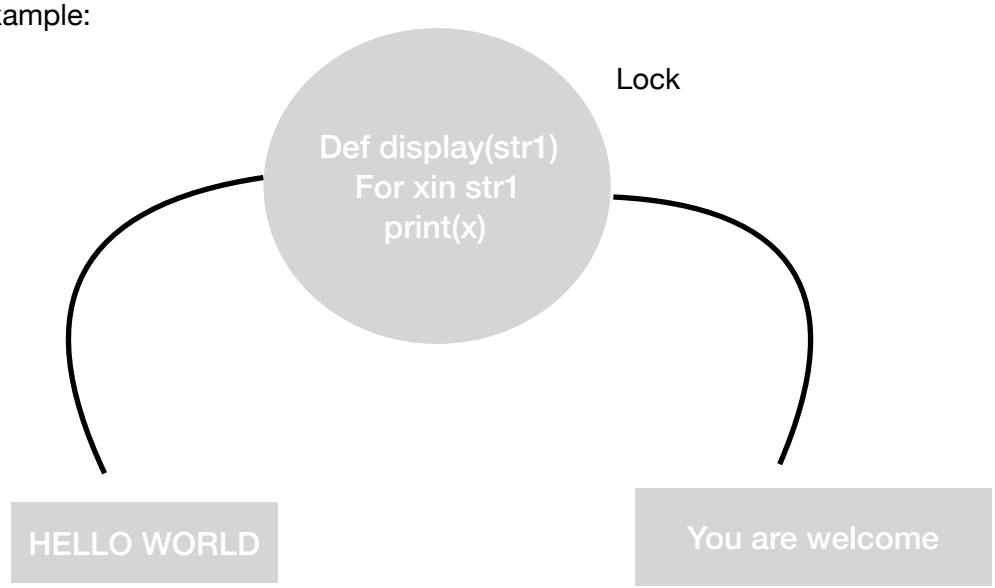
Thread Synchronisation(mutex)

If you have more than thread they may accessing to different resources then they may enter into race condition

And if we want to synchronise then we have to use mutex and semaphore.

Example:



Thread 1 Thread2

This will take HELLO WORLD to str1 and this will be taken as single single alphabets

And sending thread 2 also in the same function

Program:

```
from threading import *
from time import *

def display(str1):
    for x in str1:
        print(x)


t1 = Thread(target=display, args=('HELLO WORLD',))
t2 = Thread(target=display, args=('you are welcome',))

t1.start()
t2.start()

t1.join()
t2.join()
```

HELLO is broken down the output is mixed

- To print Hello world and you are welcome as separate string when this is mixed like this then it is called as race condition

Mutex: is a lock or variable that is used as lock and if once the thread is using display function it should put a lock and use display function and other thread is not enter into race condition actual thread should not start start using it

- And second thread use this display function only after first thread has finished it's work.

So How it is possible?

- We will create a lock called mutex
- So when the first thread has put the lock then the second thread has to wait. When the lock is released by first thread then the second thread can use it.

Program:

```
from threading import *
from time import *

def display(str1):
    l.acquire()
    for x in str1:
        print(x)
    l.release()


l = Lock()

t1 = Thread(target=display, args=('HELLO WORLD',))
t2 = Thread(target=display, args=('you are welcome',))

t1.start()
t2.start()

t1.join()
t2.join()
```

And the sleep function works slowly in discipline

```
from threading import *
from time import *

def display(str1):
    l.acquire()
    for x in str1:
        print(x)
        sleep(1)
    l.release()


l = Lock()

t1 = Thread(target=display, args=('HELLO WORLD',))
t2 = Thread(target=display, args=('you are welcome',))

t1.start()
t2.start()

t1.join()
t2.join()
```