Lec -17

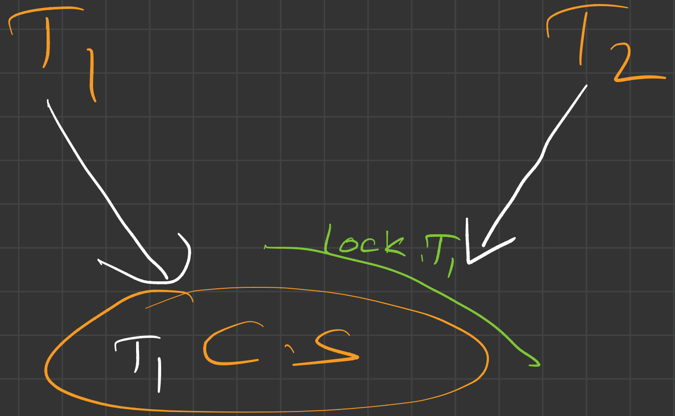
① — single flag ✗

② — Peterson's sol^n 2 threads

③ — Locks / Mutex

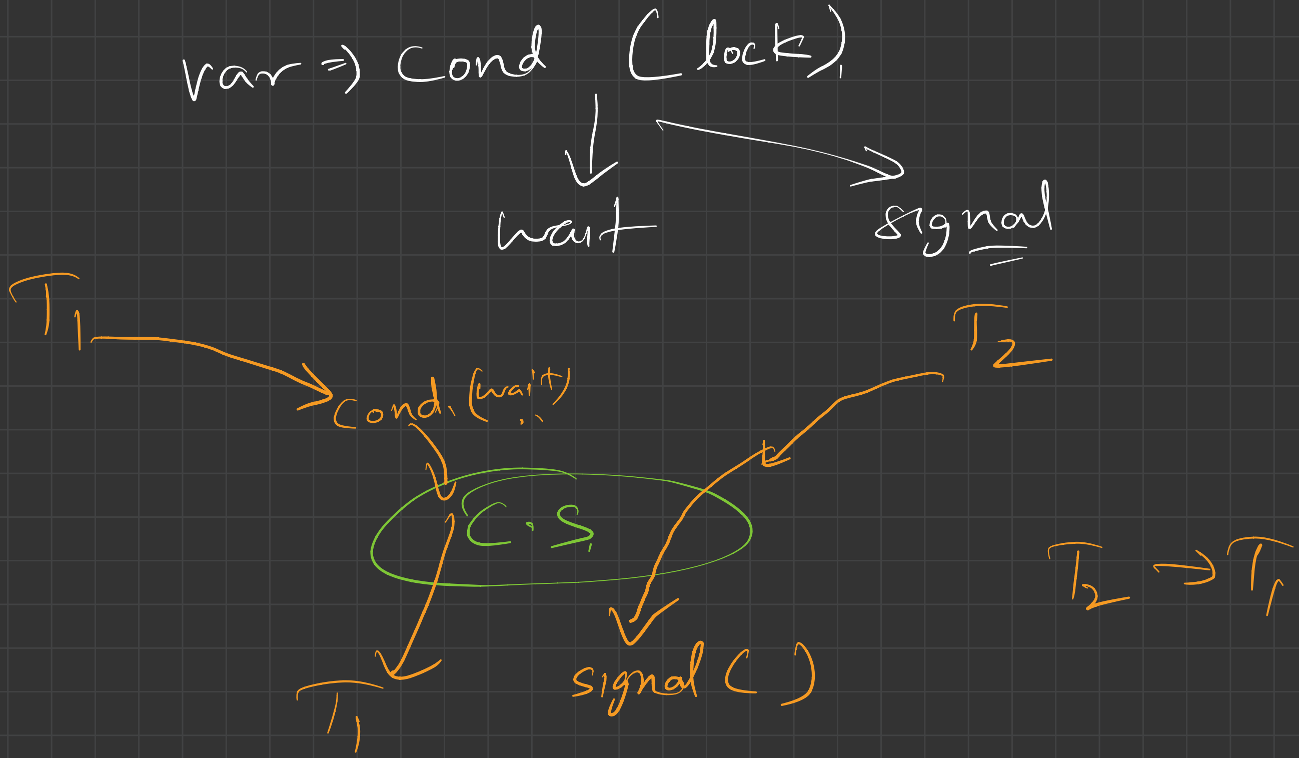
↓

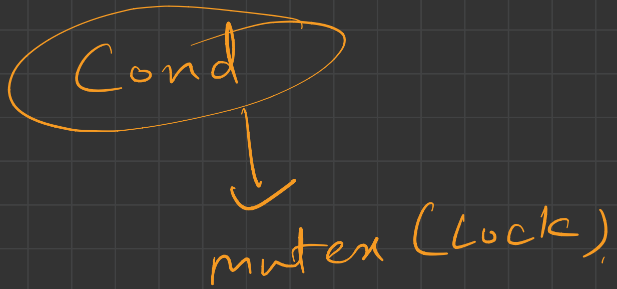
Problem → Busy waiting



while(flag)

