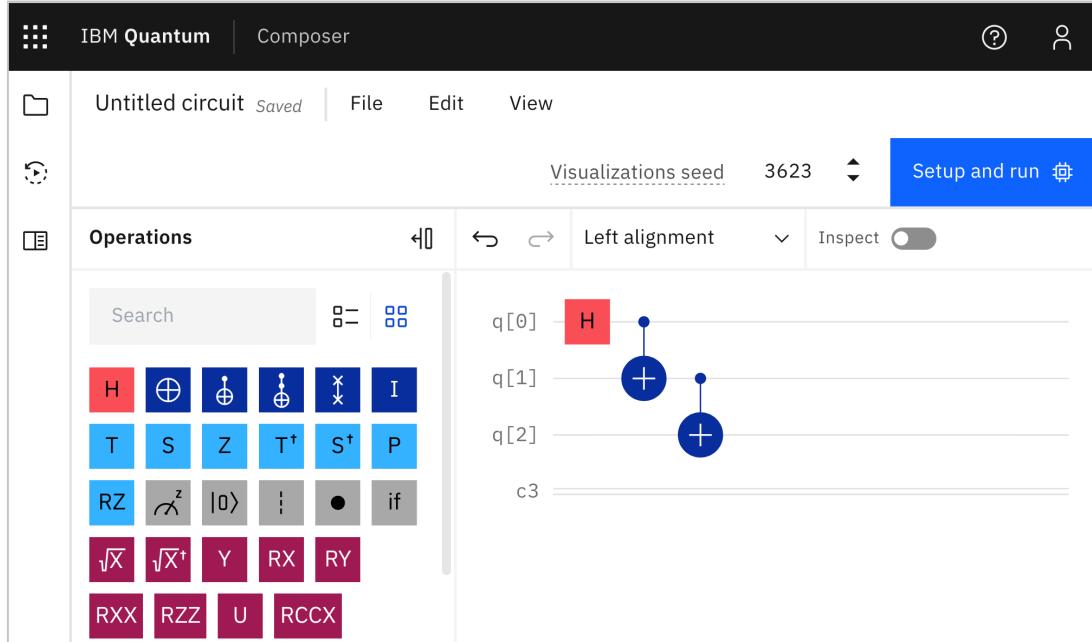


# 量子ゲート基礎 IBM Quantum Composer

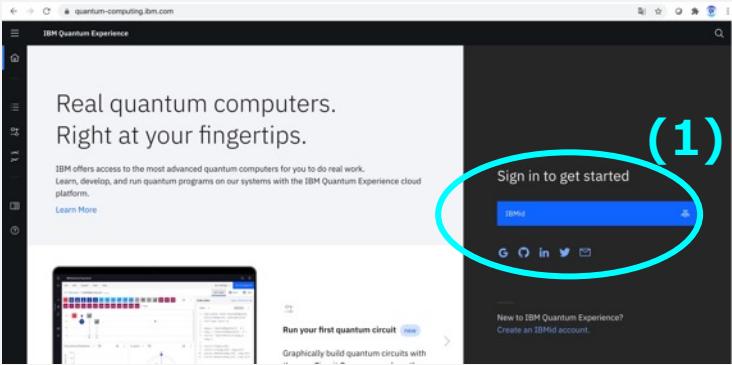
沼田 祈史

Kifumi Numata  
IBM Quantum

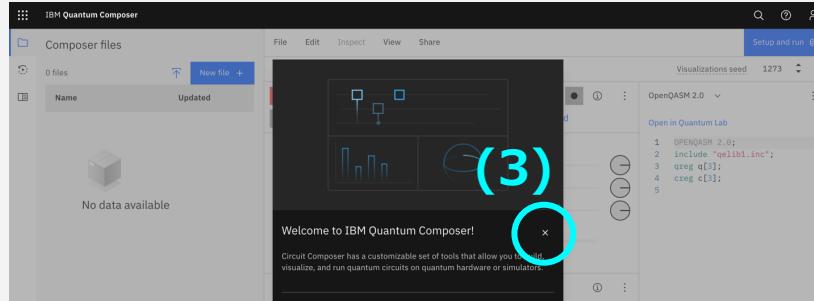


# ハンズオン：いつしょにやってみましょう！

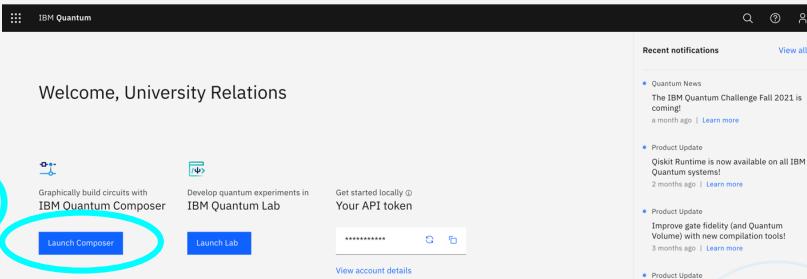
(1) IBM Quantum にログインします。URL:  
<https://quantum-computing.ibm.com/>



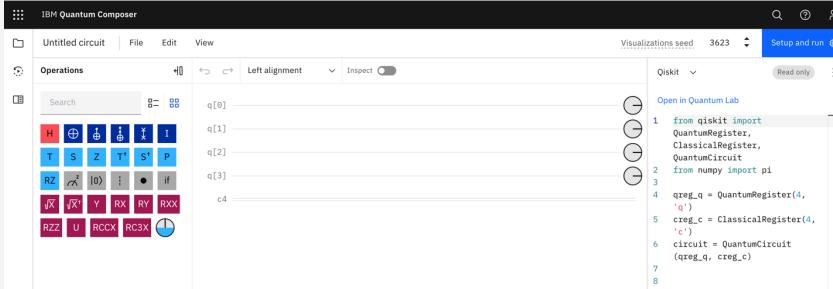
(3) ポップアップウィンドウは「x」をクリックして、閉じます。



(2) 左の青アイコン「Launch Composer」をクリック。



(4) この画面になつたら準備完了です。



# ハンズオンの資料

URL: [ibm.biz/kwskgit](https://ibm.biz/kwskgit)

The screenshot shows a GitHub repository page for 'quantum-tokyo / kawasaki-quantum-camp'. The 'Code' tab is selected. The repository is public. The main branch is 'main'. There is 1 branch and 0 tags. The commit history is as follows:

- kifumi Merge pull request #5 from quantum-tokyo/kifumi ... 0e8e409 last week 25 commits
- 2022 update readme last week
- day1 add folders last week
- day2 add folders last week
- day3 add folders last week
- .gitignore upload day1 materials last year
- README.md update readme last week

At the bottom, there is a file named 'README.md' with the text 'Kawasaki Quantum Summer Camp 2023'.

