import requests

import hmac

import hashlib

import time

import json

# Deine Bitget API-Schlüssel

api\_key = 'YOUR\_API\_KEY'

api\_secret = 'YOUR\_API\_SECRET'

passphrase = 'YOUR\_PASSPHRASE'

# API-Endpunkt

base\_url = 'https://api.bitget.com'

# Signatur für die API-Authentifizierung erstellen

def generate\_signature(api\_secret, timestamp, method, request\_path, body):

message = f"{timestamp}{method}{request\_path}{body}"

signature = hmac.new(api\_secret.encode('utf-8'), message.encode('utf-8'), hashlib.sha256).hexdigest()

return signature

# HTTP-Header erstellen

def get\_headers(api\_key, passphrase, signature, timestamp):

return {

'Content-Type': 'application/json',

'ACCESS-KEY': api\_key,

'ACCESS-SIGN': signature,

'ACCESS-TIMESTAMP': timestamp,

'ACCESS-PASSPHRASE': passphrase

}

# Order-Parameter

symbol = "BTCUSDT"

marginCoin = "USDT"

leverage = 20

amount\_usdc = 30 # Summe in USDC

limit\_price = 97000 # Limit Preis

stop\_loss\_price = 96700 # Stop Loss Preis

take\_profit\_price = 99000 # Take Profit Preis

# Timestamp erstellen

timestamp = str(int(time.time() \* 1000))

# Hebel setzen

try:

request\_path = f"/api/mix/v1/account/setLeverage"

method = "POST"

body = json.dumps({

"symbol": symbol,

"marginCoin": marginCoin,

"leverage": leverage

})

signature = generate\_signature(api\_secret, timestamp, method, request\_path, body)

headers = get\_headers(api\_key, passphrase, signature, timestamp)

response = requests.post(base\_url + request\_path, headers=headers, data=body)

print("Hebel erfolgreich gesetzt:", response.json())

except Exception as e:

print(f"Fehler beim Setzen des Hebels: {e}")

# Limit-Order platzieren

try:

request\_path = f"/api/mix/v1/order/placeOrder"

method = "POST"

quantity = round(amount\_usdc / limit\_price, 6) # Berechnung der Menge in BTC

body = json.dumps({

"symbol": symbol,

"marginCoin": marginCoin,

"size": str(quantity),

"side": "open\_long", # Long-Position

"orderType": "limit",

"price": str(limit\_price),

"timeInForceValue": "normal"

})

signature = generate\_signature(api\_secret, timestamp, method, request\_path, body)

headers = get\_headers(api\_key, passphrase, signature, timestamp)

response = requests.post(base\_url + request\_path, headers=headers, data=body)

print("Limit-Order erfolgreich platziert:", response.json())

except Exception as e:

print(f"Fehler beim Platzieren der Limit-Order: {e}")

# Take-Profit und Stop-Loss einrichten

try:

# Take-Profit-Order

request\_path\_tp = f"/api/mix/v1/order/placeOrder"

body\_tp = json.dumps({

"symbol": symbol,

"marginCoin": marginCoin,

"size": str(quantity),

"side": "close\_long", # Schließt die Long-Position

"orderType": "market",

"price": str(take\_profit\_price),

"triggerPrice": str(take\_profit\_price),

"triggerType": "fill\_price",

"timeInForceValue": "normal"

})

signature\_tp = generate\_signature(api\_secret, timestamp, method, request\_path\_tp, body\_tp)

headers\_tp = get\_headers(api\_key, passphrase, signature\_tp, timestamp)

response\_tp = requests.post(base\_url + request\_path\_tp, headers=headers\_tp, data=body\_tp)

print("Take-Profit-Order erfolgreich platziert:", response\_tp.json())

# Stop-Loss-Order

request\_path\_sl = f"/api/mix/v1/order/placeOrder"

body\_sl = json.dumps({

"symbol": symbol,

"marginCoin": marginCoin,

"size": str(quantity),

"side": "close\_long", # Schließt die Long-Position

"orderType": "market",

"price": str(stop\_loss\_price),

"triggerPrice": str(stop\_loss\_price),

"triggerType": "fill\_price",

"timeInForceValue": "normal"

})

signature\_sl = generate\_signature(api\_secret, timestamp, method, request\_path\_sl, body\_sl)

headers\_sl = get\_headers(api\_key, passphrase, signature\_sl, timestamp)

response\_sl = requests.post(base\_url + request\_path\_sl, headers=headers\_sl, data=body\_sl)

print("Stop-Loss-Order erfolgreich platziert:", response\_sl.json())

except Exception as e:

print(f"Fehler beim Platzieren von Take-Profit- oder Stop-Loss-Order: {e}")