python利用多进程多线程加速代码的执行效率

这里不谈理论知识， 什么GIL什么的，还有各种理论知识，可以自行搜索了解学习，这里我只附上一些我常用的代码方法。

1. python多线程

多线程适用于IO密集型的程序，能够很大程度上提升代码的执行效率，但是面对计算密集型的程序就有点无能为力了（这里我在实际中进行图像处理时，试过，用多进程更好）

这里附上一个多线程的一个套路代码

```
import multithreading
def image_processing(imageName,args1, args2):
    '''这个函数放置具体的图像处理代码, args1,args2为其他必要的参数
    '''
    pass


# 设置锁
threadLock = threading.Lock()

def get_imgName(imgNames):
    '''利用锁，获得每一张图片，这里imgNames为一个迭代器
    '''
    print ('thread %s is running...' % threading.current_thread().name)
    while True:
        try:
            with threadLock:
                img_file_name = next(imgNames)  # 获取原图的文件名
        except StopIteration:
            return None
        image_processing(img_file_name, args1, args2,)


if __name__ == '__main__':
        # #线程数
        thread_num = opt.thread_num
        ts = []
        for i in range(0, thread_num):
            t = threading.Thread(target=get_imgName,
name='_parallel_save_imageThread %s' % i,
                args=(imgNames,))
            # t.setDaemon(True)   #把子进程设置为守护线程，必须在start()之前设置
            t.start()
            ts.append(t)
        for t in ts:
            t.join()
```

2. 多进程简单代码

```
from multiprocessing import Pool


def image_processing(imageName,args1, args2):
    '''这个函数放置具体的图像处理代码, args1,args2为其他必要的参数
    '''
    pass


if __name__ == "__main__":
    # #进程数
    process_num = opt.process_num
    processing_pool = Pool(processes=process_num)


    #============================================================#
    #                    多进程主要代码                           #
    #============================================================#
    for img_file_name in  files:
       processing_pool.apply_async(
                func = image_processing,
                args = (img_file_name,args1, args2, )
            )
    processing_pool.close()
    processing_pool.join()
```

3. 多进程向同一个list或者dict写元素

```
import multiprocessing

def image_processing(imageName,syn_info, args1, args2):
    '''这个函数放置具体的图像处理代码, args1,args2为其他必要的参数
    '''
    pass


if __name__ == '__main__':

   # 进程数
    processing_num = opt.processing_num
    # 进程池
    processing_pool = multiprocessing.Pool(processes=processing_num)
    # 最终保存的字典, 需要采用多进程通信的字典格式
    syn_info = multiprocessing.Manager().dict()
    # syn_info = multiprocessing.Manager().list()

    for img_file_name in range(files):
        processing_pool.apply_async(
                func = image_processing,
                args = (img_file_name, syn_info,args1, args2, )
            )
    processing_pool.close()
    processing_pool.join()
```

这里记录一下基本的框架代码，更多的multithreading与multiprocessign这俩包的细节知识，其实我也不大懂，都是用着学着，查着