# 1量子ビット回路

The screenshot shows the IBM Quantum Composer interface. At the top, it says "IBM Quantum Composer". Below that is a toolbar with "Untitled circuit", "File", "Edit", "View", and "Visu". The main area is titled "Operations" and includes a search bar, alignment tools, and an "Inspect" toggle. On the left is a palette of quantum gates: H, CNOT, Toffoli, Swap, Z, T, S, RZ, RY, RY<sup>†</sup>, V, P, P<sup>†</sup>, PYY, PYY<sup>†</sup>, and If. There are four horizontal lines representing qubits: q[0], q[1], q[3], and c4. q[0] has a red H gate applied. q[1] has a blue CNOT gate applied. q[3] has a blue Swap gate applied. c4 is a classical register. A red arrow points to the trash bin icon next to q[1].

マウスでq[1]をクリックするとゴミ箱マークが出てくるので、  
クリックして消します。  
q[0]だけにして、1量子ビット回路の準備をします。

# 1量子ビット回路

The screenshot shows the IBM Quantum Composer interface. On the left, there's a toolbar labeled "量子ゲート" (Quantum Gates) with various gate icons. A red circle highlights the Hadamard gate (H). A red arrow points from this highlighted gate to the "Operations" section of the main workspace, which contains a search bar and a grid of quantum operations including H, T, RZ, and others. Another red arrow points from the "Operations" section to the workspace itself, where a single-qubit quantum register q[0] and a classical register c4 are shown. The workspace has a "Left alignment" button and an "Inspect" toggle. On the right, a large red arrow points from the workspace to a code editor window titled "Qiskit". The code editor displays the following Qiskit code:

```
from qiskit import QuantumRegister, ClassicalRegister, QuantumCircuit
from numpy import pi
qreg_q = QuantumRegister(1, 'q')
creg_c = ClassicalRegister(4, 'c')
circuit = QuantumCircuit(qreg_q, creg_c)
```

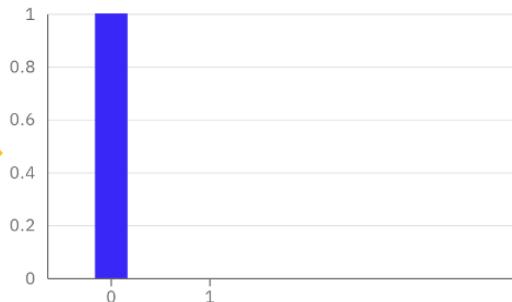
量子ゲートをマウスでドラッグ&ドロップして、  
量子回路を作ります。  
右側には、Qiskitのコードが自動生成されます。

# 1. Xゲート(NOTゲート)

図の回路を作つてみてください。下に表示される棒グラフの変化を確認しましょう。



初期状態は $|0\rangle$



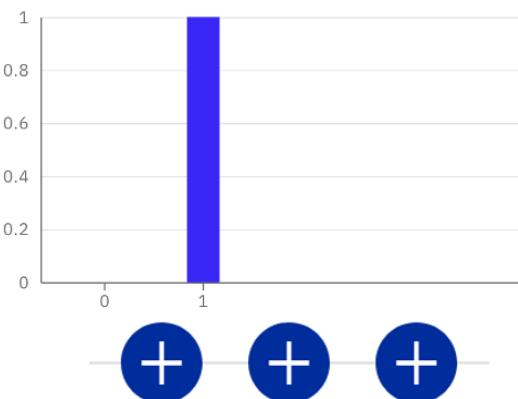
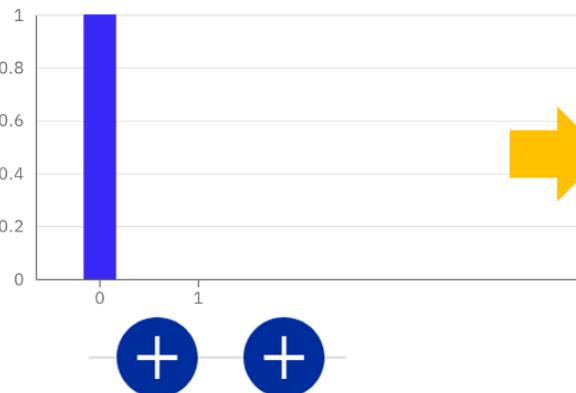
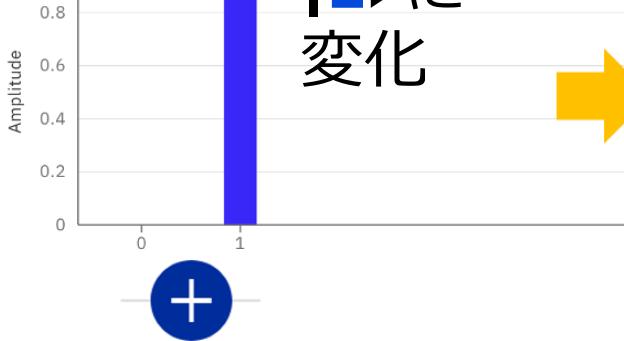
棒グラフ (Statevector 表示) は  
量子ビットの状態

$\alpha \times |0\rangle + \beta \times |1\rangle$   
の  $\alpha, \beta$  (確率振幅) です。

