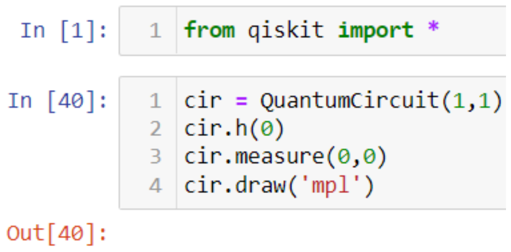
Assignment - 1 (CS F316)

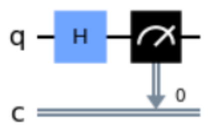
Name: **Abhay Kamble(2019A7PS0128G)**

1. Hadamard Gate (H Gate)

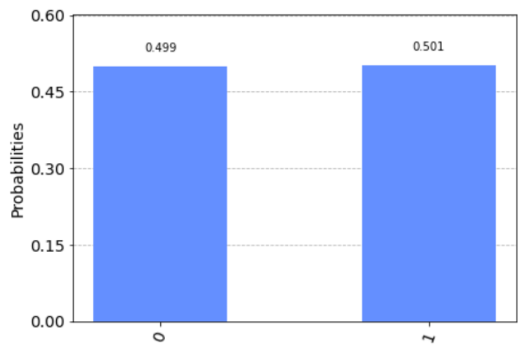
The H Gate is a single-qubit gate. It is a pi rotation about the X+Z axis, and is used to change the basis between { |0⟩, |1⟩ } and { |+⟩, |-⟩ }. This gate is used to create superposition of two quantum states, |0⟩ and |1⟩ with equal probability of each state, and is the most widely used gate. Here, the H-gate is implemented as follows, where the initial state is |0⟩, followed by H-gate and then by a measurement of the circuit and storing the measured value in the Classical Register.



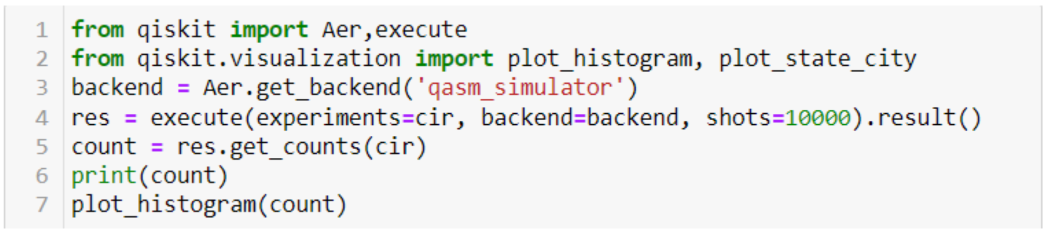
```
In [1]:  from qiskit import *

In [40]: cir = QuantumCircuit(1,1)
         cir.h(0)
         cir.measure(0,0)
         cir.draw('mpl')

Out[40]:
```



```
{'1': 5012, '0': 4988}
```

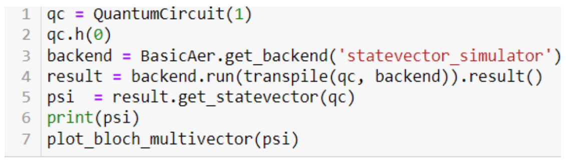


The above graph shows that if the circuit is executed 10,000 times then - around 50% of the time the outcome would be either 1 or 0. As we can see from the graph, the probability of occurrence of 0 or 1 is near 0.5 (zero - 0.499, one - 0.501), telling us that H-gate creates superposition. The below code is used to simulate the circuit(i.e. Backend definition code):



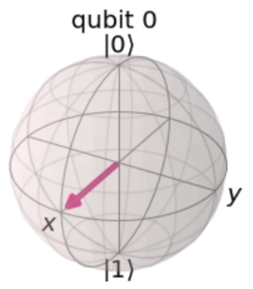
```
from qiskit import Aer,execute
from qiskit.visualization import plot_histogram, plot_state_city
backend = Aer.get_backend('qasm_simulator')
res = execute(experiments=cir, backend=backend, shots=10000).result()
count = res.get_counts(cir)
print(count)
plot_histogram(count)
```

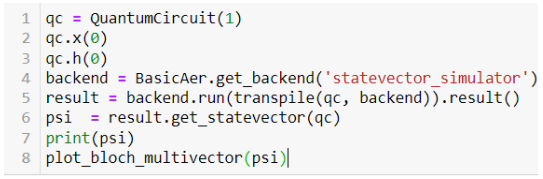
To find the state vector of the qubit we can do as follows, which shows us that the state is |+⟩ when initial state is |0⟩ and is |-⟩ when the initial state is |1⟩, showing that superposition occurs.



```
qc = QuantumCircuit(1)
qc.h(0)
backend = BasicAer.get_backend('statevector_simulator')
result = backend.run(transpile(qc, backend)).result()
psi  = result.get_statevector(qc)
print(psi)
plot_bloch_multivector(psi)
```

