In **quantum computing**, we can perform **simple addition using qubits** with **basic quantum gates**, usually with a **quantum ripple-carry adder** approach. For a beginner-friendly example, let's create a circuit to **add two 1-bit numbers**.

Simple 1-Bit Addition Using Quantum Gates

Goal: Add two numbers a and b (each 0 or 1) and get:

- Sum (s)
- Carry (c)

Truth table for 1-bit addition:

a	b	Sum (s)	Carry (c)
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

Step 1: Quantum Circuit Setup

We need **3 qubits**:

1. a → first number
2. b → second number
3. carry/sum → output

And **2 classical bits** to read the outputs.

Step 2: Gates for 1-Bit Adder

1. **CNOT gate** → calculates **sum modulo 2** (XOR)
   - sum = a XOR b
2. **CCNOT (Toffoli) gate** → calculates **carry**
   - carry = a AND b

3. **Quantum Circuit: Step-by-Step Explanation**
4. **1. Qubits and Classical Bits**

qc = QuantumCircuit(3,2)

3 qubits:

1. q0 → represents input **a**
2. q1 → represents input **b**
3. q2 → represents **sum/carry output**

 2 classical bits:

1. c0 → stores **sum**
2. c1 → stores **carry**

qc.x(0) # a=1
qc.x(1) # b=1

X gate flips a qubit from |0⟩ → |1⟩.

Here, we set a=1 and b=1 for this example.

If inputs were 0, we would not apply X.

Compute Sum (XOR)

qc.cx(0,2)

- qc.cx(1,2)

CNOT (controlled-NOT) gate: flips the target qubit if the control qubit is 1.

- **Step explanation:**
  1. First CNOT(0,2) → sum qubit flips if a=1
  2. Second CNOT(1,2) → sum qubit flips again if b=1
- **Result:** sum = a XOR b

Example:

- a=1, b=1 → sum qubit flips twice → ends up as 0 → correct sum bit.

Compute Carry (AND)

- qc.ccx(0,1,2)
- **CCX (Toffoli gate):** controlled-controlled-NOT
- Flips the target qubit **only if both control qubits are 1**.
- **Purpose:** compute carry = a AND b

Example:

- a=1, b=1 → both controls are 1 → sum/carry qubit flips → represents carry bit.

Measurement

```
qc.measure(2,0)  # sum
qc.measure(1,1)  # carry
```

Measures qubits into classical bits.

q2 → c0 → sum

q1 → c1 → carry

Now the outputs can be read as classical binary numbers.

Draw the Circuit

```
qc.draw('text')
```

1-Bit Quantum Subtractor Logic

Inputs: a (minuend), b (subtrahend)

Outputs:

- difference = a XOR b
- borrow = NOT a AND b

Truth Table:

a	b	Difference	Borrow
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0

Quantum Circuit Steps in Quirk

Step 1: Qubits

- Use **3 qubits**:
  - q0 → a (minuend)
  - q1 → b (subtrahend)
  - q2 → difference/borrow output

Step 2: Difference Calculation (XOR)

- Same as adder:

difference = a XOR b

In Quirk:

1. CNOT(a -> q2)
2. CNOT(b -> q2)

Step 3: Borrow Calculation

- Borrow occurs if b = 1 and a = 0:

borrow = b AND (NOT a)

- Steps in Quirk:
  1. Apply X gate on a (to create NOT a)

2. Apply CCX (Toffoli) gate:
    - Controls: NOT a (q0) and b (q1)
    - Target: q2 or another qubit for borrow

This flips the target qubit only when borrow is needed.

Step 4: Measurement

- Measure:
    - Difference qubit → classical bit c0
    - Borrow qubit → classical bit c1