```
#Setup
Apikey= "API key"
Secret= "Secret key"

from binance import Client, ThreadedWebsocketManager, ThreadedDepthCacheManager
import pandas as pd
pd.set_option('display.max_rows', 3000)
pd.set_option('display.max_columns', 3000)
pd.set_option('display.width', 1000)
pd.set_option('display.max_rows', 3000)
pd.set_option('display.max_columns', 3000)
pd.set_option('display.width', 3000)
import numpy as np
import mplfinance as mpf

#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)
    if "BUSD" in i:
        list_of_symbols.append (i)

print (list_of_symbols)

#Getting the price of each symbol

list_cryptocurrencies = []
list_values = []

for i in list_of_symbols:

    try:

            historical = client.get_historical_klines(""+i+"",
Client.KLINE_INTERVAL_1DAY, "02 Jan 2022") # the day before
            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"]
            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)
            hist_df_reduced = pd.DataFrame (hist_df, columns = ["Open time",
"Close"])

            historical_reduced_1h = client.get_historical_klines( "" + i + "",
Client.KLINE_INTERVAL_1HOUR, "02 Jan 2022" )  # of the day
            historical_reduced_1h_df = pd.DataFrame( historical_reduced_1h )
            historical_reduced_1h_df.columns = ["Open time 1h", "Open", "High",
"Low", "Close 1h", "Volume", "Close time 1h", "Quote asset volume", "Number of
trades","Taker buy base asset volume", "Taker buy quote asset volume","Can be
ignored"]
            historical_reduced_1h_df["Open time 1h"] = pd.to_datetime(
historical_reduced_1h_df["Open time 1h"] / 1000, unit="s" )
            historical_reduced_1h_df["Close time 1h"] =
pd.to_datetime(historical_reduced_1h_df["Close time 1h"] / 1000, unit="s" )
            numeric_columns_2 = ["Open", "High", "Low", "Close 1h", "Volume",
"Quote asset volume","Taker buy base asset volume", "Taker buy quote asset volume"]
            historical_reduced_1h_df[numeric_columns_2] =
historical_reduced_1h_df[numeric_columns_2].apply(pd.to_numeric, axis=1 )
            historical_reduced_1h_df_reduced = pd.DataFrame(
historical_reduced_1h_df,columns=["Open time 1h", "High", "Low"] )

            highp_list = [] # FALTARÍA HACER RESTAR UNA HORA A LA HORA ACTUAL /
AROONUP [1] AROONUP [0]
            for p in historical_reduced_1h_df_reduced ["High"]
[len(historical_reduced_1h_df_reduced)-14:len (historical_reduced_1h_df_reduced)]:
                highp_list.append (p)
            max_highp = max (highp_list )
            highp_list_index = highp_list.index( max_highp ) + 1
            substraction_highp = 14 - highp_list_index
            substraction_highp_len_highp_list = 14 - substraction_highp
            Aroon_up = (substraction_highp_len_highp_list / 14) * 100

            #INTENTO DE AROONUP DE LA HORA ANTERIOR PARA EL MISMO PERÍODO DE 14
DÍAS
            highp_list_one_less= []
            for t in historical_reduced_1h_df_reduced ["High"]
[len(historical_reduced_1h_df_reduced)-15:len
(historical_reduced_1h_df_reduced)-1]:
                highp_list_one_less.append (t)
            max_highp_one_less = max (highp_list_one_less)
            highp_list_index_one_less = highp_list_one_less.index
(max_highp_one_less) + 1
            substraction_highp_one_less = 14 - highp_list_index_one_less
            substraction_highp_len_highp_list_one_less = 14 -
```

```
substraction_highp_one_less
            Aroon_up_one_less = (substraction_highp_len_highp_list_one_less / 14) *
100

            lowp_list = []  # FALTARÍA HACER RESTAR UNA HORA A LA HORA ACTUAL /
AROONDOWN [1] AROONDOWN [0]
            for j in historical_reduced_1h_df_reduced ["Low"]
[len(historical_reduced_1h_df_reduced)-14:len (historical_reduced_1h_df_reduced)]:
                lowp_list.append( j )
            min_lowp = min( lowp_list )
            lowp_list_index = lowp_list.index( min_lowp ) + 1
            substraction_lowp =  14 - lowp_list_index
            substraction_lowp_len_lowp_list = 14 - substraction_lowp
            Aroon_down = (substraction_lowp_len_lowp_list / 14) * 100

            lowp_list_one_less = []
            for r in historical_reduced_1h_df_reduced["Low"][len(
historical_reduced_1h_df_reduced )-15:len( historical_reduced_1h_df_reduced)-1]:
                lowp_list_one_less.append( r )
            min_lowp_one_less = min (lowp_list_one_less)
            lowp_list_index_one_less = lowp_list_one_less.index (min_lowp_one_less)
+ 1
            substraction_lowp_one_less = 14 - lowp_list_index_one_less
            substraction_lowp_len_lowp_list_one_less = 14 -
substraction_lowp_one_less
            Aroon_down_one_less = (substraction_lowp_len_lowp_list_one_less / 14) *
100

            try:
                difference_close_price_1_day =  hist_df_reduced ["Close"] [1] -
hist_df_reduced ["Close"] [0]
                porcentage_difference_close_price_1_day = 100 - (hist_df_reduced
["Close"] [0] * 100 / hist_df_reduced ["Close"] [1])

                number_1 = Aroon_up
                number_2 = Aroon_down
                number_1_one_less = Aroon_up_one_less
                number_2_one_less = Aroon_down_one_less

            except KeyError:
                continue

            if porcentage_difference_close_price_1_day >= 0:

                if number_2_one_less > number_1_one_less:
                    negative_number_2_one_less = number_2_one_less * (-1)
                    if number_2 < number_1:
                        positive_number_1 = number_1
                        difference_close_price_1_hour_one_less_now =
positive_number_1 - negative_number_2_one_less
```

```
                        if difference_close_price_1_hour_one_less_now > 50:
                            list_cryptocurrencies.append( i )

list_values.append(difference_close_price_1_hour_one_less_now)
                            print( i, difference_close_price_1_hour_one_less_now )

                        elif difference_close_price_1_hour_one_less_now < 50:
                            continue

                    elif number_2 > number_1:
                        continue

                elif number_2_one_less < number_1_one_less: #ESTO PUEDE REVISARSE
DADO QUE PUEDE PASAR DE 0 A 90
                    continue

            elif porcentage_difference_close_price_1_day < 0:
                continue

    except ValueError:
       continue

excel_file = pd.DataFrame(list(zip(list_cryptocurrencies, list_values)), columns=
["Cryptocurrencies", "Values"])
#print (excel_file)

with pd.ExcelWriter
('C:\\Users\lucia\Desktop\Luciano\Programación\Criptocurrencies_30_31_diciembre.xls
x') as writer:
    excel_file.to_excel (writer, sheet_name="02-12-2021", index=False)
```