Name: Gudur Krishna Chaitanya Roll no: 102117049 email used:
gchaitanya_be21@thapar.edu

```
from qiskit_ibm_provider import IBMProvider

provider = IBMProvider(token='f55702335547d565b44eb80fd6708f3b82b4d0147236062

active_account = provider.active_account()

print("Active Account Details:")

print(active_account)
```

```
Active Account Details:
{'channel': 'ibm_quantum', 'token': 'f55702335547d565b44eb80fd6708f3b82b4d0
```

```
Start coding or generate with AI.
```

Quantum Teleportation:

1.Entanglement: Quantum teleportation relies on the phenomenon of entanglement, where two or more quantum particles become correlated in such a way that the state of one particle instantly influences the state of another, regardless of the distance between them.

2.Initialization: The process begins with the preparation of an entangled pair of qubits, often referred to as a Bell pair. This pair of qubits is shared between the sender (Alice) and the receiver (Bob).

3.State Preparation: Alice wants to teleport an arbitrary quantum state to Bob. She prepares the state she wishes to teleport by applying operations to a qubit in her possession.

4.Measurement and Communication: Alice then performs measurements on the qubits she possesses, including the qubit containing the unknown state and her half of the entangled pair. She obtains classical bits as a result of these measurements.

5.Classical Communication: Alice sends the classical bits resulting from her measurements to Bob through classical communication channels.

6.Conditional Operations: Upon receiving the classical bits from Alice, Bob performs conditional quantum operations on his half of the entangled pair based on the classical information he received.

7.State Reconstruction: After applying the appropriate quantum operations, Bob's qubit assumes the state originally possessed by Alice's qubit. However, the original qubit held by Alice is destroyed in the process due to the no-cloning theorem, meaning the state has been faithfully transferred rather than copied.

Now let us implement the code:

```
import numpy as np
from qiskit import QuantumCircuit, QuantumRegister, ClassicalRegister
from qiskit import transpile
from qiskit.visualization import plot_histogram, plot_bloch_multivector, arr
from qiskit.result import marginal_counts
from qiskit.quantum_info import random_statevector
import qiskit_aer
```

```
qr = QuantumRegister(2, name="Alice")
qr1=QuantumRegister(1, name="Bob")
crz = ClassicalRegister(1, name="c0")
crx = ClassicalRegister(1, name="c1")
teleportation_circuit = QuantumCircuit(qr,qr1, crz, crx)
```

```
def create_bell_pair(qc, a, b):
    qc.h(a)
    qc.cx(a,b)
```

```
qr = QuantumRegister(2, name="Alice")
qr1=QuantumRegister(1, name="Bob")
crz, crx = ClassicalRegister(1, name="c0"), ClassicalRegister(1, name="c1")
teleportation_circuit = QuantumCircuit(qr,qr1, crz, crx)
create_bell_pair(teleportation_circuit, 1, 2)
teleportation_circuit.draw(output='text',style='bw')
```

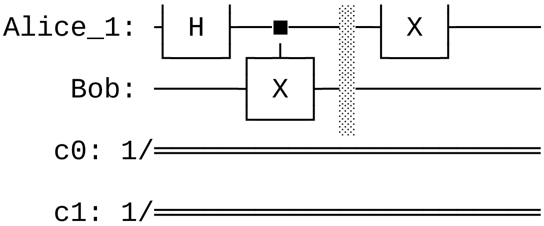
```
Alice_0: ────────────
         ┌───┐
Alice_1: ┤ H ├──■───
         └───┘┌─┴─┐
    Bob: ─────┤ X ├
              └───┘
c0: 1/═════════════

c1: 1/═════════════
```

```
def alice_gates(qc, psi, a):
    qc.cx(psi, a)
    qc.h(psi)
```

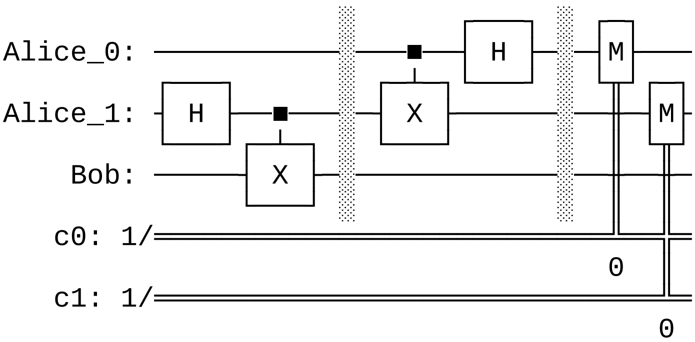
```
qr = QuantumRegister(2, name="Alice")
qr1=QuantumRegister(1, name="Bob")
crz, crx = ClassicalRegister(1, name="c0"), ClassicalRegister(1, name="c1")
teleportation_circuit = QuantumCircuit(qr,qr1,crz, crx)
create_bell_pair(teleportation_circuit, 1, 2)
teleportation_circuit.barrier()
alice_gates(teleportation_circuit, 0, 1)
teleportation_circuit.draw(output='text',style='bw')
```

```
                      ░      ┌───┐
Alice_0: ─────────────░───■──┤ H ├
         ┌───┐        ░ ┌─┴─┐└───┘
```