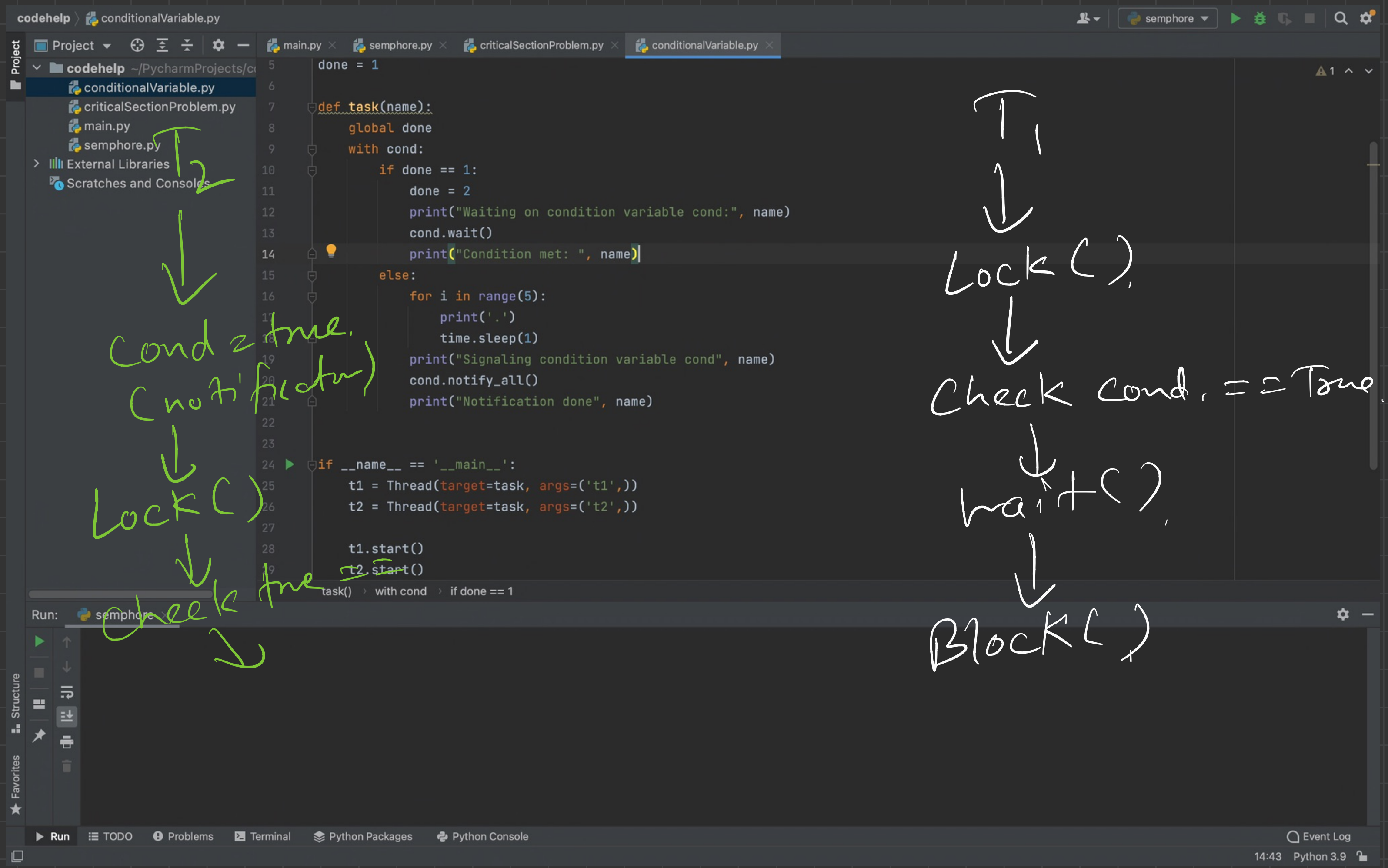
* Conditional Variables :-





Cond

mutex (Lock)



```python
done = 1

def task(name):
    global done
    with cond:
        if done == 1:
            done = 2
            print("Waiting on condition variable cond:", name)
            cond.wait()
            print("Condition met: ", name)
        else:
            for i in range(5):
                print('.')
                time.sleep(1)
            print("Signaling condition variable cond", name)
            cond.notify_all()
            print("Notification done", name)


if __name__ == '__main__':
    t1 = Thread(target=task, args=('t1',))
    t2 = Thread(target=task, args=('t2',))

    t1.start()
    t2.start()
```

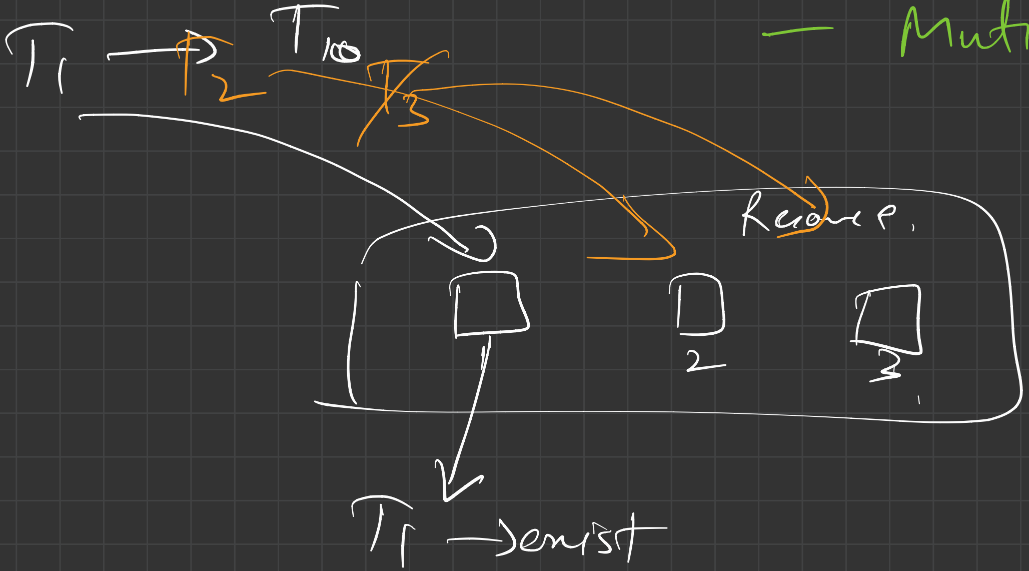
T2
↓
Cond = true.
(notification)
↓
Lock()
↓
check true
↓

T1
↓
Lock().
↓
Check cond. == True.
↓
wait().
↓
Block()

Semaphores → integer

Resource → Single X

— Multiple instances



Sem = ~~3~~ ~~2~~ ~~1~~ ~~0~~ N

↓

<0 X

wait()

C.S

Signal.

```
wait ( S)
  {
    S -> value --;
    if (S -> value < 0)
      {
        add to S -> block List
        Block();
      }
```

```
Signal(s)
{
  S -> value ++
  if ( S -> value <= 0)
  {
    remove P from S -> Block List
    wakup(P)
  }
}
```

Semaphore S(2);

T1 -> wait() -> S -> val = 1 -> After C.S -> Signal

T2 -> wait() -> S -> val = 0 ->

T3 -> wait() -> S -> val = -1 -> Block()

CPU.

Semaphore → n → 1 → Binary semaphore,

Semaphore → >1 → Counting semaphore