In [1]:

```
#Description: This program uses the dual moving average crossover to determine when to buy and sel
l stock
```

In [2]:

```
#Import the libraries
import pandas as pd
import numpy as np
from datetime import datetime
import matplotlib.pyplot as plt
plt.style.use('fivethirtyeight')
```

In [3]:

```
#Load the data
AAPL = pd.read_csv("data/AAPL.csv")
```

In [4]:

```
#Show the data
AAPL
```

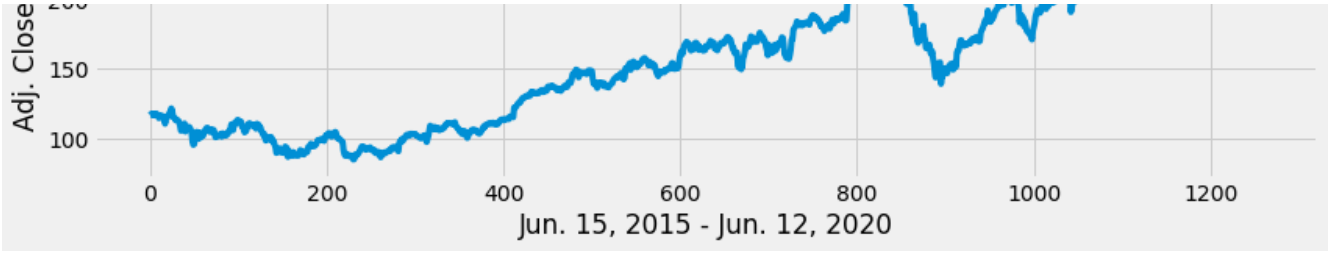
Out[4]:

	Date	Open	High	Low	Close	Adj Close	Volume
0	2015-06-15	126.099998	127.239998	125.709999	126.919998	116.970581	43988900
1	2015-06-16	127.029999	127.849998	126.370003	127.599998	117.597267	31494100
2	2015-06-17	127.720001	127.879997	126.739998	127.300003	117.320793	32918100
3	2015-06-18	127.230003	128.309998	127.220001	127.879997	117.855324	35407200
4	2015-06-19	127.709999	127.820000	126.400002	126.599998	116.675667	54716900
...	...	...	...	...	...	...	...
1254	2020-06-08	330.250000	333.600006	327.320007	333.459991	333.459991	23913600
1255	2020-06-09	332.140015	345.609985	332.010010	343.989990	343.989990	36928100
1256	2020-06-10	347.899994	354.769989	346.089996	352.839996	352.839996	41662900
1257	2020-06-11	349.309998	351.059998	335.480011	335.899994	335.899994	50415600
1258	2020-06-12	344.720001	347.799988	334.220001	338.799988	338.799988	50001500

1259 rows × 7 columns

In [5]:

```
#Visualize the data
plt.figure(figsize=(12.5, 4.5))
plt.plot(AAPL['Adj Close'], label = 'AAPL')
plt.title('Apple Adj. CLose Price History')
plt.xlabel('Jun. 15, 2015 - Jun. 12, 2020')
plt.ylabel('Adj. Close Price USD ($)')
plt.legend(loc='upper left')
plt.show()
```



In [6]:

```
#Create the simple moving average with a 30 day window
SMA30 = pd.DataFrame()
SMA30['Adj Close'] = AAPL['Adj Close'].rolling(window= 30).mean()
SMA30
```

Out[6]:

	Adj Close
0	NaN
1	NaN
2	NaN
3	NaN
4	NaN
...	...
1254	309.782894
1255	311.835709
1256	314.336111
1257	315.967670
1258	317.494108

1259 rows × 1 columns

In [7]:

```
#Create a simple moving 100 day average
SMA100 = pd.DataFrame()
SMA100['Adj Close'] = AAPL['Adj Close'].rolling(window= 100).mean()
SMA100
```

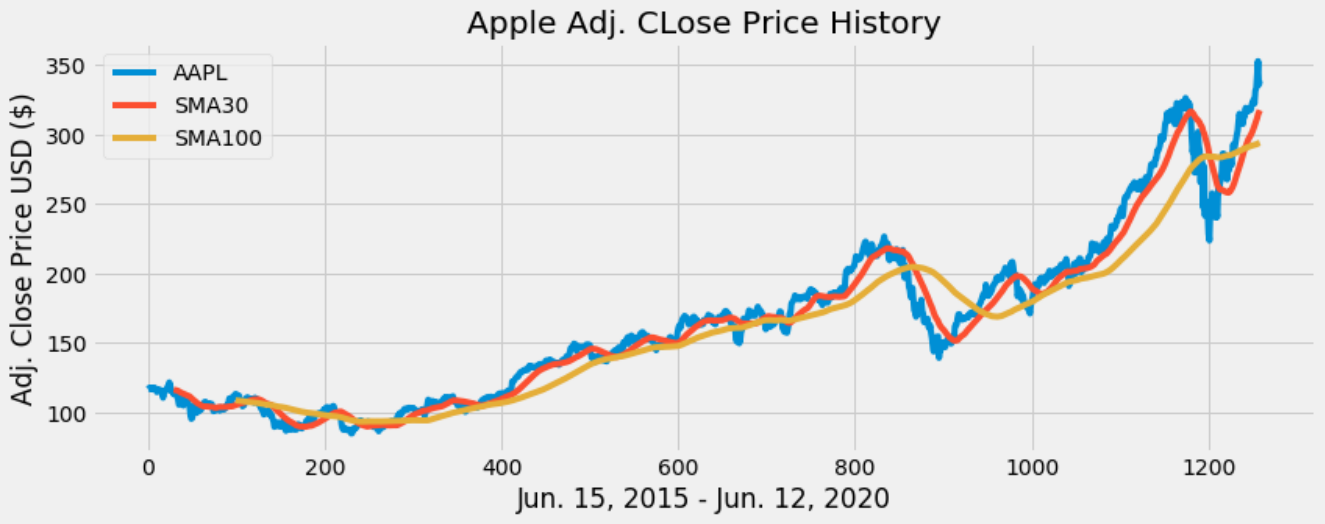
Out[7]:

	Adj Close
0	NaN
1	NaN
2	NaN
3	NaN
4	NaN
...	...
1254	292.935496
1255	293.277753
1256	293.669707
1257	293.857538
1258	294.095859

1259 rows × 1 columns

In [8]:

```
#Visualize the data
plt.figure(figsize=(12.5, 4.5))
plt.plot(AAPL['Adj Close'], label = 'AAPL')
plt.plot(SMA30['Adj Close'], label = 'SMA30')
plt.plot(SMA100['Adj Close'], label = 'SMA100')
plt.title('Apple Adj. CLose Price History')
plt.xlabel('Jun. 15, 2015 - Jun. 12, 2020')
plt.ylabel('Adj. Close Price USD ($)')
plt.legend(loc='upper left')
plt.show()
```



In [9]:

```
#Create a new data frame to store all the data
data = pd.DataFrame()
data['AAPL'] = AAPL['Adj Close']
data['SMA30'] = SMA30['Adj Close']
data['SMA100'] = SMA100['Adj Close']
data
```

Out[9]:

	AAPL	SMA30	SMA100
0	116.970581	NaN	NaN
1	117.597267	NaN	NaN
2	117.320793	NaN	NaN
3	117.855324	NaN	NaN
4	116.675667	NaN	NaN
...	...	...	...
1254	333.459991	309.782894	292.935496
1255	343.989990	311.835709	293.277753
1256	352.839996	314.336111	293.669707
1257	335.899994	315.967670	293.857538
1258	338.799988	317.494108	294.095859

