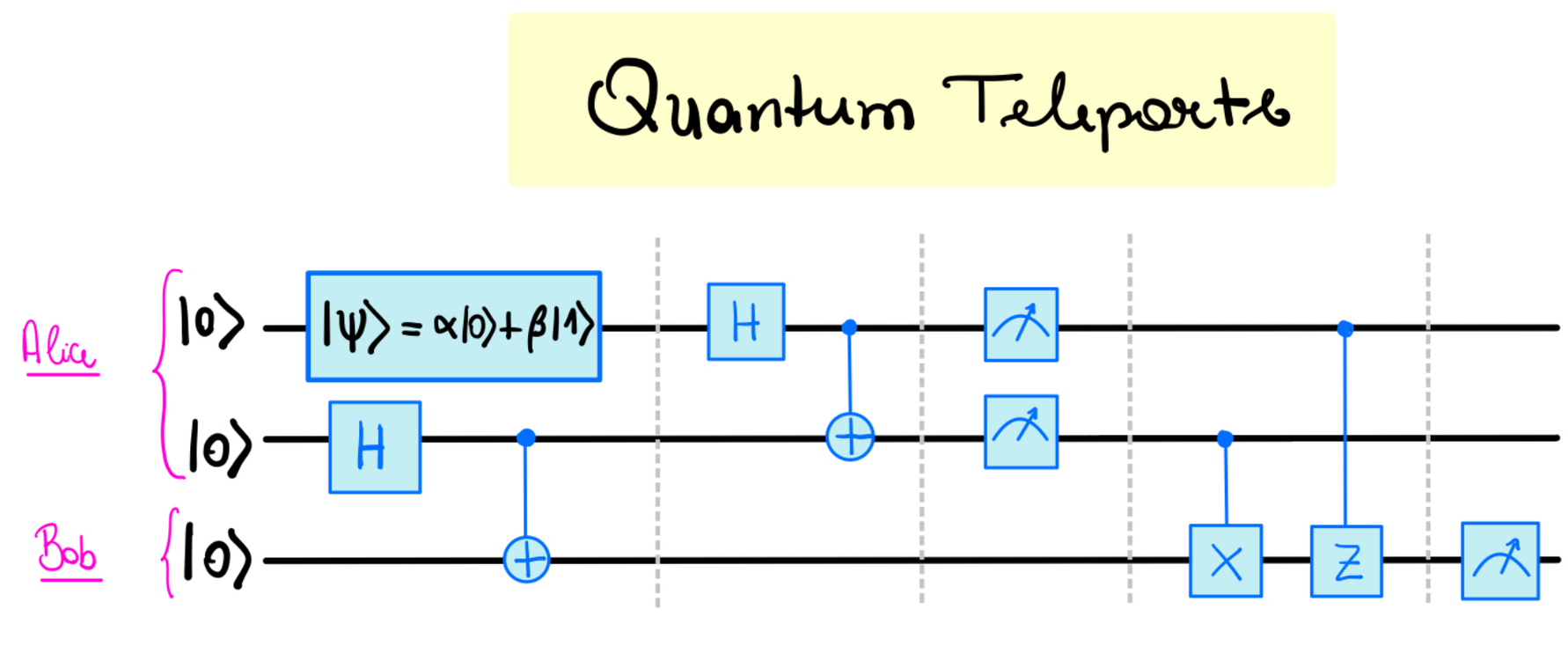
```
# !pip install --trusted-host pypi.org --trusted-host files.pythonhosted.org Ipython
```

```
from IPython.display import display, Latex
from IPython.display import Image
from qiskit import *
from qiskit.visualization import *
from qiskit.quantum_info import Statevector


import numpy as np
import warnings

# warnings.filterwarnings("ignore", category=DepriciationWarning)
warnings.filterwarnings("ignore", category=UserWarning)
```

Protocolo de teleporte quântico



Passos do protocolo:

1. Estado inicial do sistema:

$$|\phi_0\rangle = |00\rangle_{Alice} \otimes |0\rangle_{Bob}$$

1. Alice prepara o estado quântico $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$

$$|\psi\rangle = \begin{bmatrix} \alpha \\ \beta \end{bmatrix}$$

O estado é atualizado para:

$$|\phi_1\rangle = [\alpha|00\rangle + \beta|10\rangle]_{Alice} \otimes |0\rangle_{Bob}$$

1. Alice e Bob possuem um estado emaranhado entre eles:

$$|\psi_{AB}\rangle = \frac{1}{\sqrt{2}}[|00\rangle + |11\rangle]$$

O estado atualizado é:

$$|\phi_2\rangle = [\alpha|0\rangle + \beta|1\rangle] \otimes |\psi_{AB}\rangle$$

$$|\phi_2\rangle = \frac{1}{\sqrt{2}}[\alpha|0\rangle + \beta|1\rangle] \otimes [|00\rangle + |11\rangle]$$

$$|\phi_2\rangle = \frac{1}{\sqrt{2}}[\alpha|0\rangle \otimes (|00\rangle + |11\rangle) + \beta|1\rangle \otimes (|00\rangle + \beta|11\rangle)]$$