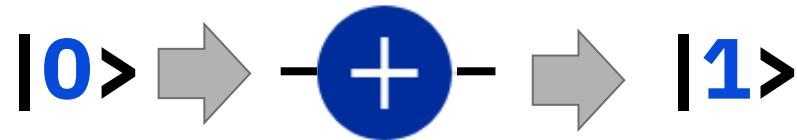
$$|0\rangle = 1 \times |0\rangle + 0 \times |1\rangle$$

$$|1\rangle = 0 \times |0\rangle + 1 \times |1\rangle$$

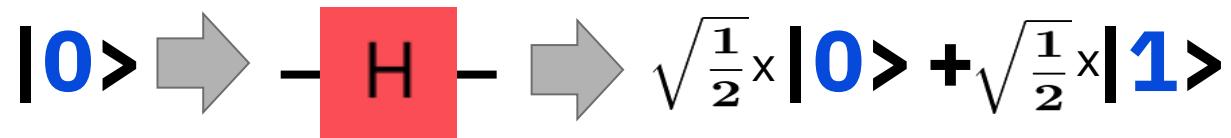
$|1\rangle$ に  
変化



# 量子コンピューターの計算方法

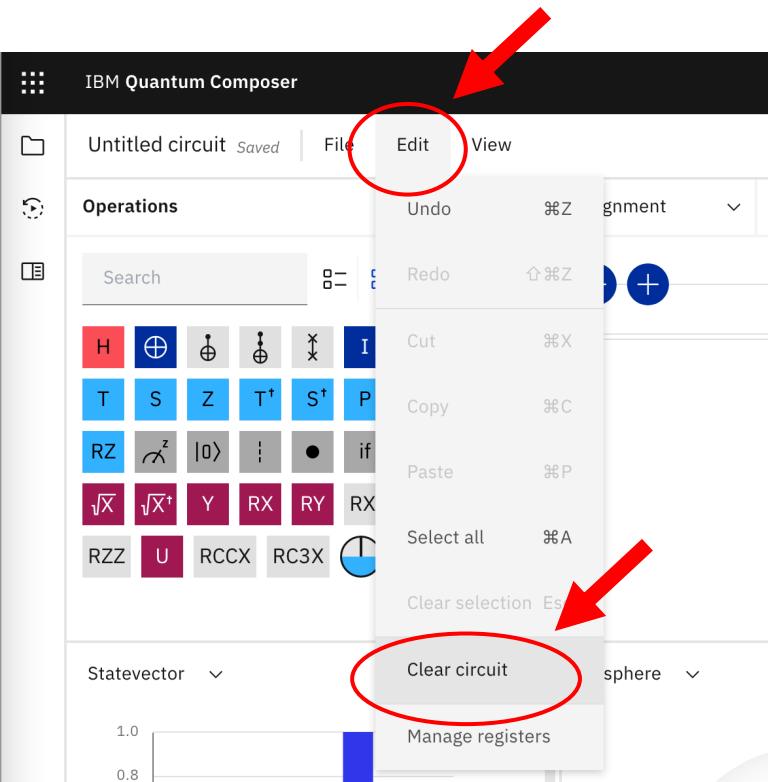


ノット（反転）ゲート

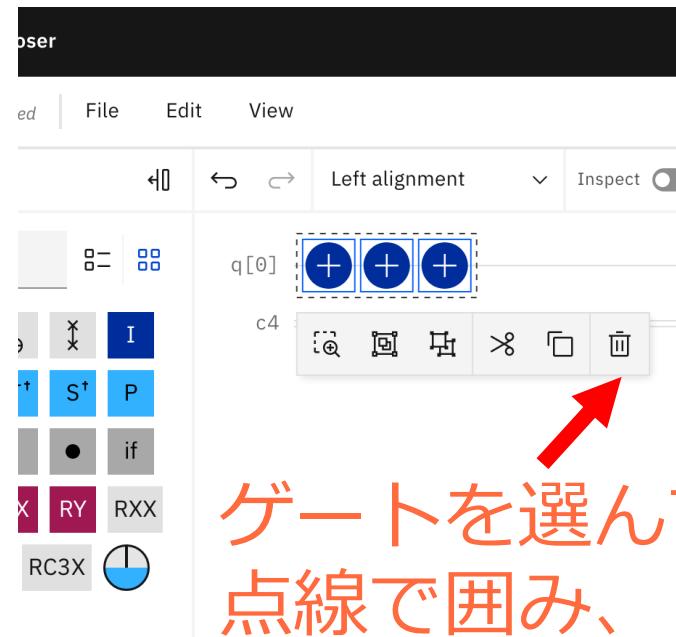


重ね合わせをつくる

# 置いたゲートを取り除く



または

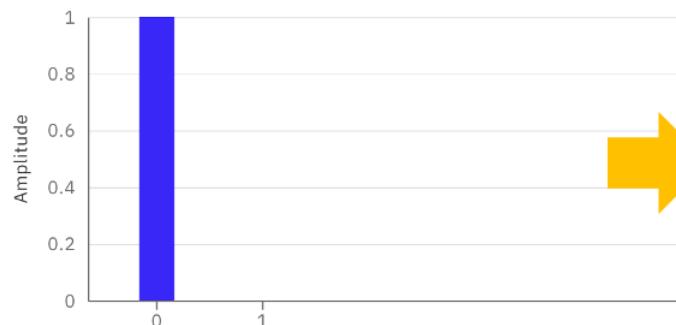
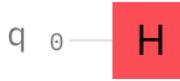


ゲートを選んで  
点線で囲み、  
ゴミ箱マークを  
クリック

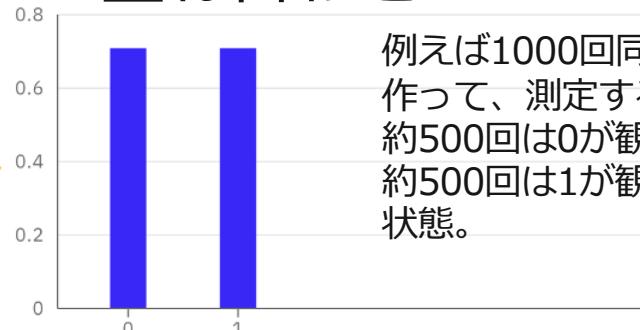
## 2. Hゲート

図の回路を作ってみてください。下に表示される棒グラフの変化を確認しましょう。

2-1)



重ね合わせ



例えば1000回同じ状態を作つて、測定すると約500回は0が観測され、約500回は1が観測される状態。

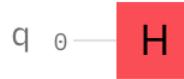
$$|0\rangle = 1 \times |0\rangle + 0 \times |1\rangle$$

$$\begin{aligned} & 0.707 \times |0\rangle + 0.707 \times |1\rangle \\ &= \frac{1}{\sqrt{2}} \times |0\rangle + \frac{1}{\sqrt{2}} \times |1\rangle \end{aligned}$$

