```python
"""
This Python script implements the BB84 quantum key distribution protocol using
Qiskit and IBM's quantum computers.

The BB84 protocol is a quantum cryptography protocol developed by Charles
Bennett and Gilles Brassard in 1984 for secure communication. It leverages the
properties of quantum mechanics to create a secure key for encryption between
two parties, Alice and Bob.

In this script, Alice generates a random set of bits and encodes them into
quantum states (qubits) using a random set of bases. She then sends these
qubits to Bob, who measures them using a random set of bases. After the
transmission, Alice and Bob publicly share their bases and keep the bits
where they used the same base, forming a secure key.

This script runs the BB84 protocol on an IBM quantum computer or simulator
specified by the 'backend'.

Note: Running circuits on real quantum computers may take some time due to the
queue.
Author: Abraham Reines
Date: July 4, 2023
"""


from qiskit import QuantumCircuit, execute, IBMQ
import numpy as np

# Load IBM Q account
IBMQ.save_account('YOUR_API_KEY')  # replace 'YOUR_API_KEY' with your actual token
provider = IBMQ.load_account()
backend = provider.get_backend('ibmq_qasm_simulator')  # or another backend

# Define the quantum circuit
def bb84_circuit(bit, base):
    circuit = QuantumCircuit(1, 1)  # One qubit and one classical bit

    # Prepare qubit in the correct state
    if bit == 1:
        circuit.x(0)
    if base == 1:
        circuit.h(0)

    circuit.barrier()
    return circuit

# Alice generates bits
n = 100
alice_bits = np.random.randint(2, size=n)

# Alice generates random bases
alice_bases = np.random.randint(2, size=n)

# Bob generates random bases
bob_bases = np.random.randint(2, size=n)

# Alice sends qubits to Bob one at a time, and Bob measures each qubit
alice_key = []
bob_key = []
for i in range(n):
    # Alice prepares a qubit
    alice_circuit = bb84_circuit(alice_bits[i], alice_bases[i])

    # Bob measures the qubit in his chosen base
    bob_circuit = bb84_circuit(0, bob_bases[i])
    bob_circuit.measure(0, 0)
    total_circuit = alice_circuit.compose(bob_circuit)  # Use compose instead of +

    job = execute(total_circuit, backend, shots=1)
    result = job.result()
    counts = result.get_counts()
    bob_bit = int(list(counts.keys())[0])

    # Alice and Bob discard the bit if they used different bases
    if alice_bases[i] == bob_bases[i]:
        alice_key.append(alice_bits[i]
                         )
        bob_key.append(bob_bit)

# Check if keys are the same
print(alice_key == bob_key)
```