In [7]:
```
!pip install matplotlib
```

```
Requirement already satisfied: matplotlib in c:\users\shivtej jagtap\anaconda3\lib\site-packages (3.5.2)
Requirement already satisfied: kiwisolver>=1.0.1 in c:\users\shivtej jagtap\anaconda3\lib\site-packages (from matplotlib) (1.4.2)
Requirement already satisfied: pillow>=6.2.0 in c:\users\shivtej jagtap\anaconda3\lib\site-packages (from matplotlib) (9.2.0)
Requirement already satisfied: cycler>=0.10 in c:\users\shivtej jagtap\anaconda3\lib\site-packages (from matplotlib) (0.11.0)
Requirement already satisfied: packaging>=20.0 in c:\users\shivtej jagtap\anaconda3\lib\site-packages (from matplotlib) (21.3)
Requirement already satisfied: python-dateutil>=2.7 in c:\users\shivtej jagtap\anaconda3\lib\site-packages (from matplotlib) (2.8.2)
Requirement already satisfied: pyparsing>=2.2.1 in c:\users\shivtej jagtap\anaconda3\lib\site-packages (from matplotlib) (3.0.9)
Requirement already satisfied: fonttools>=4.22.0 in c:\users\shivtej jagtap\anaconda3\lib\site-packages (from matplotlib) (4.25.0)
Requirement already satisfied: numpy>=1.17 in c:\users\shivtej jagtap\anaconda3\lib\site-packages (from matplotlib) (1.21.5)
Requirement already satisfied: six>=1.5 in c:\users\shivtej jagtap\anaconda3\lib\site-packages (from python-dateutil>=2.7->matplotlib) (1.16.0)
```

In [9]:
```
!pip install plotly
```

```
Requirement already satisfied: plotly in c:\users\shivtej jagtap\anaconda3\lib\site-packages (5.9.0)
Requirement already satisfied: tenacity>=6.2.0 in c:\users\shivtej jagtap\anaconda3\lib\site-packages (from plotly) (8.0.1)
```

In [10]:
```
import plotly
```

In [15]:
```
!pip install Chart_studio
```

```
Collecting Chart_studio
  Downloading Chart_studio-1.1.0-py3-none-any.whl (64 kB)
     ---------------------------------------- 64.4/64.4 kB 3.4 MB/s eta 0:00:00
Requirement already satisfied: plotly in c:\users\shivtej jagtap\anaconda3\lib\site-packages (from Chart_studio) (5.9.0)
Requirement already satisfied: six in c:\users\shivtej jagtap\anaconda3\lib\site-packages (from Chart_studio) (1.16.0)
Requirement already satisfied: requests in c:\users\shivtej jagtap\anaconda3\lib\site-packages (from Chart_studio) (2.28.1)
Collecting retrying>=1.3.3
  Downloading retrying-1.3.4-py3-none-any.whl (11 kB)
Requirement already satisfied: tenacity>=6.2.0 in c:\users\shivtej jagtap\anaconda3\lib\site-packages (from plotly->Chart_studio) (8.0.1)
Requirement already satisfied: urllib3<1.27,>=1.21.1 in c:\users\shivtej jagtap\anaconda3\lib\site-packages (from requests->Chart_studio) (1.26.11)
Requirement already satisfied: certifi>=2017.4.17 in c:\users\shivtej jagtap\anaconda3\lib\site-packages (from requests->Chart_studio) (2022.9.14)
Requirement already satisfied: idna<4,>=2.5 in c:\users\shivtej jagtap\anaconda3\lib\site-packages (from requests->Chart_studio) (3.3)
Requirement already satisfied: charset-normalizer<3,>=2 in c:\users\shivtej jagtap\anaconda3\lib\site-packages (from requests->Chart_studio) (2.0.4)
Installing collected packages: retrying, Chart_studio
Successfully installed Chart_studio-1.1.0 retrying-1.3.4
```

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline

import chart_studio.plotly as py
import plotly.graph_objs as go
from plotly.offline import plot

#from offline plotting
from plotly.offline import download_plotlyjs, init_notebook_mode, plot, iplot
init_notebook_mode(connected=True)
```

In [3]:
```
Stock = pd.read_csv('Stock.csv')
Stock.head()
```

Out[3]:

	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 [38]:
```
Stock = pd.read_csv('Stock.csv')
Stock.head()
```

Out[38]:

	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 [4]:
```
Stock.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 [5]:
```
Stock['Date'] = pd.to_datetime(Stock['Date'])
```

In [6]:
```
print(f'Datafra,e contains stock prices between {Stock.Date.min()} {Stock.Date.max()}')
print(f'total days = {(Stock.Date.max() - Stock.Date.min()).days} days')
```

```
Datafra,e contains stock prices between 2018-02-05 00:00:00 2022-02-04 00:00:00
total days = 1460 days
```

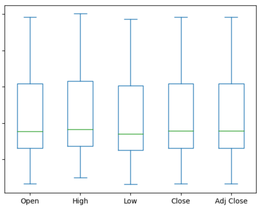
In [7]:
```
Stock.describe()
```

Out[7]:

	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 [8]:
```
Stock[['Open','High','Low','Close','Adj Close']].plot(kind ='box')
```

Out[8]:
```
<AxesSubplot:>
```



In [21]:
```
#Setting the layout for our plot
layout = go.Layout(
    title='Prices of Stock',
    xaxis=dict(
        title='Date',
        titlefont=dict(
            family='Courier New, monospace',
            size=18,
            color='#7f7f7f'
        )
    ),
     yaxis=dict(
         title='Price',
         titlefont=dict(
             family='Courier New, monospace',
             size=18,
             color='#7f7f7f'
         )
     )
)
Stock_data = [{'x':Stock['Date'], 'y':Stock['Close']}]
plot = go.Figure(data=Stock_data, layout=layout)
```

In [10]:
```
#plot(plot) #plotting offline
iplot(plot)
```

In [22]:
```
#building the regression model
from sklearn.model_selection import train_test_split

#for preprocessing
from sklearn.preprocessing import MinMaxScaler
from sklearn.preprocessing import StandardScaler

#for model evaluation
from sklearn.metrics import mean_squared_error as mse
from sklearn.metrics import r2_score
```

In [12]:
```
#split the data into train and test sets
X = np.array(Stock.index).reshape(-1,1)
Y = Stock['Close']
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.3, random_state=101)
```

In [16]:
```
#Feature scaling
scaler = StandardScaler().fit(X_train)
```

In [17]:
```
from sklearn.linear_model import LinearRegression
```

In [15]:
```
#creating a linear model
lm = LinearRegression()
lm.fit(X_train, Y_train)
```

Out[15]:
```
LinearRegression()
```

In [18]:
```
#Plot actual and predicted values for train dataset
trace0 = go.Scatter(
    x = X_train.T[0],
    y = Y_train,
    mode = 'markers',
    name = 'Actual'
)
trace1 = go.Scatter(
    x = X_train.T[0],
    y = lm.predict(X_train).T,
    mode = 'lines',
    name = 'Predicted'
)
Stock_data = [trace0,trace1]
layout.xaxis.title.text = 'Day'
plot2 = go.Figure(data=Stock_data, layout=layout)
```

In [19]:
```
iplot(plot2)
```

In [20]:
```
#Calculate score for model evaluation
scores = f'''
{'Metric'.ljust(10)}{'Train'.center(20)}{"Test" .center(20)}
{"r2_score".ljust(10)}{r2_score(Y_train, lm.predict(X_train))}\t{r2_score(Y_test,lm.predict(X_test))}
{'MSE' .ljust(10)}{mse(Y_train, lm.predict(X_train))}\t{mse(Y_test, lm.predict(X_test))}
'''
print(scores)
```

```
Metric              Train                Test
r2_score  0.6993686952944175    0.7261648566948495
MSE       3403.003880002517     3460.9885809580633
```

In [ ]:
```

```