```
# utils.py

import pandas as pd

import numpy as np

import ta

import smtplib

from email.mime.text import MIMEText

from email.mime.multipart import
MIMEMultipart

from config import GMAIL_USER,
GMAIL_PASS, BITGET_API_KEY,
BITGET_API_SECRET,
BITGET_PASSPHRASE

from bitget_api_client import BitgetClient
```

```
import os

import requests

import time

client = BitgetClient(BITGET_API_KEY,
BITGET_API_SECRET,
BITGET_PASSPHRASE)

def get_ohlcv(symbol, limit=100,
interval='1m'):

    url = f"https://api.bitget.com/api/spot/
v1/market/candles?symbol={symbol}
&period={interval}&limit={limit}"

    try:

        response = requests.get(url,
```

```
timeout=5)

        data = response.json().get('data', [])

        if not data:

            return None

        df = pd.DataFrame(data,
columns=['timestamp', 'open', 'high', 'low',
'close', 'volume'])

        df['close'] = df['close'].astype(float)

        df['volume'] = df['volume'].astype(float)

        return df[::-1].reset_index(drop=True)

    except Exception as e:

        print(f"[OHLCV] فشل في جلب بيانات
```

```
("{e} :الشموع

        return None

def analyze_liquidity(token, onchain=False):

    try:

        df = get_ohlcv(token)

        if df is None or len(df) < 20:

            return False

# تحليل الفوليوم: تأكد أن هناك نشاط تداول جيد

        avg_volume =
df['volume'].rolling(window=10).mean().iloc
[-1]

        if avg_volume < 1000: # ،رقم افتراضي
```

```
يمكن تعديله حسب الحاجة

            print(f"[Liquidity] حجم تداول ضعيف لـ
{token}")

            return False

# تحليل الاتجاه Trend: صعود أم هبوط

        if df['close'].iloc[-1] < df['close'].iloc[-5]:

            print(f"[Trend] الاتجاه هابط لـ {token}")

            return False

        return True

    except Exception as e:

        print(f"[Liquidity/Trend] خطأ: {e}")
```

```
        return False

def technical_analysis(token,
onchain=False):

    try:

        df = get_ohlcv(token)

        if df is None or len(df) < 20:

            return False

        df['rsi'] =
ta.momentum.RSIIndicator(df['close'],
14).rsi()

        macd = ta.trend.MACD(df['close'])

        df['macd'] = macd.macd()
```

```
        df['macd_signal'] =
macd.macd_signal()

        latest_rsi = df['rsi'].iloc[-1]

        return 30 < latest_rsi < 70 and
df['macd'].iloc[-1] > df['macd_signal'].iloc[-1]

    except Exception as e:

        print(f"[TA] خطأ أثناء التحليل الفني: {e}")

        return False

def execute_spot_order(symbol,
amount_usd):

    try:

        ticker = client.get_ticker(symbol)
```

```
        if not ticker:

            return False

        price = float(ticker['last'])

        quantity = round(amount_usd / price,
6)

        order = client.place_order(symbol,
None, quantity, 'buy', 'market')

        if order and 'orderId' in order:

            log_trade(symbol, amount_usd)

            return True

        return False

    except Exception as e:
```

```
        print(f"[Order] خطأ في تنفيذ أمر شراء {symbol}: {e}")
        return False

def place_sell_order(symbol):
    try:
        ticker = client.get_ticker(symbol)
        if not ticker:
            return False
        price = float(ticker['last'])
        quantity = round(50 / price, 6) # بيع نفس الكمية التي اشتريت بها (مقدار 50 دولار)
```

```
        order = client.place_order(symbol, None, quantity, 'sell', 'market')

        if order and 'orderId' in order:

            print(f"[Sell Order] بيع ناجح لـ {symbol}")

            return True

        return False

    except Exception as e:

        print(f"[Sell Order] خطأ في تنفيذ أمر بيع {symbol}: {e}")

        return False

def log_trade(symbol, amount):
```

```
    os.makedirs("logs", exist_ok=True)

    with open("logs/trades.log", "a") as f:

        f.write(f"{symbol},{amount},
{time.time()}\n")

def remove_trade(symbol):

    if not os.path.exists("logs/trades.log"):

        return

    with open("logs/trades.log", "r") as f:

        lines = f.readlines()

    with open("logs/trades.log", "w") as f:

        for line in lines:
```

```
            if not line.startswith(symbol + ','):
                f.write(line)
def monitor_open_trades():
    try:
        if not os.path.exists("logs/trades.log"):
            return
        with open("logs/trades.log", "r") as f:
            lines = f.readlines()
        for line in lines:
            symbol, amount, timestamp = line.strip().split(',')
```

```
        df = get_ohlcv(symbol)

        if df is None or len(df) < 20:

            continue

        bb =
ta.volatility.BollingerBands(df['close'])

        upper = bb.bollinger_hband().iloc[-1]

        lower = bb.bollinger_lband().iloc[-1]

        close = df['close'].iloc[-1]

# تنفيذ Take Profit عند تجاوز الحد الأعلى

        if close > upper:

            if place_sell_order(symbol):
```

```
                remove_trade(symbol)

                send_email_notification(f"بيع تلقائي لـ {symbol}", "تم تحقيق هدف الربح (Take Profit)")

            continue

        # تنفيذ Stop Loss عند كسر الحد الأدنى

        if close < lower:

            if place_sell_order(symbol):

                remove_trade(symbol)

                send_email_notification(f"بيع تلقائي لـ {symbol}", "تم تفعيل وقف الخسارة (Stop Loss)")

    except Exception as e:
```

```
        print(f"[Monitor] خطأ في مراقبة الصفقات: {e}")

def send_email_notification(subject, body):
    try:
        msg = MIMEMultipart()
        msg['From'] = GMAIL_USER
        msg['To'] = GMAIL_USER
        msg['Subject'] = subject
        msg.attach(MIMEText(body, 'plain'))
        server = smtplib.SMTP('smtp.gmail.com', 587)
```

```
        server.starttls()

        server.login(GMAIL_USER,
GMAIL_PASS)

        server.sendmail(GMAIL_USER,
GMAIL_USER, msg.as_string())

        server.quit()

    except Exception as e:

        print(f"[Email] Error: {e}")
```