In [1]:
```
import pandas as pd

df = pd.read_csv('MSFT.csv')

df
```

Out[1]:

| | Date | Open | High | Low | Close | Adj Close | Volume |
|---|---|---|---|---|---|---|---|
| **0** | 1986-03-13 | 0.088542 | 0.101563 | 0.088542 | 0.097222 | 0.060524 | 1031788800 |
| **1** | 1986-03-14 | 0.097222 | 0.102431 | 0.097222 | 0.100694 | 0.062686 | 308160000 |
| **2** | 1986-03-17 | 0.100694 | 0.103299 | 0.100694 | 0.102431 | 0.063767 | 133171200 |
| **3** | 1986-03-18 | 0.102431 | 0.103299 | 0.098958 | 0.099826 | 0.062145 | 67766400 |
| **4** | 1986-03-19 | 0.099826 | 0.100694 | 0.097222 | 0.098090 | 0.061065 | 47894400 |
| **...** | ... | ... | ... | ... | ... | ... | ... |
| **9420** | 2023-07-31 | 336.920013 | 337.700012 | 333.359985 | 335.920013 | 335.920013 | 25446000 |
| **9421** | 2023-08-01 | 335.190002 | 338.540009 | 333.700012 | 336.339996 | 336.339996 | 18311900 |
| **9422** | 2023-08-02 | 333.630005 | 333.630005 | 326.359985 | 327.500000 | 327.500000 | 27761300 |
| **9423** | 2023-08-03 | 326.000000 | 329.880005 | 325.950012 | 326.660004 | 326.660004 | 18253700 |
| **9424** | 2023-08-04 | 331.880005 | 335.140015 | 327.239990 | 327.779999 | 327.779999 | 23727700 |

9425 rows × 7 columns

In [2]:
```
df = df[['Date', 'Close']]

df
```

Out[2]:

| | Date | Close |
|---|---|---|
| **0** | 1986-03-13 | 0.097222 |
| **1** | 1986-03-14 | 0.100694 |
| **2** | 1986-03-17 | 0.102431 |
| **3** | 1986-03-18 | 0.099826 |
| **4** | 1986-03-19 | 0.098090 |
| **...** | ... | ... |
| **9420** | 2023-07-31 | 335.920013 |
| **9421** | 2023-08-01 | 336.339996 |
| **9422** | 2023-08-02 | 327.500000 |
| **9423** | 2023-08-03 | 326.660004 |
| **9424** | 2023-08-04 | 327.779999 |

9425 rows × 2 columns

In [3]:
```
df['Date']
```

Out[3]:
```
0       1986-03-13
1       1986-03-14
2       1986-03-17
3       1986-03-18
4       1986-03-19
           ...
9420    2023-07-31
9421    2023-08-01
9422    2023-08-02
9423    2023-08-03
9424    2023-08-04
Name: Date, Length: 9425, dtype: object
```

In [4]:
```
import datetime

def str_to_datetime(s):
  split = s.split('-')
  year, month, day = int(split[0]), int(split[1]), int(split[2])
  return datetime.datetime(year=year, month=month, day=day)

datetime_object = str_to_datetime('1986-03-19')
datetime_object
```

Out[4]:
```
datetime.datetime(1986, 3, 19, 0, 0)
```

In [5]:
```
df
```

Out[5]:

| | Date | Close |
|---|---|---|
| **0** | 1986-03-13 | 0.097222 |
| **1** | 1986-03-14 | 0.100694 |
| **2** | 1986-03-17 | 0.102431 |
| **3** | 1986-03-18 | 0.099826 |
| **4** | 1986-03-19 | 0.098090 |
| **...** | ... | ... |
| **9420** | 2023-07-31 | 335.920013 |
| **9421** | 2023-08-01 | 336.339996 |
| **9422** | 2023-08-02 | 327.500000 |
| **9423** | 2023-08-03 | 326.660004 |
| **9424** | 2023-08-04 | 327.779999 |

9425 rows × 2 columns

In [6]:
```
df['Date'] = df['Date'].apply(str_to_datetime)
df['Date']
```

```
C:\Users\aruna pc\AppData\Local\Temp\ipykernel_15712\2565755782.py:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy (https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)
  df['Date'] = df['Date'].apply(str_to_datetime)
```

Out[6]:
```
0      1986-03-13
1      1986-03-14
2      1986-03-17
3      1986-03-18
4      1986-03-19
          ...
9420   2023-07-31
9421   2023-08-01
9422   2023-08-02
9423   2023-08-03
9424   2023-08-04
Name: Date, Length: 9425, dtype: datetime64[ns]
```

