1. 在策略中，一般不直接调用sendOrder()， 而且用四个二次封装函数函数，这些都是在class CtaTemplate中定义的，这里面主要区别就是sendorder()函数的中ordertype制定不一样，用来区分是买开卖开等交易类型。

返回一个vtOrderIDList, 这个list里面包含vtOrderID，这个是个内部给号，可以用做追踪同一个order的状态。

2.    接下来我们看看那sendOrder()源码，还在class CtaTemplate中定义；如果stop为True是本地停止单，这个停止单并没有发送给交易所，而是存储在内部，使用ctaEngine.sendStopOrder()函数；否则这直接发送到交易所，使用ctaEngine.sendStopOrder函数。这里会返回一个vtOrderIDList, 这个list里面包含vtOrderID，和上面说的一样。这里补充一下，对于StopOrder真是交易通常是涨停价或者跌停价发出的市价单（Market price），参数price只是触发条件；而普通sendOrder是真正按照参数price的限价单（Limit price）

3.    这里我们首先看看ctaEngine.sendStopOrder()函数，在class CtaEngine中定义的，首先实例初始化时候定义了两个字典，用来存放stoporder，区别一个是停止单撤销后删除，一个不会删除；还定义了一个字典，策略对应的所有orderID。

然后在函数sendStopOrder中，首先记录给本地停止单一个专门编号，就是前缀加上顺序编号，其中STOPORDERPREFIX是 'CtaStopOrder.'，那么第一条本地编码就是'CtaStopOrder.1'，后面是这个单据信息；这里可以发现orderType其实是一个direction和offset的组合。交易方向direction有Long、short两个情况，交易对offset有open和close两个情况。组合就是上面买开，卖平等等。然后把这个stoporder放入字典，等待符合价格情况到达触发真正的发单。这里返回本地编码作为vtOrderIDList。

4.     下面是processStopOrder()函数，也在classCtaEngine中定义的，主要是当行情符合时候如何发送真正交易指令，因为stopOrderID不是tick交易重点，这里简单讲讲，具体请看源码。

当接收到tick时候，会查看tick.vtSymbol，是不是存在workingStopOrderDict的so.vtSymbol有一样的，如果有，再看tick.lastPrice价格是否可以满足触发阈值，如果满足，根据原来so的交易Direction，Long按照涨停价，Short按照跌停价发出委托。然后从workingStopOrderDic和strategyOrderDict移除该so，并更新so状态，并触发事件onStopOrder(so).

这里发现，so只是只是按照涨停价发单给交易所，并没有确保成绩，而且市价委托的实际交易vtOrderID也没有返回；从tick交易角度，再收到tick后再发送交易，本事也是有了延迟一tick。所以一般tick级别交易不建议使用stoporder。

5.  前面说了这么多，终于到了正主sendOrder(),也在class CtaEngine中定义的。代码较长，下面做了写缩减。

1）通过mainEngine.getContract获得这个品种的合约的信息，包括这个合约的名称，接口名gateway（国内期货就是ctp），交易所，最小价格变动等信息；

2）创建一个class VtOrderReq的对象req，在vtObject.py中，这个py包括很多事务类的定义；然后赋值，包括contract获得信息，交易手数，和price，和priceType，这里只有限价单。

3）根据orderType赋值direction和offset，之前sendStopOrder中已经说了，就不重复。

4）然后跳到mainEngine.convertOrderReq(req)，这里代码比较跳，分析下来，如果之前没有持仓，或者是直接返回[req]；如果有持仓就调用PositionDetail.convertOrderReq(req)，这个时候如果是平仓操作，就分析持仓量，和平今和平昨等不同操作返回拆分的出来[reqTd,reqYd]，这里不展开。

5) 如果上一部没有返回[req]，则委托有问题，直接返回控制。如果有[req]，因为存在多个req情况，就遍历每个req，使用mainEngine.sendOrder发单，并保存返回的vtOrderID到orderStrategyDict[],strategyOrderDict[]两个字典；然后把vtOrderIDList返回。

6.    在mainEngine.sendOrder中，这里不列举代码了，首先进行风控，如果到阈值就不发单，然后看gateway是否存在，如果存在，就调用gateway.sendOrder(orderReq)方法；下面用ctpgateway说明。class CtpGateway(VtGateway)是VtGateway是继承，把主要发单，返回上面都实现，同时对于不同的接口，比如外汇，数字货币，只要用一套接口标准就可以，典型继承使用。

CtpGateway.sendOrder实际是调用class CtpTdApi(TdApi)的，这个就是一套ctp交易交口，代码很简单，最后是调用封装好C++的ctp接口reqOrderInsert()。最关键返回的vtOrderID是接口名+顺序数。

整个流程下来，不考虑stoporder，是ctaTemplate -> CtaEngine->mainEngine ->ctpgateway ->CtpTdApi, 传到C++封装的接口。返回的就是vtOrderID; 因为存在平昨，平今还有锁仓，反手等拆分情况，返回的可能是一组。