```python
from AlgorithmImports import *

import numpy as np

class SimpleBreakoutExample(QCAlgorithm):

   def Initialize(self):
       # Set the cash amount you want to use for backtest
       self.SetCash(100000)

       # Start and end dates for backtest
       self.SetStartDate(2020,3,1)
       self.SetEndDate(2022,3,1)

       # added stock
       self.symbol = self.AddEquity("SPY", Resolution.Daily).Symbol

       # Lookback length
       self.lookback = 20

       # Upper/lower limit for lookback length
       self.ceiling, self.floor = 30, 10

       # Trailing stop loss
       self.initialStopRisk = 0.98
       self.trailingStopRisk = 0.9

       # function runs 20 minutes after the market opens
       self.Schedule.On(self.DateRules.EveryDay(self.symbol), \
                        self.TimeRules.AfterMarketOpen(self.symbol, 20), \
                        Action(self.EveryMarketOpen))


   def OnData(self, data):
       # stock price plotting it
       self.Plot("Data Chart", self.symbol, self.Securities[self.symbol].Close)

   def EveryMarketOpen(self):
       #  lookback length based off of the 30 day change of volatility
       close = self.History(self.symbol, 31, Resolution.Daily)["close"]
       todayvol = np.std(close[1:31])
```

```
        yesterdayvol = np.std(close[0:30])
        deltavol = (todayvol - yesterdayvol) / todayvol
        self.lookback = round(self.lookback * (1 + deltavol))

        # upper/lower limit of lookback length
        if self.lookback > self.ceiling:
            self.lookback = self.ceiling
        elif self.lookback < self.floor:
            self.lookback = self.floor

        # List of the daily highs
        self.high = self.History(self.symbol, self.lookback, Resolution.Daily)["high"]

        # Buy in case of stock price jump or breakout
        if not self.Securities[self.symbol].Invested and \
                self.Securities[self.symbol].Close >= max(self.high[:-1]):
            self.SetHoldings(self.symbol, 1)
            self.breakoutlvl = max(self.high[:-1])
            self.highestPrice = self.breakoutlvl


        # Code for stop trailing loss
        if self.Securities[self.symbol].Invested:

            # If no order exists, send stop-loss
            if not self.Transactions.GetOpenOrders(self.symbol):
                self.stopMarketTicket = self.StopMarketOrder(self.symbol, \
                                        -self.Portfolio[self.symbol].Quantity, \
                                        self.initialStopRisk * self.breakoutlvl)

            # Check if the stocks price is higher than highest Price & trailing stop
price not below initial stop price
            if self.Securities[self.symbol].Close > self.highestPrice and \
                    self.initialStopRisk * self.breakoutlvl <
self.Securities[self.symbol].Close * self.trailingStopRisk:
                # Save the new high to highestPrice
                self.highestPrice = self.Securities[self.symbol].Close
                # Update the stop price
                updateFields = UpdateOrderFields()
                updateFields.StopPrice = self.Securities[self.symbol].Close *
self.trailingStopRisk
                self.stopMarketTicket.Update(updateFields)
```

```
               # Print the new stop price
               self.Debug(updateFields.StopPrice)

           # Plot stop trailing loss
           self.Plot("Data Chart", "Stop Price",
self.stopMarketTicket.Get(OrderField.StopPrice))

# This algorithm was created with the help of TradeOptionsWithME
```