1. Aplicar Hadamar no primeiro qubit da Alice (q_0):

$$|\phi_3\rangle = \frac{1}{2}[\alpha(|0\rangle + |1\rangle) \otimes (|00\rangle + |11\rangle) + \beta(|0\rangle - |1\rangle) \otimes (|00\rangle + \beta|11\rangle)]$$

Ficamos com:

$$|\phi_3\rangle = \frac{1}{2}[\alpha|000\rangle + \alpha|011\rangle + \alpha|100\rangle + \alpha|111\rangle + \\ \beta|000\rangle + \beta|011\rangle - \beta|100\rangle - \beta|111\rangle]$$

Reorganizando temos:

$$|\phi_3\rangle = \frac{1}{2}[|00\rangle \otimes (\alpha|0\rangle + \beta|1\rangle) + \\ |01\rangle \otimes (\alpha|1\rangle + \beta|0\rangle) + \\ |10\rangle \otimes (\alpha|0\rangle - \beta|1\rangle) + \\ |11\rangle \otimes (\alpha|1\rangle - \beta|0\rangle)]$$

1. Alice faz uma medição em seus estados e pode obter como resultado os seguintes

$$|00\rangle, |01\rangle, |10\rangle \text{ ou } |11\rangle$$

O resultado da medida da Alice irá definir se Bob precisará ou não fazer uma correção para medir o estado.

00 $\rightarrow$ ele não faz nada

01 $\rightarrow$ aplica X gate

10 $\rightarrow$ aplica Z gate

11 $\rightarrow$ aplica ZX gate

Visualização de estados quânticos

```
qc = QuantumCircuit(3)
qc.ry (np.pi/4, 0) # optional, we want to transfer state 1 in this example
save = Statevector.from_instruction(qc)
qc.draw('mpl')
```


A atuação de R_y é dada por:

$$R_y(\theta)|0\rangle = e^{\frac{-i\pi}{2}}$$

O Estado deles acima é dado por:

$$|\psi\rangle = \left[\cos\frac{\pi}{8}|0\rangle + \sin\frac{\pi}{8}|1\rangle\right] \otimes |00\rangle$$

$$|\psi\rangle = \cos\pi/8|000\rangle + \sin\pi/8|100\rangle$$

Onde o ordenamento dos qubits é dado por:

$$|\psi\rangle = |q_0\rangle \otimes |q_1\rangle \otimes |q_2\rangle$$

$$|\psi\rangle = \begin{bmatrix} 000 \\ 100 \\ 010 \\ 110 \\ 001 \\ 101 \\ 011 \\ 111 \end{bmatrix} = \begin{bmatrix} 0.92388 \\ 0.38268 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

No estado acima o resultado da medição da Alice foi 00 E a amplitude é acima é dada por