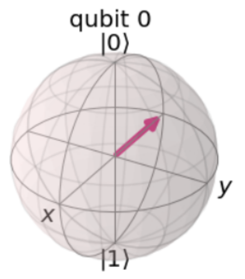
```
[0.70710678+0.j 0.70710678+0.j]
```





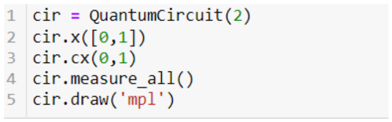
```
qc = QuantumCircuit(1)
qc.x(0)
qc.h(0)
backend = BasicAer.get_backend('statevector_simulator')
result = backend.run(transpile(qc, backend)).result()
psi  = result.get_statevector(qc)
print(psi)
plot_bloch_multivector(psi)
```

```
[ 0.70710678+0.00000000e+00j -0.70710678-8.65956056e-17j]
```

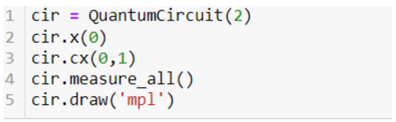


2. **Controlled NOT Gate (CNOT Gate)**

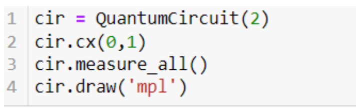
The CNOT gate is a two-qubit gate.In the computational basis, this gate flips the target qubit if the control qubit is in the $|1\rangle$ state, and is similar to the classical XOR Gate. This is also one of the most used gates as this gate is used to create entanglement. The following circuit is made using the CNOT gate ,and the various outputs from this circuit are as follows -
(all the figures and outputs are in the below order of inputs i.e. 11,10,00,01)



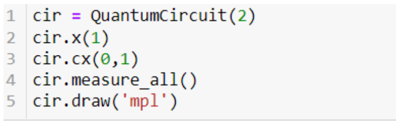
```
cir = QuantumCircuit(2)
cir.x([0,1])
cir.cx(0,1)
cir.measure_all()
cir.draw('mpl')
```



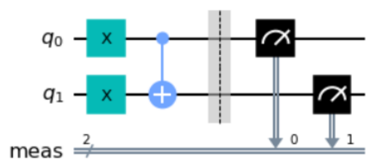
```
cir = QuantumCircuit(2)
cir.x(0)
cir.cx(0,1)
cir.measure_all()
cir.draw('mpl')
```



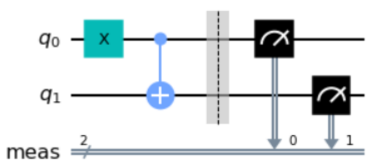
```
cir = QuantumCircuit(2)
cir.cx(0,1)
cir.measure_all()
cir.draw('mpl')
```



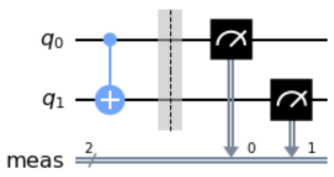
```
cir = QuantumCircuit(2)
cir.x(1)
cir.cx(0,1)
cir.measure_all()
cir.draw('mpl')
```