In [7]:
```
df.index = df.pop('Date')
df
```

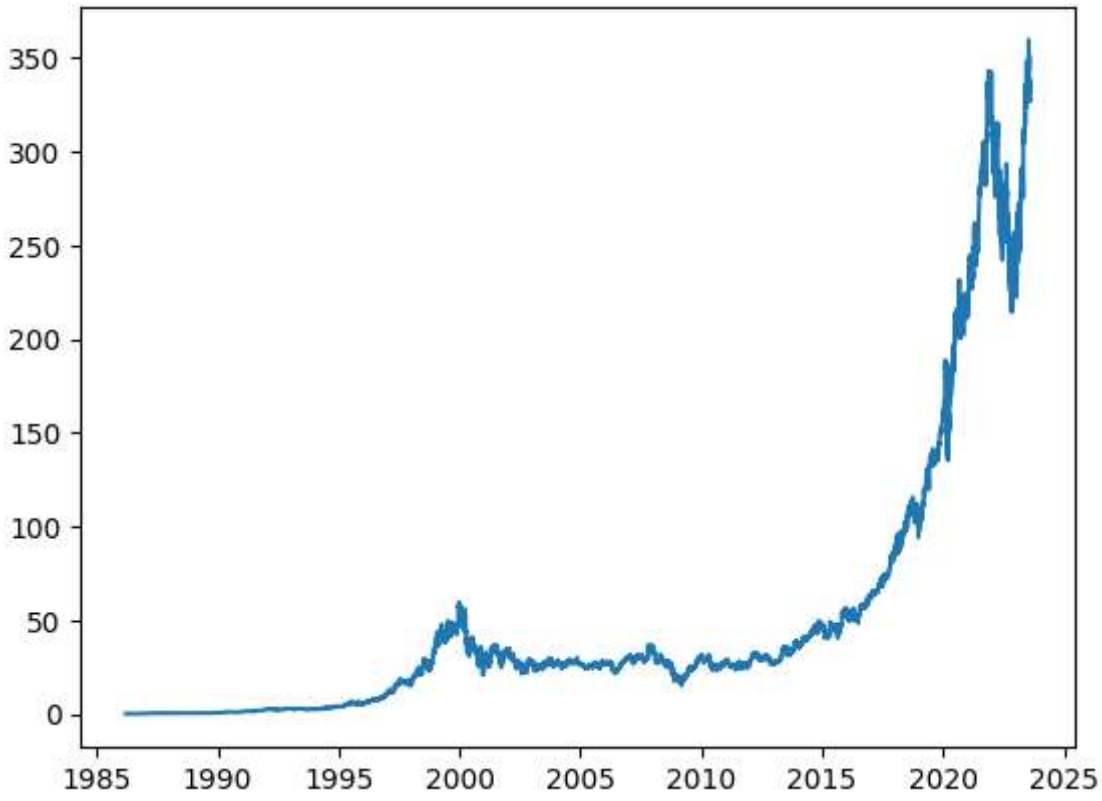
Out[7]:

| Date | Close |
|---|---|
| **1986-03-13** | 0.097222 |
| **1986-03-14** | 0.100694 |
| **1986-03-17** | 0.102431 |
| **1986-03-18** | 0.099826 |
| **1986-03-19** | 0.098090 |
| **...** | ... |
| **2023-07-31** | 335.920013 |
| **2023-08-01** | 336.339996 |
| **2023-08-02** | 327.500000 |
| **2023-08-03** | 326.660004 |
| **2023-08-04** | 327.779999 |

9425 rows × 1 columns

In [8]:
```
import matplotlib.pyplot as plt

plt.plot(df.index, df['Close'])
```

Out[8]:
```
[<matplotlib.lines.Line2D at 0x1ed23d44e50>]
```



In [9]:
```
import numpy as np

def df_to_windowed_df(dataframe, first_date_str, last_date_str, n=3):
  first_date = str_to_datetime(first_date_str)
  last_date  = str_to_datetime(last_date_str)

  target_date = first_date

  dates = []
  X, Y = [], []

  last_time = False
  while True:
    df_subset = dataframe.loc[:target_date].tail(n+1)

    if len(df_subset) != n+1:
      print(f'Error: Window of size {n} is too large for date {target_date}')
      return

    values = df_subset['Close'].to_numpy()
    x, y = values[:-1], values[-1]

    dates.append(target_date)
    X.append(x)
    Y.append(y)

    next_week = dataframe.loc[target_date:target_date+datetime.timedelta(days=
    next_datetime_str = str(next_week.head(2).tail(1).index.values[0])
    next_date_str = next_datetime_str.split('T')[0]
    year_month_day = next_date_str.split('-')
    year, month, day = year_month_day
    next_date = datetime.datetime(day=int(day), month=int(month), year=int(yea

    if last_time:
      break

    target_date = next_date

    if target_date == last_date:
      last_time = True

  ret_df = pd.DataFrame({})
  ret_df['Target Date'] = dates

  X = np.array(X)
  for i in range(0, n):
    X[:, i]
    ret_df[f'Target-{n-i}'] = X[:, i]

  ret_df['Target'] = Y

  return ret_df

# Start day second time around: '2021-03-25'
windowed_df = df_to_windowed_df(df,
                                '2022-08-04',
                                '2023-08-04',
```

```
                                    n=3)
windowed_df
```