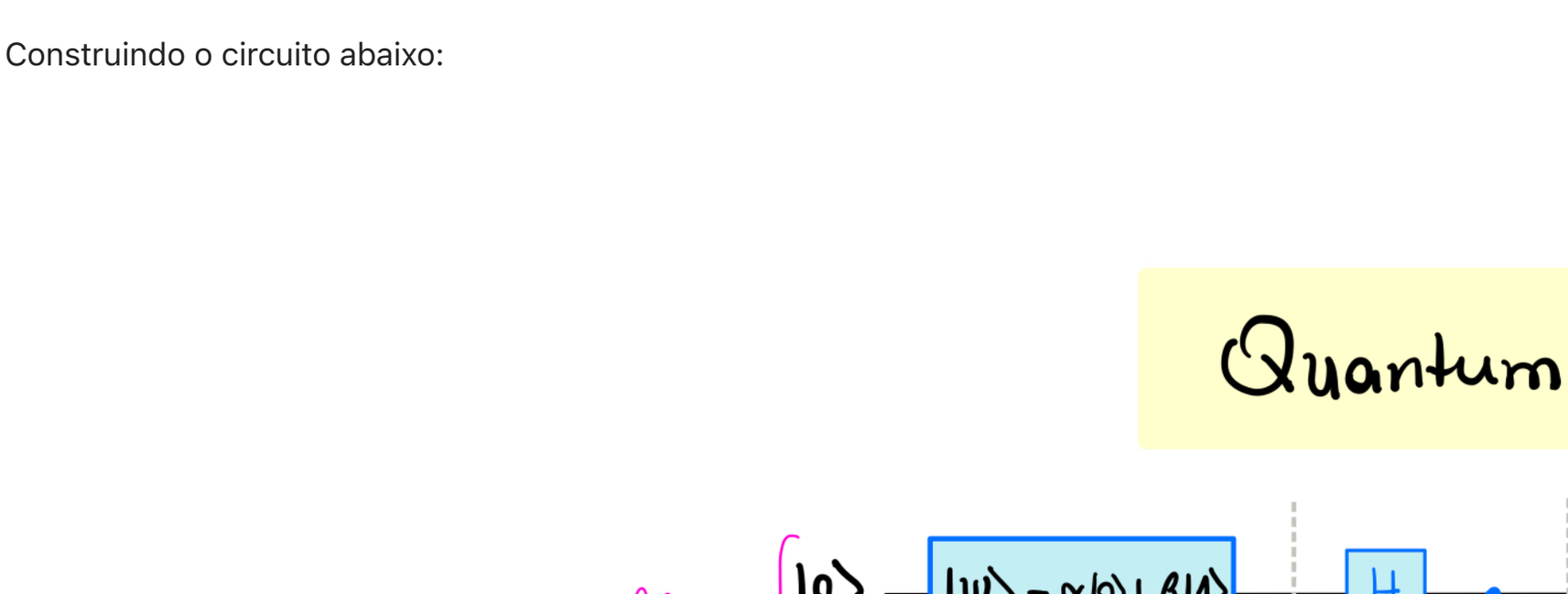
$$\alpha = \cos\frac{\pi}{8}$$

$$\beta = \sin\frac{\pi}{8}$$

Mostrando a posiçao de cada qubit na esfera de Bloch

```
simulator = Aer.get_backend('statevector_simulator')
result = execute(qc, backend= simulator, shots = 100000).result().get_statevector()

print(f'Amplitudes alpha e beta são: [{np.round(np.cos(np.pi/8)**2,5)} {np.round(np.sin(np.pi/8)**2,5)}]')


plot_bloch_multivector(result)
```

```
Amplitudes alpha e beta são: [0.85355 0.14645]
```



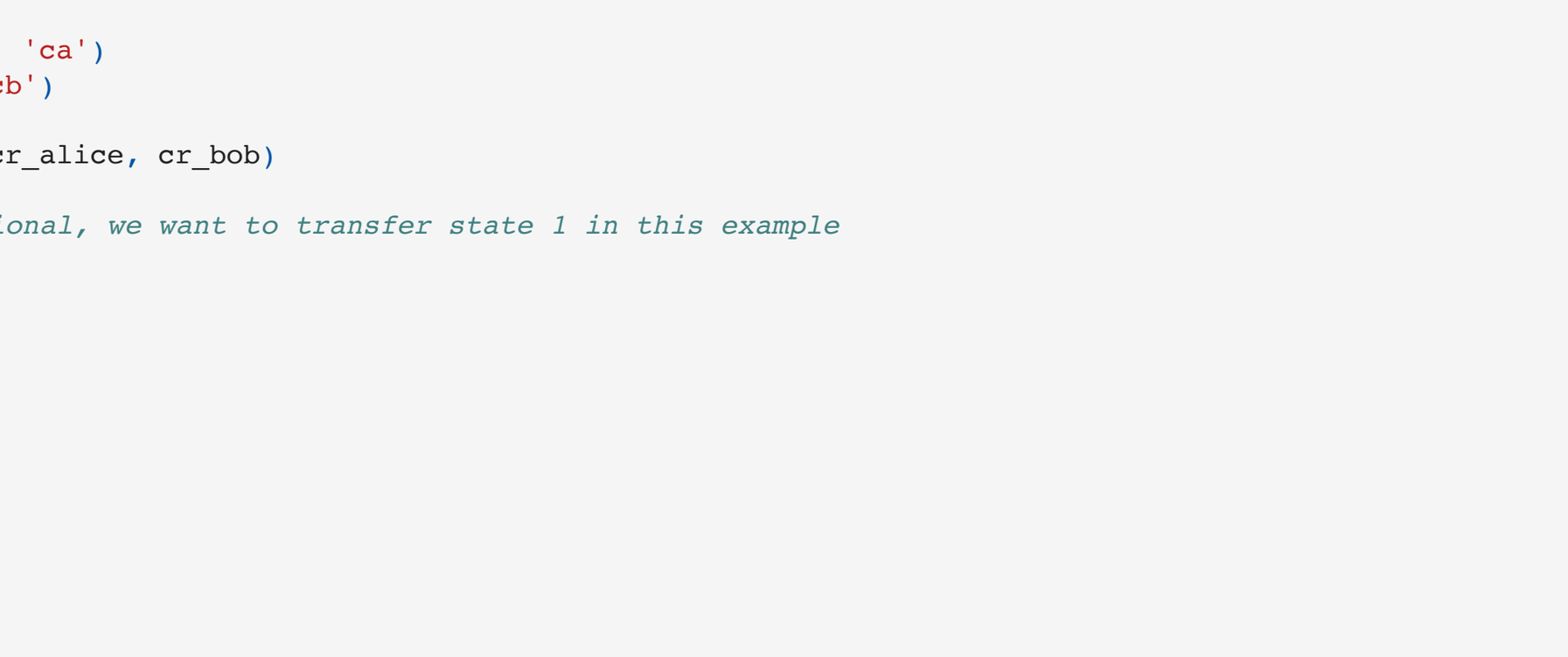
$$|q_0\rangle = \cos\frac{\pi}{8}|0\rangle + \sin\frac{\pi}{8}|1\rangle$$