1259 rows × 3 columns

In [10]:

```
#Create a funcation to signal when to buy and sell the asset
def buy_sell(data):
    sigPriceBuy = []
    sigPriceSell = []
    flag = -1

    for i in range(len(data)):
```

```
        if data['SMA30'][i] > data['SMA100'][i]:
            if flag != 1:
                sigPriceBuy.append(data['AAPL'][i])
                sigPriceSell.append(np.nan)
                flag = 1
            else:
                sigPriceBuy.append(np.nan)
                sigPriceSell.append(np.nan)
        elif data['SMA30'][i] < data['SMA100'][i]:
            if flag != 0:
                sigPriceBuy.append(np.nan)
                sigPriceSell.append(data['AAPL'][i])
                flag = 0
            else:
                sigPriceBuy.append(np.nan)
                sigPriceSell.append(np.nan)
        else:
            sigPriceBuy.append(np.nan)
            sigPriceSell.append(np.nan)
    return(sigPriceBuy, sigPriceSell)
```

In [11]:

```
#Store the buy and sell data into a variable
buy_sell = buy_sell(data)
data['Buy_Signal_Price'] = buy_sell[0]
data['Sell_Signal_Price'] = buy_sell[1]
```

In [12]:

```
#Show the data
data
```

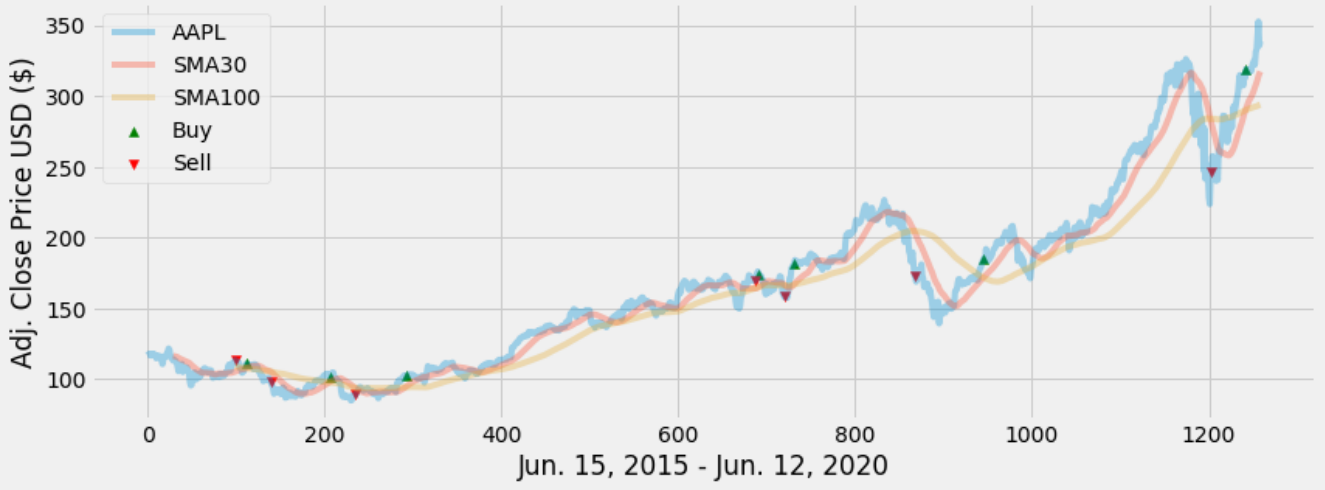
Out[12]:

	AAPL	SMA30	SMA100	Buy_Signal_Price	Sell_Signal_Price
0	116.970581	NaN	NaN	NaN	NaN
1	117.597267	NaN	NaN	NaN	NaN
2	117.320793	NaN	NaN	NaN	NaN
3	117.855324	NaN	NaN	NaN	NaN
4	116.675667	NaN	NaN	NaN	NaN
...	...	...	...	...	...
1254	333.459991	309.782894	292.935496	NaN	NaN
1255	343.989990	311.835709	293.277753	NaN	NaN
1256	352.839996	314.336111	293.669707	NaN	NaN
1257	335.899994	315.967670	293.857538	NaN	NaN
1258	338.799988	317.494108	294.095859	NaN	NaN