Out[9]:

| | Target Date | Target-3 | Target-2 | Target-1 | Target |
|---|---|---|---|---|---|
| **0** | 2022-08-04 | 278.010010 | 274.820007 | 282.470001 | 283.649994 |
| **1** | 2022-08-05 | 274.820007 | 282.470001 | 283.649994 | 282.910004 |
| **2** | 2022-08-08 | 282.470001 | 283.649994 | 282.910004 | 280.320007 |
| **3** | 2022-08-09 | 283.649994 | 282.910004 | 280.320007 | 282.299988 |
| **4** | 2022-08-10 | 282.910004 | 280.320007 | 282.299988 | 289.160004 |
| **...** | ... | ... | ... | ... | ... |
| **247** | 2023-07-31 | 337.769989 | 330.720001 | 338.369995 | 335.920013 |
| **248** | 2023-08-01 | 330.720001 | 338.369995 | 335.920013 | 336.339996 |
| **249** | 2023-08-02 | 338.369995 | 335.920013 | 336.339996 | 327.500000 |
| **250** | 2023-08-03 | 335.920013 | 336.339996 | 327.500000 | 326.660004 |
| **251** | 2023-08-04 | 336.339996 | 327.500000 | 326.660004 | 327.779999 |

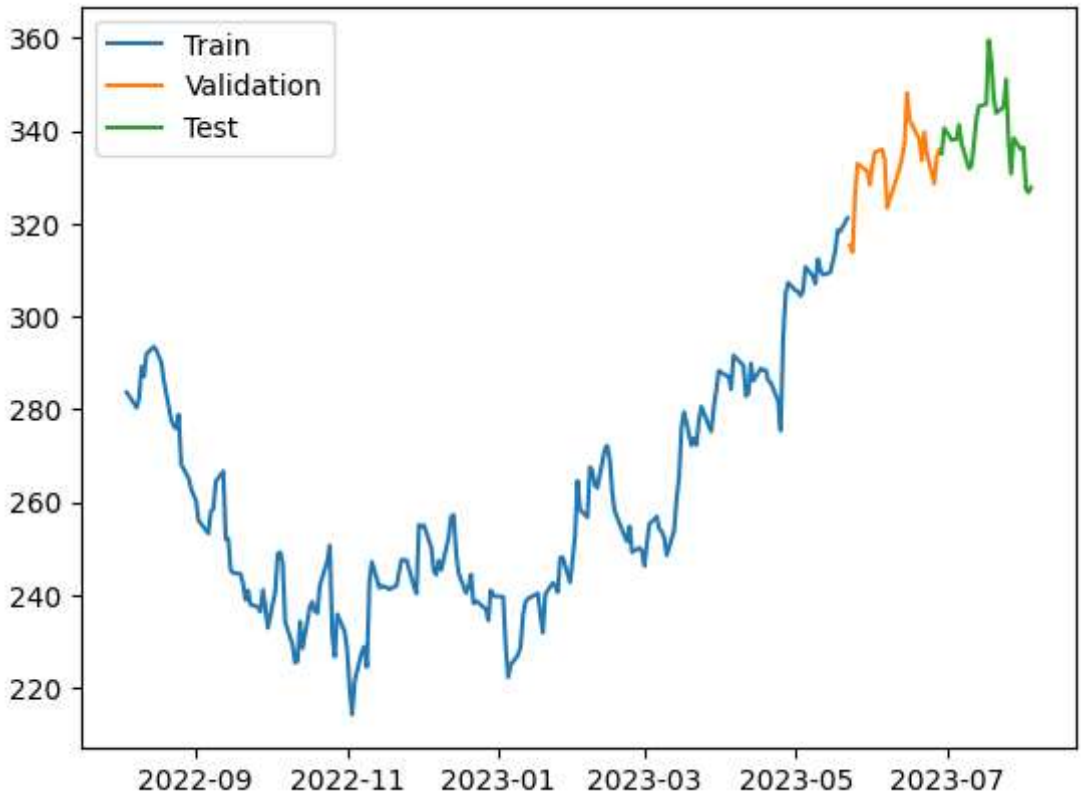
252 rows × 5 columns

In [12]:
```
def windowed_df_to_date_X_y(windowed_dataframe):
  df_as_np = windowed_dataframe.to_numpy()

  dates = df_as_np[:, 0]

  middle_matrix = df_as_np[:, 1:-1]
  X = middle_matrix.reshape((len(dates), middle_matrix.shape[1], 1))

  Y = df_as_np[:, -1]

  return dates, X.astype(np.float32), Y.astype(np.float32)

dates, X, y = windowed_df_to_date_X_y(windowed_df)

dates.shape, X.shape, y.shape
```

Out[12]:
```
((252,), (252, 3, 1), (252,))
```

In [13]:
```
q_80 = int(len(dates) * .8)
q_90 = int(len(dates) * .9)

dates_train, X_train, y_train = dates[:q_80], X[:q_80], y[:q_80]

dates_val, X_val, y_val = dates[q_80:q_90], X[q_80:q_90], y[q_80:q_90]
dates_test, X_test, y_test = dates[q_90:], X[q_90:], y[q_90:]

plt.plot(dates_train, y_train)
plt.plot(dates_val, y_val)
plt.plot(dates_test, y_test)

plt.legend(['Train', 'Validation', 'Test'])
```

Out[13]:
```
<matplotlib.legend.Legend at 0x1ed23deaad0>
```



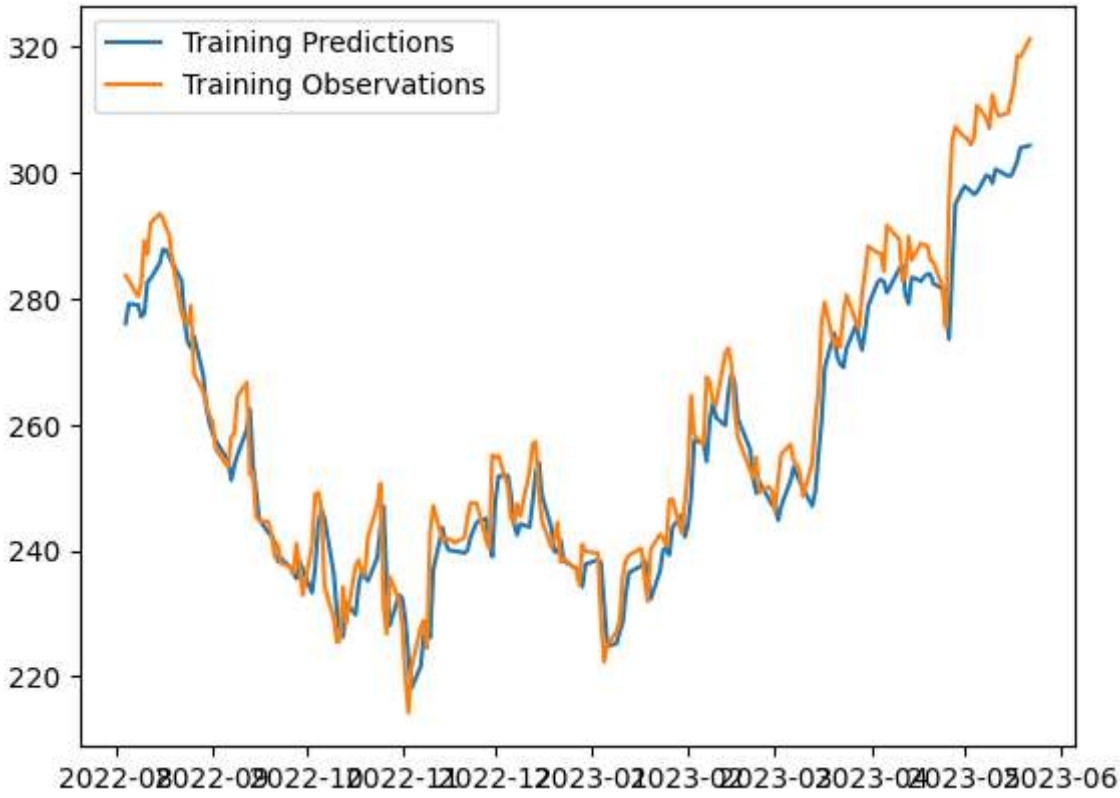
In [14]:
```
from tensorflow.keras.models import Sequential
from tensorflow.keras.optimizers import Adam
from tensorflow.keras import layers

model = Sequential([layers.Input((3, 1)),
                    layers.LSTM(64),
                    layers.Dense(32, activation='relu'),
                    layers.Dense(32, activation='relu'),
                    layers.Dense(1)])

model.compile(loss='mse',
              optimizer=Adam(learning_rate=0.001),
              metrics=['mean_absolute_error'])

model.fit(X_train, y_train, validation_data=(X_val, y_val), epochs=100)
```

```
5.9637
Epoch 72/100
7/7 [==============================] - 0s 10ms/step - loss: 87.6449 - mean
_absolute_error: 7.2736 - val_loss: 1310.3844 - val_mean_absolute_error: 3
5.5898
Epoch 73/100
7/7 [==============================] - 0s 9ms/step - loss: 72.9369 - mean_
absolute_error: 6.4132 - val_loss: 1193.4985 - val_mean_absolute_error: 3
3.8937
Epoch 74/100
7/7 [==============================] - 0s 11ms/step - loss: 72.5064 - mean
_absolute_error: 6.5251 - val_loss: 1098.6882 - val_mean_absolute_error: 3
2.4558
Epoch 75/100
7/7 [==============================] - 0s 10ms/step - loss: 68.3693 - mean
_absolute_error: 6.4615 - val_loss: 1112.8199 - val_mean_absolute_error: 3
2.7080
Epoch 76/100
7/7 [==============================] - 0s 8ms/step - loss: 59.0438 - mean_
absolute_error: 5.8998 - val_loss: 1035.0845 - val_mean_absolute_error: 3
```

