```
from iqoptionapi.stable_api import IQ_Option
import time
from configobj import ConfigObj
import json, sys
from datetime import datetime, timedelta
from catalogador import catag
from tabulate import tabulate
from colorama import init, Fore, Back
from iqoptionapi.constants import ACTIVES

init(autoreset=True)
green = Fore.GREEN
yellow = Fore.YELLOW
red = Fore.RED
white = Fore.WHITE
greenf = Back.GREEN
yellowf = Back.YELLOW
redf = Back.RED
blue = Fore.BLUE

'''+print(green

  ╚═══════╝  ╚══════╝
╚══════╝  ╚══╝   ╚══╝  ╚═══════╝
╚══════╝   ╚══════╝   ╚══════╝
╚═══════╝ '''+yellow+'''

    https://www.youtube.com/
@lucascode

('''

print(yellow +
'****************************************************
*********************************************\n\n')

#### CREATING CONFIGURATION FILE ###
config = ConfigObj('config.txt')
email = config['LOGIN']['email']
password = config['LOGIN']['senha']
account_type = config['AJUSTES']['tipo']
entry_amount = float(config['AJUSTES']
['valor_entrada'])
```

```
stop_win = float(config['AJUSTES']
['stop_win'])
stop_loss = float(config['AJUSTES']
['stop_loss'])
total_profit = 0
stop = True

if config['MARTINGALE']
:'['usar_martingale'].upper() == 'S
martingale = int(config['MARTINGALE']
['niveis_martingale'])
:else
martingale = 0
martingale_factor =
float(config['MARTINGALE']
['fator_martingale'])

if config['SOROS']['usar_soros'].upper() ==
:''S
soros = True
soros_levels = int(config['SOROS']
['niveis_soros'])
current_soros_level = 0
:else
soros = False
soros_levels = 0
current_soros_level = 0

soros_value = 0
current_trade_profit = 0

analyze_averages = config['AJUSTES']
['analise_medias']
average_candles = int(config['AJUSTES']
['velas_medias'])

print(yellow+'Starting Connection with
IQOption')
API = IQ_Option(email,password)

### Function to connect to IQOPTION ###
()check, reason = API.connect
:if check
```

```
print(green + '\nSuccessfully connected')
:else
if reason ==
'{"code":"invalid_credentials","message":"Y
ou entered the wrong credentials. Please
ensure that your login/password is
:'correct."}
print(red+'\nIncorrect email or
password')
()sys.exit
:else
print(red+ '\nThere was a connection
problem')
print(reason)
()sys.exit

Function to Select demo or real account ###
###
:while True
choice = input(green+'\n>>'+ white +'
Select the account you want to connect to:
+'\n
green+'>>'+ white +' 1 -
+'Demo\n
green+'>>'+ white +' 2 -
+'Real\n
(' '+ green+'-->'+ white

choice = int(choice)

:if choice == 1
'account = 'PRACTICE
print('Demo account selected')
break
:if choice == 2
'account = 'REAL
print('Real account selected')
break
:else
print(red+'Incorrect choice! Enter demo
or real')

API.change_balance(account)
```

```
Function to check stop win and loss ###
:()def check_stop
global stop, total_profit
if total_profit <=
:float('-'+str(abs(stop_loss)))
stop = False

print(red+'\n##########################
#')
print(red+'STOP LOSS HIT
',str(currency_symbol),str(total_profit))

print(red+'###########################')
()sys.exit

:if total_profit >= float(abs(stop_win))
stop = False

print(green+'\n########################
###')
print(green+'STOP WIN HIT
',str(currency_symbol),str(total_profit))

print(green+'##########################
#')
()sys.exit

:def payout(pair)
()profit = API.get_all_profit
()all_asset = API.get_all_open_time

:try
:if all_asset['binary'][pair]['open']
:if pair in ACTIVES
:if profit[pair]['binary'] > 0
binary = round(profit[pair]
['binary'],2) * 100
:else
binary = 0
:except
binary = 0
```

```
:try
:if all_asset['turbo'][pair]['open']
:if pair in ACTIVES
:if profit[pair]['turbo'] > 0
turbo = round(profit[pair]
['turbo'],2) * 100
:else
turbo = 0
:except
turbo = 0

:try
:if all_asset['digital'][pair]['open']
:if pair in ACTIVES
:if profit[pair]['turbo'] > 0
digital =
API.get_digital_payout(pair)
