, D.J. Quantum risk analysis. npj Quantum Inf 5, 15 (2019).

D. Egger, R. Garcia Gutierrez, J. Cahue Mestre and S. Woerner, "Credit Risk Analysis using Quantum Computers" in IEEE Transactions on Computers, vol. , no. 01, pp. 1-1, 5555, 2019.

Jacak, J.E., Jacak, W.A., Donderowicz, W.A. et al. Quantum random number generators with entanglement for public randomness testing. Sci Rep 10, 164 (2020).

Patrick Rebentrost, Brajesh Gupt, and Thomas R. Bromley. Quantum computational finance: Monte Carlo pricing of financial derivatives. Phys. Rev. A 98, 022321, 2018.

Nikitas Stamatopoulos, Daniel J. Egger, Yue Sun, Christa Zoufal, Raban Iten, Ning Shen, and Stefan Woerner. Option Pricing using Quantum Computers. Quantum 4, 291 (2020).

Panagiotis K. Barkoutsos, Giacomo Nannicini, Anton Robert, Ivano Tavernelli, and Stefan Woerner. Improving Variational Quantum Optimization using CVaR. Quantum 4, 256 (2020).