```
# from agent import Agent
# from functions import *
from agent import Agent
# from functions import *
# import sys
import pandas as pd
import keras
from keras.models import Sequential
from keras.models import load_model
from keras.layers import Dense
from keras.optimizers import Adam
import math
import numpy as np
import random
from collections import deque

# if len(sys.argv) != 4:
#   print("Usage: python train.py [stock] [window] [episodes]")
#   exit()


def formatPrice(n):
  return ("₹" if n < 0 else "₹") + "{0:.2f}".format(abs(n))

def getStockDataVec(key):
  vec = []
  lines = open(key + ".csv", "r").read().splitlines()

  for line in lines[1:]:
    if str(line.split(",")[5]) == "null":
      continue
    vec.append(float(line.split(",")[5]))
  return vec

def sigmoid(x):
  return 1 / (1 + math.exp(-x))

# returns an an n-day state representation ending at time t
def getState(data, t, n):
  d = t - n + 1
  block = data[d:t + 1] if d >= 0 else -d * [data[0]] + data[0:t + 1] # pad with t
  res = []
  for i in range(n - 1):
    res.append(sigmoid(block[i + 1] - block[i]))

  return np.array([res])




# stock_name, window_size, episode_count = sys.argv[1], float(sys.argv[2]), int(sys
window_size = 10
stock_name = "SBIN_train"
episode_count = 200
```

```
episode_count = 200
# test_size=3000
agent = Agent(window_size)
data = getStockDataVec(stock_name)
# data = pd.read_csv("^GSPC.csv")
l = len(data) - 1
batch_size = 16


for e in range(episode_count + 1):
  print("Episode " + str(e) + "/" + str(episode_count))
  state = getState(data, 0, window_size + 1)

  total_profit = 0
  agent.inventory = []

  for t in range(l):
    action = agent.act(state)

    # sit
    next_state = getState(data, t + 1, window_size + 1)
    reward = 0

    if action == 1: # buy
      agent.inventory.append(data[t])
      print("Buy: " + formatPrice(data[t]))

    elif action == 2 and len(agent.inventory) > 0: # sell
      bought_price = agent.inventory.pop(0)
      reward = max(data[t] - bought_price, 0)
      total_profit += data[t] - bought_price
      print("Sell: " + formatPrice(data[t]) + " | Profit: " + formatPrice(data[t]

    done = True if t == l - 1 else False
    agent.memory.append((state, action, reward, next_state, done))
    state = next_state

    if done:
      print("--------------------------------")
      print("Total Profit: " + formatPrice(total_profit))
      print("--------------------------------")

    if len(agent.memory) > batch_size:
      agent.expReplay(batch_size)

  if e % 10 == 0:
    agent.model.save("models/model_ep" + str(e))


#############EVALUATE ON TEST SET#######################


  # from agent import Agent
  # from functions import *
  from agent import Agent
  # from functions import *
```

```
  import sys
  import pandas as pd
  import keras
  from keras.models import Sequential
  from keras.models import load_model
  from keras.layers import Dense
  from keras.optimizers import Adam
  import math
  import numpy as np
  import random
  from collections import deque

  def formatPrice(n):
    return ("₹" if n < 0 else "₹") + "{0:.2f}".format(abs(n))


  def sigmoid(x):
    return 1 / (1 + math.exp(-x))

# returns an an n-day state representation ending at time t
def getState(data, t, n):
  d = t - n + 1
  block = data[d:t + 1] if d >= 0 else -d * [data[0]] + data[0:t + 1] # pad with t
  res = []
  for i in range(n - 1):
    res.append(sigmoid(block[i + 1] - block[i]))

  return np.array([res])

  def getStockDataVec(key):
      vec = []
      lines = open(key + ".csv", "r").read().splitlines()

      for line in lines[1:]:
        if str(line.split(",")[5]) == "null":
          continue
        vec.append(float(line.split(",")[5]))
      return vec

  # import keras
  # from keras.models import load_model

  # from agent import Agent
  # # from functions import *
  # import sys

  # if len(sys.argv) != 3:
  #   print "Usage: python evaluate.py [stock] [model]"
  #   exit()

  # stock_name, model_name = sys.argv[1], sys.argv[2]
  model_name = "model_ep0"
  model = load_model("models/" + model_name)
  window_size = model.layers[0].input.shape.as_list()[1]

  agent = Agent(window_size, True, model_name)
```

```
agent = Agent(window_size, True, model_name)
stock_name_test = "RELIANCE_test"
data = getStockDataVec(stock_name_test)
l = len(data) - 1
batch_size = 16
buy_arr = np.zeros((l,1))
sell_arr = np.zeros((l,1))

state = getState(data, 0, window_size + 1)
total_profit = 0
agent.inventory = []

for t in range(l):
  action = agent.act(state)

  # sit
  next_state = getState(data, t + 1, window_size + 1)
  reward = 0

  if action == 1: # buy
    buy_arr[t] = data[t]
    agent.inventory.append(data[t])
    print("Buy: " + formatPrice(data[t]))

  elif action == 2 and len(agent.inventory) > 0: # sell
    sell_arr[t] = data[t]
    bought_price = agent.inventory.pop(0)
    reward = max(data[t] - bought_price, 0)
    total_profit += data[t] - bought_price
    print("Sell: " + formatPrice(data[t]) + " | Profit: " + formatPrice(data[t]


  done = True if t == l - 1 else False
  agent.memory.append((state, action, reward, next_state, done))
  state = next_state

  if done:
    print("--------------------------------")
    print(stock_name_test + " Total Profit: " + formatPrice(total_profit))
    print("--------------------------------")
```

Buy: ₹925.14
Sell: ₹958.79 | Profit: ₹33.65
Buy: ₹959.73
Buy: ₹955.88
Sell: ₹965.37 | Profit: ₹5.63
Buy: ₹969.77
Sell: ₹977.77 | Profit: ₹21.89
Sell: ₹975.15 | Profit: ₹5.39
Buy: ₹967.59
Sell: ₹945.06 | Profit: ₹22.54
Buy: ₹916.00
Buy: ₹901.67
Sell: ₹905.22 | Profit: ₹10.77
Sell: ₹910.12 | Profit: ₹8.45
Buy: ₹1003.77
Sell: ₹984.49 | Profit: ₹19.27
Buy: ₹1007.42
Sell: ₹1019.98 | Profit: ₹12.55
Buy: ₹992.80
Buy: ₹967.59
Buy: ₹960.53
Buy: ₹939.75
Sell: ₹967.09 | Profit: ₹25.70
Sell: ₹955.31 | Profit: ₹12.28
Buy: ₹965.95
Sell: ₹984.65 | Profit: ₹24.12
Sell: ₹959.19 | Profit: ₹19.44
Sell: ₹972.17 | Profit: ₹6.22
Buy: ₹1070.27
Buy: ₹1086.33
Sell: ₹1086.53 | Profit: ₹16.26
Sell: ₹1098.77 | Profit: ₹12.43
Buy: ₹1114.13
Buy: ₹1104.83
Buy: ₹1109.01
Sell: ₹1104.53 | Profit: ₹9.60
Sell: ₹1123.63 | Profit: ₹18.80
Sell: ₹1145.06 | Profit: ₹36.05
Buy: ₹1161.92
Sell: ₹1170.47 | Profit: ₹8.55
Buy: ₹1211.24
Sell: ₹1197.57 | Profit: ₹13.67
Buy: ₹1181.16
Buy: ₹1203.93
Sell: ₹1194.19 | Profit: ₹13.03
Buy: ₹1197.12
Sell: ₹1228.10 | Profit: ₹24.17
Sell: ₹1240.33 | Profit: ₹43.21
Buy: ₹1234.81
Buy: ₹1222.38
Sell: ₹1235.11 | Profit: ₹0.30
Sell: ₹1220.24 | Profit: ₹2.14
Buy: ₹1254.45
Sell: ₹1271.56 | Profit: ₹17.11
Buy: ₹1210.45
Buy: ₹1204.08
Sell: ₹1210.79 | Profit: ₹0.35
Sell: ₹1225.26 | Profit: ₹21.18
Buy: ₹1198.46
Sell: ₹1116.07 | Profit: ₹82.39
Buy: ₹1043.07
Sell: ₹1103.29 | Profit: ₹60.22

Sell: ₹1105.29 | Profit: ₹60.22
Buy: ₹1096.03
Buy: ₹1081.81
Buy: ₹1120.35
Sell: ₹1133.47 | Profit: ₹37.44
Sell: ₹1157.39 | Profit: ₹75.58
Sell: ₹1144.96 | Profit: ₹24.61
Buy: ₹1095.23
Sell: ₹1056.80 | Profit: ₹38.44
Buy: ₹1039.15
Sell: ₹1081.81 | Profit: ₹42.66
Buy: ₹1087.43
Buy: ₹1074.05
Sell: ₹1093.40 | Profit: ₹5.97
Sell: ₹1091.90 | Profit: ₹17.85
Buy: ₹1121.19
Sell: ₹1143.67 | Profit: ₹22.48
Buy: ₹1106.77
Sell: ₹1096.78 | Profit: ₹9.99
Buy: ₹1161.12
Buy: ₹1150.08
Sell: ₹1145.95 | Profit: ₹15.17
Sell: ₹1148.79 | Profit: ₹1.29
Buy: ₹1127.56
Buy: ₹1084.25
Buy: ₹1091.51
Sell: ₹1104.38 | Profit: ₹23.17
Sell: ₹1100.95 | Profit: ₹16.71
Sell: ₹1106.07 | Profit: ₹14.57
Buy: ₹1122.24
Sell: ₹1094.14 | Profit: ₹28.09
Buy: ₹1083.15
Sell: ₹1092.30 | Profit: ₹9.15
Buy: ₹1115.07
Buy: ₹1114.83
Sell: ₹1100.31 | Profit: ₹14.77
Sell: ₹1086.73 | Profit: ₹28.09
Buy: ₹1104.63
Sell: ₹1101.40 | Profit: ₹3.23
Buy: ₹1128.20
Buy: ₹1177.83
Sell: ₹1230.88 | Profit: ₹102.68
Sell: ₹1228.35 | Profit: ₹50.52
Buy: ₹1222.78
Buy: ₹1203.98
Buy: ₹1189.11
Sell: ₹1220.39 | Profit: ₹2.39
Sell: ₹1243.07 | Profit: ₹39.08
Sell: ₹1283.79 | Profit: ₹94.68
Buy: ₹1303.03
Sell: ₹1283.29 | Profit: ₹19.74
Buy: ₹1237.60
Sell: ₹1213.38 | Profit: ₹24.22
Buy: ₹1209.40
Sell: ₹1227.55 | Profit: ₹18.15
Buy: ₹1225.51
Buy: ₹1213.53
Sell: ₹1216.76 | Profit: ₹8.75
Sell: ₹1224.27 | Profit: ₹10.74
Buy: ₹1257.83
Sell: ₹1263.25 | Profit: ₹5.42
Buy: ₹1296.92

Sell: ₹1324.02 | Profit: ₹27.10
Buy: ₹1314.37
Buy: ₹1342.61
Sell: ₹1368.97 | Profit: ₹54.60
Sell: ₹1367.88 | Profit: ₹25.26
Buy: ₹1334.36
Buy: ₹1317.16
Sell: ₹1359.72 | Profit: ₹25.36
Sell: ₹1341.82 | Profit: ₹24.66
Buy: ₹1382.05
Buy: ₹1367.63
Buy: ₹1345.60
Sell: ₹1346.44 | Profit: ₹35.60
Sell: ₹1321.93 | Profit: ₹45.70
Buy: ₹1327.10
Sell: ₹1324.07 | Profit: ₹21.53
Buy: ₹1339.38
Sell: ₹1335.70 | Profit: ₹8.60
Buy: ₹1332.77
Sell: ₹1336.35 | Profit: ₹3.03
Sell: ₹1378.32 | Profit: ₹45.55
Buy: ₹1356.34
Sell: ₹1381.85 | Profit: ₹25.51
Buy: ₹1377.27
Sell: ₹1336.10 | Profit: ₹41.17
Buy: ₹1292.29
Sell: ₹1249.53 | Profit: ₹42.76
Buy: ₹1225.26
Sell: ₹1253.51 | Profit: ₹28.24
Buy: ₹1322.43
Sell: ₹1322.82 | Profit: ₹0.40
Buy: ₹1352.71
Sell: ₹1344.21 | Profit: ₹8.50
Buy: ₹1320.04
Sell: ₹1307.66 | Profit: ₹12.38
Buy: ₹1310.29
Buy: ₹1275.24
Sell: ₹1273.95 | Profit: ₹36.35
Sell: ₹1270.32 | Profit: ₹4.92
Buy: ₹1255.45
Buy: ₹1288.71
Sell: ₹1287.02 | Profit: ₹31.58
Sell: ₹1267.13 | Profit: ₹21.58
Buy: ₹1246.20
Sell: ₹1261.86 | Profit: ₹15.66
Buy: ₹1245.15
Sell: ₹1273.05 | Profit: ₹27.90
Buy: ₹1273.45
Buy: ₹1269.07
Sell: ₹1285.88 | Profit: ₹12.43
Sell: ₹1274.79 | Profit: ₹5.72
Buy: ₹1242.12
Buy: ₹1273.45
Sell: ₹1266.54 | Profit: ₹24.41
Buy: ₹1252.17
Sell: ₹1224.72 | Profit: ₹48.73
Buy: ₹1207.12
Sell: ₹1204.28 | Profit: ₹47.88
Sell: ₹1174.40 | Profit: ₹32.72
Buy: ₹1278.00
Sell: ₹1292.60 | Profit: ₹14.60
Buy: ₹1275.85

```
Buy: ₹1275.85
Sell: ₹1266.80 | Profit: ₹9.05
Buy: ₹1274.85
Sell: ₹1263.30 | Profit: ₹11.55
Buy: ₹1241.75
Sell: ₹1248.55 | Profit: ₹6.80
Buy: ₹1198.60
Sell: ₹1222.50 | Profit: ₹23.90
Buy: ₹1278.70
Sell: ₹1279.55 | Profit: ₹0.85
Buy: ₹1296.80
Sell: ₹1309.05 | Profit: ₹12.25
Buy: ₹1358.00
Sell: ₹1364.15 | Profit: ₹6.15
--------------------------------
RELIANCE_test Total Profit: ₹668.79
--------------------------------
```

```
buy_arr.size
```

441

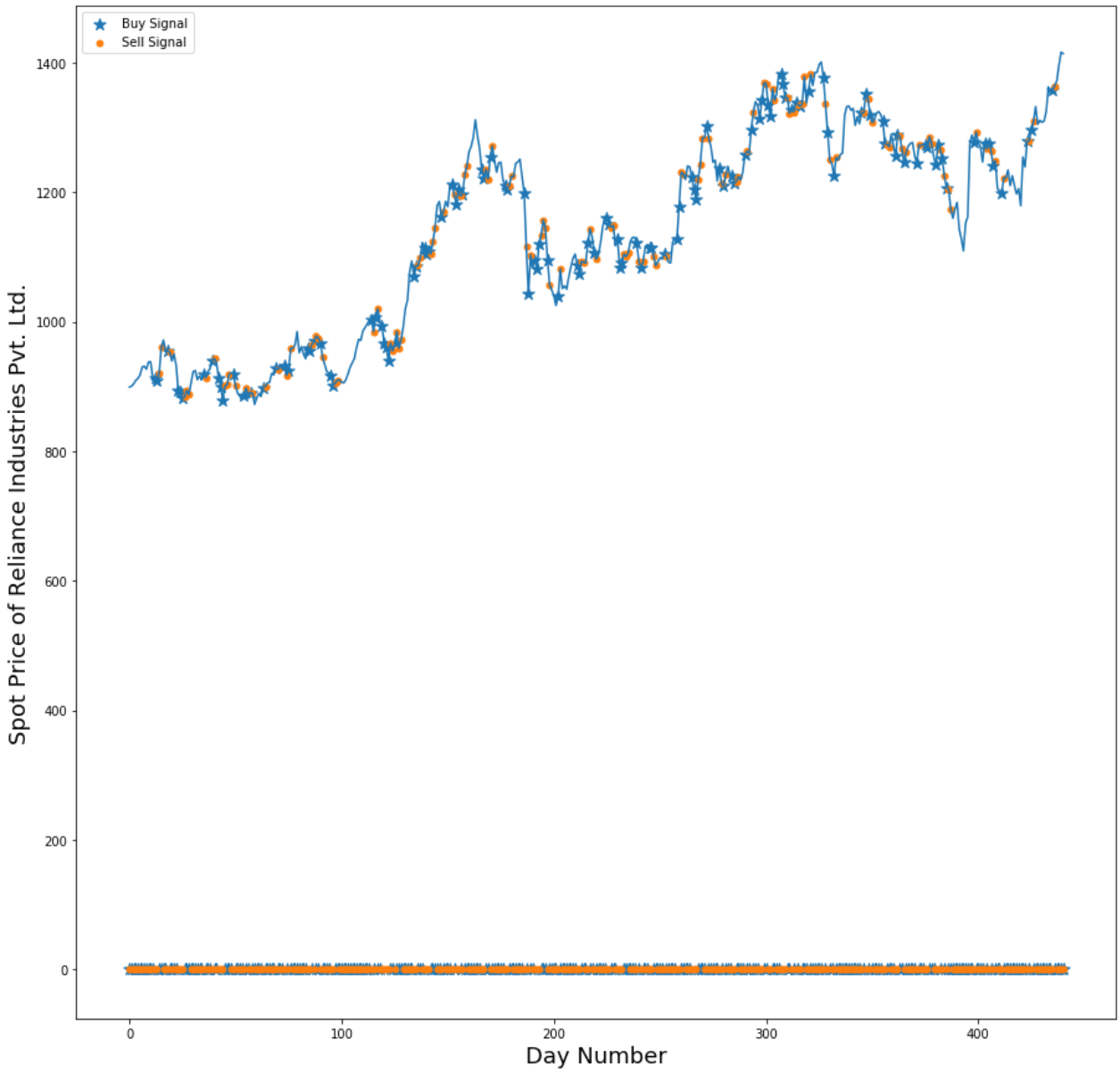
```
sell_arr.size
```

441

```
import matplotlib.pyplot as plt


stock_name_test = "RELIANCE_test"
data = getStockDataVec(stock_name_test)
arr = range(l)
plt.plot(arr,data[0:441])
plt.scatter(arr,buy_arr,marker='*',s=100,label='Buy Signal')
plt.scatter(arr,sell_arr,marker='.',s=100,label='Sell Signal')
plt.rcParams['figure.figsize'] = [15, 15]
plt.legend()
plt.xlabel('Day Number',fontsize=18)
plt.ylabel('Spot Price of Reliance Industries Pvt. Ltd.',fontsize=18)
```

```
plt.savefig('main.png')
from google.colab import files
files.download('main.png')
```




```
buy_arr[buy_arr>0].size
```

116

```
buy_nonzero=buy_arr[buy_arr>0]
```

```
array([ 913.773438,  909.029175,  954.693115,  893.857178,  891.6828  ,
        881.749329,  918.567139,  939.471802,  913.378052,  900.281738,
        879.426636,  918.27063 ,  886.493652,  890.644897,  897.662537,
        927.956909,  931.366882,  925.140015,  959.733887,  955.87915 ,
        969.766174,  967.591675,  915.997375,  901.665588, 1003.767029,
       1007.424072,  992.795837,  967.591675,  960.530762,  939.745911,
        965.950745, 1070.272949, 1086.334106, 1114.130249, 1104.831665,
       1109.008545, 1161.915527, 1211.24231 , 1181.158936, 1203.932861,
       1197.120605, 1234.81189 , 1222.380737, 1254.453125, 1210.446777,
       1204.082031, 1198.463135, 1043.073608, 1096.030396, 1081.809204,
       1120.345825, 1095.234863, 1039.145508, 1087.427979, 1074.052124,
       1121.191162, 1106.770996, 1161.119995, 1150.081055, 1127.555908,
       1084.245728, 1091.505493, 1122.235229, 1083.151733, 1115.074951,
       1114.826294, 1104.632813, 1128.202148, 1177.827393, 1222.778564,
       1203.982666, 1189.114868, 1303.034058, 1237.596436, 1209.402588,
       1225.513428, 1213.529785, 1257.834473, 1296.917969, 1314.371338,
       1342.61499 , 1334.360596, 1317.155762, 1382.046509, 1367.626343,
       1345.598389, 1327.100708, 1339.382813, 1332.769409, 1356.338867,
       1377.272949, 1292.293457, 1225.264771, 1322.426636, 1352.708984,
       1320.039917, 1310.293945, 1275.238037, 1255.447632, 1288.713379,
       1246.19873 , 1245.154663, 1273.447876, 1269.072144, 1242.121338,
       1273.447876, 1252.165771, 1207.115356, 1278.      , 1275.849976,
       1274.849976, 1241.75    , 1198.599976, 1278.699951, 1296.800049,
       1358.      ])
```

```
import numpy as np
from PIL import Image

im = Image.open('download (2).png')
im = im.convert('RGBA')
data = np.array(im)
# just use the rgb values for comparison
rgb = data[:,:,:3]
color = [246, 213, 139]   # Original value
black = [0,0,0, 255]
white = [255,255,255,255]
mask = np.all(rgb == color, axis = -1)
# change all pixels that match color to white
data[mask] = white

# change all pixels that don't match color to black
##data[np.logical_not(mask)] = black
new_im = Image.fromarray(data)
new_im.save('download (2).png')
```

```
data.size
```

```
1401600
```

```
!ls
```

```
agent.py  models       RELIANCE_test.csv   sample_data
main.png    pycache    RELIANCE_train.csv
im = Image.open('main.png')


  # from agent import Agent
  # from functions import *
  from agent import Agent
  # from functions import *
  import sys
  import pandas as pd
  import keras
  from keras.models import Sequential
  from keras.models import load_model
  from keras.layers import Dense
  from keras.optimizers import Adam
  import math
  import numpy as np
  import random
  from collections import deque

  def formatPrice(n):
    return ("₹" if n < 0 else "₹") + "{0:.2f}".format(abs(n))


  def sigmoid(x):
    return 1 / (1 + math.exp(-x))

# returns an an n-day state representation ending at time t
def getState(data, t, n):
  d = t - n + 1
  block = data[d:t + 1] if d >= 0 else -d * [data[0]] + data[0:t + 1] # pad with t
  res = []
  for i in range(n - 1):
    res.append(sigmoid(block[i + 1] - block[i]))

  return np.array([res])

  def getStockDataVec(key):
      vec = []
      lines = open(key + ".csv", "r").read().splitlines()

      for line in lines[1:]:
        if str(line.split(",")[5]) == "null":
          continue
        vec.append(float(line.split(",")[5]))
      return vec

  # import keras
  # from keras.models import load_model

  # from agent import Agent
  # # from functions import *
  # import sys
```

```
# if len(sys.argv) != 3:
#   print "Usage: python evaluate.py [stock] [model]"
#   exit()

# stock_name, model_name = sys.argv[1], sys.argv[2]
model_name = "model_ep0"
model = load_model("models/" + model_name)
window_size = model.layers[0].input.shape.as_list()[1]

agent = Agent(window_size, True, model_name)
stock_name_test = "SBIN_test"
data = getStockDataVec(stock_name_test)
l = len(data) - 1
batch_size = 8
buy_arr = np.zeros((l,1))
sell_arr = np.zeros((l,1))

state = getState(data, 0, window_size + 1)
total_profit = 0
agent.inventory = []

for t in range(l):
  action = agent.act(state)

  # sit
  next_state = getState(data, t + 1, window_size + 1)
  reward = 0

  if action == 1: # buy
    buy_arr[t] = data[t]
    agent.inventory.append(data[t])
    print("Buy: " + formatPrice(data[t]))

  elif action == 2 and len(agent.inventory) > 0: # sell
    sell_arr[t] = data[t]
    bought_price = agent.inventory.pop(0)
    reward = max(data[t] - bought_price, 0)
    total_profit += data[t] - bought_price
    print("Sell: " + formatPrice(data[t]) + " | Profit: " + formatPrice(data[t]


  done = True if t == l - 1 else False
  agent.memory.append((state, action, reward, next_state, done))
  state = next_state

  if done:
    print("--------------------------------")
    print(stock_name_test + " Total Profit: " + formatPrice(total_profit))
    print("--------------------------------")
```