python3对多线程join的理解

2019年01月16日 18:19:20 阿常呓语 阅读数：424

python3 对多线程join 的理解

多线程编程个人比较难理解,虽然 工作中用一直用多线程编程, 难免有时候也会遇到问题, 这里就简单谈一谈 线程里面join 如何使用的?

当然参考了别人的博客,如果有什么问题,请留言反馈, 一起交流.

这里所说的是 threading.Thread 这种方式 创建的多线程.

例如下面的代码,
有两个线程, 一个速度快, 一个速度慢

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
@Time    : 2019/1/16 10:28
@File    : test_join.py
@Author  : frank.chang@shoufuyou.com


refer :
# Youtube video tutorial: https://www.youtube.com/channel/UCdyjiB5H8Pu7aDTNVXTTpcg
# Youku video tutorial: http://i.youku.com/pythontutorial

https://github.com/MorvanZhou/tutorials/blob/master/threadingTUT/thread3_join.py

"""

import threading
import time


def T1_job():
    print('T1 start')
    for i in range(10):
        print('begin sleep 0.1s')
        time.sleep(0.1)
    print('T1 finish')


def T2_job():
    print('T2 start')
    print('T2 finish')


def main():
    print('---main begin----')
    t1 = threading.Thread(target=T1_job, name='T1')
    t2 = threading.Thread(target=T2_job, name='T2')
    t1.start()
    t2.start()

    print('---main end----')


if __name__ == '__main__':
    main()

```

打印结果如下:

```
---main begin----
---main end----
T2 start
T1 start
begin sleep 0.1s
T2 finish
begin sleep 0.1s
begin sleep 0.1s
begin sleep 0.1s
begin sleep 0.1s
begin sleep 0.1s
begin sleep 0.1s
begin sleep 0.1s
begin sleep 0.1s
begin sleep 0.1s
T1 finish

```

分析: 主线程,main 很快就结束了, 子线程 T2先开始,然后 T1开始, T2很快就完成任务了, 然后经过漫长的等待最后也完成任务了. 这是没有用join 的情况. 即使 主线程main 停止了, 两个子线程 并不会直接退出. 而是要把自己各自的任务完成, 才会退出.

> 那么 问题来了,有的时候可能我们不想 看到这样的效果, 我们有可能要看到, T1结束,T2结束,然后main 线程 结束,整个程序结束. 这个时候怎么办呢?

这个时候就需要用join 来解决这个问题.

```
def main():
    print('---main begin----')
    t1 = threading.Thread(target=T1_job, name='T1')
    t2 = threading.Thread(target=T2_job, name='T2')
    t1.start()
    t2.start()

    t1.join()
    print("t1 done")
    t2.join()
    print("t2 done")

    print('---main end----')
```

打印结果如下:

```
---main begin----
T2 start
T2 finish
T1 start
begin sleep 0.1s
begin sleep 0.1s
begin sleep 0.1s
begin sleep 0.1s
begin sleep 0.1s
begin sleep 0.1s
begin sleep 0.1s
begin sleep 0.1s
begin sleep 0.1s
begin sleep 0.1s
T1 finish
t1 done
t2 done
```

```
---main end----
```

分析: 通过join ,可以让线程在这里等着,阻塞在这里,等线程跑完了, 在继续跑下面的代码.

t1.join() 跑完之后, t1 就完成任务了. 之后 t2.join(), 等待t2线程跑完, 它早已经跑完了.

这样之后打印 main end .

```
def main():
    print('---main begin----')
    t1 = threading.Thread(target=T1_job, name='T1')
    t2 = threading.Thread(target=T2_job, name='T2')
    t1.start()
    t2.start()

    t2.join()
    print("t2 done")

    t1.join()
    print("t1 done")

    print('---main end----')

```

结果如下:

```
---main begin----
T1 start
begin sleep 0.1s
T2 start
T2 finish
t2 done
begin sleep 0.1s
begin sleep 0.1s
begin sleep 0.1s
begin sleep 0.1s
begin sleep 0.1s
begin sleep 0.1s
begin sleep 0.1s
begin sleep 0.1s
begin sleep 0.1s
T1 finish
t1 done
---main end----

```

分析: 这里 改了一下 join的顺序 , t2.join() , 这样一定 是t2 先跑完, 之后才会往下跑下面的代码.
但是 T1, T2 这两个线程 哪个先开始 我们好像没有办法确定,有时候是 T1 ,有时候T2 .

join 有一个参数 timeout 意思是等待时间, 如果超过等待的时间, 任务还没有完成那就不等待了,直接继续跑下面的代码.
如果任务 完成的时候, 没有超过等待时间, 那么就正常往下跑.

timeout 是一个float number 单位是秒

```
join(timeout=None)
Wait until the thread terminates. This blocks the calling thread until the thread whose join() method is called term
```

```

import threading
import time

```

```
def T1_job():
    print('T1 start')
    for i in range(10):
        print('begin sleep 0.5s')
        time.sleep(0.5)
    print('T1 finish')


def T2_job():
    print('T2 start')
    print('T2 finish')


def main():
    print('---main begin----')
    t1 = threading.Thread(target=T1_job, name='T1')
    t2 = threading.Thread(target=T2_job, name='T2')
    t1.start()
    t2.start()

    t1.join(timeout=3)
    print("t1 done")

    t2.join()
    print("t2 done")

    print('---main end----')
```

结果如下:

```
---main begin----
T1 start
begin sleep 0.5s
T2 start
T2 finish
begin sleep 0.5s
begin sleep 0.5s
begin sleep 0.5s
begin sleep 0.5s
begin sleep 0.5s
t1 done     # t1 继续往下跑
t2 done
---main end----
begin sleep 0.5s
begin sleep 0.5s
begin sleep 0.5s
begin sleep 0.5s
T1 finish
```

分析: 从打印结果看, t1 睡了大概3s,之后没有继续等待,直接跑下面的代码. 之后main end 完成后, T1 还有4次打印 才完成任务,然后退出的.

> 我们都知道,只要thread.start() 线程就开始了, 至于线程怎么跑, t1, t2谁先跑完,取决于 每个线程的任务量,或者说执行时间.所以在实际的编程中, t1, t2是很难评估哪个先 跑完. 为此才会有join 这个操作, 这个操作的意思:

阻塞线程,把线程的任务完成后,才会往下继续执行下面的代码.

总结

用join 就是用来线程的执行顺序. 如果需要显示控制线程的 执行顺序, 我们需要在 线程start 后, 显示 的join来阻塞当前线程.
个人任务, 在实际编程中如何这两个线程任务 之间没有关系, 不需要关系两个线程之间的先后顺序, 这个时候其实 可以不用join ,因为这两个线程任务 是没有 关系的所以不用join.

如果 这两个线程 一定是 要 T1 先完成, T2 在完成,之后才能进行 下面的操作, 之后时候用join,来控制 进程的执行进度.

参考文档

参考链接
https://blog.csdn.net/zhiyuan_2007/article/details/48807761
https://github.com/MorvanZhou/tutorials/blob/master/threadingTUT/thread3_join.py
https://morvanzhou.github.io/tutorials/python-basic/threading/3-join/
https://docs.python.org/3.7/library/threading.html#threading.Thread.join

分享快乐,留住感动. 2019-01-16 19:53:28 --frank