1259 rows × 5 columns

In [13]:

```
#Visualize the data and the strategy to buy and sell the stock
plt.figure(figsize=(12.6, 4.6))
plt.plot(data['AAPL'], label = 'AAPL', alpha = 0.35)
plt.plot(data['SMA30'], label = 'SMA30', alpha = 0.35)
plt.plot(data['SMA100'], label = 'SMA100', alpha = 0.35)
plt.scatter(data.index, data['Buy_Signal_Price'], label = 'Buy', marker = '^', color = 'green')
plt.scatter(data.index, data['Sell_Signal_Price'], label = 'Sell', marker = 'v', color = 'red')
plt.title('Apple Adj. Close Price History Buy and Sell Signals')
plt.xlabel('Jun. 15, 2015 - Jun. 12, 2020')
plt.ylabel('Adj. Close Price USD ($)')
plt.legend(loc='upper left')
plt.show()
```



In [14]:

```
#Next section - Trading strategy technical anlaysis using Python, RSI
```

In [15]:

```
#Store the data
FB = pd.read_csv('data/FB.csv')
FB
```

Out[15]:

	Date	Open	High	Low	Close	Adj Close	Volume
0	2015-06-15	80.550003	80.930000	80.070000	80.709999	80.709999	18805100
1	2015-06-16	80.820000	81.510002	80.449997	81.059998	81.059998	13693700
2	2015-06-17	81.760002	82.220001	81.339996	81.790001	81.790001	18350300
3	2015-06-18	81.639999	83.190002	81.570000	82.910004	82.910004	26782600
4	2015-06-19	82.830002	82.980003	82.120003	82.510002	82.510002	23353200
...	...	...	...	...	...	...	...
1254	2020-06-08	229.029999	231.550003	227.410004	231.399994	231.399994	15466500
1255	2020-06-09	231.520004	239.770004	230.410004	238.669998	238.669998	27462900
1256	2020-06-10	240.960007	241.210007	235.279999	236.729996	236.729996	20720700
1257	2020-06-11	229.940002	232.889999	223.550003	224.429993	224.429993	26708200
1258	2020-06-12	229.899994	231.660004	224.500000	228.580002	228.580002	22071700

1259 rows × 7 columns

In [16]:

```
#Set the date as the index for the data
FB = FB.set_index(pd.DatetimeIndex(FB['Date'].values))
FB
```

Out[16]:

	Date	Open	High	Low	Close	Adj Close	Volume
2015-06-15	2015-06-15	80.550003	80.930000	80.070000	80.709999	80.709999	18805100
2015-06-16	2015-06-16	80.820000	81.510002	80.449997	81.059998	81.059998	13693700
2015-06-17	2015-06-17	81.760002	82.220001	81.339996	81.790001	81.790001	18350300
2015-06-18	2015-06-18	81.639999	83.190002	81.570000	82.910004	82.910004	26782600
2015-06-19	2015-06-19	82.830002	82.980003	82.120003	82.510002	82.510002	23353200
...	...	...	...	...	...	...	...
2020-06-08	2020-06-08	229.029999	231.550003	227.410004	231.399994	231.399994	15466500

	Date	Open	High	Low	Close	Adj Close	Volume
2020-06-09	2020-06-09	231.520004	239.770004	230.410004	238.669998	238.669998	27462900
2020-06-10	2020-06-10	240.960007	241.210007	235.279999	236.729996	236.729996	20720700
2020-06-11	2020-06-11	229.940002	232.889999	223.550003	224.429993	224.429993	26708200
2020-06-12	2020-06-12	229.899994	231.660004	224.500000	228.580002	228.580002	22071700

