***For automatic strike price selection + condition***

import requests

import schedule

import time

# API Key and Base URL

api\_key = "YOUR\_API\_KEY"

base\_url = "https://smartapi.angelbroking.com"

# Authentication Headers

headers = {

"Authorization": f"Bearer {api\_key}"

}

# Store high and low of last 2 candles

high\_of\_last\_2\_candles = None

low\_of\_last\_2\_candles = None

# Option symbol

option\_symbol = None

# Shares in the position

shares\_in\_position = 0

def fetch\_option\_symbol():

global option\_symbol

# Fetch list of option instruments

endpoint = "/instruments/optidx" # Option instruments for Nifty index

params = {

"expiry": "2023-08-25", # Replace with desired expiry date

"optionType": "CE", # Call option

"strike": "ATM", # ATM strike

}

response = requests.get(base\_url + endpoint, headers=headers, params=params)

data = response.json()

# Extract option symbol for nearest ATM call option

if data and "data" in data:

option\_symbol = data["data"][0]["symbol"]

def fetch\_market\_data():

global high\_of\_last\_2\_candles, low\_of\_last\_2\_candles

# Fetch historical candle data for the option

endpoint = "/quote/historical/{symbol}" # Replace {symbol} with the option symbol

params = {

"interval": "1", # 1-minute interval

"fromDate": "2023-08-23T09:15:00", # Replace with your desired start date and time

"toDate": "2023-08-23T15:15:00", # Replace with your desired end date and time

}

response = requests.get(base\_url + endpoint, headers=headers, params=params)

data = response.json()

# Extract high and low of last 2 candles

high\_of\_last\_2\_candles = max(data[-1]["high"], data[-2]["high"])

low\_of\_last\_2\_candles = min(data[-1]["low"], data[-2]["low"])

def start\_trading():

global option\_symbol, high\_of\_last\_2\_candles, low\_of\_last\_2\_candles, shares\_in\_position

if not option\_symbol:

fetch\_option\_symbol()

# Fetch recent market data for the option

fetch\_market\_data()

# Fetch live LTP (Last Traded Price) for the option

ltp\_endpoint = f"/quote/ltp/{option\_symbol}"

ltp\_response = requests.get(base\_url + ltp\_endpoint, headers=headers)

ltp\_data = ltp\_response.json()

ltp = ltp\_data["data"][0]["last\_price"]

if ltp > high\_of\_last\_2\_candles and shares\_in\_position == 0:

# Place a buy order for 2 contracts of the option using the API

# Implement your buy order logic for options here

shares\_in\_position = 2

print("Placing Buy Order for 2 option contracts")

elif shares\_in\_position > 0:

# Check if LTP increased by 7% or crossed below the low of last 2 candles

if (ltp - ltp\_data["data"][0]["open"]) / ltp\_data["data"][0]["open"] >= 0.07:

# Place a sell order for 1 contract of the option

# Implement your sell order logic for options here

shares\_in\_position -= 1

print("Placing Sell Order for 1 option contract (7% increase)")

elif ltp < low\_of\_last\_2\_candles:

# Place a sell order for all remaining contracts of the option

# Implement your sell order logic for options here

shares\_in\_position = 0

print("Placing Sell Order for remaining option contracts (LTP below low of last 2 candles)")

def stop\_trading():

global shares\_in\_position

# Implement logic to close any open positions and stop trading

shares\_in\_position = 0

print("Stopping Trading")

# Schedule trading session

schedule.every().day.at("09:15").do(fetch\_option\_symbol) # Fetch option symbol before trading

schedule.every().day.at("09:16").do(fetch\_market\_data) # Fetch initial data before trading

schedule.every().day.at("09:17").do(start\_trading)

schedule.every().day.at("15:15").do(stop\_trading)

while True:

schedule.run\_pending()

time.sleep(1)