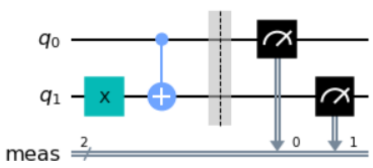
$|11\rangle$ as input



$|10\rangle$ as input

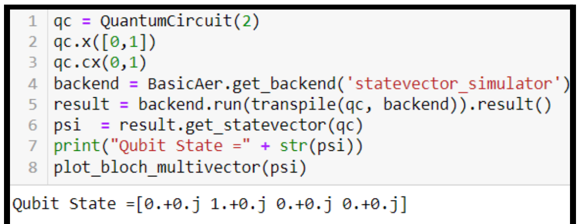


$|00\rangle$ as input



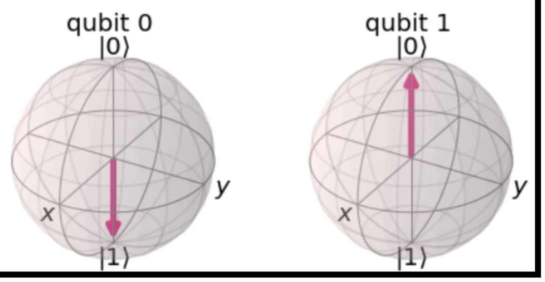
$|01\rangle$ as input

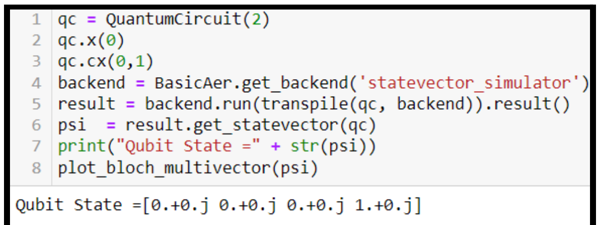
The various inputs can be $|11\rangle$,$|10\rangle$,$|00\rangle$,$|01\rangle$ & the respective state vector and bloch sphere outputs are::



```
qc = QuantumCircuit(2)
qc.x([0,1])
qc.cx(0,1)
backend = BasicAer.get_backend('statevector_simulator')
result = backend.run(transpile(qc, backend)).result()
psi  = result.get_statevector(qc)
print("Qubit State =" + str(psi))
plot_bloch_multivector(psi)
```

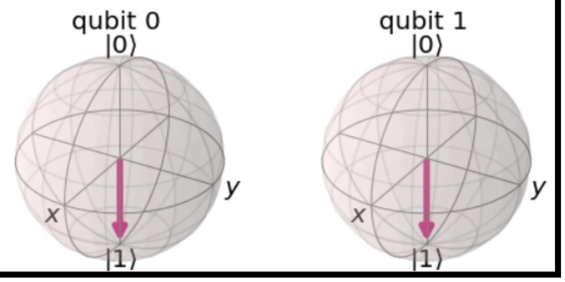
```
Qubit State =[0.+0.j 1.+0.j 0.+0.j 0.+0.j]
```

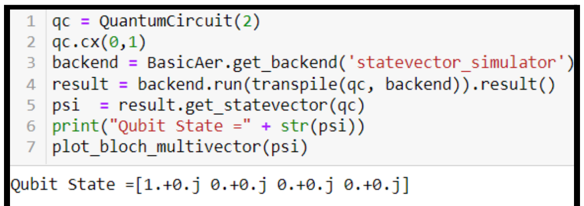




```
qc = QuantumCircuit(2)
qc.x(0)
qc.cx(0,1)
backend = BasicAer.get_backend('statevector_simulator')
result = backend.run(transpile(qc, backend)).result()
psi  = result.get_statevector(qc)
print("Qubit State =" + str(psi))
plot_bloch_multivector(psi)
```

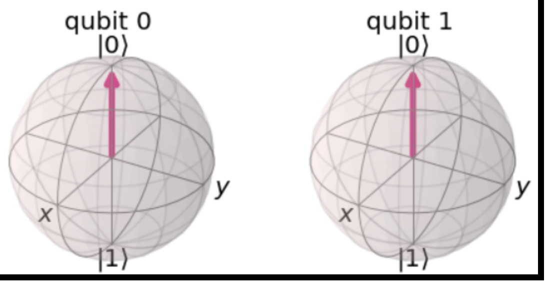
```
Qubit State =[0.+0.j 0.+0.j 0.+0.j 1.+0.j]
```

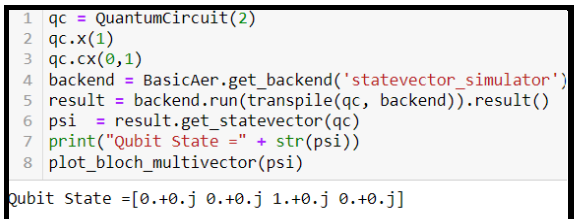




```
qc = QuantumCircuit(2)
qc.cx(0,1)
backend = BasicAer.get_backend('statevector_simulator')
result = backend.run(transpile(qc, backend)).result()
psi  = result.get_statevector(qc)
print("Qubit State =" + str(psi))
plot_bloch_multivector(psi)
```

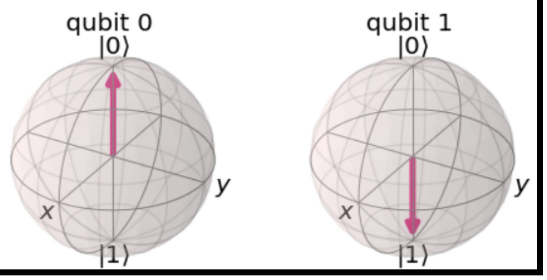
```
Qubit State =[1.+0.j 0.+0.j 0.+0.j 0.+0.j]
```





```
qc = QuantumCircuit(2)
qc.x(1)
qc.cx(0,1)
backend = BasicAer.get_backend('statevector_simulator')
result = backend.run(transpile(qc, backend)).result()
psi  = result.get_statevector(qc)
print("Qubit State =" + str(psi))
plot_bloch_multivector(psi)
```

```
Qubit State =[0.+0.j 0.+0.j 1.+0.j 0.+0.j]
```



The CNOT gate was applied and it was seen how the control qubit affects the target qubit, and results in the qubits getting entangled. The probability value is ‘1’, for each output state..