## 2. Hゲート

図の回路を作ってみてください。下に表示される棒グラフの変化を確認しましょう。

2-1)



2-2)



2-3)



# 量子コンピューターの計算方法

$$|0\rangle \xleftarrow{H} \sqrt{\frac{1}{2}}|0\rangle + \sqrt{\frac{1}{2}}|1\rangle$$



$$|1\rangle \xleftarrow{H} \sqrt{\frac{1}{2}}|0\rangle - \sqrt{\frac{1}{2}}|1\rangle$$

# 3. Zゲート

図の回路を作ってみてください。下に表示される棒グラフの変化を確認しましょう。

3-1)



3-2)



3-3)



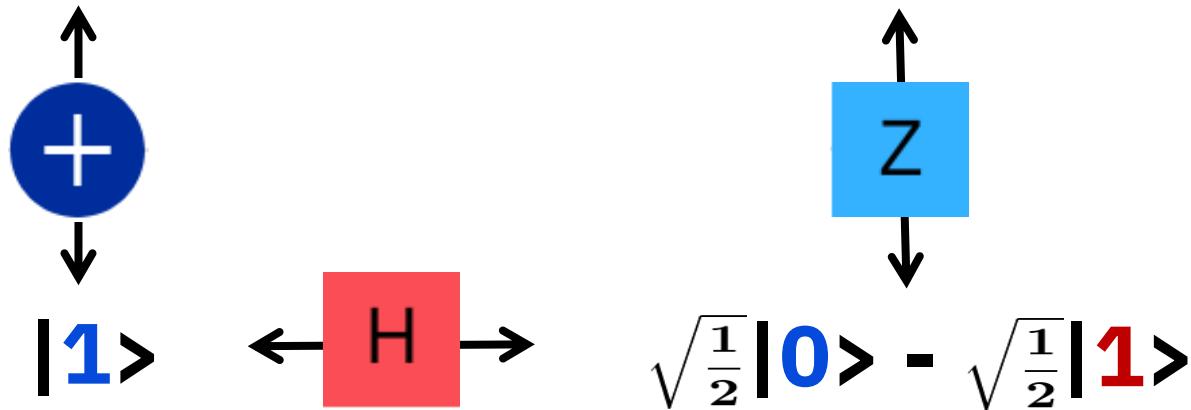
# 量子コンピューターの計算方法



1 の符号を反転する

# 量子コンピューターの計算方法

$$|0\rangle \xleftarrow{H} \sqrt{\frac{1}{2}}|0\rangle + \sqrt{\frac{1}{2}}|1\rangle$$


$$\begin{array}{c} \uparrow \\ + \\ \downarrow \end{array} \quad |1\rangle \xleftarrow{H} \sqrt{\frac{1}{2}}|0\rangle - \sqrt{\frac{1}{2}}|1\rangle$$

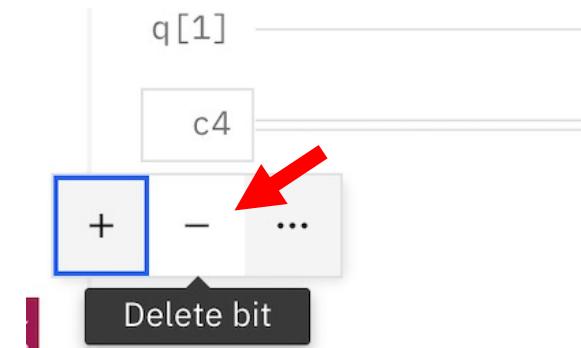
# 量子ビットを増やす

q[0]をクリックして、さらに「+」マークをクリックして、2量子ビットの回路を準備します。



2量子ビット回路  
になります

次に、c4をクリックして、「-」マークをクリックするを2回繰り返して、2古典ビットにします。



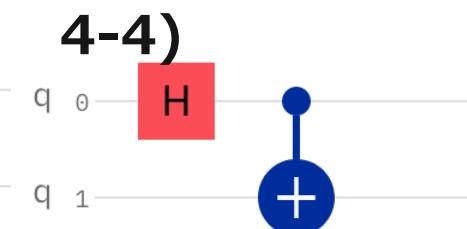
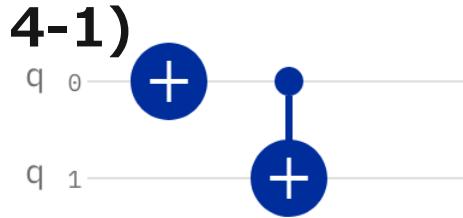
2古典ビットになります

# 4. CNOTゲート(制御Xゲート)

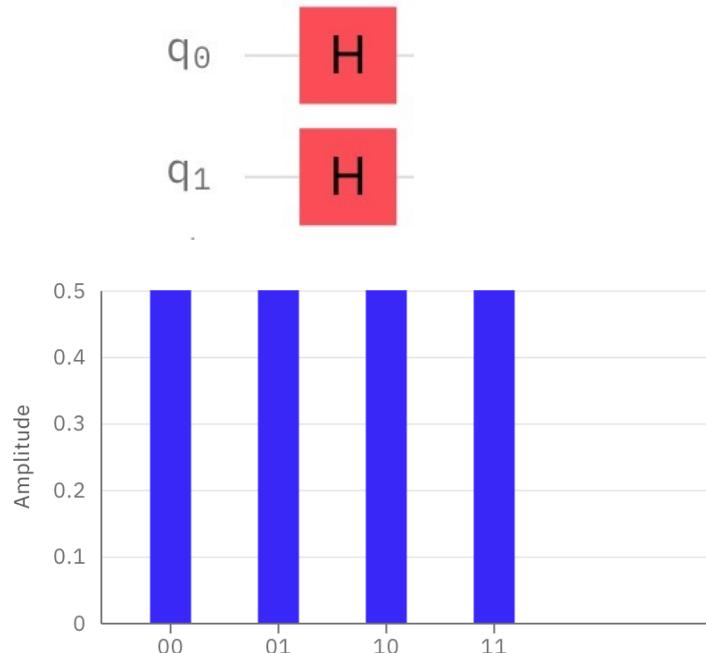
制御ビットが $|1\rangle$ のときのみ、目標ビットを反転（NOT）するゲートです。