In [15]:
```
train_predictions = model.predict(X_train).flatten()

plt.plot(dates_train, train_predictions)
plt.plot(dates_train, y_train)
plt.legend(['Training Predictions', 'Training Observations'])
```

```
7/7 [==============================] - 0s 2ms/step
```

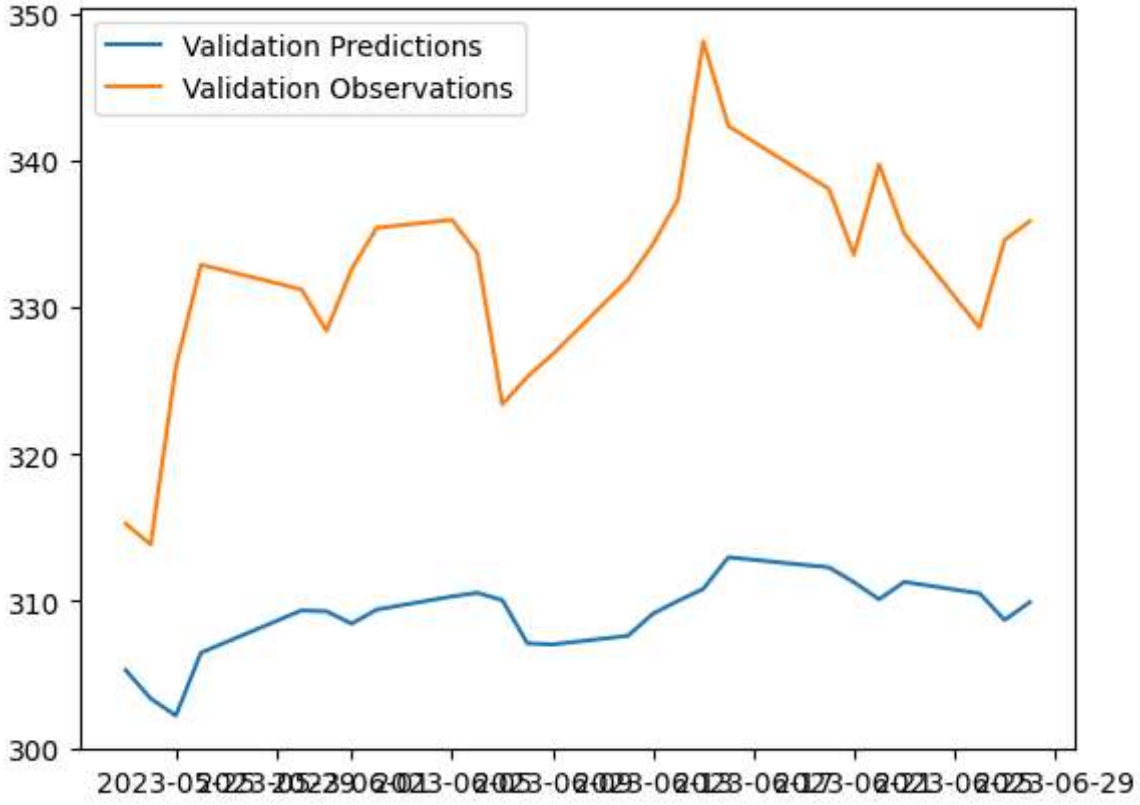
Out[15]:
```
<matplotlib.legend.Legend at 0x1ed532f23d0>
```



