파이썬 기반 거래 시스템 API 호출 메소드 조사 결과

바이낸스 파이썬 SDK

공식 라이브러리

- **binance-sdk-spot**: 현물 거래용 공식 SDK
- **binance-connector-python**: 통합 커넥터 (2025년 8월 최신 업데이트)

설치 방법

```
pip install binance-sdk-spot
```

기본 설정 및 초기화

```
 import os
import logging
from binance_common.configuration import ConfigurationRestAPI
from binance_common.constants import SPOT_REST_API_PROD_URL
from binance_sdk_spot.spot import Spot
from binance_sdk_spot.rest_api.models import NewOrderSideEnum, NewOrderTypeEnum

# 로깅 설정
logging.basicConfig(level=logging.INFO)

# API 설정
configuration = ConfigurationRestAPI(
    api_key="your-api-key",
    api_secret="your-api-secret",
    base_path=SPOT_REST_API_PROD_URL
)

# 클라이언트 초기화
client = Spot(config_rest_api=configuration)
```

주문 생성 예제 (new_order)

```
def new_order():
    try:
        response = client.rest_api.new_order(
            symbol="BNBUSDT",
            side=NewOrderSideEnum["BUY"].value,
            type=NewOrderTypeEnum["MARKET"].value,
        )

        data = response.data()
        logging.info(f"new_order() response: {data}")

    except Exception as e:
        logging.error(f"new_order() error: {e}")
```

지원하는 주문 타입

- LIMIT: 지정가 주문
- MARKET: 시장가 주문
- STOP_LOSS: 손절 주문
- STOP_LOSS_LIMIT: 손절 지정가 주문
- TAKE_PROFIT: 익절 주문
- TAKE_PROFIT_LIMIT: 익절 지정가 주문
- LIMIT_MAKER: 메이커 전용 지정가 주문

주문 취소

```
def delete_order():
    try:
        response = client.rest_api.delete_order(
            symbol="BNBUSDT",
            order_id=12345
        )
        data = response.data()
        logging.info(f"delete_order() response: {data}")
    except Exception as e:
        logging.error(f"delete_order() error: {e}")
```

계정 정보 조회

```
def account_info():
    try:
        response = client.rest_api.account()
        data = response.data()
        logging.info(f"account() response: {data}")
    except Exception as e:
        logging.error(f"account() error: {e}")
```

바이비트 파이썬 SDK

공식 라이브러리

- **pybit**: 바이비트 공식 파이썬 SDK

설치 방법

```
pip install pybit
```

기본 설정 및 초기화

```
from pybit.unified_trading import HTTP

# 메인넷
session = HTTP(
    testnet=False,
    api_key="your_api_key",
    api_secret="your_api_secret",
)

# 테스트넷
session = HTTP(
    testnet=True,
    api_key="your_testnet_api_key",
    api_secret="your_testnet_api_secret",
)
```

주문 생성 (V5 API)

```
def place_order():
    try:
        result = session.place_order(
            category="spot",  # spot, linear, inverse, option
            symbol="BTCUSDT",
            side="Buy",  # Buy, Sell
            orderType="Market",  # Market, Limit
            qty="0.01",
        )
        print(result)
    except Exception as e:
        print(f"Error: {e}")
```

지정가 주문

```
def place_limit_order():
    try:
        result = session.place_order(
            category="linear",  # 선물 거래
            symbol="BTCUSDT",
            side="Buy",
            orderType="Limit",
            qty="0.1",
            price="50000",
            timeInForce="GTC"  # GTC, IOC, FOK, PostOnly
        )
        print(result)
    except Exception as e:
        print(f"Error: {e}")
```

조건부 주문 (Stop Loss / Take Profit)

```
def place_conditional_order():
    try:
        result = session.place_order(
            category="linear",
            symbol="BTCUSDT",
            side="Buy",
            orderType="Limit",
            qty="0.1",
            price="50000",
            triggerPrice="49000",  # 트리거 가격
            stopLoss="48000",      # 손절가
            takeProfit="52000"     # 익절가
        )
        print(result)
    except Exception as e:
        print(f"Error: {e}")
```

주문 취소

```
def cancel_order():
    try:
        result = session.cancel_order(
            category="spot",
            symbol="BTCUSDT",
            orderId="order_id_here"
        )
        print(result)
    except Exception as e:
        print(f"Error: {e}")
```

포지션 정보 조회

```
def get_positions():
    try:
        result = session.get_positions(
            category="linear",
            symbol="BTCUSDT"
        )
        print(result)
    except Exception as e:
        print(f"Error: {e}")
```

계정 잔고 조회

```
def get_wallet_balance():
    try:
        result = session.get_wallet_balance(
            accountType="UNIFIED"  # UNIFIED, CONTRACT, SPOT
        )
        print(result)
    except Exception as e:
        print(f"Error: {e}")
```

WebSocket 실시간 데이터

바이낸스 WebSocket

```python
import asyncio
from binance_common.configuration import ConfigurationWebSocketAPI
from binance_common.constants import SPOT_WS_API_PROD_URL

configuration_ws_api = ConfigurationWebSocketAPI(
    api_key="your-api-key",
    api_secret="your-api-secret",
    stream_url=SPOT_WS_API_PROD_URL
)

client = Spot(config_ws_api=configuration_ws_api)

async def websocket_example():
    connection = None
    try:
        connection = await client.websocket_api.create_connection()

        response = await connection.account_commission(
            symbol="BNBUSDT",
        )
        result = response.data()
        print(f"WebSocket Response: {result}")

    except Exception as e:
        print(f"WebSocket error: {e}")
    finally:
        if connection:
            await connection.close_connection(close_session=True)

# 실행
asyncio.run(websocket_example())
```

바이비트 WebSocket

```python
from pybit.unified_trading import WebSocket

def handle_message(message):
    print(f"Received: {message}")

# WebSocket 연결
ws = WebSocket(
    testnet=False,
    channel_type="private",  # private, public
    api_key="your_api_key",
    api_secret="your_api_secret"
)

# 구독
ws.order_stream(callback=handle_message)
```

에러 처리

바이낸스 에러 타입

- ClientError: SDK 클라이언트 오류
- RequiredError: 필수 파라미터 누락
- UnauthorizedError: 인증 오류
- ForbiddenError: 접근 금지
- TooManyRequestsError: 요청 한도 초과
- RateLimitBanError: IP 차단
- ServerError: 서버 내부 오류
- NetworkError: 네트워크 연결 오류

바이비트 에러 처리

```
from pybit.exceptions import InvalidRequestError

try:
    result = session.place_order(...)
except InvalidRequestError as e:
    print(f"Invalid request: {e}")
except Exception as e:
    print(f"Unexpected error: {e}")
```

테스트넷 사용

바이낸스 테스트넷

```
from binance_common.constants import SPOT_REST_API_TESTNET_URL

configuration = ConfigurationRestAPI(
    api_key="testnet-api-key",
    api_secret="testnet-api-secret",
    base_path=SPOT_REST_API_TESTNET_URL
)
```

바이비트 테스트넷

```
session = HTTP(
    testnet=True,
    api_key="testnet_api_key",
    api_secret="testnet_api_secret"
)
```

보안 고려사항

1. **API 키 관리**: 환경변수 사용 권장

```
import os
api_key = os.getenv('BINANCE_API_KEY')
api_secret = os.getenv('BINANCE_API_SECRET')
```

1. **권한 설정**: 필요한 최소 권한만 부여
2. 현물 거래: Spot Trading 권한
3. 선물 거래: Futures Trading 권한
4. 읽기 전용: Read Info 권한
5. **IP 화이트리스트**: API 키에 IP 제한 설정
6. **요청 제한**: Rate Limit 준수
7. 바이낸스: 1200 requests/minute
8. 바이비트: 120 requests/minute (일반)

실제 구현 시 고려사항

1. **비동기 처리**: asyncio 사용 권장
2. **재시도 로직**: 네트워크 오류 대응
3. **로깅**: 모든 거래 기록 저장
4. **백테스팅**: 전략 검증 후 실거래 적용
5. **리스크 관리**: 손절/익절 로직 필수 구현