```
#Detecting resistance and support from the finacial data
```

In this program we will try to see how we could use python to detect resistance and support for Eur vs USD

```
import pandas as pd
df = pd.read_csv("EURUSD_Candlestick_1_D_ASK_05.05.2003-30.06.2021.csv")
df.tail()
```

	Local time	open	high	low	close	volume
6627	26.06.2021 00:00:00.000 GMT+0300	1.19392	1.19392	1.19392	1.19392	0.00000
6628	27.06.2021 00:00:00.000 GMT+0300	1.19392	1.19392	1.19392	1.19392	0.00000
6629	28.06.2021 00:00:00.000 GMT+0300	1.19380	1.19447	1.19025	1.19260	85154.26000
[illegible]	29.06.2021 00:00:00.000	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]

```
df=df[df['volume']!=0]
df.reset_index(drop=True, inplace=True)
```

df.reset_index() is a method used to reset the index of a DataFrame. By default, it moves the current index to a new column and assigns a new numerical index to the DataFrame.

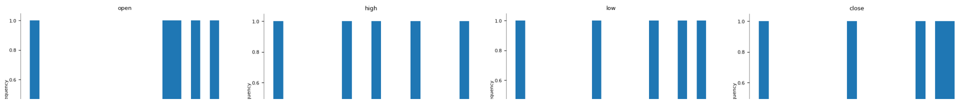
drop=True means that the old index will be removed. If set to False, the old index will be kept as a new column in the DataFrame.

inplace=True means that the operation will be performed on the DataFrame itself, rather than returning a modified copy. This means that df will be updated with the new index.

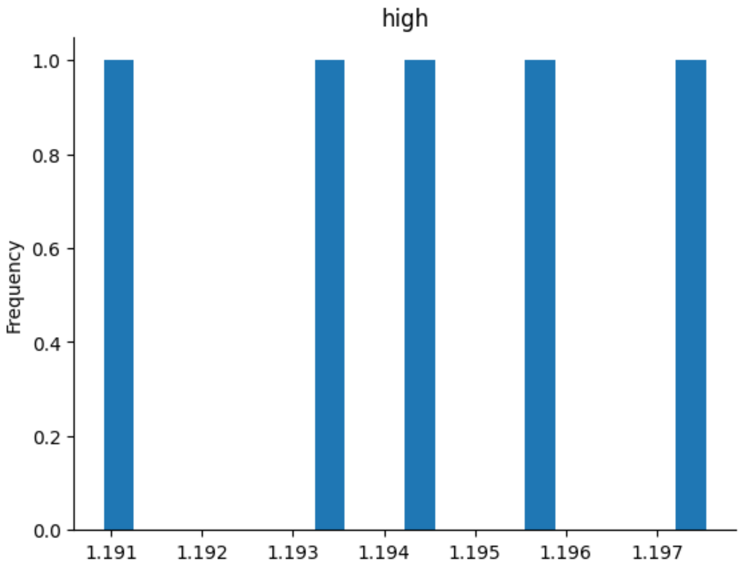
```
df.isna().sum()
df.tail()
#df.isna().sum() counts the number of True values in each column, which corresponds to the number of missing values in each column.
```

	Local time	open	high	low	close	volume
4729	24.06.2021 00:00:00.000 GMT+0300	1.19267	1.19565	1.19178	1.19322	85152.21000
4730	25.06.2021 00:00:00.000 GMT+0300	1.19322	1.19754	1.19264	1.19392	77837.64500
4731	28.06.2021 00:00:00.000 GMT+0300	1.19380	1.19447	1.19025	1.19260	85154.26000
4732	29.06.2021 00:00:00.000 GMT+0300	1.19297	1.19334	1.18779	1.18973	98898.57000
4733	30.06.2021 00:00:00.000 GMT+0300	1.18973	1.19092	1.18452	1.18589	4301.30191

