In [1]:

```python
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

In [26]:

```python
data = pd.read_csv("NFLX.csv")
df = pd.DataFrame(data)
```

In [27]:

```python
df.head()
```

Out[27]:

	Date	Open	High	Low	Close	Adj Close	Volume
0	2018-02-05	262.000000	267.899994	250.029999	254.259995	254.259995	11896100
1	2018-02-06	247.699997	266.700012	245.000000	265.720001	265.720001	12595800
2	2018-02-07	266.579987	272.450012	264.329987	264.559998	264.559998	8981500
3	2018-02-08	267.079987	267.619995	250.000000	250.100006	250.100006	9306700
4	2018-02-09	253.850006	255.800003	236.110001	249.470001	249.470001	16906900

In [28]:

```python
df.describe()
```

Out[28]:

	Open	High	Low	Close	Adj Close	Volume
count	1009.000000	1009.000000	1009.000000	1009.000000	1009.000000	1.009000e+03
mean	419.059673	425.320703	412.374044	419.000733	419.000733	7.570685e+06
std	108.537532	109.262960	107.555867	108.289999	108.289999	5.465535e+06
min	233.919998	250.649994	231.229996	233.880005	233.880005	1.144000e+06
25%	331.489990	336.299988	326.000000	331.619995	331.619995	4.091900e+06
50%	377.769989	383.010010	370.880005	378.670013	378.670013	5.934500e+06
75%	509.130005	515.630005	502.529999	509.079987	509.079987	9.322400e+06
max	692.349976	700.989990	686.090027	691.690002	691.690002	5.890430e+07

In [29]:

```python
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1009 entries, 0 to 1008
Data columns (total 7 columns):
 #   Column     Non-Null Count  Dtype  
---  ------     --------------  -----  
 0   Date       1009 non-null   object 
 1   Open       1009 non-null   float64
 2   High       1009 non-null   float64
 3   Low        1009 non-null   float64
 4   Close      1009 non-null   float64
 5   Adj Close  1009 non-null   float64
 6   Volume     1009 non-null   int64  
dtypes: float64(5), int64(1), object(1)
memory usage: 55.3+ KB
```

In [30]:

```python
df.isnull()
```

Out[30]:

	Date	Open	High	Low	Close	Adj Close	Volume
0	False	False	False	False	False	False	False
1	False	False	False	False	False	False	False
2	False	False	False	False	False	False	False
3	False	False	False	False	False	False	False
4	False	False	False	False	False	False	False
...	...	...	...	...	...	...	...
1004	False	False	False	False	False	False	False
1005	False	False	False	False	False	False	False
1006	False	False	False	False	False	False	False
1007	False	False	False	False	False	False	False
1008	False	False	False	False	False	False	False

1009 rows × 7 columns

In [31]:

```python
df.sum()
```

Out[31]:

```
Date         2018-02-052018-02-062018-02-072018-02-082018-0...
Open                                              422831.209918
High                                              429148.589408
Low                                               416085.410196
Close                                             422771.739517
Adj Close                                         422771.739517
Volume                                               7638821200
dtype: object
```

In [32]:

```python
df.isnull().sum()
```

Out[32]:

```
Date         0
Open         0
High         0
Low          0
Close        0
Adj Close    0
Volume       0
dtype: int64
```

In [33]:

```python
for i in df.columns:
  print(i,"\t-\t", df[i].isna().mean()*100)
```

```
Date	-	 0.0
Open	-	 0.0
High	-	 0.0
Low	-	 0.0
Close	-	 0.0
Adj Close	-	 0.0
Volume	-	 0.0
```

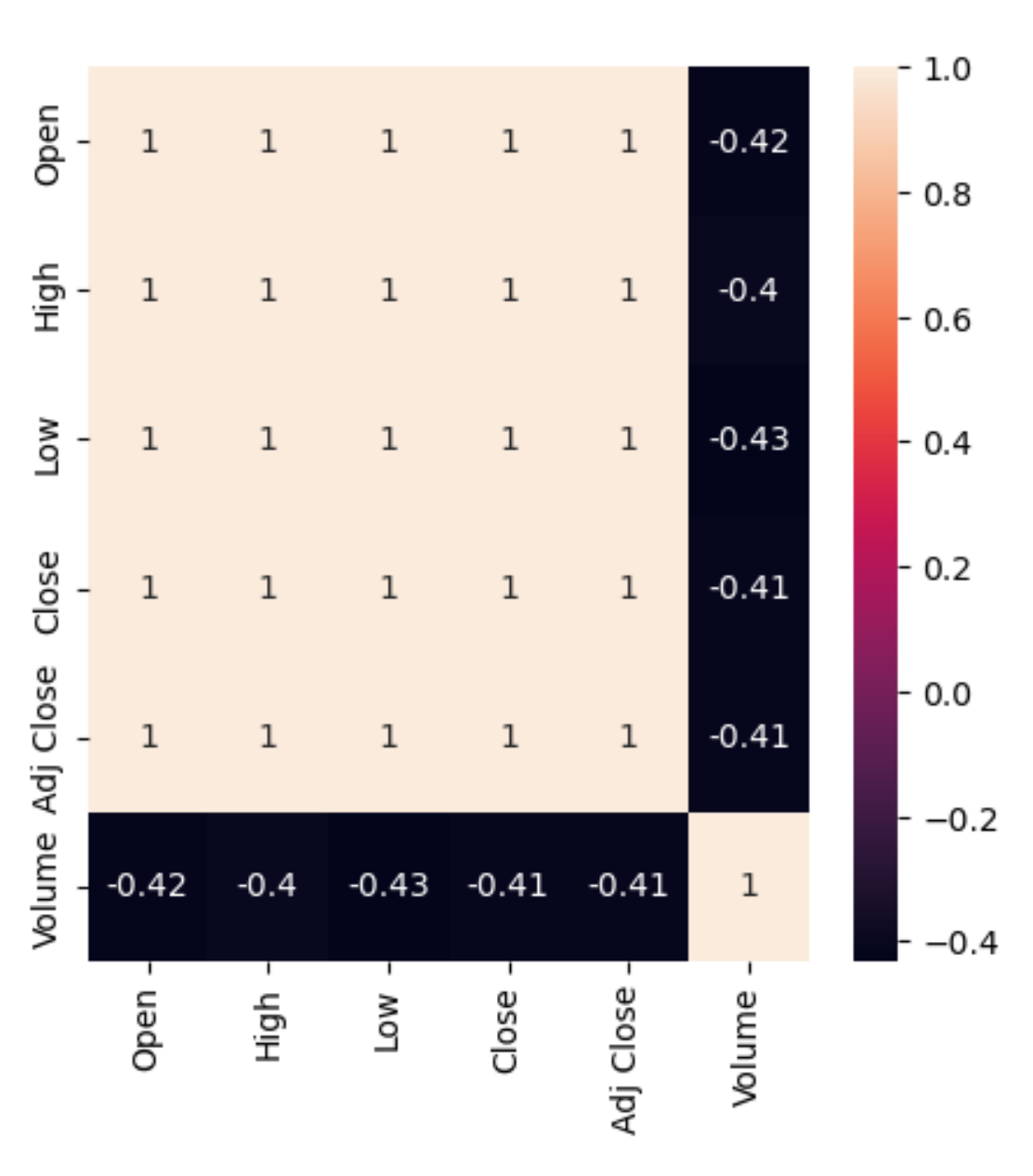
In [37]:

```python
f = df[df['Volume']=='AAP']   # Choosin stock values for any company

cormap = df.corr()
fig, ax = plt.subplots(figsize=(5,5))
sns.heatmap(cormap, annot = True)
```

Out[37]:

```
<AxesSubplot:>
```



In [48]:

```python
def get_corelated_col(cor_dat, threshold):
  # Cor_data to be column along which corelation to be measured
  #Threshold be the value above which of corelation to considered
  feature=[]
  value=[]

  for i ,index in enumerate(cor_dat.index):
    if abs(cor_dat[index]) > threshold:
      feature.append(index)
      value.append(cor_dat[index])

  df = pd.DataFrame(data = value, index = feature, columns=['corr value'])
  return df
```

In [49]:

```python
top_corelated_values = get_corelated_col(cormap['Close'], 0.60)
top_corelated_values
```

Out[49]:

	corr value
Open	0.996812
High	0.998551
Low	0.998544
Close	1.000000
Adj Close	1.000000

In [50]:

```python
df = df[top_corelated_values.index]
df.head()
```

Out[50]:

	Open	High	Low	Close	Adj Close
0	262.000000	267.899994	250.029999	254.259995	254.259995
1	247.699997	266.700012	245.000000	265.720001	265.720001
2	266.579987	272.450012	264.329987	264.559998	264.559998
3	267.079987	267.619995	250.000000	250.100006	250.100006
4	253.850006	255.800003	236.110001	249.470001	249.470001

In [52]:

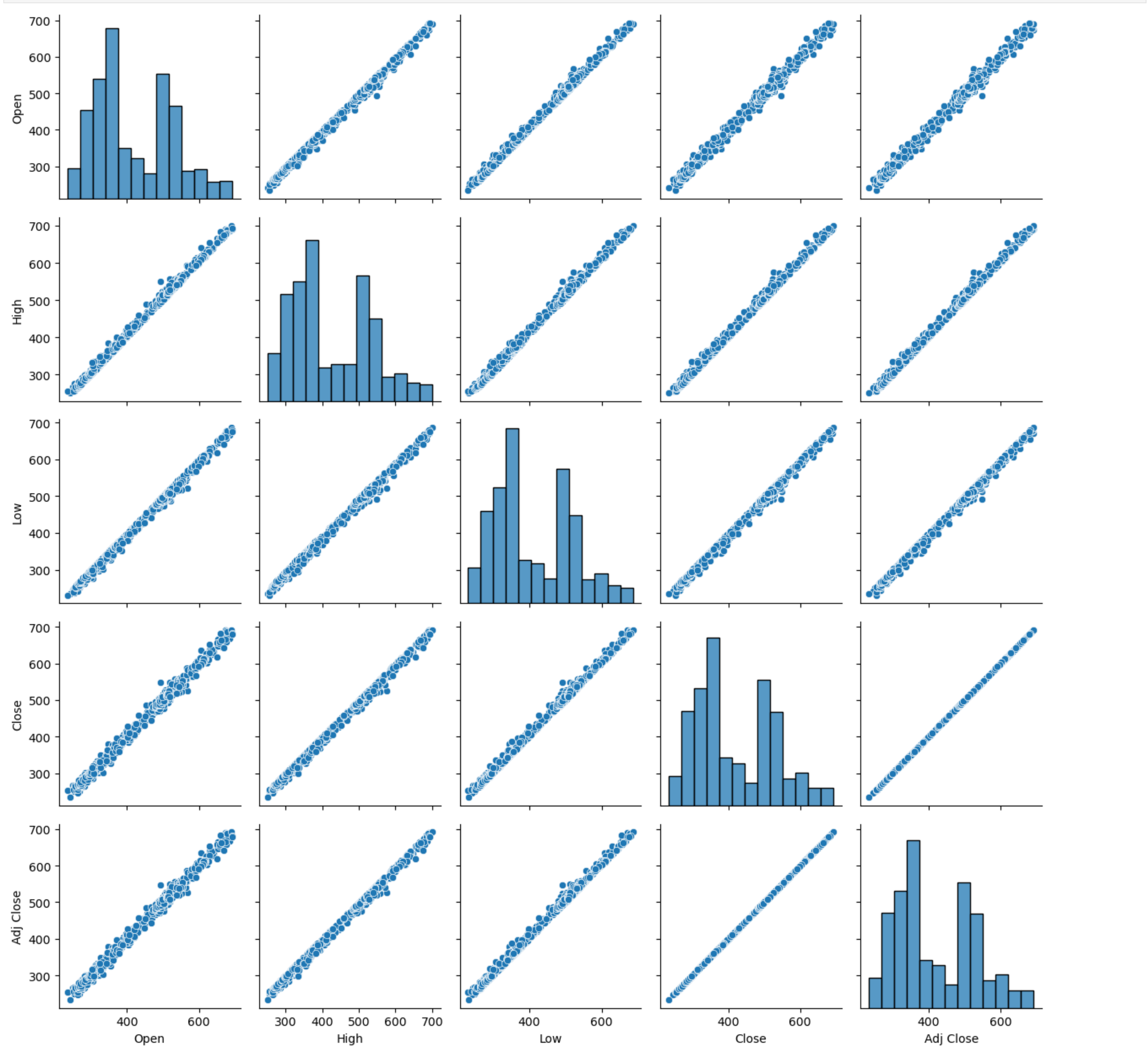
```python
df.shape
```

Out[52]:

```
(1009, 5)
```

In [53]:

```python
sns.pairplot(df)
plt.tight_layout()
```



In [54]:

```python
X = df.drop(['Close'], axis=1)
y = df['Close']
```

In [65]:

```python
X
```

Out[65]:

	Open	High	Low	Adj Close
0	0.061253	0.038304	0.041331	0.044516
1	0.030059	0.035640	0.030273	0.069548
2	0.071243	0.048408	0.072770	0.067015
3	0.072334	0.037683	0.041265	0.035430
4	0.043474	0.011436	0.010729	0.034053
...	...	...	...	...
1004	0.366577	0.393147	0.367080	0.422140
1005	0.434178	0.461496	0.427186	0.487648
1006	0.467531	0.447062	0.429253	0.427251
1007	0.409048	0.396611	0.380447	0.375090
1008	0.378226	0.359995	0.363650	0.385072

1009 rows × 4 columns

In [66]:

```python
y
```

Out[66]:

```
0       254.259995
1       265.720001
2       264.559998
3       250.100006
4       249.470001
           ...    
1004    427.140015
1005    457.130005
1006    429.480011
1007    405.600006
1008    410.170013
Name: Close, Length: 1009, dtype: float64
```

In [55]:

```python
from sklearn.preprocessing import MinMaxScaler

scaler = MinMaxScaler()
X = pd.DataFrame(scaler.fit_transform(X), columns=X.columns)
X.head()
```

Out[55]:

	Open	High	Low	Adj Close
0	0.061253	0.038304	0.041331	0.044516
1	0.030059	0.035640	0.030273	0.069548
2	0.071243	0.048408	0.072770	0.067015
3	0.072334	0.037683	0.041265	0.035430
4	0.043474	0.011436	0.010729	0.034053

In [67]:

```python
X.tail()
```

Out[67]:

	Open	High	Low	Adj Close
1004	0.366577	0.393147	0.367080	0.422140
1005	0.434178	0.461496	0.427186	0.487648
1006	0.467531	0.447062	0.429253	0.427251
1007	0.409048	0.396611	0.380447	0.375090
1008	0.378226	0.359995	0.363650	0.385072

In [56]:

```python
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, shuffle=False)

Acc = []
```

In [68]:

```python
X_train
```

Out[68]:

	Open	High	Low	Adj Close
0	0.061253	0.038304	0.041331	0.044516
1	0.030059	0.035640	0.030273	0.069548
2	0.071243	0.048408	0.072770	0.067015
3	0.072334	0.037683	0.041265	0.035430
4	0.043474	0.011436	0.010729	0.034053
...	...	...	...	...
802	0.700107	0.675534	0.675593	0.668705
803	0.676766	0.672470	0.684672	0.688801
804	0.690662	0.669117	0.677747	0.682947
805	0.682722	0.679020	0.690982	0.700203
806	0.699125	0.694831	0.692675	0.689566

807 rows × 4 columns

In [69]:

```python
X_test
```

Out[69]:

	Open	High	Low	Adj Close
807	0.597867	0.588022	0.598800	0.600730
808	0.610562	0.584692	0.592094	0.600467
809	0.600070	0.575232	0.592424	0.593412
810	0.595162	0.576964	0.597481	0.603788
811	0.607945	0.582538	0.600954	0.593412
...	...	...	...	...
1004	0.366577	0.393147	0.367080	0.422140
1005	0.434178	0.461496	0.427186	0.487648
1006	0.467531	0.447062	0.429253	0.427251
1007	0.409048	0.396611	0.380447	0.375090
1008	0.378226	0.359995	0.363650	0.385072

202 rows × 4 columns

In [70]:

```python
y_train
```

Out[70]:

```
0      254.259995
1      265.720001
2      264.559998
3      250.100006
4      249.470001
          ...    
802    540.020020
803    549.219971
804    546.539978
805    554.440002
806    549.570007
Name: Close, Length: 807, dtype: float64
```

In [71]:

```python
y_test
```

Out[71]:

```
807     508.899994
808     508.779999
809     505.549988
810     510.299988
811     505.549988
           ...    
1004    427.140015
1005    457.130005
1006    429.480011
1007    405.600006
1008    410.170013
Name: Close, Length: 202, dtype: float64
```

In [57]:

```python
from sklearn.linear_model import LinearRegression

# model training

model_1 = LinearRegression()
model_1.fit(X_train, y_train)
```

Out[57]:

```
LinearRegression()
```

In [58]:

```python
# prediction
y_pred_1 = model_1.predict(X_test)
pred_df = pd.DataFrame({'Actual': y_test, 'Predicted': y_pred_1})
pred_df.head()
```

Out[58]:

	Actual	Predicted
807	508.899994	508.899994
808	508.779999	508.779999
809	505.549988	505.549988
810	510.299988	510.299988
811	505.549988	505.549988

In [59]:

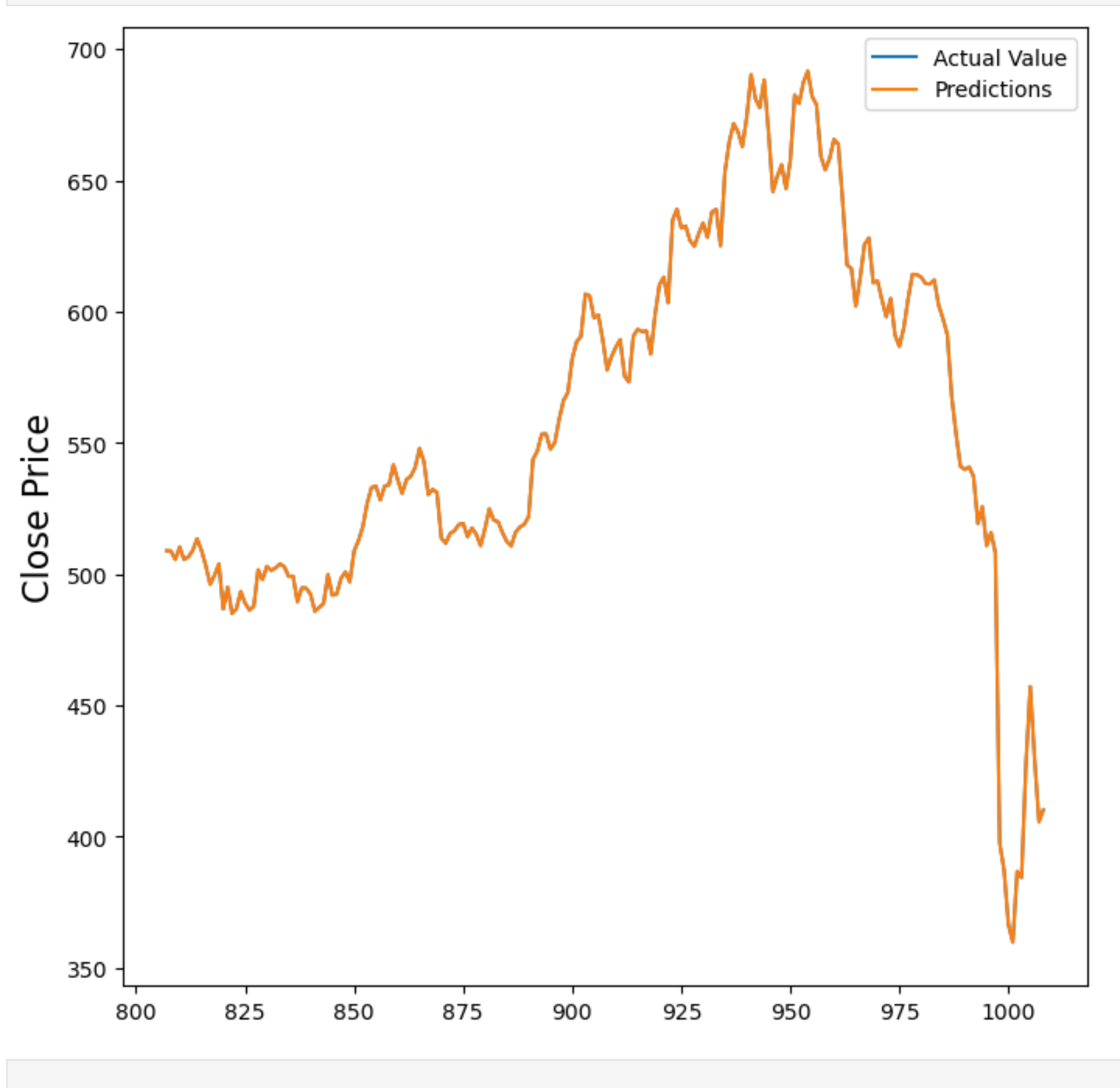
```python
from sklearn.metrics import r2_score

print("Accuracy score of the predictions: {0}".format(r2_score(y_test, y_pred_1)))
Acc.append(r2_score(y_test, y_pred_1))
```

```
Accuracy score of the predictions: 1.0
```

In [60]:

```python
plt.figure(figsize=(8,8))
plt.ylabel('Close Price', fontsize=16)
plt.plot(pred_df)
plt.legend(['Actual Value', 'Predictions'])
plt.show()
```



In [ ]: