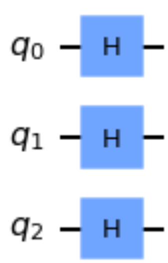
In [2]:
```
import numpy as np
import math
# Importing standard Qiskit libraries
from qiskit import QuantumCircuit, transpile, Aer, IBMQ, assemble
from qiskit.tools.jupyter import *
from qiskit.visualization import *
from ibm_quantum_widgets import *
from math import pi
# Loading your IBM Quantum account(s)
provider = IBMQ.load_account()
```

```
ibmqfactory.load_account:WARNING:2021-07-10 11:47:42,489: Credentials
are already in use. The existing account in the session will be repla
ced.
```

In [3]:
```
qc = QuantumCircuit(3)
# Apply H-gate to each qubit:
for qubit in range(3):
    qc.h(qubit)
# See the circuit:
qc.draw()
```

Out[3]:



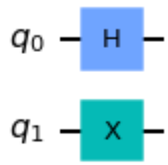
In [4]:
```
# Let's see the result
svsim = Aer.get_backend('aer_simulator')
qc.save_statevector()
qobj = assemble(qc)
final_state = svsim.run(qobj).result().get_statevector()

# In Jupyter Notebooks we can display this nicely using Latex.
# If not using Jupyter Notebooks you may need to remove the
# array_to_latex function and use print(final_state) instead.
from qiskit.visualization import array_to_latex
array_to_latex(final_state, prefix="\\text{Statevector} = ")
```

Out[4]:

$$\text{Statevector} = \begin{bmatrix} \frac{1}{\sqrt{8}} & \frac{1}{\sqrt{8}} & \frac{1}{\sqrt{8}} & \frac{1}{\sqrt{8}} & \frac{1}{\sqrt{8}} & \frac{1}{\sqrt{8}} & \frac{1}{\sqrt{8}} & \frac{1}{\sqrt{8}} \end{bmatrix}$$

In [5]:
```
qc = QuantumCircuit(2)
qc.h(0)
qc.x(1)
qc.draw()
```

Out[5]:



In [6]:
```
usim = Aer.get_backend('aer_simulator')
qc.save_unitary()
qobj = assemble(qc)
unitary = usim.run(qobj).result().get_unitary()
```

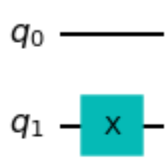
In [7]:
```
# In Jupyter Notebooks we can display this nicely using Latex.
# If not using Jupyter Notebooks you may need to remove the
# array_to_latex function and use print(unitary) instead.
from qiskit.visualization import array_to_latex
array_to_latex(unitary, prefix="\\text{Circuit = }\n")
```

Out[7]:

$$\text{Circuit} = \begin{bmatrix} 0 & 0 & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ 0 & 0 & \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 0 & 0 \\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} & 0 & 0 \end{bmatrix}$$

In [8]:
```
qc = QuantumCircuit(2)
qc.x(1)
qc.draw()
```

Out[8]:



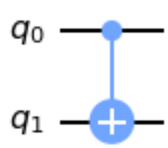
In [9]:
```
# Simulate the unitary
usim = Aer.get_backend('aer_simulator')
qc.save_unitary()
qobj = assemble(qc)
unitary = usim.run(qobj).result().get_unitary()
# Display the results:
array_to_latex(unitary, prefix="\\text{Circuit = } ")
```

Out[9]:

$$\text{Circuit} = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}$$

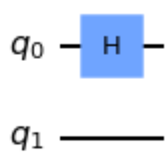
In [10]:
```
qc = QuantumCircuit(2)
# Apply CNOT
qc.cx(0,1)
# See the circuit:
qc.draw()
```

Out[10]:



In [11]:
```
qc = QuantumCircuit(2)
# Apply H-gate to the first:
qc.h(0)
qc.draw()
```

Out[11]:



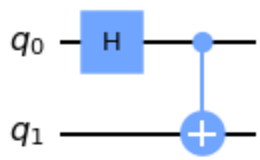
In [12]:
```
# Let's see the result:
svsim = Aer.get_backend('aer_simulator')
qc.save_statevector()
qobj = assemble(qc)
final_state = svsim.run(qobj).result().get_statevector()
# Print the statevector neatly:
array_to_latex(final_state, prefix="\\text{Statevector = }")
```

Out[12]:

$$\text{Statevector} = \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 0 & 0 \end{bmatrix}$$

In [13]:
```
qc = QuantumCircuit(2)
# Apply H-gate to the first:
qc.h(0)
# Apply a CNOT:
qc.cx(0,1)
qc.draw()
```

Out[13]:



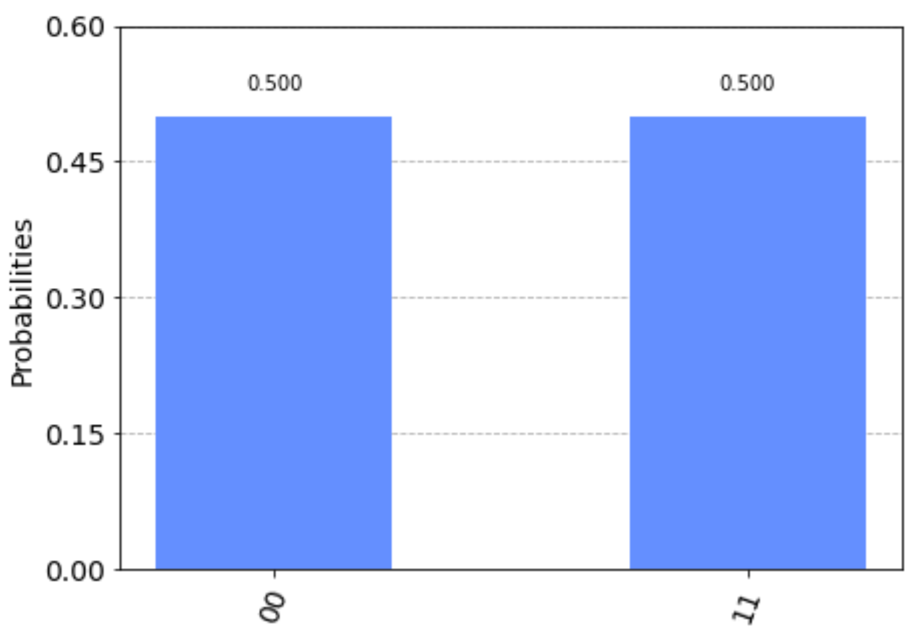
In [14]:
```
# Let's get the result:
# 2 qubits have values, 00, 01, 10, 11, these have been subjected to ab
ove gates
qc.save_statevector()
qobj = assemble(qc)
result = svsim.run(qobj).result()
# Print the statevector neatly:
final_state = result.get_statevector()
array_to_latex(final_state, prefix="\\text{Statevector = }")
#00 and 11 have only values, as see in statevector output
```

Out[14]:

$$\text{Statevector} = \begin{bmatrix} \frac{1}{\sqrt{2}} & 0 & 0 & \frac{1}{\sqrt{2}} \end{bmatrix}$$

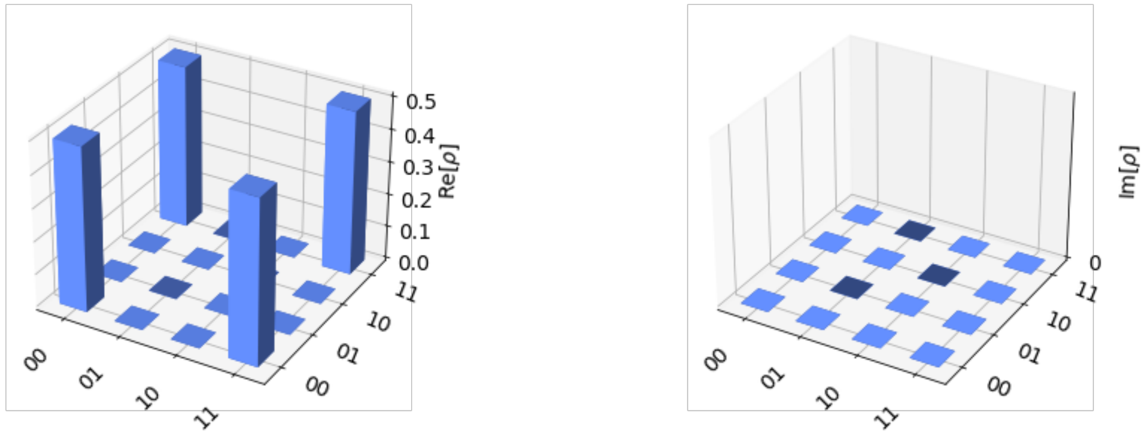
In [15]:
```
plot_histogram(result.get_counts())
```

Out[15]:



In [16]:
```
# Plot the Real and Imaginary parts of the statevector of the 2 qubits,
in this case, img part is zero so no pillars in second plot
from qiskit.visualization import plot_state_city
plot_state_city(final_state)
```

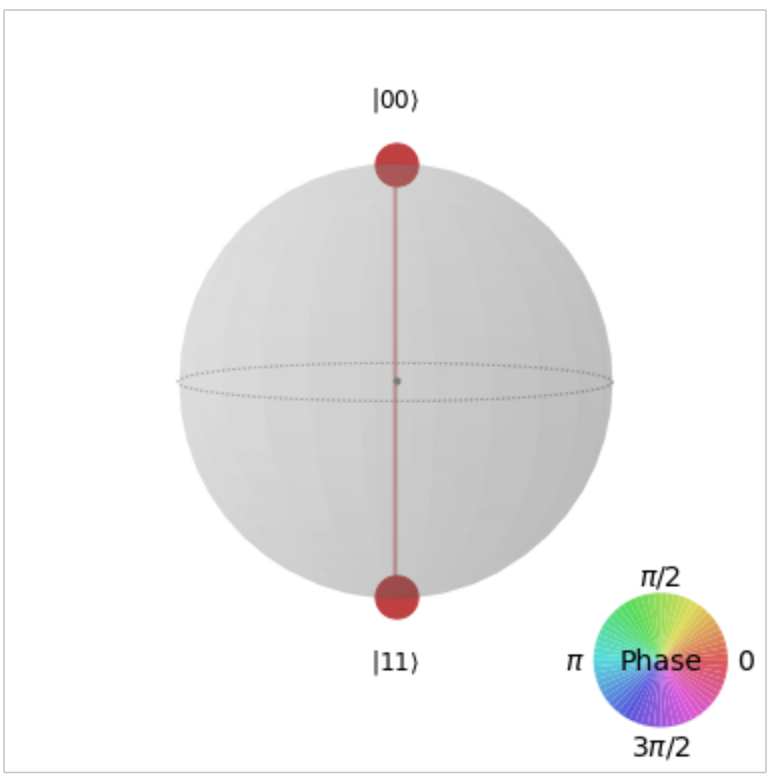
Out[16]:



In [17]:
```
from qiskit.visualization import plot_state_qsphere
plot_state_qsphere(final_state)
```

```
/opt/conda/lib/python3.8/site-packages/qiskit/visualization/state_vis
ualization.py:705: MatplotlibDeprecationWarning:
The M attribute was deprecated in Matplotlib 3.4 and will be removed
two minor releases later. Use self.axes.M instead.
  xs, ys, _ = proj3d.proj_transform(xs3d, ys3d, zs3d, renderer.M)
```

Out[17]:



In [ ]:
```
# Executed by Bhadale IT, in IBM Quantum Lab for multi-bit state repres
entation
#superposition state representation using amplitude
```