（1）python多线程与多进程的区别

**线程：线程**是操作系统能够进行运算调度的最小单位。它被包含在进程之中，是进程中的实际运作单位。一条线程指的是进程中一个单一顺序的控制流，一个进程中可以并发多个线程，每条线程并行执行不同的任务。一个线程是一个execution context（执行上下文），即一个cpu执行时所需要的一串指令。

假设你正在读一本书，没有读完，你想休息一下，但是你想在回来时恢复到当时读的具体进度。有一个方法就是记下页数、行数与字数这三个数值，这些数值就是execution context。如果你的室友在你休息的时候，使用相同的方法读这本书。你和她只需要这三个数字记下来就可以在交替的时间共同阅读这本书了。

线程的工作方式与此类似。CPU会给你一个在同一时间能够做多个运算的幻觉，实际上它在每个运算上只花了极少的时间，本质上CPU同一时刻只干了一件事。它能这样做就是因为它有每个运算的execution context。就像你能够和你朋友共享同一本书一样，多任务也能共享同一块CPU。

**进程：**一个程序的执行实例就是一个进程。每一个进程提供执行程序所需的所有资源

（进程本质上是资源的集合）一个进程有一个虚拟的地址空间、可执行的代码、操作系统的接口、安全的上下文（记录启动该进程的用户和权限等等）、唯一的进程ID、环境变量、优先级类、最小和最大的工作空间（内存空间），还要有至少一个线程。

每一个进程启动时都会最先产生一个线程，即主线程。然后主线程会再创建其他的子线程。

**进程与线程区别：**

1.同一个进程中的线程共享同一内存空间，但是进程之间是独立的。  
2.同一个进程中的所有线程的数据是共享的（进程通讯），进程之间的数据是独立的。  
3.对主线程的修改可能会影响其他线程的行为，但是父进程的修改（除了删除以外）不会影响其他子进程。  
4.线程是一个上下文的执行指令，而进程则是与运算相关的一簇资源。  
5.同一个进程的线程之间可以直接通信，但是进程之间的交流需要借助中间代理来实现。  
6.创建新的线程很容易，但是创建新的进程需要对父进程做一次复制。  
7.一个线程可以操作同一进程的其他线程，但是进程只能操作其子进程。  
8.线程启动速度快，进程启动速度慢（但是两者运行速度没有可比性）。

（2）Python是如何进行内存管理的？

Python引用了一个内存池(memory pool)机制，即Pymalloc机制(malloc:n.分配内存)，用于管理对小块内存的申请和释放  
内存池（memory pool）的概念：  
　　当 创建大量消耗小内存的对象时，频繁调用new/malloc会导致大量的内存碎片，致使效率降低。内存池的概念就是预先在内存中申请一定数量的，大小相等 的内存块留作备用，当有新的内存需求时，就先从内存池中分配内存给这个需求，不够了之后再申请新的内存。这样做最显著的优势就是能够减少内存碎片，提升效率。  
内存池的实现方式有很多，性能和适用范围也不一样。   
python中的内存管理机制——Pymalloc：  
　　python中的内存管理机制都有两套实现，一套是针对小对象，就是大小小于256bits时,pymalloc会在内存池中申请内存空间；当大于256bits，则会直接执行new/malloc的行为来申请内存空间。  
　　关于释放内存方面，当一个对象的引用计数变为0时，python就会调用它的析构函数。在析构时，也采用了内存池机制，从内存池来的内存会被归还到内存池中，以避免频繁地释放动作。

(3) Python里面search()和match()的区别

match()函数只检测RE是不是在string的开始位置匹配，search()会扫描整个string查找匹配, 也就是说match()只有在0位置匹配成功的话才有返回，如果不是开始位置匹配成功的话，match()就返回none

(4) 谈下python的GIL

GIL 是python的全局解释器锁，同一进程中假如有多个线程运行，一个线程在运行python程序的时候会霸占python解释器（加了一把锁即GIL），使该进程内的其他线程无法运行，等该线程运行完后其他线程才能运行。如果线程运行过程中遇到耗时操作，则解释器锁解开，使其他线程运行。所以在多线程中，线程的运行仍是有先后顺序的，并不是同时进行。

多进程中因为每个进程都能被系统分配资源，相当于每个进程有了一个python解释器，所以多进程可以实现多个进程的同时运行，缺点是进程系统资源开销大