1259 rows × 7 columns

In [17]:

```
#Visually show the price
plt.figure(figsize=(12.2, 4.5))
plt.plot(FB.index, FB['Adj Close'], label='Adj Close')
plt.title('Adj. Close Price History')
plt.xlabel('Jun 15, 2015 - Jun 12, 2020', fontsize = 18)
plt.ylabel('Adj. Close Price USD ($)', fontsize = 18)
plt.show()
```

In [18]:

```
#Prepare the data to calculate the RSI

#Get the difference in price from the previous day
delta = FB['Adj Close'].diff(1)
delta
```

Out[18]:

```
2015-06-15          NaN
2015-06-16     0.349999
2015-06-17     0.730003
2015-06-18     1.120003
2015-06-19    -0.400002
                 ...
2020-06-08     0.629990
2020-06-09     7.270004
2020-06-10    -1.940002
2020-06-11   -12.300003
2020-06-12     4.150009
Name: Adj Close, Length: 1259, dtype: float64
```

In [19]:

```
#Get ride of NaN
delta = delta.dropna()
delta
```

Out[19]:

```
2015-06-16     0.349999
2015-06-17     0.730003
```

```
2015-06-18     1.120003
2015-06-19    -0.400002
2015-06-22     2.229996
                 ...
2020-06-08     0.629990
2020-06-09     7.270004
2020-06-10    -1.940002
2020-06-11   -12.300003
2020-06-12     4.150009
Name: Adj Close, Length: 1258, dtype: float64
```

In [20]:

```
#Get the positive gains (up) and the negative gains (down)
up = delta.copy()
down = delta.copy()

up[up<0]=0
down[down>0]=0
```

In [21]:

```
#Get the time period
period = 14
#Calculate the average gain and the average loss
AVG_Gain = up.rolling(window=period).mean()
AVG_Loss = abs(down.rolling(window=period).mean())
```

In [24]:

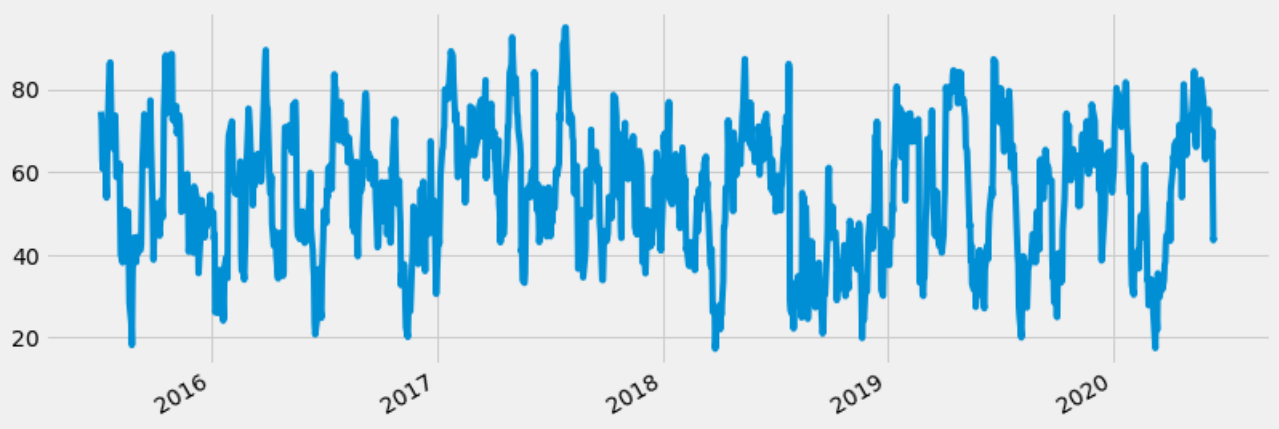
```
#Calculate the RSI

#Calculate the Relative Strength (RS)
RS=AVG_Gain/AVG_Loss
```

In [25]:

```
#Calculate the Relative Strength Index(RSI)
RSI = 100.0 - (100.0/(1.0 + RS))
```