Distributions



```
from matplotlib import pyplot as plt
_df_1['high'].plot(kind='hist', bins=20, title='high')
plt.gca().spines[['top', 'right',]].set_visible(False)
```



```
def support(df1, l, n1, n2): #n1 and n2 is the numbers of candle anfter candle l
  for i in range(l-n1+1, l+1):
    if(df1.low[i]>df1.low[i-1]):
      return 0
  for i in range(l+1, l+n2+1):
    if(df1.low[i]<df1.low[i-1]):
      return 0
  return 1

def resistance (df1, l, n1, n2):
  for i in range(l-n1+1, l+1):
    if(df1.high[i-1]>df1.high[i]):
      return 0
  for i in range(l+1, l+n2+1):
    if(df1.high[i]>df1.high[i-1]):
      return 0
  return 1
```

```
length = len(df)
high = list(df['high'])
low = list(df['low'])
close = list(df['close'])
open = list(df['open'])
bodydiff = [0]*length

highdiff = [0]*length
lowdiff = [0]*length
ratio1 = [0]*length
ratio2 = [0]*length
```

```
def isEngulfing(l):
    row=l
    bodydiff[row] = abs(open[row]-close[row])
    if bodydiff[row]<0.000001:
        bodydiff[row]=0.000001

    bodydiffmin = 0.002
    if (bodydiff[row]>bodydiffmin and bodydiff[row-1]>bodydiffmin and
        open[row-1]<close[row-1] and
        open[row]>close[row] and
        (open[row]-close[row-1])>=-0e-5 and close[row]<open[row-1]): #+0e-5 -5e-5
        return 1

    elif(bodydiff[row]>bodydiffmin and bodydiff[row-1]>bodydiffmin and
        open[row-1]>close[row-1] and
        open[row]<close[row] and
        (open[row]-close[row-1])<=+0e-5 and close[row]>open[row-1]):#-0e-5 +5e-5
        return 2
    else:
        return 0

def isStar(l):
    bodydiffmin = 0.0020
    row=l
    highdiff[row] = high[row]-max(open[row],close[row])
    lowdiff[row] = min(open[row],close[row])-low[row]
    bodydiff[row] = abs(open[row]-close[row])
    if bodydiff[row]<0.000001:
        bodydiff[row]=0.000001
    ratio1[row] = highdiff[row]/bodydiff[row]
    ratio2[row] = lowdiff[row]/bodydiff[row]

    if (ratio1[row]>1 and lowdiff[row]<0.2*highdiff[row] and bodydiff[row]>bodydiffmin):# and open[row]>close[row]):
        return 1
    elif (ratio2[row]>1 and highdiff[row]<0.2*lowdiff[row] and bodydiff[row]>bodydiffmin):# and open[row]<close[row]):
        return 2
    else:
        return 0

def closeResistance(l,levels,lim):
    if len(levels)==0:
        return 0
    c1 = abs(df.high[l]-min(levels, key=lambda x:abs(x-df.high[l])))<=lim
    c2 = abs(max(df.open[l],df.close[l])-min(levels, key=lambda x:abs(x-df.high[l])))<=lim
    c3 = min(df.open[l],df.close[l])<min(levels, key=lambda x:abs(x-df.high[l]))
    c4 = df.low[l]<min(levels, key=lambda x:abs(x-df.high[l]))
    if( (c1 or c2) and c3 and c4 ):
        return 1
    else:
        return 0

def closeSupport(l,levels,lim):
    if len(levels)==0:
        return 0
    c1 = abs(df.low[l]-min(levels, key=lambda x:abs(x-df.low[l])))<=lim
    c2 = abs(min(df.open[l],df.close[l])-min(levels, key=lambda x:abs(x-df.low[l])))<=lim
    c3 = max(df.open[l],df.close[l])>min(levels, key=lambda x:abs(x-df.low[l]))
    c4 = df.high[l]>min(levels, key=lambda x:abs(x-df.low[l]))
    if( (c1 or c2) and c3 and c4 ):
        return 1
    else:
        return 0
```

```
n1=2
n2=2
backCandles=45
signal = [0] * length

for row in range(backCandles, len(df)-n2):
    ss = []
    rr = []
    for subrow in range(row-backCandles+n1, row+1):
        if support(df, subrow, n1, n2):
            ss.append(df.low[subrow])
        if resistance(df, subrow, n1, n2):
            rr.append(df.high[subrow])
    #!!!! parameters
    if ((isEngulfing(row)==1 or isStar(row)==1) and closeResistance(row, rr, 150e-5) ):#and df.RSI[row]<30
        signal[row] = 1
    elif((isEngulfing(row)==2 or isStar(row)==2) and closeSupport(row, ss, 150e-5)):#and df.RSI[row]>70
        signal[row] = 2
    else:
        signal[row] = 0


df['signal']=signal
df[df['signal']==1].count()
```

```
Local time    98
open          98
high          98
low           98
close         98
volume        98
signal        98
dtype: int64
```

```
SLTPRatio = 1.1 #TP/SL Ratio
def mytarget(barsupfront, df1):
    length = len(df1)
    high = list(df1['high'])
    low = list(df1['low'])
    close = list(df1['close'])
    open = list(df1['open'])
    signal = list(df1['signal'])
    trendcat = [0] * length
    amount = [0] * length

    SL=0
    TP=0
    for line in range(backCandles, length-barsupfront-n2):

        if signal[line]==1:
            SL = max(high[line-1:line+1])#!!!!! parameters
            TP = close[line]-SLTPRatio*(SL-close[line])
            for i in range(1,barsupfront+1):
                if(low[line+i]<=TP and high[line+i]>=SL):
                    trendcat[line]=3
                    break
                elif (low[line+i]<=TP):
                    trendcat[line]=1 #win trend 1 in signal 1
                    amount[line]=close[line]-low[line+i]
                    break
                elif (high[line+i]>=SL):
                    trendcat[line]=2 #loss trend 2 in signal 1
                    amount[line]=close[line]-high[line+i]
                    break

        if signal[line]==2:
            SL = min(low[line-1:line+1])#!!!!! parameters
            TP = close[line]+SLTPRatio*(close[line]-SL)

            for i in range(1,barsupfront+1):
                if(high[line+i]>=TP and low[line+i]<=SL):
                    trendcat[line]=3
                    break
                elif (high[line+i]>=TP):
                    trendcat[line]=2 #win trend 2 in signal 2
                    amount[line]=high[line+i]-close[line]
                    break
                elif (low[line+i]<=SL):
                    trendcat[line]=1 #loss trend 1 in signal 2
                    amount[line]=low[line+i]-close[line]
                    break
    #return trendcat
    return amount
```

```
df['Trend'] = mytarget(16, df)
df['Amount'] = mytarget(16, df)
```

```
df[df['Amount']!=0]
```

	Local time	open	high	low	close	volume	signal	Trend	Amo
105	29.09.2003 00:00:00.000 GMT+0300	1.14558	1.16015	1.13941	1.15971	1.120821e+06	2	0.02425	0.02
113	09.10.2003 00:00:00.000 GMT+0300	1.18097	1.18604	1.16840	1.17457	1.122516e+06	1	0.01639	0.01
119	17.10.2003 00:00:00.000 GMT+0300	1.15860	1.16843	1.15526	1.16812	1.110120e+06	2	0.01453	0.01
157	10.12.2003 00:00:00.000 GMT+0200	1.22576	1.22649	1.21523	1.22145	1.114964e+06	1	0.01022	0.01
178	08.01.2004 00:00:00.000 GMT+0200	1.26306	1.27796	1.25590	1.27661	1.113789e+06	2	-0.04157	-0.04
...	...	...	...	...	...	...	...	...	

```
import numpy as np
conditions = [(df['Trend'] == 1) & (df['signal'] == 1), (df['Trend'] == 2) & (df['signal'] == 2)]
values = [1, 2]
df['result'] = np.select(conditions, values)

trendId=1
print(df[df['result']==trendId].result.count()/df[df['signal']==trendId].signal.count())
df[ (df['Trend']!=trendId) & (df['Trend']!=3) & (df['signal']==trendId) ] # false positives
```

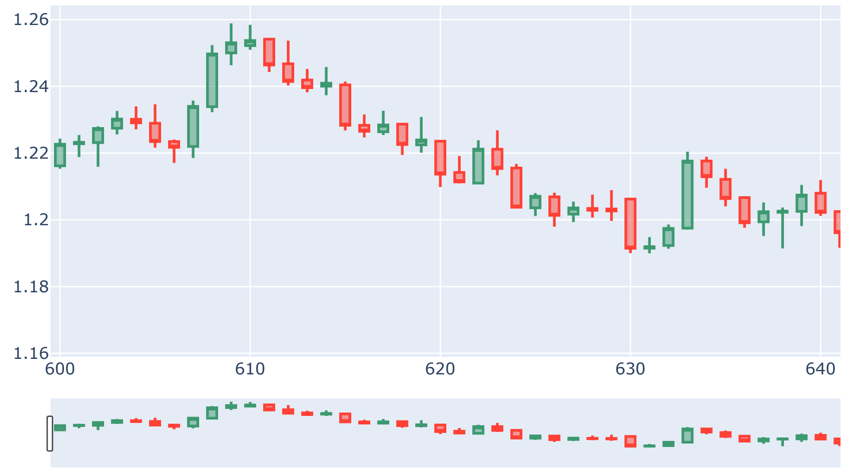
```
0.0
```

	Local time	open	high	low	close	volume	signal	Trend	Amo
113	09.10.2003 00:00:00.000 GMT+0300	1.18097	1.18604	1.16840	1.17457	1.122516e+06	1	0.01639	0.01
157	10.12.2003 00:00:00.000 GMT+0200	1.22576	1.22649	1.21523	1.22145	1.114964e+06	1	0.01022	0.01
180	12.01.2004 00:00:00.000 GMT+0200	1.28538	1.28971	1.27342	1.27493	1.108322e+06	1	0.01879	0.01
243	08.04.2004 00:00:00.000 GMT+0300	1.21754	1.22186	1.20584	1.20853	1.133015e+06	1	0.01822	0.01
263	06.05.2004 00:00:00.000 GMT+0300	1.21783	1.21783	1.20635	1.20825	1.137840e+06	1	0.02060	0.02
...	...	...	...	...	...	...	...	...	

```
dfpl = df[600:670]
import plotly.graph_objects as go
from datetime import datetime

fig = go.Figure(data=[go.Candlestick(x=dfpl.index,
                open=dfpl['open'],
                high=dfpl['high'],
                low=dfpl['low'],
                close=dfpl['close'])])

fig.show()
```


1.26
1.24
1.22
1.2
1.18
1.16
600
610
620
630
640