$$|q_1\rangle = |0\rangle$$

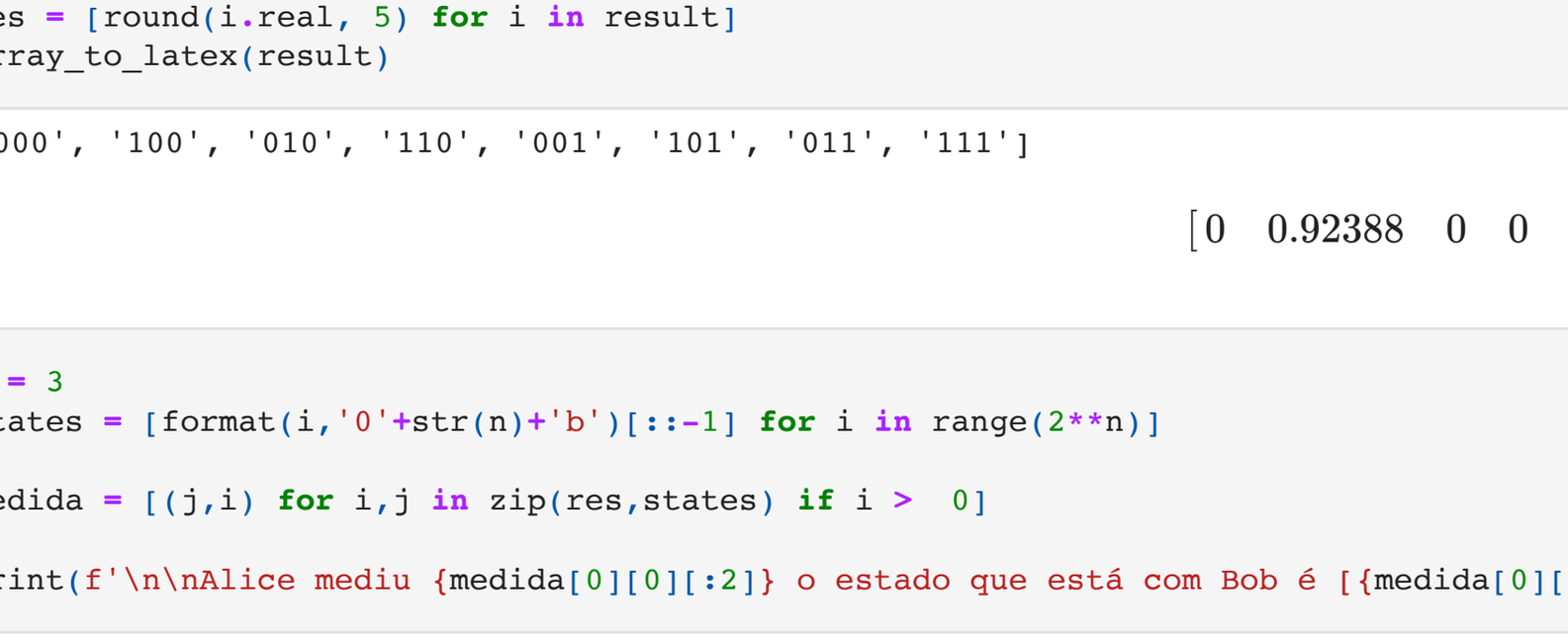
$$|q_2\rangle = |0\rangle$$

Protocolo

Construindo o circuito abaixo:




```
alice = QuantumRegister(2, 'a')
bob = QuantumRegister(1,'b')
cr_alice = ClassicalRegister(2, 'ca')
cr_bob = ClassicalRegister(1,'cb')

qc = QuantumCircuit(alice,bob,cr_alice, cr_bob)

qc.ry(np.pi/4, alice[0]) # optional, we want to transfer state 1 in this example
qc.barrier()

qc.h(alice[1])
qc.cx(alice[1],bob[0])
qc.barrier()


qc.cx(alice[0],alice[1])
qc.h(alice[0])

qc.barrier()

qc.measure(alice, cr_alice)
qc.barrier()

qc.cx(alice[1], bob[0])
qc.cz(alice[0], bob[0])
# qc.measure(bob, cr_bob)


qc.draw('mpl')
```



Resultado State Vector

```
simulator = Aer.get_backend('statevector_simulator')
result = execute(qc, backend= simulator, shots = 10000).result().get_statevector()
# plot_histogram(result)
states = [format(i,'0'+str(n)+'b')[::-1] for i in range(2**n)]

print(states)
res = [round(i.real, 5) for i in result]
array_to_latex(result)
```

```
['000', '100', '010', '110', '001', '101', '011', '111']
```

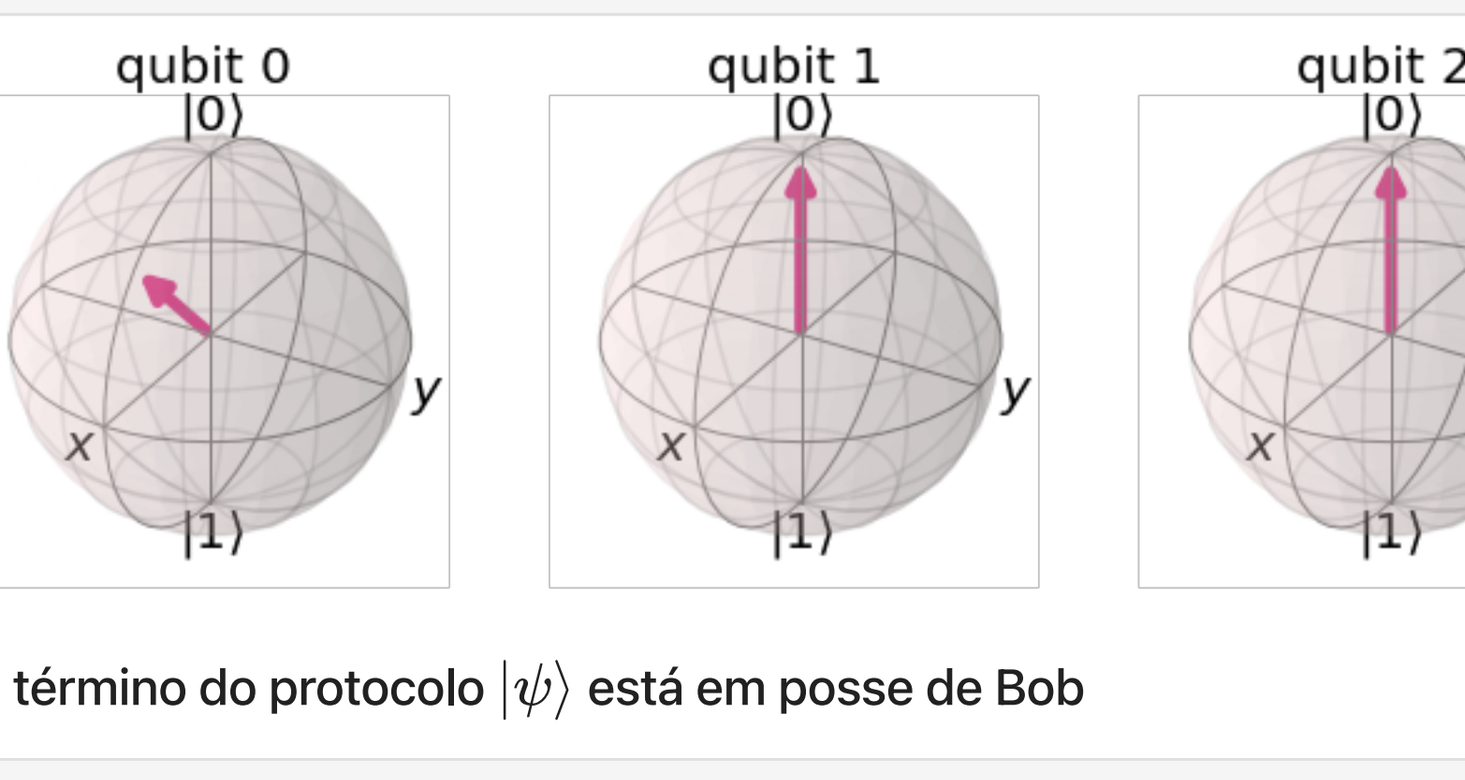
$$\begin{bmatrix} 0 & 0.92388 & 0 & 0 & 0 & 0.38268 & 0 & 0 \end{bmatrix}$$

```
n = 3
states = [format(i,'0'+str(n)+'b')[::-1] for i in range(2**n)]

medida = [(j,i) for i,j in zip(res,states) if i >  0]

print(f'\n\nAlice mediu {medida[0][0][:2]} o estado que está com Bob é [{medida[0][1]} {medida[1][1]}]\n\n')
```

```
Alice mediu 10 o estado que está com Bob é [0.92388 0.38268]
```

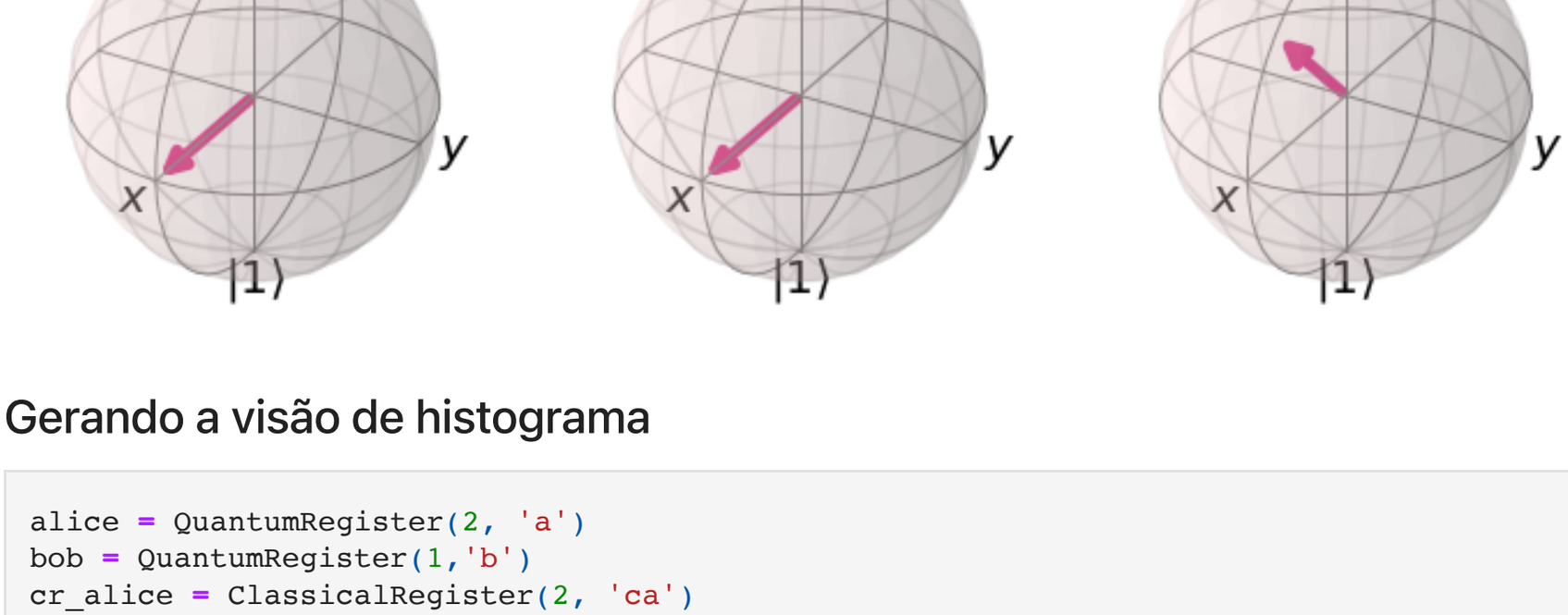
Plotando resultado na Bloch Sphere antes e depois do protocolo

```
alice = QuantumRegister(2, 'a')
bob = QuantumRegister(1,'b')
cr_alice = ClassicalRegister(2, 'ca')
cr_bob = ClassicalRegister(1,'cb')

qc = QuantumCircuit(alice,bob,cr_alice, cr_bob)

qc.ry(np.pi/4, alice[0]) # optional, we want to transfer state 1 in this example
qc.barrier()
save1 = Statevector.from_instruction(qc)


qc.h(alice[1])
qc.cx(alice[1],bob[0])
qc.barrier()


qc.cx(alice[0],alice[1])
qc.h(alice[0])

qc.barrier()


# qc.measure(alice, cr_alice)
qc.barrier()

qc.cx(alice[1], bob[0])
qc.cz(alice[0], bob[0])
save2 = Statevector.from_instruction(qc)


qc.measure(bob, cr_bob)


qc.draw('mpl')
```



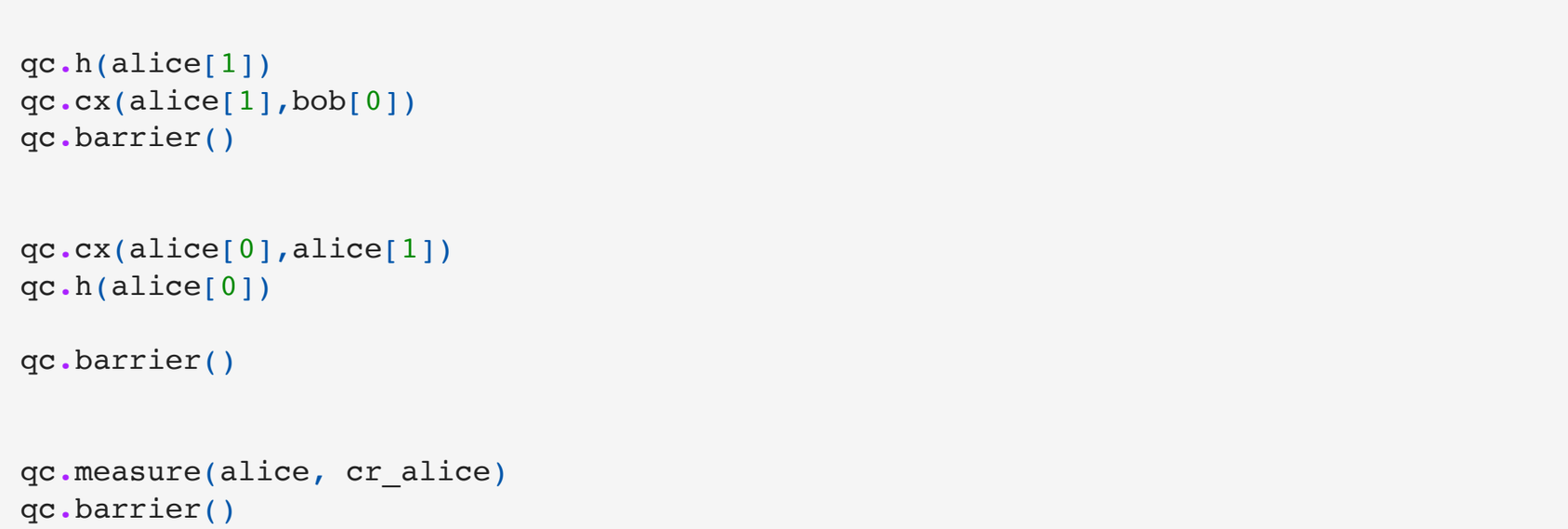
No início $|\psi\rangle$ está em posse da Alice

```
plot_bloch_multivector(save1)
```



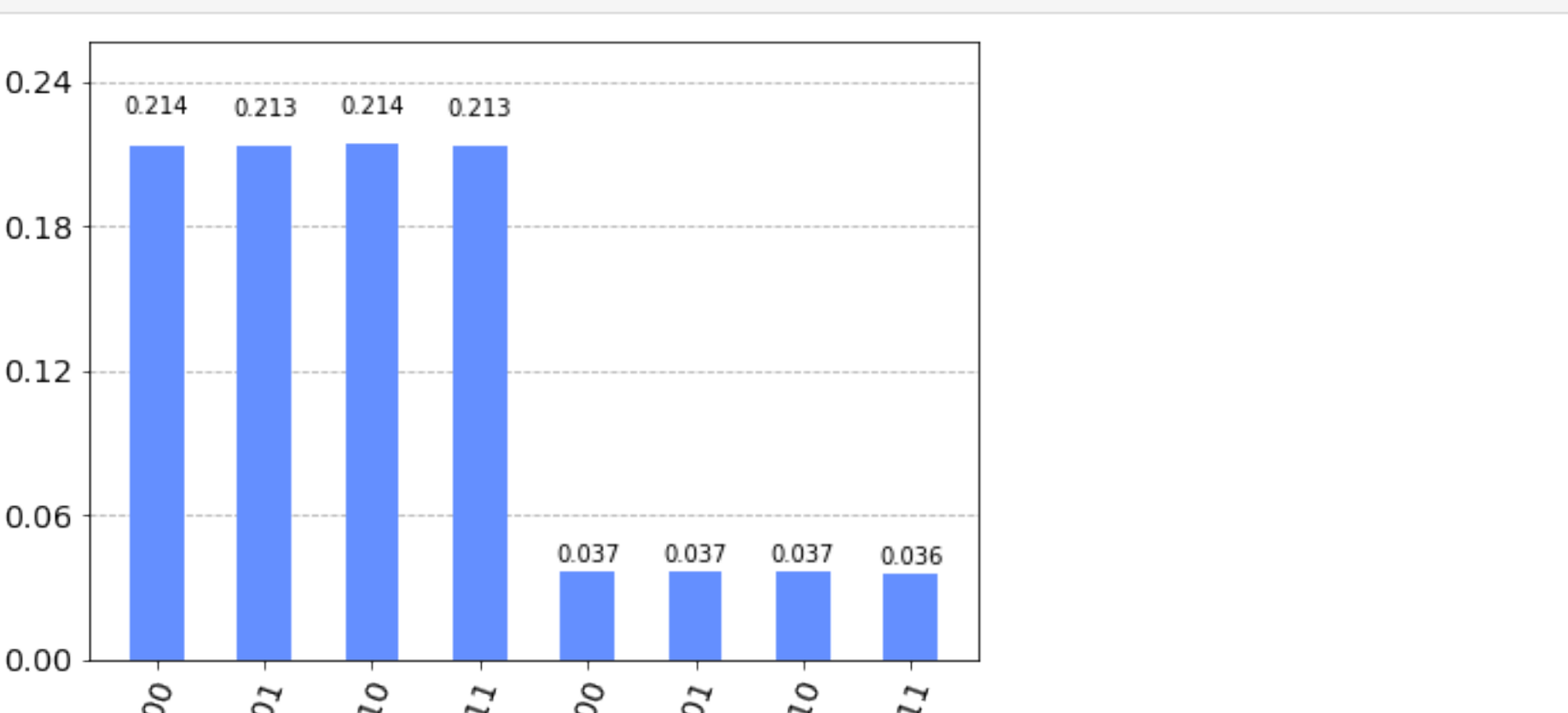
Ao término do protocolo $|\psi\rangle$ está em posse de Bob

```
plot_bloch_multivector(save2)
```