:else
digital = 0
:except
digital = 0

return binary, turbo, digital

Function to open order and check result ###
###
def buy(asset, entry_amount, direction,
:expiration, trade_type)
global stop, total_profit,
current_soros_level, soros_levels,
soros_value, current_trade_profit

:if soros
:if current_soros_level == 0
entry = entry_amount

if current_soros_level >=1 and
soros_value > 0 and current_soros_level <=
:soros_levels
entry = entry_amount + soros_value

:if current_soros_level > soros_levels
current_trade_profit = 0
```

```
soros_value = 0
entry = entry_amount
current_soros_level = 0
:else
entry = entry_amount

:for i in range(martingale + 1)
:if stop == True
:'if trade_type == 'digital
check, id =
API.buy_digital_spot_v2(asset, entry,
direction, expiration)
:else
check, id = API.buy(entry, asset,
direction, expiration)

:if check
:if i == 0
print(yellow + '\n>>'+white+'
Order opened \n'+yellow+'>>'+white+'
Pair:',asset,'\n'+yellow+'>>
'+white+'Timeframe:',expiration,'\n'+yellow+'
>>'+white+' Entry
amount:',currency_symbol,entry)
:if i >= 1
print(yellow + '\n>>'+white+'
Order opened for
martingale',str(i),'\n'+yellow+'>>'+white+'
Pair:',asset,'\n'+yellow+'>>
'+white+'Timeframe:',expiration,'\n'+yellow+'
>>'+white+' Entry
amount:',currency_symbol,entry)

:while True
time.sleep(0.1)
status, result =
API.check_win_digital_v2(id) if trade_type
== 'digital' else API.check_win_v4(id)

:if status
total_profit += round(result,2)
soros_value += round(result,2)
current_trade_profit +=
round(result,2)

:if result > 0
:if i == 0
print(green+ '\n>> Result:
WIN \n'+white+'>> Profit:', round(result,2),
'\n>> Pair:', asset, '\n>> Total profit: ',
round(total_profit,2))
:if i >= 1
print(green+ '\n>> Result:
WIN in martingale',str(i)+white+'\n>> Profit:',
round(result,2), '\n>> Pair:', asset, '\n>> Total
profit: ', round(total_profit,2))

:elif result == 0
:if i == 0
print(yellow +'\n>> Result:
DRAW \n'+white+'\>> Profit:', round(result,2),
'\n>> Pair:', asset, '\n>> Total profit: ',
round(total_profit,2))

:if i >= 1
print(yellow+'\n>> Result:
DRAW in martingale',str(i),'\n'+white+'>>
Profit:', round(result,2), '\n>> Pair:', asset,
'\n>> Total profit: ', round(total_profit,2))

:if i+1 <= martingale
martingale_amount =
float(entry)
entry =
round(abs(martingale_amount), 2)

:else
:if i == 0
print(red+'\n>> Result:
LOSS \n'+white+'>> Profit:', round(result,2),
'\n>> Pair:', asset, '\n>> Total profit: ',
round(total_profit,2))
:if i >= 1
print(red+'\n>> Result:
LOSS in martingale',str(i), '\n'+white+'>>
Profit:', round(result,2), '\n>> Pair:', asset,
'\n>> Total profit: ', round(total_profit,2))

:if i+1 <= martingale
martingale_amount =
float(entry) * float(martingale_factor)
entry =
round(abs(martingale_amount), 2)

()check_stop
break

:if result > 0
break

:else
print('Error opening order,', id, asset)

:if soros
:if current_trade_profit > 0
current_soros_level += 1
current_trade_profit = 0
:else
soros_value = 0
current_soros_level = 0
current_trade_profit = 0

### Function to get broker time ###
:()def get_time
now =
datetime.fromtimestamp(API.get_server_-
timestamp())
return now

:def moving_averages(candles)
sum = 0
:for i in candles
sum += i['close']
average = sum / average_candles

:if average > candles[-1]['close']
'trend = 'put
:else
'trend = 'call
```

```
return trend

MHI Strategy Analysis Function ###
:()def mhi_strategy
global account_type

:'if account_type == 'automatico
binary, turbo, digital = payout(asset)
print(binary, turbo, digital)
:if digital > turbo
print('Your entries will be made in
digital options')
'account_type = 'digital
:elif turbo > digital
print('Your entries will be made in
binary options')
'account_type = 'binary
:else
print('Pair closed, choose another')
()sys.exit

:while True
time.sleep(0.1)

### IQOption time ###
minutes =