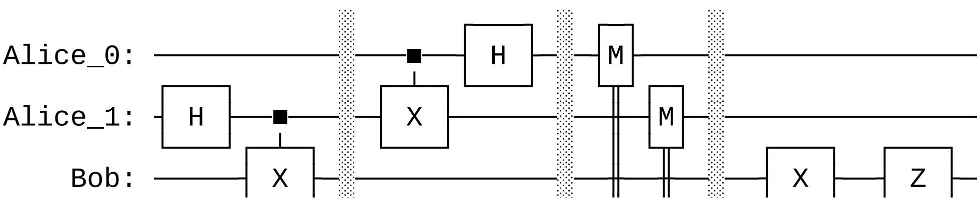

```
def measure_and_send(qc, a, b):
    qc.barrier()
    qc.measure(a,0)
    qc.measure(b,1)
```

```
qr = QuantumRegister(2, name="Alice")
qr1=QuantumRegister(1, name="Bob")
crz, crx = ClassicalRegister(1, name="c0"), ClassicalRegister(1, name="c1")
teleportation_circuit = QuantumCircuit(qr,qr1,crz, crx)
create_bell_pair(teleportation_circuit, 1, 2)
teleportation_circuit.barrier()
alice_gates(teleportation_circuit, 0, 1)
measure_and_send(teleportation_circuit, 0 ,1)
teleportation_circuit.draw(output='text',style='bw')
```



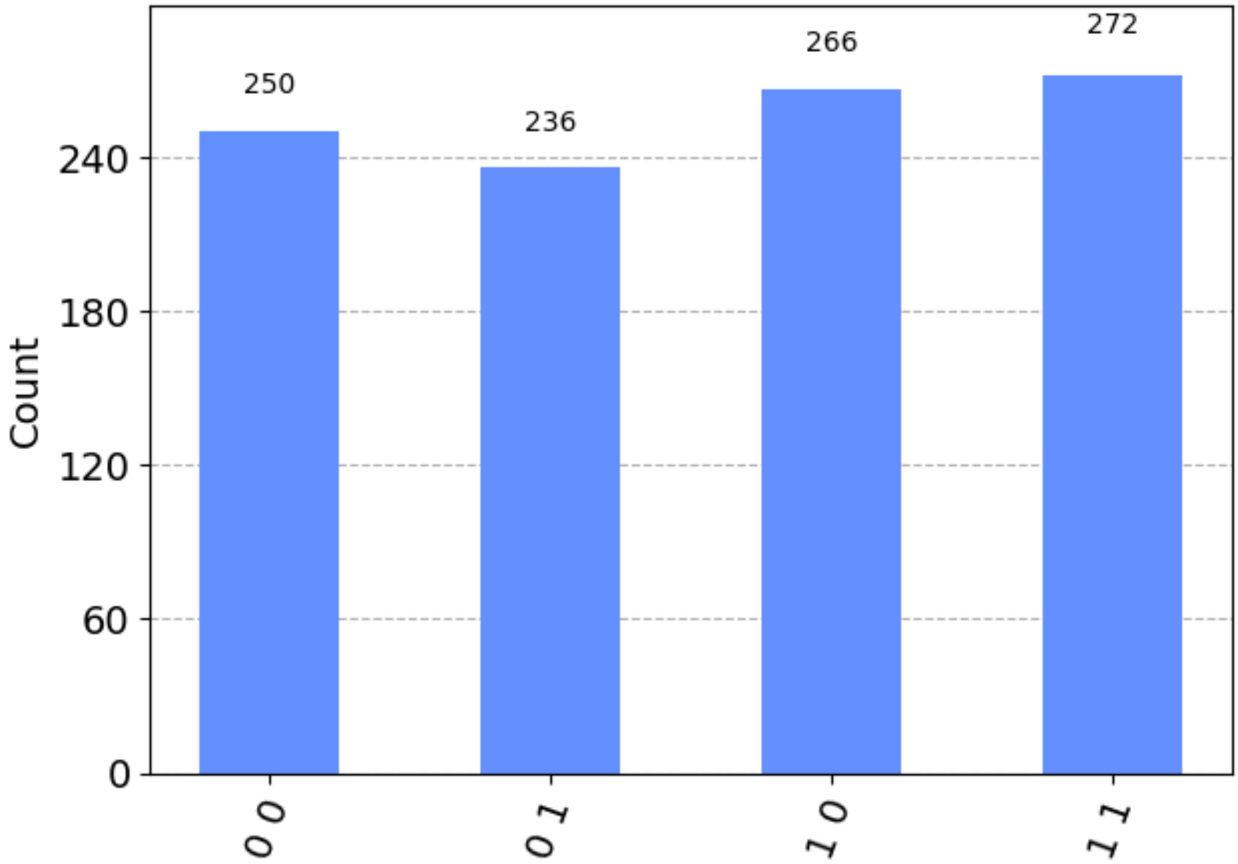

```
def bob_gates(qc, qubit, crz, crx):
    qc.x(qubit).c_if(crx, 1)
    qc.z(qubit).c_if(crz, 1)
```

```
qr = QuantumRegister(2, name="Alice")
qr1=QuantumRegister(1, name="Bob")
crz, crx = ClassicalRegister(1, name="c0"), ClassicalRegister(1, name="c1")
teleportation_circuit = QuantumCircuit(qr,qr1,crz, crx)
create_bell_pair(teleportation_circuit, 1, 2)
teleportation_circuit.barrier()
alice_gates(teleportation_circuit, 0, 1)
measure_and_send(teleportation_circuit, 0, 1)
teleportation_circuit.barrier()
bob_gates(teleportation_circuit, 2, crz, crx)
teleportation_circuit.draw(output='text',style='bw')
```



```
c0: 1/═══════════════════════════════╩═══╬═════════╬═══╡ 0x1 ╞
                                     0   ║         ║
c1: 1/═══════════════════════════════════╩════╡ 0x1 ╞══════════
                                         0
```

```
aer_sim = qiskit_aer.Aer.get_backend('aer_simulator')
results = aer_sim.run(teleportation_circuit).result()
answer = results.get_counts()
plot_histogram(answer)
```




```
Start coding or generate with AI.
```