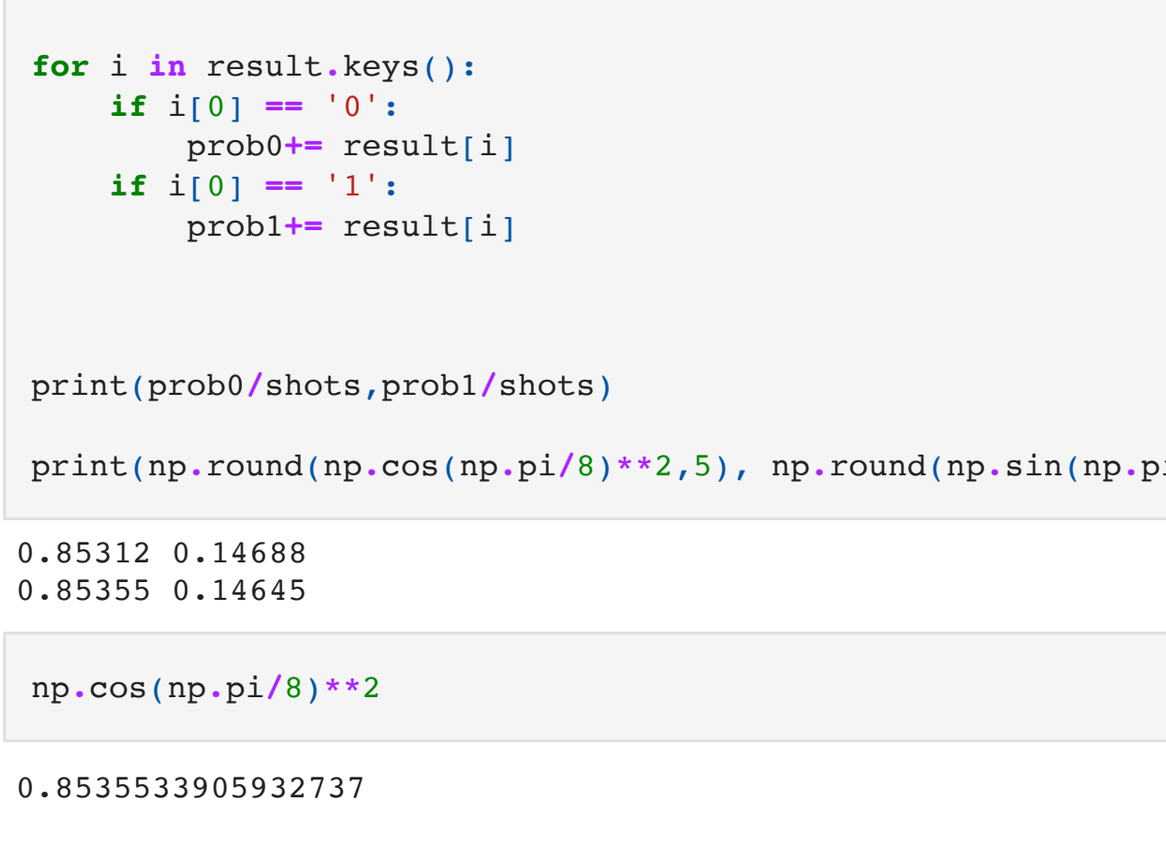


Gerando a visão de histograma

```
alice = QuantumRegister(2, 'a')
bob = QuantumRegister(1,'b')
cr_alice = ClassicalRegister(2, 'ca')
cr_bob = ClassicalRegister(1,'cb')

qc = QuantumCircuit(alice,bob,cr_alice, cr_bob)

qc.ry(np.pi/4, alice[0]) # optional, we want to transfer state 1 in this example
qc.barrier()


qc.h(alice[1])
qc.cx(alice[1],bob[0])
qc.barrier()


qc.cx(alice[0],alice[1])
qc.h(alice[0])

qc.barrier()


qc.measure(alice, cr_alice)
qc.barrier()

qc.cx(alice[1], bob[0])
qc.cz(alice[0], bob[0])


qc.measure(bob, cr_bob)


qc.draw('mpl')
```




```
simulator = Aer.get_backend('qasm_simulator')
shots = 100000
result = execute(qc, backend=simulator, shots=shots).result().get_counts()

from qiskit.visualization import plot_histogram
plot_histogram(result)
```



Pós processamento de informação

```
prob0 = 0
prob1 = 0



for i in result.keys():
    if i[0] == '0':
        prob0+= result[i]
    if i[0] == '1':
        prob1+= result[i]



print(prob0/shots,prob1/shots)

print(np.round(np.cos(np.pi/8)**2,5), np.round(np.sin(np.pi/8)**2,5))
```

```
0.85312 0.14688
0.85355 0.14645
```

```
np.cos(np.pi/8)**2
```

```
0.8535533905932737
```

```
simulator = Aer.get_backend('statevector_simulator')
result = execute(qc, backend= simulator, shots = 10000).result().get_statevector()
# plot_histogram(result)

array_to_latex(result, prefix= '\\vert \psi \\rangle = ')
```

$$|\psi\rangle = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}$$