float(datetime.fromtimestamp(API.get_serve
r_timestamp()).strftime('%M.%S')[1:])

entry_time = True if (minutes >= 4.59
and minutes <= 5.00) or minutes >= 9.59 else
False

print('Waiting for entry time', minutes,
end='\r')

:if entry_time
print('\n>> Starting MHI strategy
analysis')

direction = False
```

```
timeframe = 60
candle_count = 3

:'if analyze_averages == 'S
candles = API.get_candles(asset,
timeframe, average_candles, time.time())
trend = moving_averages(candles)
:else
candles = API.get_candles(asset,
timeframe, candle_count, time.time())

candles[-1] = 'Green' if candles[-1]
['open'] < candles[-1]['close'] else 'Red' if
candles[-1]['open'] > candles[-1]['close']
'else 'Doji
candles[-2] = 'Green' if candles[-2]
['open'] < candles[-2]['close'] else 'Red' if
candles[-2]['open'] > candles[-2]['close']
'else 'Doji
candles[-3] = 'Green' if candles[-3]
['open'] < candles[-3]['close'] else 'Red' if
candles[-3]['open'] > candles[-3]['close']
'else 'Doji

colors = candles[-3], candles[-2],
candles[-1]

if colors.count('Green') >
colors.count('Red') and colors.count('Doji')
'== 0: direction = 'put
if colors.count('Green') <
colors.count('Red') and colors.count('Doji')
'== 0: direction = 'call

:'if analyze_averages =='S
:if direction == trend
pass
:else
'direction = 'abort

if direction == 'put' or direction ==
:''call
print('Candles: ',candles[-3],
candles[-2], candles[-1], ' - Entry for',
direction)
buy(asset, entry_amount, direction,
1, account_type)
print('\n')
:else
:'if direction == 'abort
print('Candles: ',candles[-3],
candles[-2], candles[-1])
print('Entry aborted - Against
Trend.')
:else
print('Candles: ',candles[-3],
candles[-2], candles[-1])
print('Entry aborted - A Doji was
found in the analysis.')

time.sleep(2)


print('\n############################
####################################
########\n')

Twin Towers Strategy Analysis Function ###
:()def twin_towers_strategy
global account_type

:'if account_type == 'automatico
binary, turbo, digital = payout(asset)
print(binary, turbo, digital)
:if digital > turbo
print('Your entries will be made in
digital options')
'account_type = 'digital
:elif turbo > digital
print('Your entries will be made in
binary options')
'account_type = 'binary
:else
print('Pair closed, choose another')
()sys.exit
```

```
:while True
time.sleep(0.1)

minutes =
float(datetime.fromtimestamp(API.get_serve
r_timestamp()).strftime('%M.%S')[1:])

entry_time = True if (minutes >= 3.59
and minutes <= 4.00) or (minutes >= 8.59
and minutes <= 9.00) else False

print('Waiting for entry time', minutes,
end='\r')

:if entry_time
print('\n>> Starting Twin Towers
strategy analysis')

direction = False

timeframe = 60
candle_count = 4

:'if analyze_averages == 'S
candles = API.get_candles(asset,
timeframe, average_candles, time.time())
trend = moving_averages(candles)
:else
candles = API.get_candles(asset,
timeframe, candle_count, time.time())

candles[-4] = 'Green' if candles[-4]
['open'] < candles[-4]['close'] else 'Red' if
candles[-4]['open'] > candles[-4]['close']
'else 'Doji

colors = candles[-4]

if colors.count('Green') >
colors.count('Red') and colors.count('Doji')
'== 0: direction = 'call
if colors.count('Green') <
colors.count('Red') and colors.count('Doji')
```

```
'== 0: direction = 'put

:'if analyze_averages =='S
:if direction == trend
pass
:else
'direction = 'abort

if direction == 'put' or direction ==
:''call
print('Candles: ',candles[-3],
candles[-2], candles[-1], ' - Entry for',
direction)
buy(asset, entry_amount, direction,
1, account_type)
print('\n')
:else
:'if direction == 'abort
print('Candles: ',candles[-3],
candles[-2], candles[-1])
print('Entry aborted - Against
Trend.')
:else
print('Candles: ',candles[-3],
candles[-2], candles[-1])
print('Entry aborted - A Doji was
found in the analysis.')

time.sleep(2)


print('\n################################
######################################
########\n')

MHI M5 Strategy Analysis Function ###
:()def mhi_m5_strategy
global account_type

:'if account_type == 'automatico
binary, turbo, digital = payout(asset)
print(binary, turbo, digital)
:if digital > turbo
```

```
print('Your entries will be made in
digital options')
'account_type = 'digital
:elif turbo > digital
print('Your entries will be made in
binary options')
'account_type = 'binary
:else
print('Pair closed, choose another')
()sys.exit

:while True
time.sleep(0.1)

minutes =
float(datetime×fromtimestamp(API×get_serv
er_timestamp()).strftime('%M.%S'))

entry_time = True if (minutes >= 29.59
and minutes <= 30.00) or minutes == 59.59
else False

print('Waiting for entry time', minutes,
end='\r')

:if entry_time
print('\n>> Starting MHI M5 strategy
analysis')

direction = False

timeframe = 300
candle_count = 3

:'if analyze_averages == 'S
candles = API.get_candles(asset,
timeframe, average_candles, time.time())
trend = moving_averages(candles)
:else
candles = API.get_candles(asset,
timeframe, candle_count, time.time())

candles[-1] = 'Green' if candles[-1]
```

```
['open'] < candles[-1]['close'] else 'Red' if
 candles[-1]['open'] > candles[-1]['close']
                                         'else 'Doji
 candles[-2] = 'Green' if candles[-2]
['open'] < candles[-2]['close'] else 'Red' if
 candles[-2]['open'] > candles[-2]['close']
                                         'else 'Doji
 candles[-3] = 'Green' if candles[-3]
['open'] < candles[-3]['close'] else 'Red' if
 candles[-3]['open'] > candles[-3]['close']
                                         'else 'Doji

   colors = candles[-3], candles[-2],
                                       candles[-1]

             if colors.count('Green') >
colors.count('Red') and colors.count('Doji')
                             '== 0: direction = 'put
             if colors.count('Green') <
colors.count('Red') and colors.count('Doji')
                             '== 0: direction = 'call

            :'if analyze_averages =='S
               :if direction == trend
                           pass
                              :else
               'direction = 'abort

   if direction == 'put' or direction ==
                                            :''call
        print('Candles: ',candles[-3],
        candles[-2], candles[-1], ' - Entry for',
                                         direction)
buy(asset, entry_amount, direction,
                                 5, account_type)
                                print('\n')
                                        :else
              :'if direction == 'abort
      print('Candles: ',candles[-3],
                        candles[-2], candles[-1])
   print('Entry aborted - Against
                                             Trend.')
                                     :else
```

```
print('Candles: ',candles[-3],
candles[-2], candles[-1])
print('Entry aborted - A Doji was
found in the analysis.')

time.sleep(2)



print('\n############################
##################################
########\n')

INPUT DEFINITION AT THE START OF ###
### THE ROBOT

profile =
json×loads(json×dumps(API×get_pro-
file_ansyc()))
currency_symbol =
str(profile['currency_char'])
name = str(profile['name'])

account_value = float(API×get_balance())

print(yellow+'\n######################
##################################
##############')
print('\nHello, ',name, '\nWelcome to Lucas
Channel Robot.')
print('\nYour balance in account',choice, 'is',
currency_symbol,account_value)
print('\nYour entry amount
is',currency_symbol,entry_amount)
print('\nStop
win:',currency_symbol,stop_win)
print('\nStop
loss:',currency_symbol,'-',stop_loss)
print(yellow+'\n######################
##################################
##############\n\n')

print('>> Starting cataloging')
catalog_list, line = catag(API)
```

```
print(yellow+ tabulate(catalog_list,
headers=['STRATEGY','PAIR','WIN','MARTIN-
GALE1','MARTINGALE2']))

strategy = catalog_list[0][0]
asset = catalog_list[0][1]
accuracy = catalog_list[0][line]

print('\n>> Best pair: ', asset, ' | Strategy:
',strategy,' | Accuracy: ', accuracy)
print('\n')

### Function to choose strategy ###
:while True
strategy_choice = input(green+'\n>>'+
+'white +' Select the desired strategy:\n
green+'>>'+ white +' 1 -
+'MHI\n
green+'>>'+ white +' 2 - Twin
+'Towers\n
green+'>>'+ white +' 3 - MHI
+'M5\n
(' '+ green+'-->'+ white

strategy_choice = int(strategy_choice)

:if strategy_choice == 1
break
:if strategy_choice == 2
break
:if strategy_choice == 3
break
:else
print(red+'Incorrect choice! Enter 1 to 3')

asset = input(green+ '\n>>'+white+' Enter the
()asset you want to trade: ').upper
print('\n')

:if strategy_choice == 1
()mhi_strategy
:if strategy_choice == 2
```

```
()twin_towers_strategy
:if strategy_choice == 3
()mhi_m5_strategy
```