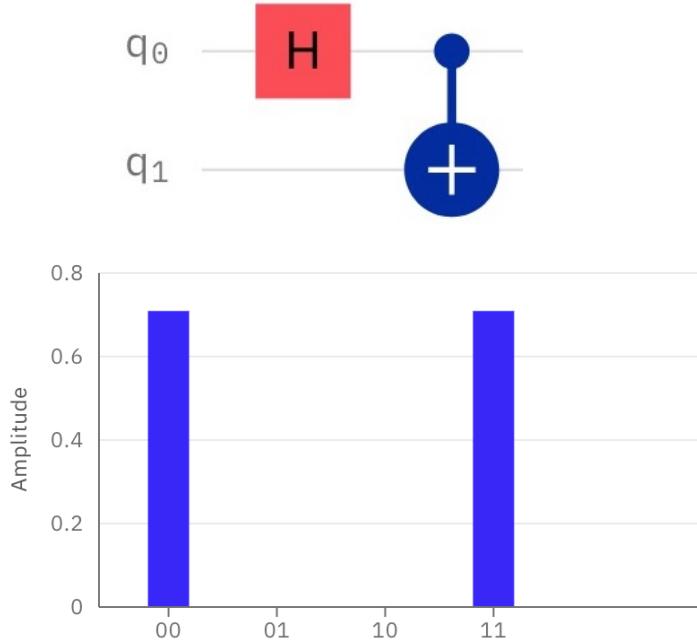
入力		出力	
制御ビット	目標ビット	制御ビット	目標ビット
0	0	0	0
0	1	0	1
1	0	1	1
1	1	1	0



# 量子重ね合わせ

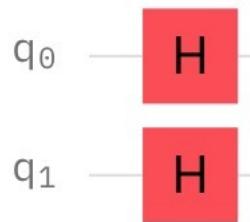


# 量子もつれ (エンタングルメント)



CNOTゲートは、  
エンタングルメントを作ります。

## 量子重ね合わせ



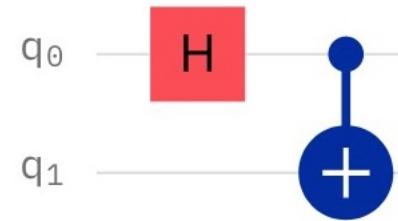
**0 0** ... 25%

**0 1** ... 25%

**1 0** ... 25%

**1 1** ... 25%

## 量子もつれ (エンタングルメント)



**0 0** ... 50%

**0 1** ... 0%

**1 0** ... 0%

**1 1** ... 50%

1量子ビット目が0だと分かったら  
2量子ビット目も0

# 5. 測定

ファイル名を変更  
(Entanglementなど)

IBM Quantum Compose

Untitled circuit Saved

Operations

Search

Operations palette:

- q[0]: H
- q[1]: +
- c2:  $\text{z}$

Measurement gate (q[1] to c2)

Visualizations seed: 3623

Qiskit

Open in Quantum Lab

```
1 from qiskit import QuantumRegister, ClassicalRegister, QuantumCircuit
2 from numpy import pi
3
4 qreg_q = QuantumRegister(2, 'q')
5 creg_c = ClassicalRegister(2, 'c')
6 circuit = QuantumCircuit(qreg_q, creg_c)
7
8 circuit.h(qreg_q[0])
9 circuit.cx(qreg_q[0], qreg_q[1])
10 circuit.measure(qreg_q[0], creg_c[0])
11 circuit.measure(qreg_q[1], creg_c[1])
```

(1) 測定ゲートを追加

(2) ファイル名を変更  
(Entanglementなど)

(3) クリック

# まず量子シミュレーターで実験

Set up and run your circuit

Step 1  
**Choose a system or simulator**

Search by system or simulator name

Total pending jobs 1

63 Qubits

**ibmq\_qasm\_simulator** See details

Simulator status • Online

Total pending jobs 1

32 Qubits

**simulator\_statevector** See details

Simulator status • Online

Total pending jobs 1

(4) スクロール

Step 2  
**Choose your settings**

Provider ibmq-internal/deployed/default

Shots \* 1024

Job name e.g. Untitled circuit job

(5) **ibmq\_qasm\_simulator** を選択

(6) Run on **ibmq\_qasm\_simulator**

Close

IBM Quantum Composer

Untitled circuit Saved File Edit View Visualizations seed 3623 Setup and run

Composer jobs

(7) Jobの確認

Search

Operations

q[0] H c2

q[1] + c2

z

Qiskit

Open in Quantum Lab

```
1 from qiskit import QuantumRegister, ClassicalRegister, QuantumCircuit
2 from numpy import pi
3
4 qreg_q = QuantumRegister(2, 'q')
```



IBM Quantum Composer

Untitled circuit Saved File Edit View Visualizations seed 3623 Setup and run

Composer jobs

View all jobs →

Search jobs

Completed: Jul 21, 2022 8:49 PM ID: 62d93d60aaf3a771... | ibmq\_qasm\_simulator

(8)

Operations

q[0] H

q[1] +

Qiskit

Open in Quantum Lab

```
1 from qiskit import QuantumRegister, ClassicalRegister,
```

Jobs /  
62d93d60aaaf3a771842b9ede

See more details ↗

Completed  
Jul 21, 2022 8:50 PM (in 8.3s)

Backend  
ibmq\_qasm\_simulator

Status timeline Completed

Details

Result - histogram

Untitled circuit Saved File Edit View

Visualizations seed

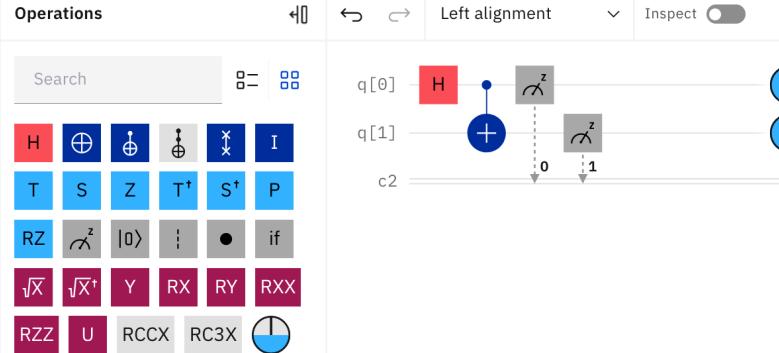
3623

Setup and run

Read only

(10)

