1.Implement an order book class: here is a sample python API, and a unit test to add bids, asks, show the orderbook afterwards, and show the trades

# The orderbook contains bid orders and ask orders,  
# Each order has the following members:  
# Time: the time at which the order was sent  
# Side: Bid/Ask - bid means buy, and ask means sell  
# Size: the amount to trade  
# Price: the price at which to trade  
# OrderId: a unique tracking number for the order  
# Name: the name/Id of the person doing the order  
# A trade occurs when an order was entered that intersects an existing order,  
# Each Trade has the following members:  
# Time  
# Price  
# Size  
# BuyerId  
# SellerId  
# for example:  
# if the orderbook contains a bid order at price=100, size=10  
# and we just receive an ask order at price=99, size = 12  
# a trade will happen at price 100, for size 10,  
# and the orderbook will now contain an ask order with price=99, size=2  
# when a trade occurs, the subscribers (traders) need to be notified of the trade, and the trade should be logged  
  
class OrderBook:  
 def \_\_init\_\_(self):  
 self.bids = {}  
 self.asks = {}  
 # feel free to add more members as required  
  
 # Returns the highest bid and lowest ask orders  
 def show\_top(self):  
 # please implement  
 best\_bid = 0  
 best\_ask = 0  
 return best\_bid, best\_ask  
  
 # Returns the list of trades sorted by time  
 def show\_trades(self):  
 # please implement  
 return  
  
 # print the current state orderbook in a human readable format  
 def show\_orderbook(self):  
 # please implement  
 return  
   
 # called by add\_order, when a trade has occurred, notify subscribers of the trade  
 def notify\_trade(self, trade\_event, subscribers):  
 #please implement as required  
 return   
  
 # Add an order, notify if a trade has occurred  
 # record the order (and trade) in a log  
 def add\_order(self, order, sender\_id):  
 #please implement  
 return   
  
 # remove an order  
 # record the removal order in the log   
 def remove\_order(self, orderId, sender\_id):  
 #please implement  
 return