生成器(generator),yield关键字：函数中的yield起的作用就是把这个函数变成一个generator。调用这个函数不会返回函数值，而是返回一个iterable对象。下次调用时，代码从yield的下一条语句继续执行，直到再次遇到yield(每次执行到一个yield语句就会中断，并返回一个迭代值:gen.next()，下次执行时从yield的下一个语句继续执行)。获取迭代器的值：next(gen)

全局变量引用与修改：若需要在函数中修改全局变量，那么在函数钟需要使用global var

classmethod修饰符：classmethod修饰的函数不需要实例化，不需要self参数，但修饰的函数的第一个参数需要是表示自身类的cls参数，可以来调用类的属性，类的方法

多线程：通过threading提供对线程的支持。通过继承threading.Thread类，并实例化后调用start()方法启动新线程，即调用了线程的run()方法。包括了run(),start(),join()等

Python线程同步：threadlock = threading.Lock() threadlock.acquire() threadlock.release()

线程优先级队列(Queue):提供了同步的，线程安全的队列类，包括FIFO队列， LIFO(后入先出)，优先队列。这些队列都实现了锁原语，能够在多线程中直接使用，可以使用队列实现线程间的同步。

异常处理：python中提供了两个功能处理程序运行中的异常和错误：异常处理，断言。

当脚本发生异常时，要捕获处理它，否则程序会终止执行。

捕获异常使用try/except语句。

try的工作原理：

1. 当开始一个try语句时，python就在当前程序的上下文做标记，这样当异常出现时就可以回到这里。如果当try语句发生异常，python就跳回到第一个匹配该异常的except语句，异常处理完毕，继续执行后续语句。
2. 当try语句发生了异常，却没有匹配的except子句，异常将被递交到上层的try，或者到程序的最上层，直至程序退出。
3. 如果try子句中没发生异常，python将执行else语句(如果有else语句)，然后执行后续语句。