```
#Setup
import stats
import ta.momentum

import Ta

Apikey= "Insert APIKey"
Secret= "Insert SecretKey"

from binance import Client, ThreadedWebsocketManager, ThreadedDepthCacheManager
import pandas as pd
import ta.momentum as TAM
from scipy import stats
pd.set_option('display.max_rows', 3000)
pd.set_option('display.max_columns', 3000)
pd.set_option('display.width', 1000)

#Authenticate
client = Client (Apikey, Secret)

#Get tickers
tickers = client.get_all_tickers()
tickers_df = pd.DataFrame (tickers, columns = ["symbol", "price"])

#List of symbols
list_of_symbols = []
for i in tickers_df ["symbol"]:
    if "USDT" in i:
       list_of_symbols.append(i)

print (list_of_symbols)
nueva = []
for m in range(101):
    nueva.append(0)

nueva_df = pd.DataFrame(nueva)
nueva_df.columns = ["relleno"]

for i in list_of_symbols:

    try:

        #Get Historical Data
        historical = client.get_historical_klines(""+i+"",
Client.KLINE_INTERVAL_1HOUR, "01 Apr 2022")

        hist_df = pd.DataFrame (historical)
        hist_df.columns = ["Open_time", "Open", "High", "Low", "Close", "Volume",
"Close time", "Quote asset volume", "Number of trades", "Taker buy base asset
volume", "Taker buy quote asset volume", "Can be ignored"]
```

```
        #Preprocess Historical Data
        hist_df ["Open_time"] = pd.to_datetime(hist_df ["Open_time"]/1000,
unit="s")
        hist_df ["Close time"] = pd.to_datetime(hist_df ["Close time"]/1000,
unit="s")
        numeric_columns = ["Open", "High", "Low", "Close", "Volume", "Quote asset
volume", "Taker buy base asset volume", "Taker buy quote asset volume"]
        hist_df [numeric_columns] = hist_df [numeric_columns].apply(pd.to_numeric,
axis=1)

        #RSI = TAM.rsi(hist_df ["Close"],3)

        RSI = TAM.rsi(hist_df ["Close"], 3)
        RSI_df = pd.DataFrame (RSI)
        RSI_df.columns = ["RSI"]


        #UpDownLength

        Column_UPL = []
        Alt_list_Up = []
        Alt_list_Down = []
        Alt_list_Zero = []

        a=1
        while a < len (hist_df ["Close"]):

            if hist_df ["Close"] [a] > hist_df ["Close"] [a-1]:
                Alt_list_Down.clear()
                Alt_list_Zero.clear()
                Value1 = 1
                Alt_list_Up.append (Value1)
                Column_UPL.append(len(Alt_list_Up))

            if hist_df ["Close"] [a] < hist_df ["Close"] [a-1]:
                Alt_list_Up.clear()
                Alt_list_Zero.clear()
                Value2 = 1
                Alt_list_Down.append(Value2)
                Column_UPL.append(len(Alt_list_Down)*-1)

            if hist_df ["Close"] [a] == hist_df ["Close"] [a-1]:
                Alt_list_Up.clear()
                Alt_list_Down.clear()
                Value3 = 1
                Alt_list_Zero.append(Value3)
                Column_UPL.append(len(Alt_list_Zero))

            a+=1
```

```
        RSI_UpDownLength_df = pd.DataFrame (Column_UPL)
        RSI_UpDownLength_df.columns = ["UDL"]
        RSI_UDL = TAM.rsi(RSI_UpDownLength_df ["UDL"], 2)
        RSI_UDL_df = pd.DataFrame (RSI_UDL)
        RSI_UDL_df.columns = ["RSI_UDL"]

        #Percent_rank - prince change (1ª)

        list1= []
        w=1
        while w < len (hist_df ["Close"]):
            Values_w = (hist_df ["Close"] [w] - hist_df ["Close"] [w-1]) / hist_df
["Close"] [w-1]
            list1.append(Values_w*100)
            w+=1


        #Percent rank - 2ª

        list2= []
        list3= []
        e=100
        while e < len (list1):
            for r in list1 [e-100:e]:
                if r < list1 [e]:
                    list2.append(r)
            list3.append(len(list2))
            list2.clear()
            e+=1


        Percent_rank = pd.DataFrame (list3)
        Percent_rank.columns = ["Percent Rank"]

        Percent_rank_real = pd.concat([nueva_df, Percent_rank],
axis=0).reset_index()
        Percent_rank_real_df = pd.DataFrame (Percent_rank_real ["Percent Rank"])

        Concat2 = pd.concat([RSI_df, RSI_UpDownLength_df, RSI_UDL_df,
Percent_rank_real_df ["Percent Rank"]], axis=1)

        LV_CRSI = (RSI_df ["RSI"] [len(RSI_df)-1] + RSI_UDL_df ["RSI_UDL"] [len
(RSI_UDL_df)-1] + Percent_rank_real_df ["Percent Rank"] [len
(Percent_rank_real_df)-1])/3
        print (i, LV_CRSI)

    except KeyError:
        continue
    except ValueError:
```

```
continue
```