再度実験します

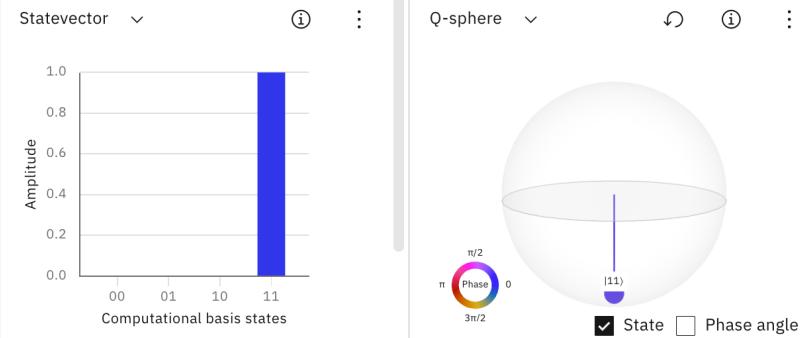
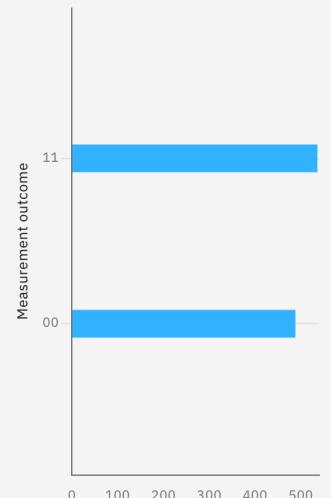


Qiskit

One in Qiskit

```
1 from qiskit import QuantumRegister, ClassicalRegister, QuantumCircuit
2 from numpy import pi
3
4 qreg_q = QuantumRegister(2, 'q')
5 creg_c = ClassicalRegister(2, 'c')
6 circuit = QuantumCircuit(qreg_q, creg_c)
7
8 circuit.h(qreg_q[0])
9 circuit.cx(qreg_q[0], qreg_q[1])
10 circuit.measure(qreg_q[0], creg_c[0])
11 circuit.measure(qreg_q[1], creg_c[1])
```

(9) 測定結果



Set up and run your circuit

# 実機の量子コンピューターで実験

X

Step 1

Choose a system or simulator

Search bar: Search by system or simulator name

ibmq\_quito (selected):  
System status: Online  
Total pending jobs: 7  
5 Qubits 16 QV 2.5K CLOPS

ibmq\_belem:  
System status: Online

(11)

Step 2

Choose your settings

Provider

ibm-q-education/ibm-3/kawasaki-camp

Shots \*

1024

Job limit: None

Optional

Name your job

(12)

Kawasaki-campを選択します

実デバイスを選択します。

- Total pending jobsが少ない

- QVが大きい

などから選んでみましょう。

Close

Run on ibmq\_belem

(13)

IBM Quantum Composer

Jobs / (14) Jobの確認

Untitled circuit Saved File Edit View Visualiz

Completed Jul 21, 2022 8:55 PM (in 1m 8s)

Backend ibmq\_qasm\_simulator

Status timeline Completed

See more details

Operations

Search

H  $\oplus$   $\ominus$   $\oplus$   $\ominus$   $\otimes$  I  
T S Z  $T^\dagger$   $S^\dagger$  P  
RZ  $r_z$   $|0\rangle$  i ● if

q[0] H  $r_z$   
q[1] +  $r_z$   
c2 0 1

Qiskit Open in Quantu

```
1 from qiskit import *
2 ClassicalRegister creg
3 from numpy import *
4 qreg_q = QuantumRegister(2)
5 creg_c = ClassicalRegister(2)
6 circuit =
```



IBM Quantum Composer

View all jobs → Composer jobs

Pending: Jul 21, 2022 9:21 PM ID: 62d944d1e59b9100e90919... | ibmq\_belem

Completed: Jul 21, 2022 8:59 PM ID: 62d93fbbaaf3a73b512b9ef4 | ibmq\_belem

Operations

Search

H  $\oplus$   $\ominus$   $\oplus$   $\ominus$   $\otimes$  I  
T S Z  $T^\dagger$   $S^\dagger$  P  
RZ  $r_z$   $|0\rangle$  i ● if

q[0] H  $r_z$   
q[1] +  $r_z$   
c2 0 1

Qiskit Open in Quantu

```
1 from qiskit import *
2 ClassicalRegister creg
3 from numpy import *
4 qreg_q = QuantumRegister(2)
```



IBM Quantum Composer



Jobs /

62d944d1e59b9100e90919d8

⋮

# 待ち時間がある場合

Backend

ibmq\_belem

See more details ↗

Pending ^

Status timeline

Created: Jul 21, 2022 9:21 PM

Transpiling: 904ms

Validating: 892ms

In queue

Running

Completed

Details

Untitled circuit Saved

File

Edit

View

Visual

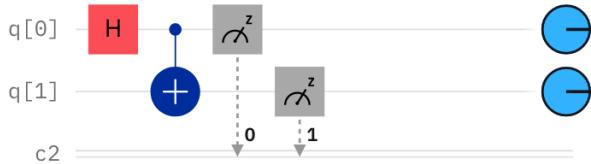
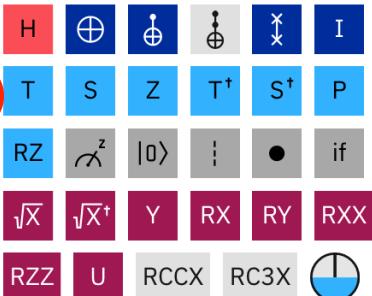
Operations

↶

↷

Left alignment

▼

Inspect Search  ⌂ ⌂

Qiskit ▾

Open in Quant

```
1 from qis
2 Classics
3 from nu
4 qreg_q :
5 creg_c :
6 circuit
7 circuit
8 circuit
9 circuit
10 circuit
11 circuit
```

[See more details](#)Untitled circuit Saved

Edit

View

Visualizations seed

3623

Setup and run

Completed

Jul 21, 2022 9:00:00 PM (in 38.2s)

Backend

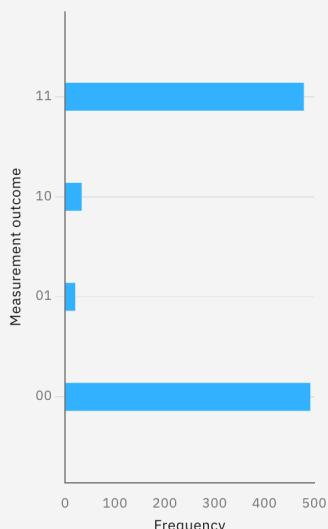
ibmq\_belem

Status timeline

Completed

Details

Result - histogram



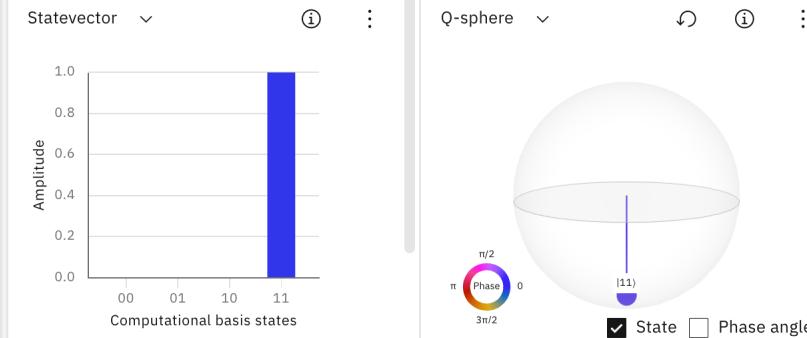
Operations

```
q[0] --H--+
      +---+
q[1] --+--+
      |   |
      0   1
c[2] ---|---
```

Search

H	$\oplus$	$\oplus$	$\otimes$	$\otimes$	$\otimes$	I
T	S	Z	$T^\dagger$	$S^\dagger$	P	
RZ	$\alpha_z$	$ 0\rangle$	$ 1\rangle$	if		
$\sqrt{X}$	$\sqrt{X}^\dagger$	Y	RX	RY	RXX	
RZZ	U	RCCX	RC3X			

## (16) 実機での結果



Qiskit

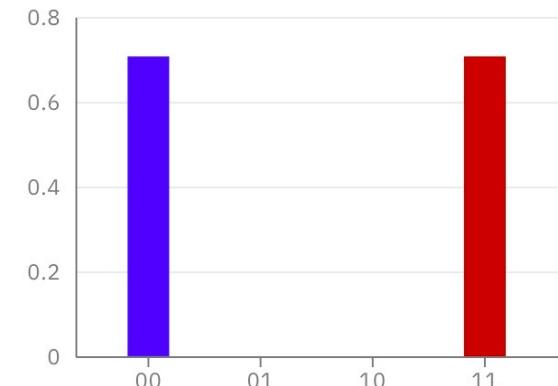
Open in Quantum Lab

```
1 from qiskit import QuantumRegister, ClassicalRegister, QuantumCircuit
2 from numpy import pi
3
4 qreg_q = QuantumRegister(2, 'q')
5 creg_c = ClassicalRegister(2, 'c')
6 circuit = QuantumCircuit(qreg_q, creg_c)
7
8 circuit.h(qreg_q[0])
9 circuit.cx(qreg_q[0], qreg_q[1])
10 circuit.measure(qreg_q[0], creg_c[0])
11 circuit.measure(qreg_q[1], creg_c[1])
```

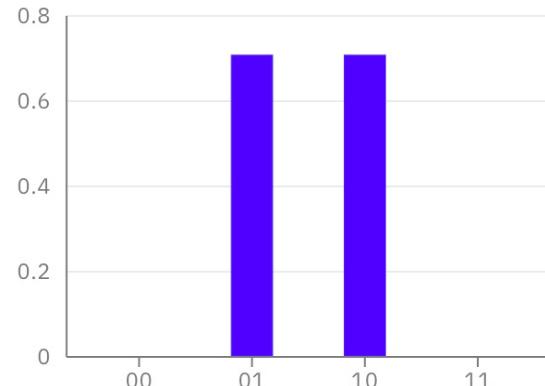
# 演習問題

2量子ビットのエンタングル状態を作ってみましょう。  
答えは一つではないので、どんな作り方でもOKです。

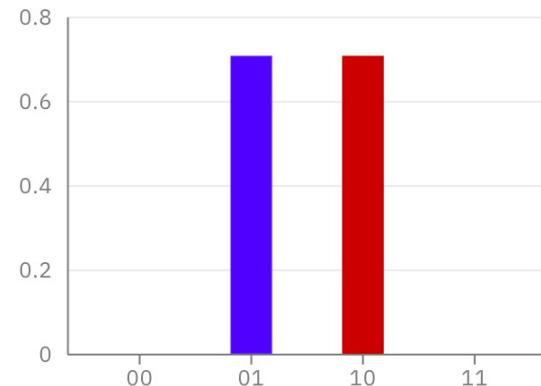
(1)  $\frac{1}{\sqrt{2}}(|00\rangle - |11\rangle)$



(2)  $\frac{1}{\sqrt{2}}(|01\rangle + |10\rangle)$



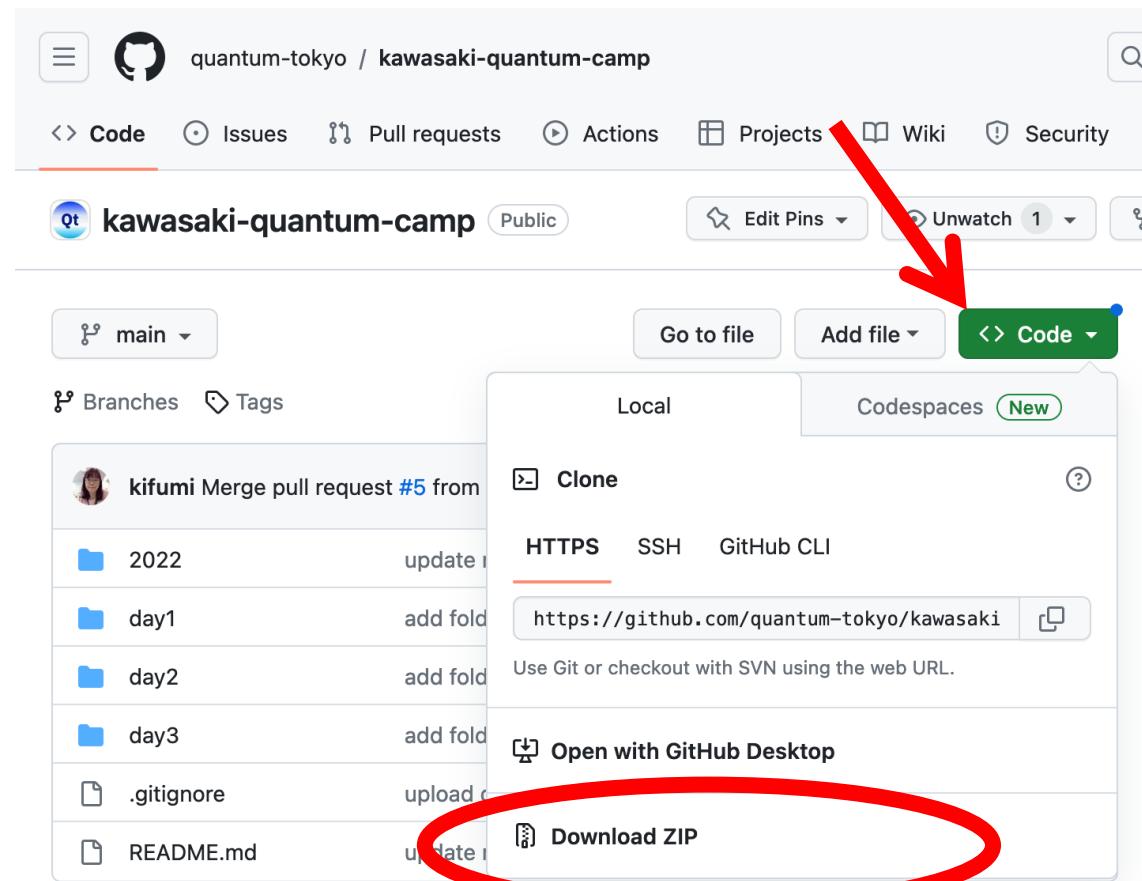
(3)  $\frac{1}{\sqrt{2}}(|01\rangle - |10\rangle)$



# ハンズオン ファイルのダウンロード

URL: [ibm.biz/kwskgit](https://ibm.biz/kwskgit)

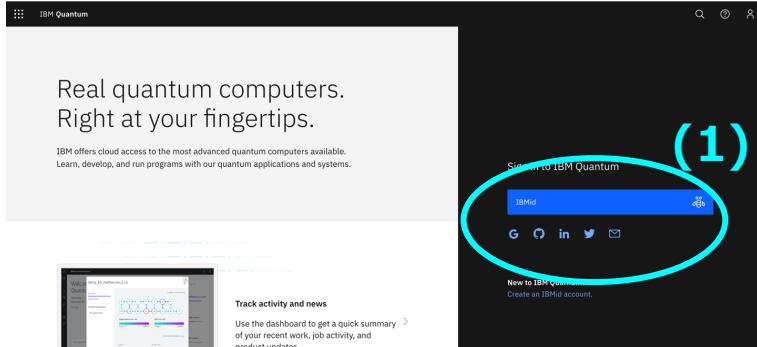
1. ハンズオンで使用するファイル(zipファイル)をダウンロードします。
2. ダウンロードしたzipファイルを解凍します。
3. 「day1」フォルダーの下のファイルを使います。



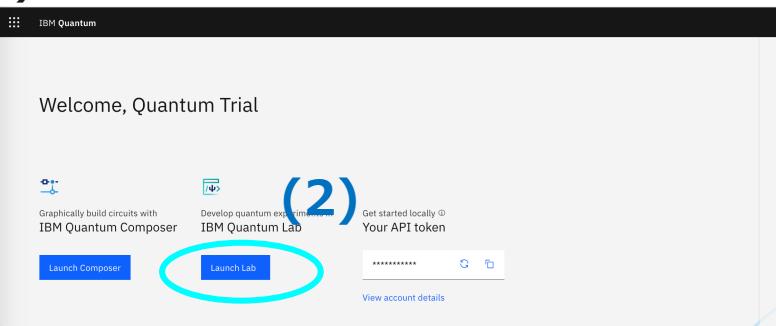
# IBM Quantum Labでの実行

(1) IBM Quantumにログインします。

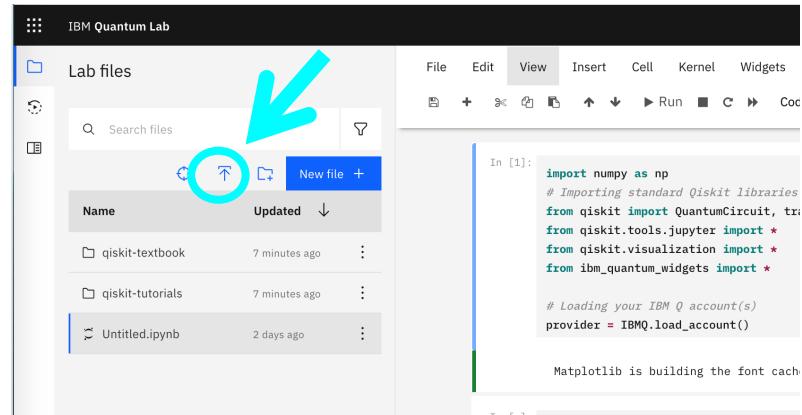
URL: <https://quantum-computing.ibm.com/>



(2) 青の右側「Launch Lab」をクリック。



(3) 左側 ↑ の「Upload file」から、ご自分のローカルに保存したハンズオンコンテンツ「20230817\_qiskit.ipynb」を探して、アップロードします。



(4) ファイル名「20230817\_qiskit.ipynb」をダブルクリックして開きます。

import numpy as np  
# Importing standard Qiskit libraries  
from qiskit import QuantumCircuit, transpile  
from qiskit.tools.jupyter import \*\nfrom qiskit.visualization import \*\nfrom ibm\_quantum\_widgets import \*\n\n# Loading your IBM Q account(s)\nprovider = IBMQ.load\_account()

Matplotlib is building the font cache