In [16]:
```
val_predictions = model.predict(X_val).flatten()

plt.plot(dates_val, val_predictions)
plt.plot(dates_val, y_val)
plt.legend(['Validation Predictions', 'Validation Observations'])
```

```
1/1 [==============================] - 0s 23ms/step
```

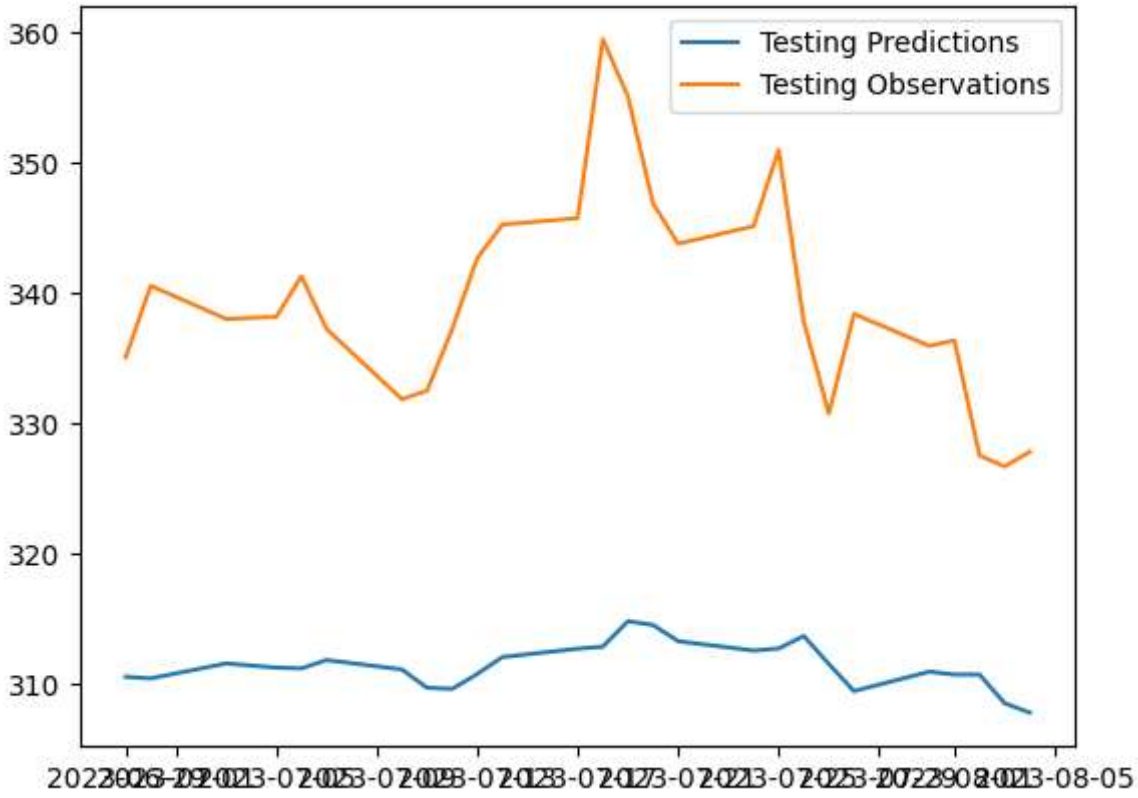
Out[16]:
```
<matplotlib.legend.Legend at 0x1ed53227290>
```



In [17]:
```
test_predictions = model.predict(X_test).flatten()

plt.plot(dates_test, test_predictions)
plt.plot(dates_test, y_test)
plt.legend(['Testing Predictions', 'Testing Observations'])
```

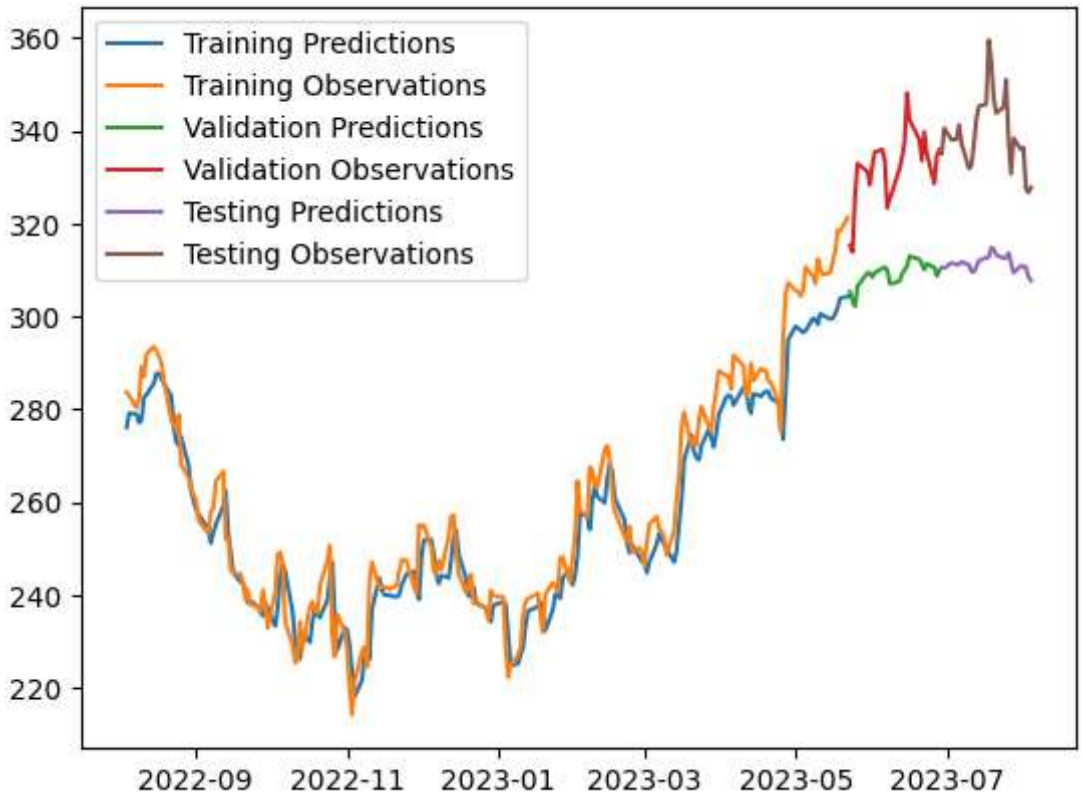
```
1/1 [==============================] - 0s 25ms/step
```

Out[17]:
```
<matplotlib.legend.Legend at 0x1ed5438d2d0>
```



In [18]:
```
plt.plot(dates_train, train_predictions)
plt.plot(dates_train, y_train)
plt.plot(dates_val, val_predictions)
plt.plot(dates_val, y_val)
plt.plot(dates_test, test_predictions)
plt.plot(dates_test, y_test)
plt.legend(['Training Predictions',
            'Training Observations',
            'Validation Predictions',
            'Validation Observations',
            'Testing Predictions',
            'Testing Observations'])
```

Out[18]: <matplotlib.legend.Legend at 0x1ed53262ad0>

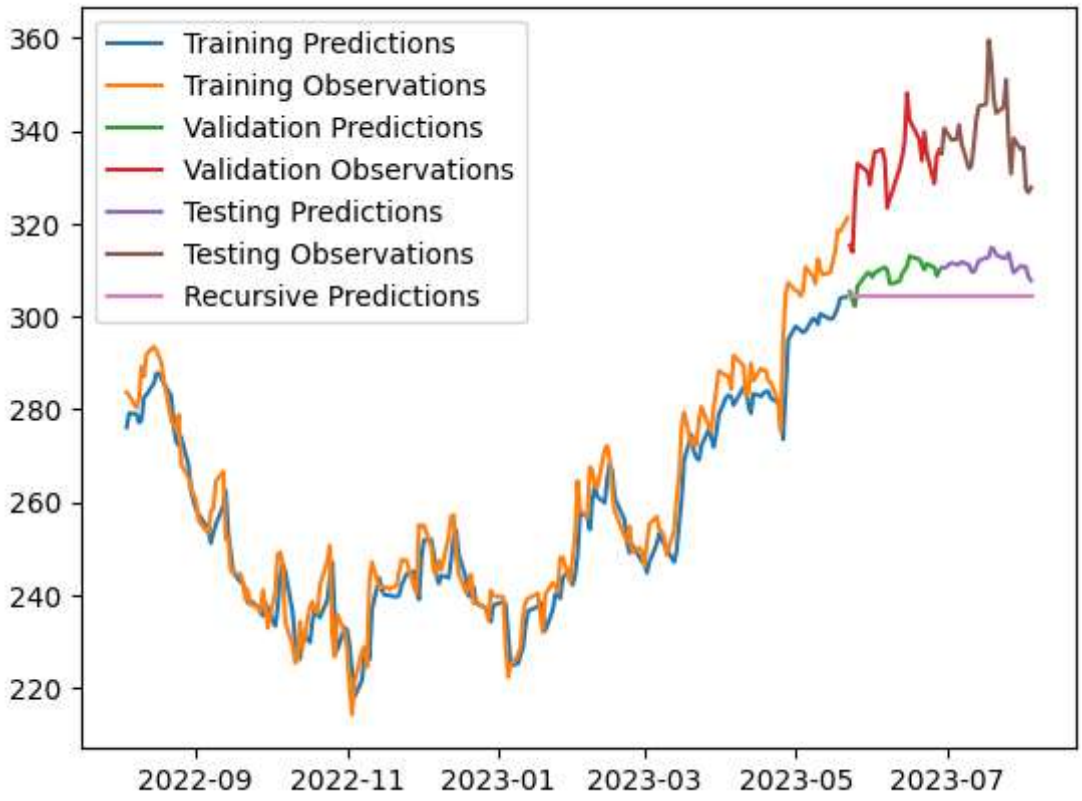


In [19]:
```
from copy import deepcopy

recursive_predictions = []
recursive_dates = np.concatenate([dates_val, dates_test])

for target_date in recursive_dates:
  last_window = deepcopy(X_train[-1])
  next_prediction = model.predict(np.array([last_window])).flatten()
  recursive_predictions.append(next_prediction)
  last_window[-1] = next_prediction
```

```
1/1 [==============================] - 0s 24ms/step
1/1 [==============================] - 0s 19ms/step
1/1 [==============================] - 0s 22ms/step
1/1 [==============================] - 0s 27ms/step
1/1 [==============================] - 0s 20ms/step
1/1 [==============================] - 0s 23ms/step
1/1 [==============================] - 0s 22ms/step
1/1 [==============================] - 0s 22ms/step
1/1 [==============================] - 0s 26ms/step
1/1 [==============================] - 0s 20ms/step
1/1 [==============================] - 0s 20ms/step
1/1 [==============================] - 0s 21ms/step
1/1 [==============================] - 0s 22ms/step
1/1 [==============================] - 0s 21ms/step
1/1 [==============================] - 0s 25ms/step
1/1 [==============================] - 0s 20ms/step
1/1 [==============================] - 0s 26ms/step
1/1 [==============================] - 0s 20ms/step
1/1 [==============================] - 0s 21ms/step
1/1 [==============================] - 0s 21ms/step
1/1 [==============================] - 0s 19ms/step
1/1 [==============================] - 0s 21ms/step
1/1 [==============================] - 0s 26ms/step
1/1 [==============================] - 0s 27ms/step
1/1 [==============================] - 0s 22ms/step
1/1 [==============================] - 0s 23ms/step
1/1 [==============================] - 0s 22ms/step
1/1 [==============================] - 0s 20ms/step
1/1 [==============================] - 0s 21ms/step
1/1 [==============================] - 0s 23ms/step
1/1 [==============================] - 0s 22ms/step
1/1 [==============================] - 0s 23ms/step
1/1 [==============================] - 0s 20ms/step
1/1 [==============================] - 0s 23ms/step
1/1 [==============================] - 0s 19ms/step
1/1 [==============================] - 0s 22ms/step
1/1 [==============================] - 0s 22ms/step
1/1 [==============================] - 0s 22ms/step
1/1 [==============================] - 0s 20ms/step
1/1 [==============================] - 0s 22ms/step
1/1 [==============================] - 0s 22ms/step
1/1 [==============================] - 0s 22ms/step
1/1 [==============================] - 0s 22ms/step
1/1 [==============================] - 0s 26ms/step
1/1 [==============================] - 0s 20ms/step
1/1 [==============================] - 0s 25ms/step
1/1 [==============================] - 0s 20ms/step
1/1 [==============================] - 0s 24ms/step
1/1 [==============================] - 0s 27ms/step
1/1 [==============================] - 0s 23ms/step
1/1 [==============================] - 0s 19ms/step
```

In [20]:
```
plt.plot(dates_train, train_predictions)
plt.plot(dates_train, y_train)
plt.plot(dates_val, val_predictions)
plt.plot(dates_val, y_val)
plt.plot(dates_test, test_predictions)
plt.plot(dates_test, y_test)
plt.plot(recursive_dates, recursive_predictions)
plt.legend(['Training Predictions',
            'Training Observations',
            'Validation Predictions',
            'Validation Observations',
            'Testing Predictions',
            'Testing Observations',
            'Recursive Predictions'])
```

Out[20]: <matplotlib.legend.Legend at 0x1ed546b5ed0>




In [ ]: