7-2 线程创建的总结

一、当多个线程运行时，可能会出现的问题及解决方案

- 通过线程执行的函数无法获取返回值——线程间如何通信：**通过队列**
- 多个线程同时修改文件可能造成数据错错乱——线程间如何避免资源抢占：**创建线程锁**
- 线程数量太多可能会造成资源不足，甚至死机等情况——如何避免创建线程数量过多：**创建线程池**

二、通过对列通信来解决

代码如下：

```
import threading,time,queue,random

def sender():
    while True:
        x=random.randint(1,10)
        print("send done:",x)
        q.put(x) 往队列当中放数据
        time.sleep(1)

def recvder():
    while True:
        x=q.get() 从队列中取数据
        print("recv done:",x*3.14)
        time.sleep(1)

if __name__ == "__main__":
    q=queue.Queue() 定义队列
    t1=threading.Thread(target=sender)
    t2=threading.Thread(target=recvder)

    t1.start()
    t2.start()

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

三、创建线程锁

在线程代码中需要加上锁的地方写加锁代码，要释放锁的地方写解锁代码即可

- 使用模块： `threading`
- 如何加锁： `threading.Lock().acquire()`
- 如何解锁： `threading.Lock().release()`

四、创建线程池

首先写出创建线程池的方法，之后往线程池中放入线程即可

- 使用的模块： `concurrent.futures`