In [27]:

```
#Show the RSI Visually
plt.figure(figsize=(12.2, 4.5))
RSI.plot()
plt.show()
```



In [30]:

```
#Put it all together

#Create a new data frame
new_df = pd.DataFrame()
```

```
new_df['Adj Close'] = FB['Adj Close']
new_df['RSI'] = RSI
new_df
```

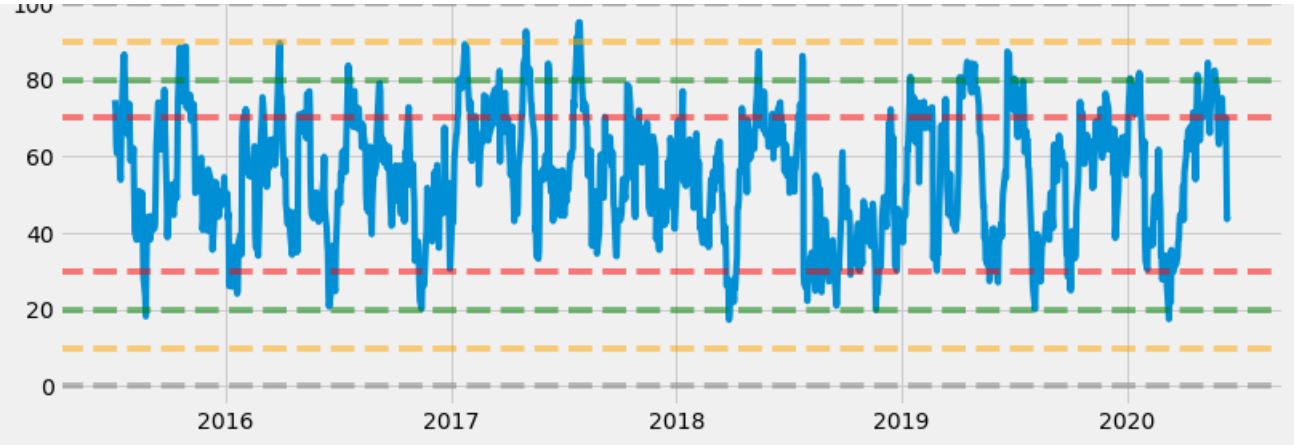
Out[30]:

	Adj Close	RSI
2015-06-15	80.709999	NaN
2015-06-16	81.059998	NaN
2015-06-17	81.790001	NaN
2015-06-18	82.910004	NaN
2015-06-19	82.510002	NaN
...	...	...
2020-06-08	231.399994	67.955022
2020-06-09	238.669998	70.068137
2020-06-10	236.729996	57.834945
2020-06-11	224.429993	43.557940
2020-06-12	228.580002	44.208603

1259 rows × 2 columns

In [35]:

```
#Visually show the adjusted close price and RSI

#Plot the adjusted close price
plt.figure(figsize = (12.2, 4.5))
plt.plot(new_df.index, new_df['Adj Close'])
plt.title('Adj. Close Price History')
plt.legend(new_df.columns.values, loc = 'upper left')
plt.show()

#Plot the corresponding RSI values and the significant levels
plt.figure(figsize=(12.2, 4.5))
plt.title('RSI Plot')
plt.plot(new_df.index, new_df['RSI'])
plt.axhline(0, linestyle = '--', alpha = 0.5, color = 'gray')
plt.axhline(10, linestyle = '--', alpha = 0.5, color = 'orange')
plt.axhline(20, linestyle = '--', alpha = 0.5, color = 'green')
plt.axhline(30, linestyle = '--', alpha = 0.5, color = 'red')
plt.axhline(70, linestyle = '--', alpha = 0.5, color = 'red')
plt.axhline(80, linestyle = '--', alpha = 0.5, color = 'green')
plt.axhline(90, linestyle = '--', alpha = 0.5, color = 'orange')
plt.axhline(100, linestyle = '--', alpha = 0.5, color = 'gray')
plt.show()
```



In [ ]: