Shared Memory Problem (Race Condition)

Because all threads share memory, this can happen:

```
import threading

counter = 0

def increase():
    global counter
    for _ in range(100000):
        counter += 1

t1 = threading.Thread(target=increase)
t2 = threading.Thread(target=increase)

t1.start()
t2.start()

t1.join()
t2.join()

print(counter)
```

You might expect: `200000`

But you may get something like: `157342`

Why?

Because both threads modify `counter` at the same time → **race condition**.

Solution: Locks (Thread Safety)

To fix this, we use a **Lock**:

```
import threading

counter = 0
lock = threading.Lock()

def increase():
    global counter
    for _ in range(100000):
        with lock:
            counter += 1

t1 = threading.Thread(target=increase)
t2 = threading.Thread(target=increase)

t1.start()
t2.start()

t1.join()
t2.join()

print(counter)   # Now reliably 200000
```

Lock means:

“Only one thread can enter this block at a time.”