**1.进程和线程的区别？什么时候用进程？主进程和子进程之间的关系？**

进程是指在系统中正在运行的一个应用程序；程序一旦运行就是进程，或者更专业化来说：进程是指程序执行时的一个实例。

线程是进程的一个实体。

进程——资源分配的最小单位，线程——程序执行的最小单位。

线程进程的区别体现在几个方面：

第一：因为进程拥有独立的堆栈空间和数据段，所以每当启动一个新的进程必须分配给它独立的地址空间，建立众多的数据表来维护它的代码段、堆栈段和数据段，这对于多进程来说十分“奢侈”，系统开销比较大，而线程不一样，线程拥有独立的堆栈空间，但是共享数据段，它们彼此之间使用相同的地址空间，共享大部分数据，比进程更节俭，开销比较小，切换速度也比进程快，效率高，但是正由于进程之间独立的特点，使得进程安全性比较高，也因为进程有独立的地址空间，一个进程崩溃后，在保护模式下不会对其它进程产生影响，而线程只是一个进程中的不同执行路径。一个线程死掉就等于整个进程死掉。

第二：体现在通信机制上面，正因为进程之间互不干扰，相互独立，进程的通信机制相对很复杂，譬如管道，信号，消息队列，共享内存，套接字等通信机制，而线程由于共享数据段所以通信机制很方便。。

3.属于同一个进程的所有线程共享该进程的所有资源，包括文件描述符。而不同过的进程相互独立。

4.线程又称为轻量级进程，进程有进程控制块，线程有线程控制块；

5.线程必定也只能属于一个进程，而进程可以拥有多个线程而且至少拥有一个线程

第四：体现在程序结构上，举一个简明易懂的列子：当我们使用进程的时候，我们不自主的使用if else嵌套来判断pid，使得程序结构繁琐，但是当我们使用线程的时候，基本上可以甩掉它，当然程序内部执行功能单元需要使用的时候还是要使用，所以线程对程序结构的改善有很大帮助。

进程与线程的选择取决以下几点：

1、需要频繁创建销毁的优先使用线程；因为对进程来说创建和销毁一个进程代价是很大的。

2、线程的切换速度快，所以在需要大量计算，切换频繁时用线程，还有耗时的操作使用线程可提高应用程序的响应

3、因为对CPU系统的效率使用上线程更占优，所以可能要发展到多机分布的用进程，多核分布用线程；

4、并行操作时使用线程，如C/S架构的服务器端并发线程响应用户的请求；

5、需要更稳定安全时，适合选择进程；需要速度时，选择线程更好。

**2.编程实现list转dict，dict转list**

class A:

    a=1

    b=100

class B:

    a=1

    b=200

class C:

    a=2

    b=200

arr=[A,B,C]

res={}

for obj in arr:

    if obj.a not in res:

        res[obj.a]=list()

    res[obj.a].append(obj.b)

**3.定义两个dict，找出同时在两个dict中共有的键，并判断所对应的值是否相等**

**4.定义一个list，打印出每个元素出现的次数**

1. mylist = ['D', 'G', 'H','D', 'G', 'H','D', 'G']
2. myset = set(mylist)  #myset是另外一个列表，里面的内容是mylist里面的无重复项
3. **for** item **in** myset:
4. **print**("the %s has found %d" %(item,mylist.count(item)))
5. times = mylist.count('D')
6. **print**(list(myset))

**5.最优化实现：查找一篇文章（一个文件），打开一个文件，输入关键词，如果匹配到5个关键词，说明整篇文章匹配到，打印5个关键词，如果匹配的关键词不到5个，打印关键词**

**6.简述面相函数编程和面相对象编程的区别？什么时候使用面相函数编程？什么时候使用面相对象编程？请举例说明**

函数式编程，顾名思义，这种编程是以函数思维做为核心，在这种思维的角度去思考问题。这种编程最重要的基础是λ演算，接受函数当作输入和输出。  
面向对象编程，这种编程是把问题看作由对象的属性与对象所进行的行为组成。基于对象的概念，以类作为对象的模板，把类和继承作为构造机制，以对象为中心，来思考并解决问题。

优点  
函数式编程：支持闭包和高阶函数，闭包是一种可以起函数的作用并可以如对象般操作的对象；而高阶函数是可以以另一个函数作为输入值来进行编程。支持惰性计算，这就可以在求值需要表达式的值得时候进行计算，而不是固定在变量时计算。还有就是可以用递归作为控制流程。函数式编程所编程出来的代码相对而言少很多，而且更加简洁明了。  
面向对象编程：面向对象有三个主要特征，分别是封装性、继承性和多态性。类的说明展现了封装性，类作为对象的模板，含有私有数据和公有数据，封装性能使数据更加安全依赖的就是类的特性，使得用户只能看到对象的外在特性，不能看到对象的内在属性，用户只能访问公有数据不能直接访问到私有数据。类的派生功能展现了继承性，继承性是子类共享父类的机制，但是由于封装性，继承性也只限于公有数据的继承（还有保护数据的继承），子类在继承的同时还可以进行派生。而多态性是指对象根据接收的信息作出的行为的多态，不同对象接收同一信息会形成多种行为。

缺点  
函数式编程：所有的数据都是不可以改变的，严重占据运行资源，导致运行速度也不够快。  
面向对象编程：为了编写可以重用的代码导致许多无用代码的产生，并且许多人为了面向对象而面向对象导致代码给后期维护带来很多麻烦。