

Shioaji

Usage Manual

SinoPac Securities, Sinotrade

Copyright © 2025 SinoPac

Table of contents

1. Shioaji	4
1.1 安裝	4
2. AI 輔助開發	5
2.1 可用資源	5
2.2 最佳實踐	6
3. 快速入門	7
3.1 總結	8
4. 環境設定	9
4.1 系統需求	9
4.2 安裝 Python 環境	9
4.3 創建專案環境	9
5. 教程 - 使用者指南	12
5.1 前置作業	12
5.2 登入	27
5.3 商品檔	31
5.4 行情資料	35
5.5 下單	67
5.6 CallBack	105
5.7 帳務	113
5.8 模擬模式	129
5.9 進階指南	130
6. 升版指南	145
6.1 Shioaji 物件	145
6.2 登入	147
6.3 證券下單	147
6.4 期貨下單	150
6.5 行情資料	153
6.6 期貨帳務資訊	155
7. 問與答	156
8. 發佈版本	159
8.1 version: 1.3.1 (2025-12-29)	159
8.2 version: 1.3.0 (2025-12-17)	159
8.3 version: 1.2.9 (2025-10-29)	159
8.4 version: 1.2.8 (2025-09-10)	159

8.5 version: 1.2.7 (2025-08-13)	159
8.6 version: 1.2.6 (2025-06-16)	159
8.7 version: 1.2.5 (2024-10-01)	160
8.8 version: 1.2.4 (2024-08-28)	160
8.9 version: 1.2.3 (2024-03-06)	160
8.10 version: 1.2.2 (2024-01-09)	160
8.11 version: 1.2.1 (2023-12-22)	160
8.12 version: 1.2.0 (2023-12-20)	161
8.13 version: 1.1.13 (2023-11-01)	161
8.14 version: 1.1.12 (2023-08-22)	161
8.15 version: 1.1.11 (2023-08-04)	161
9. 使用限制	162

1. Shioaji



[PyPI](#) - [Status](#) [PyPI](#) - [Python Version](#) [PyPI](#) - [Downloads](#) [Build](#) - [Status](#) [Coverage](#) [Binder](#) [doc](#) [Telegram](#)

Shioaji 是一個使用 Python 語言的應用程式介面，提供投資者在台灣和全球金融市場上進行交易。此外，使用者可以利用 Shioaji 為基礎整合像 NumPy、pandas、PyTorch 或 TensorFlow 等流行的 Python 套件，創造出專屬於自己的跨平台交易模型。

特色:

- 高效率: 使用 C++ 作為核心邏輯和 FPGA 作為訊息交換
- 簡單: 設計為易於使用和學習
- 快速編譯: 使用原生 Python 集成大型 Python 生態系統
- 跨平台: 台灣第一個兼容 Linux 的 Python 交易應用程式介面
- AI 輔助開發: 台灣第一個支援 [AI Coding Agent Skills](#) 的交易 API，讓 AI 助手幫你寫程式

1.1 安裝

1.1.1 Binaries

使用 pip 簡單安裝

```
pip install shioaji
```

更新 shioaji

```
pip install -U shioaji
```

1.1.2 uv

使用 uv 安裝

```
uv add shioaji
```

安裝快速版本

```
uv add shioaji --extra speed
```

1.1.3 Docker Image

在 Docker 中以互動模式執行

```
docker run -it sinotrade/shioaji:latest
```

在 Jupyter Lab 或 Jupyter Notebook 執行

```
docker run -p 8888:8888 sinotrade/shioaji:jupyter
```

2. AI 輔助開發

Shioaji 是台灣第一個支援 LLM.txt 與 AI Coding Agent Skills 的金融 API。

我們提供 AI 友善的資源，幫助您將交易 API 與 AI 程式開發助手整合。無論您使用 Claude Code、Cursor 或其他 AI 開發工具，這些資源都能讓 AI 理解並協助 Shioaji API 的開發。

2.1 可用資源

2.1.1 LLM.txt

我們提供機器可讀的文件檔案，讓 AI 助手能夠理解 Shioaji API：

檔案	說明
llms.txt	所有文件頁面的結構化連結
llms-full.txt	完整的純文字文件內容

這些檔案遵循 [llms.txt 標準](#)，並在文件更新時自動同步。

使用方式：只需將網址提供給您的 AI 助手：

```
https://sinotrade.github.io/llms-full.txt
```

2.1.2 AI Coding Agent Skills

若需更深度的整合，我們為主流 AI 程式開發助手提供 skills/plugins：

Claude Code

[Claude Code](#) 是 Anthropic 官方的 AI 輔助開發 CLI 工具。

安裝 Shioaji skill：

```
claude plugin marketplace add Sinotrade/Shioaji  
claude plugin install shioaji
```

OpenAI Codex CLI

在 Codex 工作階段中，使用 skill-installer：

```
$skill-installer install shioaji from Sinotrade/Shioaji --path skills/shioaji
```

2.1.3 Skill 功能

安裝後，AI 程式開發助手可協助您：

- 認證與登入流程
- 商品檔規格（股票、期貨、選擇權）
- 下單與委託管理
- 即時行情資料串流
- 帳戶餘額與部位查詢
- 歷史資料擷取

範例提示詞：

如何使用 Shioaji 訂閱即時股票報價？
幫我下一張台積電的限價單
顯示我的帳戶部位

新手入門提示詞

第一次使用 Shioaji？使用這個提示詞讓 AI 引導你完成完整的設定與測試流程：

我是 Shioaji 新手，想要開始進行交易。請一步一步引導我完成完整的設定流程：

1. 如何開立永豐金帳戶
2. 如何簽署 API 下單服務條款
3. 如何申請 API Key 和 Secret Key
4. 如何完成模擬模式測試（登入測試、證券下單測試、期貨下單測試）
5. 如何確認測試是否通過（signed=True）
6. 如何申請並啟用憑證
7. 如何切換到正式環境

請在每個步驟確認我的進度，並協助我排除任何問題。

這個提示詞可以幫助 AI 引導你完成整個前置作業流程，這是使用正式環境進行交易前的必要步驟。

2.2 最佳實踐

1. 提供上下文 - 告訴 AI 您正在使用 Shioaji 台灣交易 API
2. 參考文件 - 將 llms-full.txt 提供給 AI 以獲得完整知識
3. 驗證生成的程式碼 - 執行前務必檢查 AI 生成的交易程式碼
4. 使用模擬模式 - 先在模擬模式中測試 AI 生成的策略



重要提醒

AI 生成的交易程式碼應在實際使用前，於模擬模式中進行審查和測試。AI 助手根據文件提供指導，但無法保證交易結果。

3. 快速入門

只需像使用其他流行的 Python 套件一樣，導入我們的應用程式介面並新建實例即可開始使用我們的應用程式介面。



在開始前請還需完成前置作業，包含開戶、服務條款 及 Token。

3.0.1 登入並啟用憑證

version>=1.0 version<1.0

```
import shioaji as sj

api = sj.Shioaji()
accounts = api.login("YOUR_API_KEY", "YOUR_SECRET_KEY")
api.activate_ca(
    ca_path="/c/your/ca/path/Sinopac.pfx",
    ca_passwd="YOUR_CA_PASSWORD",
    person_id="Person of this Ca",
)

import shioaji as sj

api = sj.Shioaji()
accounts = api.Login("YOUR_PERSON_ID", "YOUR_PASSWORD")
api.activate_ca(
    ca_path="/c/your/ca/path/Sinopac.pfx",
    ca_passwd="YOUR_CA_PASSWORD",
    person_id="Person of this Ca",
)
```



Windows環境下複製文件路徑時用 \ 分隔文件，需要用 / 替換。

3.0.2 訂閱行情

訂閱行情需將合約帶入 `subscribe` 功能，並指定行情類型，就可以接收資料。

```
api.quote.subscribe(api.Contracts.Stocks["2330"], quote_type="tick")
api.quote.subscribe(api.Contracts.Stocks["2330"], quote_type="bidask")
api.quote.subscribe(api.Contracts.Futures["TXFCO"], quote_type="tick")
```



目前我們支持 `shioaji.constant.QuoteType` 中的兩種行情類型。最好的使用方法是直接將這個枚舉類型傳入 `subscribe` 函數。

3.0.3 下單

與上面訂閱行情的方法雷同，需將合約及定義下單資訊帶入 `place_order` 函數，然後它將返回描述您交易狀態。

```
contract = api.Contracts.Stocks["2890"]
order = api.Order(
    price=12,
    quantity=5,
    action=sj.constant.Action.Buy,
    price_type=sj.constant.StockPriceType.LMT,
    order_type=sj.constant.OrderType.ROD,
)
trade = api.place_order(contract, order)
```

3.1 總結

這個快速入門演示了我們使用原生 Python 的套件是有多簡單，與許多其他交易應用程式介面不同。我們致力於為用戶提供更具有 Python 特色的交易應用程式介面。

4. 環境設定

4.1 系統需求

在開始之前，請確保你的系統符合以下需求：

- 作業系統：Windows、MacOS 或 Linux 之 64 位元版本
- Python 版本：3.8 以上
- 使用者需要具備永豐金證券帳戶，並取得 Shioaji API 權限。

4.2 安裝 Python 環境

首先，你需要在系統上安裝 Python，推薦使用 uv，本篇教學範例將使用 uv 作為 Python 及專案環境管理工具，並在專案中使用 Shioaji API。

延伸筆記

[uv](#) 是跨平台管理 python 環境及專案環境管理工具的最佳解決方案。

4.2.1 安裝 uv

指令

Linux 與 MacOS Windows

```
curl -LsSf https://astral.sh/uv/install.sh | sh
powershell -c "irm https://astral.sh/uv/install.ps1 | iex"
```

更多安裝與使用方式請參考 [uv 官方文件](#)

4.3 創建專案環境

首先，先創建一個名為 sj-trading 的專案

```
uv init sj-trading --package --app --vcs git
cd sj-trading
```

創建出來的專案路徑如下

```
sj-trading
├── README.md
└── pyproject.toml
└── src
    └── sj_trading
        └── __init__.py
```

加入 shioaji 套件到專案中

```
uv add shioaji
```

接著打開 pyproject.toml 檔案將會看到以下內容

```
[project]
name = "sj-trading"
version = "0.1.0"
description = "Shioaji Trading"
```

```
readme = "README.md"
requires-python = ">=3.12"
dependencies = [
    "shioaji>=1.2.5",
]

[project.scripts]
hello = "sj_trading:hello"

[build-system]
requires = ["hatchling"]
build-backend = "hatchling.build"
```

專案中有一個 `hello` 指令，代表我們已經成功可以執行 `hello` 指令

執行 hello 指令

```
uv run hello
```



Hello from sj-trading!

接著打開 `src/sj_trading/__init__.py` 檔案，將以下內容複製貼上

```
import shioaji as sj

def hello():
    get_shioaji_client()

def get_shioaji_client() -> sj.Shioaji:
    api = sj.Shioaji()
    print("Shioaji API created")
    return api
```

執行指令

```
uv run hello
```



Shioaji API created

這邊最基本的環境安裝就完成可以開始使用了。

4.3.1 使用 Jupyter 環境

加入 ipykernel 到專案的開發依賴

```
uv add --dev ipykernel
```

將專案使用環境加入到 Jupyter 的 kernel

```
uv run ipython kernel install --user --name=sj-trading
```

啟動 Jupyter

```
uv run --with jupyter jupyter lab
```

在 jupyter 中創建 dev.ipynb 檔案，就可以選擇 sj-trading 的 kernel 來執行指令

剛剛我們寫好的 hello 指令就可以在這邊執行了
jupyterlab

如果已經開好戶可以跳過下一章直接前往 [金鑰與憑證申請](#) 取得 API Key 與憑證。

5. 教程 - 使用者指南

5.1 前置作業

5.1.1 開戶

使用Shioaji必須擁有永豐金帳戶。若你還沒有擁永豐金帳戶，請依據下列步驟開戶：

1. 至[開戶](#)頁面
open_acct
2. 若你沒有永豐銀行帳戶，請先開銀行帳戶當你的交割戶
open_bank_1
3. 請選取[我要開DAWHO+大戶投](#)，為開銀行戶以及證券戶
open_bank_2
4. 完成銀行及證券開戶

5.1.2 金鑰與憑證申請

在版本1.0之後，我們將使用Token作為我們的登入方式。請根據下列的步驟進行申請及使用。

申請金鑰

1. 至[理財網](#)個人服務中的API管理頁面
newweb_1
2. 點選新增API KEY
newweb_2
3. 利用手機或是信箱做雙因子驗證，驗證成功才能建立API KEY。
newweb_3
4. 進行API KEY的到期時間設定，以及勾選擬限與帳戶，並且設定IP限制。
newweb_4

權限勾選擬說明

- 行情 / 資料：可否使用行情 / 資料相關 API
- 帳務：可否使用帳務相關 API
- 交易：可否使用交易相關 API
- 正式環境：可否在正式環境中使用

注意

IP建議使用限制，能使該KEY安全性提高。

5. 新增成功會得到金鑰(API Key)與密鑰(Secret Key)
newweb_5

注意

- 請妥善保存您的鑰匙，勿將其透漏給任何人，以免造成資產損失。
- Secret Key 僅在建立成功時取得，此後再無任何方法得到，請確保以保存

憑證下載

1. 點選下載憑證按鈕
newweb_7
2. 下載完成請前往下載資料夾將憑證放置到 API 要讀取的路徑
newweb_8

確認密鑰與憑證

延續前面開好的專案 sj-trading，在專案資料夾中新增 .env 檔案，並且新增以下內容

.env 檔案內容如下

```
API_KEY=<前面申請的API Key>
SECRET_KEY=<前面申請的Secret Key>
CA_CERT_PATH=<前面設定的憑證路徑>
CA_PASSWORD=<憑證密碼>
```

專案資料夾結構如下

```
sj-trading
├── README.md
├── .env
├── pyproject.toml
└── src
    └── sj_trading
        └── __init__.py
└── uv.lock
```

加入 python-dotenv 套件來將 .env 的金鑰與憑證載入環境變數

```
uv add python-dotenv
```

在 src/sj_trading/__init__.py 中新增以下內容

```
import os
from dotenv import load_dotenv

load_dotenv()

def main():
    api = sj.Shoaji(simulation=True)
    api.login(
        api_key=os.environ["API_KEY"],
        secret_key=os.environ["SECRET_KEY"],
        fetch_contract=False
    )
    api.activate_ca(
        ca_path=os.environ["CA_CERT_PATH"],
        ca_passwd=os.environ["CA_PASSWORD"],
    )
    print("login and activate ca success")
```

在 pyproject.toml 中 [project.scripts] 新增 main 指令

```
[project.scripts]
main = "sj_trading:main"
```

執行 main 指令

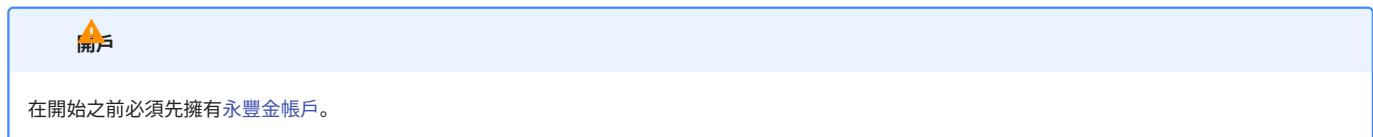
```
uv run main
```

如果看到 login and activate ca success 代表成功登入模擬環境了

接著如果你還有沒有進行 API 簽署的話，請前往下一章進行簽署與測試審核。

5.1.3 服務條款簽署

受限於台灣金融法規，新用戶首次使用需簽署相關文件並在測試模式完成測試報告才能進行正式環境的使用。



簽署文件

請參見[簽署中心](#)並在簽署前仔細閱讀文件。



證券類

一鍵簽署合併版風險預告，簽署項目如下：

風險預告書(合併)

- 權證風險預告書
- 附認股權風險預告書
- 交易外商股票風險預告書
- 交易日商股票風險預告書
- 黃金現貨風險預告書

已簽署 權證風險預告書

已簽署 現股當沖同意書

券差借貸契約書

已簽署 指數股票型ETF風險預告書

已簽署 興櫃風險預告書

已簽署 保管劃撥帳戶契約書

不可簽署 融資融券契約書

已簽署 ETN風險預告書

已簽署 轉交換債風險預告書

已簽署 交易外商股票風險預告書

已簽署 交易日商股票風險預告書

已簽署 黃金現貨風險預告書

已簽署 API 證券下單簽署

不可簽署 負面信用資料查詢同意書(信用戶)

不可簽署 負面信用資料查詢同意書(五日型借貸)

API測試

確保您完全理解如何使用，需在模擬模式完成測試報告，內容包含以下功能：

- 登入測試 `Login`
- 下單測試 `place_order`

Attention

可測試時間:

- 因應公司資訊安全規定，測試報告服務為星期一至五 08:00 ~ 20:00
- 18:00 ~ 20:00: 只允許台灣IP
- 08:00 ~ 18:00: 沒有限制

版本限制:

- 版本 ≥ 1.2 :

安裝指令: uv add shioaji or pip install -U shioaji

其他:

- API下單簽署時間須早於API測試的時間，以利審核通過
- 證券、期貨戶須各別測試
- 證券/期貨下單測試，需間隔1秒以上，以利系統留存測試紀錄

查詢使用版本

版本

```
import shioaji as sj
print(sj.__version__)
# 1.0.0
```

- 請注意**版本限制**

登入測試

登入

version ≥ 1.0 version <1.0

```
api = sj.Shioaji(simulation=True) # 模擬模式
api.login(
    api_key="金鑰", # 請修改此處
    secret_key="密鑰" # 請修改此處
)

api = sj.Shioaji(simulation=True) # 模擬模式
api.login(
    person_id="身分證字號", # 請修改此處
    passwd="密碼", # 請修改此處
)
```

- 版本 ≥ 1.0 : 使用 API Key 進行登入，若您尚未申請 API Key，可參考 [Token](#)
- 版本 < 1.0 : 使用 身分證字號 進行登入

證券下單測試


version>=1.0 version<1.0

```
# 商品檔 - 請修改此處
contract = api.Contracts.Stocks.TSE["2890"]

# 證券委託單 - 請修改此處
order = api.Order(
    price=18,                                     # 價格
    quantity=1,                                    # 數量
    action=sj.constant.Action.Buy,                 # 買賣別
    price_type=sj.constant.StockPriceType.LMT,    # 委託價格類別
    order_type=sj.constant.OrderType.ROD,          # 委託條件
    account=api.stock_account                      # 下單帳號
)

# 下單
trade = api.place_order(contract, order)
trade

# 商品檔 - 請修改此處
contract = api.Contracts.Stocks.TSE["2890"]

# 證券委託單 - 請修改此處
order = api.Order(
    price=18,                                     # 價格
    quantity=1,                                    # 數量
    action=sj.constant.Action.Buy,                 # 買賣別
    price_type=sj.constant.TFTStockPriceType.LMT, # 委託價格類別
    order_type=sj.constant.TFTOrderType.ROD,       # 委託條件
    account=api.stock_account                      # 下單帳號
)

# 下單
trade = api.place_order(contract, order)
trade
```



```
Response Code: 0 | Event Code: 0 | Info: host '218.32.76.102:80', IP 218.32.76.102:80 (host 1 of 1) (host connection attempt 1 of 1) (total connection attempt 1 of 1) | Event: Session up

Trade(
    contract=Stock(...),
    order=Order(...),
    status=OrderStatus(
        id='531e27af',
        status=<Status.Submitted: 'Submitted'>,
        status_code='00',
        order_datetime=datetime(2023, 1, 12, 11, 18, 3, 867490),
        order_quantity=1,
        deals=[]
    )
)
```

- 收到此訊息， Response Code: 0 | Event Code: 0 | Info: host '218.32.76.102:80' ...，代表您成功連結上測試伺服器。此訊息只有首次下單會顯示。若您未收到此訊息，請確認一下狀況均符合
 - a. 在 可測試時間 進行測試
 - b. 版本限制
 - c. signed 未在您的帳號中顯示
- 委託單狀態不應為 Failed，若您的委託單狀態 Failed，請正確的修改委託單然後再次執行 place_order
- 商品檔
- 證券委託下單

期貨下單測試

**verion>=1.0 verion<1.0**

```
# 商品檔 - 近月台指期貨，請修改此處
contract = min(
    [
        x for x in api.Contracts.Futures.TXF
        if x.code[-2:] not in ["R1", "R2"]
    ],
    key=lambda x: x.delivery_date
)

# 期貨委託單 - 請修改此處
order = api.Order(
    action=sj.constant.Action.Buy,          # 買賣別
    price=15000,                          # 價格
    quantity=1,                            # 數量
    price_type=sj.constant.FuturesPriceType.LMT, # 委託價格類別
    order_type=sj.constant.OrderType.ROD,   # 委託條件
    octype=sj.constant.FuturesOrderType.Auto, # 倉別
    account=api.futopt_account            # 下單帳號
)

# 下單
trade = api.place_order(contract, order)

# 商品檔 - 近月台指期貨，請修改此處
contract = min(
    [
        x for x in api.Contracts.Futures.TXF
        if x.code[-2:] not in ["R1", "R2"]
    ],
    key=lambda x: x.delivery_date
)

# 期貨委託單 - 請修改此處
order = api.Order(
    action=sj.constant.Action.Buy,          # 買賣別
    price=15000,                          # 價格
    quantity=1,                            # 數量
    price_type=sj.constant.FuturesPriceType.LMT, # 委託價格類別
    order_type=sj.constant.FuturesOrderType.ROD, # 委託條件
    octype=sj.constant.FuturesOrderType.Auto, # 倉別
    account=api.futopt_account            # 下單帳號
)

# 下單
trade = api.place_order(contract, order)
```



```
Response Code: 0 | Event Code: 0 | Info: host '218.32.76.102:80', IP 218.32.76.102:80 (host 1 of 1) (host connection attempt 1 of 1) (total connection attempt 1 of 1) | Event: Session up

Trade(
    contract=Stock(...),
    order=Order(...),
    status=OrderStatus(
        id='531e27af',
        status=<Status.Submitted: 'Submitted'>,
        status_code='00',
        order_datetime=datetime(2023, 1, 12, 11, 18, 3, 867490),
        order_quantity=1,
        deals=[]
    )
)
```

- 收到此訊息， Response Code: 0 | Event Code: 0 | Info: host '218.32.76.102:80' ...，代表您成功連結上測試伺服器。此訊息只有首次下單會顯示。若您未收到此訊息，請確認一下狀況均符合
 - a. 在 可測試時間 進行測試
 - b. 版本限制
 - c. signed 未在您的帳號中顯示
- 委託單狀態不應為 Failed，若您的委託單狀態 Failed，請正確的修改委託單然後再次執行 place_order
- 商品檔
- 期貨委託下單

查詢是否通過API測試

⚠️ Attention

在查詢前，請確認以下狀況均符合

- API下單簽署時間須早於API測試的時間，以利審核通過
- 在可測試時間進行測試
- 證券、期貨戶須各別測試
- 等待API測試審核(約5分鐘)

💡 簽署狀態

version>=1.0 version<1.0

```
import shioaji as sj

api = sj.Shioaji(simulation=False) # Production Mode
accounts = api.login(
    api_key="YOUR_API_KEY",          # edit it
    secret_key="YOUR_SECRET_KEY"     # edit it
)
accounts

import shioaji as sj

api = sj.Shioaji(simulation=False) # Production Mode
accounts = api.login(
    person_id="YOUR_PERSON_ID",      # edit it
    passwd="YOUR_PASSWORD",          # edit it
)
accounts
```

✅ OK

Response Code: 0 | Event Code: 0 | Info: host '203.66.91.161:80', hostname '203.66.91.161:80' IP 203.66.91.161:80 (host 1 of 1) (host connection attempt 1 of 1) (total connection attempt 1 of 1)
| Event: Session up

[FutureAccount(person_id='QBCCAIGJBJ', broker_id='F002000', account_id='9100020', signed=True, username='PAPIUSER01'),
StockAccount(person_id='QBCCAIGJBJ', broker_id='9A95', account_id='0504350', username='PAPIUSER01')]

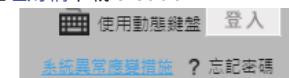
- signed=True : 恭喜完成測試! Ex: FutureAccount.
- signed=False 或 signed 未顯示: 此帳號尚未通過API測試或尚未簽署API下單文件. Ex: StockAccount.

憑證

下單前必須申請並啟用憑證

申請憑證

1. 至理財網下載 eleader



客戶服務

- 新手上路
- 我要開戶
- 問答百科
- 下載專區**
- 線上補發密碼
- 最新公告
- 與我聯絡
- 客服一點通

金融友善服務

服務收費標準

熱門下載

類型	下載內容	
申請表單	證券	基本資料變更(通訊辦理者需照會本人，更改戶籍地請檢附身分證影本)
電子交易	網路環境	完整環境安裝包(含JAVA安裝)
電子交易	網路環境	純環境調整包(無JAVA安裝)
電子交易	網路環境	JAVA安裝檔
電子交易	網路環境	新版憑證管理網頁元件安裝檔-32位元
電子交易	網路環境	新版憑證管理網頁元件安裝檔-64位元
電子交易	網路環境	複委託IE環境檢測
電子交易	網路環境	如何移除sun java
電子交易	網路環境	Acrobat Reader下載
電子交易	網路環境	API元件下單說明網頁(採申請制，請透過所屬業務同仁)
電子交易	網路環境	原太證憑證元件修正檔

搜尋結果

類型	下載內容	
電子交易	下單平台	eLeader (直接下載)
電子交易	下單平台	好神通PLUS (直接下載)
電子交易	下單平台	手機下單
電子交易	下單平台	GLeader國外期貨版 (直接下載)
電子交易	下單平台	進階豐元寶(GPM)國外期貨版 (直接下載)
電子交易	下單平台	憑證管理小AP
電子交易	下單平台	Android TV看盤版
電子交易	下單平台	MAC 總管 (供MAC電腦 申請/更新/查詢憑證)
電子交易	網路環境	完整環境安裝包(含JAVA安裝)



2. 登入 eleader

最新消息

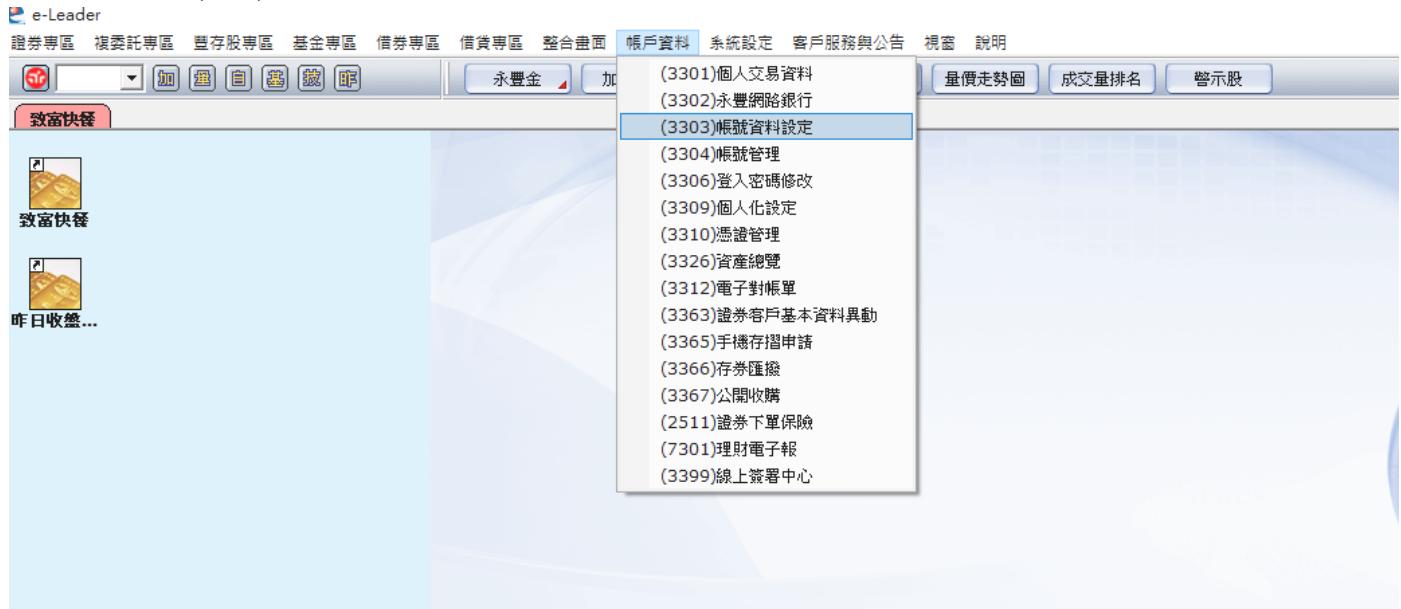
- EZTrade台股功能將於3月12日及EZTrade複委託報價將於3月20日下架通知
- 參加豐狂存股計畫抽8888元現金
- 2020/2/17~2/21，熱烈募集「復華新興亞洲3至10年期美元債券指數基金」

收單時間

收單時間	證券交易			期權交易	
	整股下單	定盤下單	零股下單	一般交易	盤後交易
下單時間	08:30~13:30	14:00~14:30	13:40~14:30	8:30~13:45	14:50~次 交易日5:00
預約單時間	14:30~次 交易日08:30	當日13:35 ~14:00	15:30~次 交易日13:40	次交易日 6:00~8:30	無

客服專線 : 02-6630-8899 ; 0800-038-123 , 客服信箱 : service.sec@sinopac.com

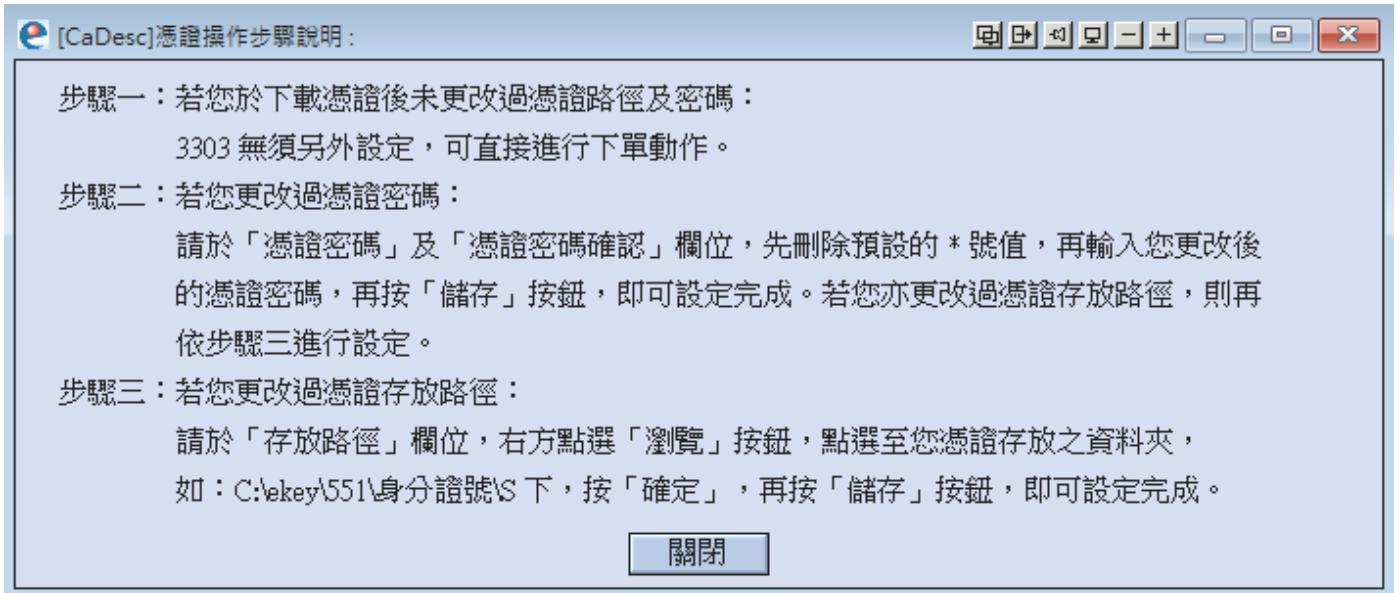
3. 從上方帳戶資料選取(3303)帳號資料設定



4. 點選"步驟說明"

The screenshot shows the '3303 Account Settings' configuration window. The title bar says '[3303] 帳號資料設定 : (0) TR<13304>'. The window contains fields for '身分證或統一編號', '憑證密碼', '憑證密碼確認', and '存放路徑'. Below these fields are buttons for '更新' (Update), '帳號管理' (Account Management), and '儲存' (Save). A large table below lists accounts with columns for '身分證號或統一編號', '姓名', '認證', and '憑證路徑'. At the bottom of the window are buttons for '登入密碼修改' (Login Password Modification) and '憑證密碼修改' (Certificate Password Modification).

5. 憑證操作步驟說明



啟用憑證

- 若是使用測試帳號無需啟用憑證
- 如果您使用macOS，可能會遇到版本上的問題。我們建議您使用 [docker](#) 去運行shioaji。



```
result = api.activate_ca(
    ca_path="/c/your/ca/path/Sinopac.pfx",
    ca_passwd="YOUR_CA_PASSWORD",
    person_id="Person ID of this Ca",
)
print(result)
# True
```



在 Windows 系統中，如果文件路徑使用 \ 來分隔文件，您需要將它替換為 /。

確認憑證效期



```
api.get_ca_expiretime("Person ID")
```

5.1.4 測試專案範例

首先，我們延伸前面在環境建立章節使用 `uv` 建立的專案 `sj-trading` 來新增測試流程的部分。

這部分完整專案的程式碼可以參考 `sj-trading` <https://github.com/Sinotrade/sj-trading-demo>。

可以使用 `git` 將整個環境複製到本地就可以直接使用

下載專案

```
git clone https://github.com/Sinotrade/sj-trading-demo.git  
cd sj-trading-demo
```

下面我們將一步一步的介紹如何新增測試流程。

SHIOAJI 版本

獲取 Shioaji 版本資訊

新增版本資訊

在 `src/sj_trading/__init__.py` 新增

```
def show_version() -> str:  
    print(f"Shioaji Version: {sj.__version__}")  
    return sj.__version__
```

新增版本指令到專案

在 `pyproject.toml` 新增 `version` 的指令

```
[project.scripts]  
version = "sj_trading:show_version"
```

執行 `uv run version` 就可以看到 Shioaji 的版本資訊

```
Shioaji Version: 1.2.0
```

現貨下單測試

新增下單測試檔案

在 src/sj_trading 新增檔案 testing_flow.py

新增以下內容

```
import shioaji as sj
from shioaji.constant import Action, StockPriceType, OrderType
import os

def testing_stock_ordering():
    # 測試環境登入
    api = sj.Shioaji(simulation=True)
    accounts = api.login(
        api_key=os.environ["API_KEY"],
        secret_key=os.environ["SECRET_KEY"],
    )
    # 顯示所有可用的帳戶
    print(f"Available accounts: {accounts}")
    api.activate_ca(
        ca_path=os.environ["CA_CERT_PATH"],
        ca_passwd=os.environ["CA_PASSWORD"],
    )

    # 準備下單的 Contract
    # 使用 2890 永豐金為例
    contract = api.Contracts.Stocks["2890"]
    print(f"Contract: {contract}")

    # 建立委託下單的 Order
    order = sj.order.StockOrder(
        action=Action.Buy, # 買進
        price=contract.reference, # 以平盤價買進
        quantity=1, # 下單數量
        price_type=StockPriceType.LMT, # 限價單
        order_type=OrderType.ROD, # 當日有效單
        account=api.stock_account, # 使用預設的帳戶
    )
    print(f"Order: {order}")

    # 送出委託單
    trade = api.place_order(contract=contract, order=order)
    print(f"Trade: {trade}")

    # 更新狀態
    api.update_status()
    print(f"Status: {trade.status}")
```

新增測試下單指令到專案

在 pyproject.toml 新增 stock_testing 的指令

```
[project.scripts]
stock_testing = "sj_trading.testing_flow:testing_stock_ordering"
```

執行 uv run stock_testing 就開始進行測試下單了

期貨下單測試

新增期貨下單測試在 `src/sj_trading/testing_flow.py` 新增以下內容

```
from shioaji.constant import (
    FuturesPriceType,
    FuturesOCTYPE,
)

def testing_futures_ordering():
    # 測試環境登入
    api = sj.Shioaji(simulation=True)
    accounts = api.login(
        api_key=os.environ["API_KEY"],
        secret_key=os.environ["SECRET_KEY"],
    )
    # 顯示所有可用的帳戶
    print(f"Available accounts: {accounts}")
    api.activate_ca(
        ca_path=os.environ["CA_CERT_PATH"],
        ca_passwd=os.environ["CA_PASSWORD"],
    )

    # 取得合約 使用台指期近月為例
    contract = api.Contracts.Futures["TXFR1"]
    print(f"Contract: {contract}")

    # 建立期貨委託單的 Order
    order = sj.order.FuturesOrder(
        action=Action.Buy, # 買進
        price=contract.reference, # 以平盤價買進
        quantity=1, # 下單數量
        price_type=FuturesPriceType.LMT, # 限價單
        order_type=OrderType.ROD, # 當日有效單
        octype=FuturesOCTYPE.Auto, # 自動選擇新平倉
        account=api.futopt_account, # 使用預設的帳戶
    )
    print(f"Order: {order}")

    # 送出委託單
    trade = api.place_order(contract=contract, order=order)
    print(f"Trade: {trade}")

    # 更新狀態
    api.update_status()
    print(f"Status: {trade.status}")
```

新增期貨下單指令到專案在 `pyproject.toml` 新增 `futures_testing` 的指令

```
[project.scripts]
futures_testing = "sj_trading.testing_flow:testing_futures_ordering"
```

執行 `uv run futures_testing` 就開始進行測試下單了

5.2 登入

登入必須擁有永豐金帳戶。若你還沒有擁永豐金帳戶，可詳見[開戶](#)。

5.2.1 登入

Token login

在1.0版本之後，我們將使用Token作為我們的登入方式，申請KEY可參見[文件](#)。當版本小於1.0，我們使用帳號密碼作為我們登入的方法。



version>=1.0 version<1.0

```
import shioaji as sj
api = sj.Shioaji()
api.login(
    api_key="YOUR_API_KEY",
    secret_key="YOUR_SECRET_KEY"
)

import shioaji as sj
api = sj.Shioaji()
api.login(
    person_id="YOUR_PERSON_ID",
    passwd="YOUR_PASSWORD",
)
```



[FutureAccount(person_id='', broker_id='', account_id='', signed=True, username=''),
StockAccount(person_id='', broker_id='', account_id='', signed=True, username='')]

- If you cannot find `signed` in your accounts, please sign the [document](#) first.

- 如果在帳號清單中找不到 `signed`，請至[服務條款](#)了解使用API服務所需要步驟。

Login Arguments

version>=1.0 version<1.0

```
api_key (str): API金鑰
secret_key (str): 密鑰
fetch_contract (bool): 是否從快取中讀取商品檔或從伺服器下載商品檔 (預設值: True)
contracts_timeout (int): 獲取商品檔 timeout (預設值: 0 ms)
contracts_cb (typing.Callable): 獲取商品檔 callback (預設值: None)
subscribe_trade (bool): 是否訂閱委託/成交回報 (預設值: True)
receive_window (int): 登入動作有效執行時間 (預設值: 30,000 毫秒)

person_id (str): 身分證字號
passwd (str): 密碼
hashed (bool): 密碼是否已經被hashed (預設值: False)
fetch_contract (bool): 是否從快取中讀取商品檔或從伺服器下載商品檔 (預設值: True)
contracts_timeout (int): 獲取商品檔 timeout (預設值: 0 ms)
contracts_cb (typing.Callable): 獲取商品檔 callback (預設值: None)
subscribe_trade (bool): 是否訂閱委託/成交回報 (預設值: True)
```



當版本大於1.0時，可能在登入時收到**Sign data is timeout**，這表示登入超過有效執行時間。可能是您的電腦時間與伺服器時間相差過大，需校準電腦的時間。或者登入執行時間超過有效時間，可將 `receive_window` 調高。

獲取商品檔Callback



version>=1.0 version<1.0

```
import shioaji as sj
api = sj.Shioaji()
api.login(
    api_key="YOUR_API_KEY",
    secret_key="YOUR_SECRET_KEY",
    contracts_cb=lambda security_type: print(f"{repr(security_type)} fetch done.")
)

import shioaji as sj
api = sj.Shioaji()
api.login(
    person_id="YOUR_PERSON_ID",
    passwd="YOUR_PASSWORD",
    contracts_cb=lambda security_type: print(f"{repr(security_type)} fetch done.")
)
```



```
[
    FutureAccount(person_id='', broker_id='', account_id='', signed=True, username=''),
    StockAccount(person_id='', broker_id='', account_id='', signed=True, username='')
]
<SecurityType.Index: 'IND'> fetch done.
<SecurityType.Future: 'FUT'> fetch done.
<SecurityType.Option: 'OPT'> fetch done.
<SecurityType.Stock: 'STK'> fetch done.
```

訂閱委託/成交回報

我們提供2個方式讓您可以調整訂閱委託/成交回報。首先是於 `login` 的參數 `subscribe_trade`，預設值為 `True`，會自動為您訂閱所有帳號的委託/成交回報。



version>=1.0 version<1.0

```
import shioaji as sj
api = sj.Shioaji()
api.login(
    api_key="YOUR_API_KEY",
    secret_key="YOUR_SECRET_KEY",
    subscribe_trade=True
)

import shioaji as sj
api = sj.Shioaji()
api.login(
    person_id="YOUR_PERSON_ID",
    passwd="YOUR_PASSWORD",
    subscribe_trade=True
)
```

另一個方式是，對特定帳號使用API `subscribe_trade` 及 `unsubscribe_trade`，即可訂閱或取消訂閱委託/成交回報。



subscribe trade

```
api.subscribe_trade(account)
```



unsubscribe trade

```
api.unsubscribe_trade(account)
```

5.2.2 帳號

帳號列表



```
accounts = api.list_accounts()
accounts
```



```
# print(accounts)
[
    FutureAccount(person_id='PERSON_ID_1', broker_id='BROKER_ID_1', account_id='ACCOUNT_ID_1', signed=True, username='USERNAME_1'),
    FutureAccount(person_id='PERSON_ID_2', broker_id='BROKER_ID_2', account_id='ACCOUNT_ID_2', username='USERNAME_2'),
    StockAccount(person_id='PERSON_ID_3', broker_id='BROKER_ID_3', account_id='ACCOUNT_ID_3', username='USERNAME_3'),
    StockAccount(person_id='PERSON_ID_4', broker_id='BROKER_ID_4', account_id='ACCOUNT_ID_4', signed=True, username='USERNAME_4')
]
```

- 若 signed 在帳號列表中未出現，如同 ACCOUNT_ID_2 及 ACCOUNT_ID_3，代表該帳號尚未簽署或者尚未完成在測試模式中的測試報告。可參見[服務條款](#)。

預設帳號



```
# 期貨預設帳號
print(api.futopt_account)
```



```
FutureAccount(person_id='PERSON_ID_1', broker_id='BROKER_ID_1', account_id='ACCOUNT_ID_1', signed=True, username='USERNAME_1')
```

設定預設帳號



```
# 預設的期貨帳號從 ACCOUNT_ID_1轉換成 ACCOUNT_ID_2
api.set_default_account(accounts[1])
print(api.futopt_account)
```



```
FutureAccount(person_id='PERSON_ID_2', broker_id='BROKER_ID_2', account_id='ACCOUNT_ID_2', username='USERNAME_2')
```

下單Order物件中需要指定帳號。更多資訊請參考[現貨和期權下單](#)。

```
order = api.Order(
    price=12,
    quantity=1,
    action=sj.constant.Action.Buy,
    price_type=sj.constant.StockPriceType.LMT,
    order_type=sj.constant.OrderType.ROD,
    order_lot=sj.constant.StockOrderLot.Common,
    account=api.stock_account
)
```

5.2.3 登出

登出功能將關閉客戶端及服務端之間的連接。為了提供優質的服務，我們從2021/08/06開始將**限制**連線數。在不使用的時候終止程式是一個良好的習慣。

```
api.logout()  
# True
```

5.3 商品檔

商品檔將在很多地方被使用，例如下單、訂閱行情...等。

取得商品檔

下方提供兩種方法取得商品檔：

- 方法1：登入成功後，將開始下載商品檔。但這個下載過程將不會影響其他的操作。若您想了解是否下載完成，可利用 Contracts.status 去得到下載狀態。contracts_timeout 設定為10000，它將等待10秒下載商品檔。



version>=1.0 version<1.0

```
import shioaji as sj
api = sj.Shioaji()
api.login(
    api_key="YOUR_API_KEY",
    secret_key="YOUR_SECRET_KEY",
    contracts_timeout=10000,
)

import shioaji as sj
api = sj.Shioaji()
api.login(
    person_id="YOUR_PERSON_ID",
    passwd="YOUR_PASSWORD",
    contracts_timeout=10000,
)
```

- 方法2：若不想在登入時下載商品檔，將 fetch_contract 設定為 False。利用 fetch_contracts 下載商品檔



version>=1.0 version<1.0

```
import shioaji as sj
api = sj.Shioaji()
api.login(
    api_key="YOUR_API_KEY",
    secret_key="YOUR_SECRET_KEY",
    fetch_contract=False,
)
api.fetch_contracts(contract_download=True)

import shioaji as sj
api = sj.Shioaji()
api.login(
    person_id="YOUR_PERSON_ID",
    passwd="YOUR_PASSWORD",
    fetch_contract=False,
)
api.fetch_contracts(contract_download=True)
```

商品檔資訊

目前我們所提供的商品包含：證券、期貨、選擇權以及指數。可從下列方法更詳細得到我們所提供的商品。



api.Contracts



Contracts(Indexs=(OTC, TSE), Stocks=(OES, OTC, TSE), Futures=(BRF, CAF, CBF, CCF, CDF, CEF, CFF, CGF, CHF, CJF, CK1, CKF, CLF, CM1, CMF, CNF, CQF, CRF, CSF, CU1, CUF, CWF, CXF, CYF, CZ1, CZF, DCF, DD1, DDF, DEF, DFF, DGF, DHF, DIF, DJF, DKF, DLF, DNF, DDF, DPF, DQF, DSF, DUF, DVF, DWF, DXF, DYF, DZF, EEF, EGF, EHF, EMF, EPF, ERF, EXF, EY1, EYF, FEF, FFF, FGF, FKF, FQF, FRF, FTF, FVF, FWF, FXF, FYF, FZF, G2F, GAF, GCF, GDF, GHF, GIF, GJF, GLF, GMF, GNF, GOF, GRF, GTF, GUF, GWF, GXF, HAF, HBF, HCF, HHF, HIF, HLF, HOF, HS1, HSF, HY1, HYF, IAI, IAF, IHF, IIF, IJF, IMF, IOF, IPF, IQF, IRF, ITF, IXF, IVF, IZF, JBF, JF, JNF, JPF, JSF, JWF, JZF, KAF, KB1, KBF, KCF, KDF, KFF, KGF, KIF, KKF, KLF, KOF, KPF, KSF, KW1, LBF, LCF, LE1, LEF, LIF, LMF, LOF, LQF, LRF, LTF, LU1, LV1, LW1, LX1, LY1, MAF, MBF, MCF, MJF, MKF, MPF, MQF, NWF, MX1, MXF, MYF, NAF, NBF, NCF, NDF, NEF, NGF, NHF, NIF, NJF, NLF, NMF, NNF, NOF, NSF, NU1, NUF, NWF, NXF, NYF, NZF, OAF, OBF, OCF, ODF, OEF, OGF, OHF, OJF, OKF, OLF, OMF, OOF, OPF, QOF, ORF, OS1, OSF, OTF, OUF, OVF, OWF, OXF, PAF, PBF, PCF, PDF, PEF, PFF, PGF, PHF, P1F, PKF, PLF, PMF, PNF, POF, PPF, PQF, RHF, RTF, SPF, TSF, TGF, TJF, TXF, UDF, UNF, XAF, XBF, XEF, XIF, XJF), Options=(CAO, CBO, CCO, CDO, CEO, CFO, CGO, CHO, CJO, CKO, CLO, CMO, CNO, CQO, CRO, CSO, CXO, CZO, DC0, DEO, DFO, DGO, DHO, DJO, DKO, DLO, DNO, DOO, DPO, DQO, DSO, DUO, DVO, DWF, DZO, G1O, G2O, HCO, IJO, LOO, NYA, NYO, NZO, OAO, OBO, OCO, OJO, OKO, OOO, OZO, RHO, RTO, TEO, TFO, TGO, TX1, TXO))

證券



```
api.Contracts.Stocks["2890"]
# or api.Contracts.Stocks.TSE.TSE2890
```



```
Stock(
    exchange=<Exchange.TSE: 'TSE'>,
    code='2890',
    symbol='TSE2890',
    name='永豐金',
    category='17',
    unit=1000,
    limit_up=19.1,
    limit_down=15.7,
    reference=17.4,
    update_date='2023/01/17',
    day_trade=<DayTrade.Yes: 'Yes'>
)
```



exchange (Exchange): 交易所 {OES, OTC, TSE ...等}
code (str): 商品代碼
symbol (str): 符號
name (str): 商品名稱
category (str): 類別
unit (int): 單位
limit_up (float): 漲停價
limit_down (float): 跌停價
reference (float): 參考價
update_date (str): 更新日期
margin_trading_balance (int): 融資餘額
short_selling_balance (int): 融券餘額
day_trade (DayTrade): 可否當沖 {Yes, No, OnlyBuy}

期貨



```
api.Contracts.Futures["TXFA3"]
# or api.Contracts.Futures.TXF.TXF202301
```

Cut

```
Future(
    code='TXFA3',
    symbol='TF202301',
    name='臺股期貨01',
    category='TF',
    delivery_month='202301',
    delivery_date='2023/01/30',
    underlying_kind='I',
    unit=1,
    limit_up=16417.0,
    limit_down=13433.0,
    reference=14925.0,
    update_date='2023/01/18'
)
```

Future

code (**str**): 商品代碼
 symbol (**str**): 符號
 name (**str**): 商品名稱
 category (**str**): 類別
 delivery_month (**str**): 交割月份
 delivery_date (**str**): 結算日
 underlying_kind (**str**): 標的類型
 unit (**int**): 單位
 limit_up (**float**): 漲停價
 limit_down (**float**): 跌停價
 reference (**float**): 參考價
 update_date (**str**): 更新時間

選擇權

Info

```
api.Contracts.Options["TX018000R3"]
# or api.Contracts.Options.TX0.TX020230618000P
```

Cut

```
Option(
    code='TX018000R3',
    symbol='TX020230618000P',
    name='臺指選擇權06月 18000P',
    category='TXO',
    delivery_month='202306',
    delivery_date='2023/06/21',
    strike_price=18000,
    option_right=<OptionRight.Put: 'P'>,
    underlying_kind='I',
    unit=1,
    limit_up=4720.0,
    limit_down=1740.0,
    reference=3230.0,
    update_date='2023/01/18'
)
```

Option

code (**str**): 商品代碼
 symbol (**str**): 符號
 name (**str**): 商品名稱
 category (**str**): 類型
 delivery_month (**str**): 交割月份
 delivery_date (**str**): 交割日期
 strike_price (**int** or **float**): 屢約價
 option_right (**OptionRight**): 買賣權別
 underlying_kind (**str**): 標的類型
 limit_up (**float**): 漲停價
 limit_down (**float**): 跌停價
 reference (**float**): 參考價
 update_date (**str**): 更新時間

指數

Indexes 物件顯示所有可以支援的指數商品，其他類別亦然。指數類的商品不支援下單，但允許訂閱行情。



```
api.Contracts.Indexes.TSE
```



```
TSE(TSE001, TSE015, TSE016, TSE017, TSE018, TSE019, TSE020, TSE022, TSE023, TSE024, TSE025, TSE026, TSE028, TSE029, TSE030, TSE031, TSE032, TSE033, TSE035, TSE036, TSE037, TSE038, TSE039, TSE040, TSE041, TSE042, TSE043, TSE004, TSE005)
```



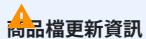
```
api.Contracts.Indexes.TSE["001"]
# or api.Contracts.Indexes.TSE.TSE001
```



```
Index(
    exchange=<Exchange.TSE: 'TSE'>,
    code='001',
    symbol='TSE001',
    name='加權指數'
)
```



``` python  
 exchange (Exchange): 交易所{OES, OTC, TSE ...等} code (str): 商品代碼 symbol (str): 符號 name (str): 商品名稱



- 07:50 期貨商品檔更新
- 08:00 全市場商品檔更新
- 14:45 期貨夜盤商品檔更新
- 17:15 期貨夜盤商品檔更新

## 5.4 行情資料

### 5.4.1 即時行情

#### 證券

利用訂閱商品檔的方式去取得即時行情。

#### Subscribe

```
>> api.quote.subscribe?

Signature:
api.quote.subscribe(
 contract:shioaji.contracts.Contract,
 quote_type:shioaji.constant.QuoteType=<QuoteType.Tick: 'tick'>,
 intraday_odd:<bool>=False,
 version: shioaji.constant.QuoteVersion.v0: 'v0'>,
)
```

#### Quote Parameters:

quote\_type: 訂閱類型 {'tick', 'bidask'}  
intraday\_odd: 盤中零股 {True, False}  
version: 行情版本 {'v1', 'v0'}

#### TICK

##### 整股

#### i

```
api.quote.subscribe(
 api.Contracts.Stocks["2330"],
 quote_type = sj.constant.QuoteType.Tick,
 version = sj.constant.QuoteVersion.v1
)
```



### QuoteVersion.v1    QuoteVersion.v0

Response Code: 200 | Event Code: 16 | Info: TIC/v1/STK/\*/TSE/2330 | Event: Subscribe or Unsubscribe ok

```
Exchange.TSE
Tick(
 code = '2330',
 datetime = datetime.datetime(2021, 7, 2, 13, 16, 35, 92970),
 open = Decimal('590'),
 avg_price = Decimal('589.05'),
 close = Decimal('590'),
 high = Decimal('593'),
 low = Decimal('587'),
 amount = Decimal('590000'),
 total_amount = Decimal('8540101000'),
 volume = 1,
 total_volume = 14498,
 tick_type = 1,
 chg_type = 4,
 price_chg = Decimal('-3'),
 pct_chg = Decimal('-.505902'),
 bid_side_total_vol= 6638,
 ask_side_total_vol= 7860,
 bid_side_total_cnt = 2694,
 ask_side_total_cnt = 2705,
 closing_ oddlot_shares = 0,
 fixed_trade_vol = 0,
 suspend = 0,
 simtrade = 0,
 intraday_ odd = 0
)
```

Response Code: 200 | Event Code: 16 | Info: MKT/\*/TSE/2330 | Event: Subscribe or Unsubscribe ok

```
MKT/1dcdmzpcr01/TSE/2330
{
 'AmountSum': [1688787000.0],
 'Close': [593.0],
 'Date': '2021/07/01',
 'TickType': [2],
 'Time': '09:10:20.628620',
 'VolSum': [2837],
 'Volume': [1]
}
```

盤中零股



```
api.quote.subscribe(
 api.Contracts.Stocks["2330"],
 quote_type = sj.constant.QuoteType.Tick,
 version = sj.constant.QuoteVersion.v1,
 intraday_ odd = True
)
```

 Out**QuoteVersion.v1    QuoteVersion.v0**

Response Code: 200 | Event Code: 16 | Info: TIC/v1/ODD/\*/TSE/2330 | Event: Subscribe or Unsubscribe ok

```
Exchange.TSE
Tick(
 code = '2330',
 datetime = datetime.datetime(2021, 7, 2, 13, 16, 55, 544646),
 open = Decimal('591'),
 avg_price = Decimal('590.24415'),
 close = Decimal('590'),
 high = Decimal('591'),
 low = Decimal('589'),
 amount = Decimal('276120'),
 total_amount = Decimal('204995925'),
 volume = 468,
 total_volume = 347307,
 tick_type = 1,
 chg_type = 4,
 price_chg = Decimal('-3'),
 pct_chg = Decimal('-.0505902'),
 bid_side_total_vol = 68209,
 ask_side_total_vol = 279566,
 bid_side_total_cnt = 28,
 ask_side_total_cnt = 56,
 closing_ oddlot_shares = 0,
 fixed_trade_vol = 0,
 suspend = 0,
 simtrade = 1,
 intraday_odd = 1
)
```

Response Code: 200 | Event Code: 16 | Info: TIC/v2/\*/TSE/2330/ODDLLOT | Event: Subscribe or Unsubscribe ok

```
TIC/v2/replay/TSE/2330/ODDLLOT
{
 'Date': '2021/07/01',
 'Time': '09:23:36.880878',
 'Close': '593',
 'TickType': 1,
 'Shares': 1860,
 'SharesSum': 33152,
 'Simtrade': 1
}
```

屬性

 Tick**QuoteVersion.v1    QuoteVersion.v0**

code (**str**): 商品代碼  
 datetime (**datetime**): 時間  
 open (**decimal**): 開盤價  
 avg\_price (**decimal**): 均價  
 close (**decimal**): 成交價  
 high (**decimal**): 最高價(自開盤)  
 low (**decimal**): 最低價(自開盤)  
 amount (**decimal**): 成交額 (NTD)  
 total\_amount (**decimal**): 總成交額 (NTD)  
 volume (**int**): 成交量 (整股:張, 盤中零股:股)  
 total\_volume (**int**): 總成交量 (整股:張, 盤中零股:股)  
 tick\_type (**int**): 內外盤別{1: 外盤, 2: 內盤, 0: 無法判定}  
 chg\_type (**int**): 漲跌註記{1: 漲停, 2: 漲, 3: 平盤, 4: 跌, 5: 跌停}  
 price\_chg (**decimal**): 漲跌  
 pct\_chg (**decimal**): 漲跌幅  
 bid\_side\_total\_vol (**int**): 買盤成交總量 (整股:張, 盤中零股:股)  
 ask\_side\_total\_vol (**int**): 賣盤成交總量 (整股:張, 盤中零股:股)  
 bid\_side\_total\_cnt (**int**): 買盤成交筆數  
 ask\_side\_total\_cnt (**int**): 賣盤成交筆數  
 closing\_ oddlot\_shares (**int**): 盤後零股成交股數(股)  
 fixed\_trade\_vol (**int**): 定盤成交量 (整股:張, 盤中零股:股)  
 suspend (**bool**): 暫停交易  
 simtrade (**bool**): 試撮  
 intraday\_odd (**int**): 盤中零股 {0: 整股, 1:盤中零股}

AmountSum (:List:**float**): 總成交額  
 Close (:List:**float**): 成交價  
 Date (**str**): 日期 (yyyy/MM/dd)  
 TickType (:List:**int**): 內外盤別{1: 外盤, 2: 內盤, 0: 無法判定}  
 Time (**str**): 時間 (HH:mm:ss.fffffffff)  
 VolSum (:List:**int**): 總成交量 (張)  
 Volume (:List:**int**): 成交量 (張)

**BIDASK**

整股



```
api.quote.subscribe(
 api.Contracts.Stocks["2330"],
 quote_type = sj.constant.QuoteType.BidAsk,
 version = sj.constant.QuoteVersion.v1
)
```

**QuoteVersion.v1    QuoteVersion.v0**

Response Code: 200 | Event Code: 16 | Info: QU0/v1/STK/\*/TSE/2330 | Event: Subscribe or Unsubscribe ok

```
Exchange.TSE
BidAsk(
 code = '2330',
 datetime = datetime.datetime(2021, 7, 1, 9, 9, 54, 36828),
 bid_price = [Decimal('593'), Decimal('592'), Decimal('591'), Decimal('590'), Decimal('589')],
 bid_volume = [248, 180, 258, 267, 163],
 diff_bid_vol = [3, 0, 0, 0, 0],
 ask_price = [Decimal('594'), Decimal('595'), Decimal('596'), Decimal('597'), Decimal('598')],
 ask_volume = [1457, 531, 506, 90, 259],
 diff_ask_vol = [0, 0, 0, 0, 0],
 suspend = 0,
 simtrade = 0,
 intraday_odd = 0
)
```

Response Code: 200 | Event Code: 16 | Info: QUT/\*/TSE/2330 | Event: Subscribe or Unsubscribe ok

```
QUT/idcdmzpcr01/TSE/2330
{
 'AskPrice': [594.0, 595.0, 596.0, 597.0, 598.0],
 'AskVolume': [1465, 532, 507, 92, 258],
 'BidPrice': [593.0, 592.0, 591.0, 590.0, 589.0],
 'BidVolume': [254, 178, 255, 268, 163],
 'Date': '2021/07/01',
 'Time': '09:09:48.447219'
}
```

**盤中零股**

```
api.quote.subscribe(
 api.Contracts.Stocks["2330"],
 quote_type = sj.constant.QuoteType.BidAsk,
 version = sj.constant.QuoteVersion.v1
 intraday_odd=True
)
```



### QuoteVersion.v1    QuoteVersion.v0

Response Code: 200 | Event Code: 16 | Info: QU0/v1/ODD/\*/TSE/2330 | Event: Subscribe or Unsubscribe ok

```
Exchange.TSE
BidAsk(
 code = '2330',
 datetime = datetime.datetime(2021, 7, 2, 13, 17, 45, 743299),
 bid_price = [Decimal('589'), Decimal('588'), Decimal('587'), Decimal('586'), Decimal('585')],
 bid_volume = [59391, 224490, 74082, 68570, 125246],
 diff_bid_vol = [49874, 101808, 23863, 38712, 77704],
 ask_price = [Decimal('590'), Decimal('591'), Decimal('592'), Decimal('593'), Decimal('594')],
 ask_volume = [26355, 9680, 18087, 11773, 3568],
 diff_ask_vol = [13251, -14347, 39249, -20397, -10591],
 suspend = 0,
 simtrade = 1,
 intraday_odd = 1
)
```

Response Code: 200 | Event Code: 16 | Info: QU0/v2/\*/TSE/2330/ODDLLOT | Event: Subscribe or Unsubscribe ok

```
QU0/v2/replay/TSE/2330/ODDLLOT
{
 'Date': '2021/07/01',
 'Time': '09:43:47.143789',
 'BidPrice': ['592', '591', '590', '589', '588'],
 'AskPrice': ['593', '594', '595', '596', '597'],
 'BidShares': [16979, 12009, 45045, 5501, 12956],
 'AskShares': [17276, 14823, 26518, 23388, 10527],
 'Simtrade': 1
}
```

## 屬性



### QuoteVersion.v1    QuoteVersion.v0

```
code (:str): 商品代碼
datetime (:datetime): 時間
bid_price (:List:decimal): 委買價
bid_volume (:List:int): 委買量 (張)
diff_bid_vol (:List:int): 買價增減量 (張)
ask_price (:List:decimal): 委賣價
ask_volume (:List:int): 委賣量
diff_ask_vol (:List:int): 賣價增減量 (張)
suspend (:bool): 暫停交易
simtrade (:bool): 試撮

AskPrice (:List:float): 委賣價
AskVolume (:List:int): 委賣量
BidPrice (:List:float): 委買價
BidVolume (:List:int): 委買量
Date (datetime.date): 日期 (yyyy/MM/dd)
Time (time): 時間 (HH:mm:ss.fffffffff)
```

## QUOTE

### 整股



```
api.quote.subscribe(
 api.Contracts.Stocks["2330"],
 quote_type = sj.constant.QuoteType.Quote,
 version = sj.constant.QuoteVersion.v1
)
```

## Out

```
Response Code: 200 | Event Code: 16 | Info: QU0/v2/STK/*/TSE/2330 | Event: Subscribe or Unsubscribe ok

Exchange.TSE,
Quote(
 code='2330',
 datetime=datetime.datetime(2022, 7, 1, 10, 43, 15, 430329),
 open=Decimal('471.5'),
 avg_price=Decimal('467.91'),
 close=Decimal('461'),
 high=Decimal('474'),
 low=Decimal('461'),
 amount=Decimal('461000'),
 total_amount=Decimal('11834476000'),
 volume=1,
 total_volume=25292,
 tick_type=2,
 chg_type=4,
 price_chg=Decimal('-15'),
 pct_chg=Decimal('-3.15'),
 bid_side_total_vol=9350,
 ask_side_total_vol=15942,
 bid_side_total_cnt=2730,
 ask_side_total_cnt=2847,
 closing_oddlot_shares=0,
 closing_oddlot_close=Decimal('0.0'),
 closing_oddlot_amount=Decimal('0'),
 closing_oddlot_bid_price=Decimal('0.0'),
 closing_oddlot_ask_price=Decimal('0.0'),
 fixed_trade.vol=0,
 fixed_trade.amount=Decimal('0'),
 bid_price=[Decimal('461'), Decimal('460.5'), Decimal('460'), Decimal('459.5'), Decimal('459')],
 bid_volume=[220, 140, 994, 63, 132],
 diff_bid.vol=[-1, 0, 0, 0, 0],
 ask_price=[Decimal('461.5'), Decimal('462'), Decimal('462.5'), Decimal('463'), Decimal('463.5')],
 ask_volume=[115, 101, 103, 147, 91],
 diff_ask.vol=[0, 0, 0, 0, 0],
 avail_borrowing=9579699,
 suspend=0,
 simtrade=0
)
```

## 屬性

## Quote

```
code (str): 商品代碼
datetime (datetime): 時間
open (decimal): 開盤價
avg_price (decimal): 均價
close (decimal): 成交價
high (decimal): 最高價(自開盤)
low (decimal): 最低價(自開盤)
amount (decimal): 成交額 (NTD)
total_amount (decimal): 總成交額 (NTD)
volume (int): 成交量
total_volume (int): 總成交量
tick_type (int): 內外盤別
chg_type (int): 漲跌註記
price_chg (decimal): 漲跌價
pct_chg (decimal): 漲跌率
bid_side_total_vol (int): 買盤成交總量 (張)
ask_side_total_vol (int): 賣盤成交總量 (張)
bid_side_total_cnt (int): 買盤成交筆數
ask_side_total_cnt (int): 賣盤成交筆數
closing_oddlot_shares (int): 盤後零股成交股數
closing_oddlot_close (decimal): 盤後零股成交價
closing_oddlot_amount (decimal): 盤後零股成交額
closing_oddlot_bid_price (decimal): 盤後零股買價
closing_oddlot_ask_price (decimal): 盤後零股賣價
fixed_trade.vol (int): 定盤成交量 (張)
fixed_trade.amount (decimal): 定盤成交額
bid_price (:List:decimal): 買價
bid.volume (:List:int) 買量
diff_bid.vol (:List:int) 買價增減量
ask.price (:List:decimal): 賣價
ask.volume (:List:int) 賣量
diff_ask.vol (:List:int) 賣價增減量
avail_borrowing (int): 借券可用餘額
suspend (bool): 暫停交易
simtrade (bool): 試撮
```

## CALLBACK

預設狀況下我們將即時行情使用 print 的方式呈現。可根據個人需求修改函數。請避免在函數內進行運算。

## Tick

## decorator方式

## QuoteVersion.v1    QuoteVersion.v0

```
from shioaji import TickSTKv1, Exchange

@api.on_tick_stk_v1()
def quote_callback(exchange: Exchange, tick:TickSTKv1):
 print(f"Exchange: {exchange}, Tick: {tick}")

@api.quote.on_quote
def quote_callback(topic: str, quote: dict):
 print(f"Topic: {topic}, Quote: {quote}")
```

## 傳統方式

## QuoteVersion.v1    QuoteVersion.v0

```
from shioaji import TickSTKv1, Exchange

def quote_callback(exchange: Exchange, tick:TickSTKv1):
 print(f"Exchange: {exchange}, Tick: {tick}")

api.quote.set_on_tick_stk_v1_callback(quote_callback)

def quote_callback(topic: str, quote: dict):
 print(f"Topic: {topic}, Quote: {quote}")

api.quote.set_quote_callback(quote_callback)
```

## Cut

## QuoteVersion.v1    QuoteVersion.v0

```
Exchange: Exchange.TSE, Tick: Tick(code='2330', datetime=datetime.datetime(2021, 7, 2, 13, 16, 35, 92970), open=Decimal('590'), avg_price=Decimal('589.05'), close=Decimal('590'),
high=Decimal('593'), low=Decimal('587'), amount=Decimal('590000'), total_amount=Decimal('8540101000'), volume=1, total_volume=14498, tick_type=1, chg_type=4, price_chg=Decimal('-3'),
pct_chg=Decimal('-0.505902'), trade_bid_volume=6638, ask_side_total_vol=7860, bid_side_total_cnt=2694, ask_side_total_cnt=2705, closing_ordinal_shares=0, fixed_trade_vol=0, suspend=0, simtrade=0,
intraday_odd=0)

Topic: MKT/*/TSE/2330, Quote: {'AmountSum': [4739351000.0], 'Close': [596.0], 'Date': '2021/03/30', 'TickType': [2], 'Time': '10:01:33.349431', 'VolSum': [7932], 'Volume': [1]}
```

## BidAsk

## decorator方式

## QuoteVersion.v1    QuoteVersion.v0

```
from shioaji import BidAskSTKv1, Exchange

@api.on_bidask_stk_v1()
def quote_callback(exchange: Exchange, bidask:BidAskSTKv1):
 print(f"Exchange: {exchange}, BidAsk: {bidask}")

@api.quote.on_quote
def quote_callback(topic: str, quote: dict):
 print(f"Topic: {topic}, Quote: {quote}")
```

### 傳統方式

#### QuoteVersion.v1    QuoteVersion.v0

```
from shioaji import BidAskSTKv1, Exchange

def quote_callback(exchange: Exchange, bidask:BidAskSTKv1):
 print(f"Exchange: {exchange}, BidAsk: {bidask}")

api.quote.set_on_bidask_stk_v1_callback(quote_callback)

def quote_callback(topic: str, quote: dict):
 print(f"Topic: {topic}, Quote: {quote}")

api.quote.set_quote_callback(quote_callback)
```

### Cat

#### QuoteVersion.v1    QuoteVersion.v0

```
Exchange: Exchange.TSE, BidAsk: BidAsk(code='2330', datetime=datetime.datetime(2021, 7, 2, 13, 17, 29, 726428), bid_price=[Decimal('589'), Decimal('588'), Decimal('587'), Decimal('586'), Decimal('585')], bid_volume=[223, 761, 1003, 809, 1274], diff_bid_vol=[0, 0, 0, 0], ask_price=[Decimal('590'), Decimal('591'), Decimal('592'), Decimal('593'), Decimal('594')], ask_volume=[304, 232, 183, 242, 131], diff_ask_vol=[1, 0, 0, 0], suspend=0, simtrade=0, intraday_odd=0)

Topic: QUT/idcdmzpcr01/TSE/2330, Quote: {'AskPrice': [590.0, 591.0, 592.0, 593.0, 594.0], 'AskVolume': [303, 232, 183, 242, 131], 'BidPrice': [589.0, 588.0, 587.0, 586.0, 585.0], 'BidVolume': [224, 762, 1003, 809, 1274], 'Date': '2021/07/02', 'Time': '13:17:26.391840'}
```

### Quote

### decorator方式

```
from shioaji import QuoteSTKv1, Exchange

@api.on_quote_stk_v1()
def quote_callback(exchange: Exchange, quote:QuoteSTKv1):
 print(f"Exchange: {exchange}, Quote: {quote}")
```

### 傳統方式

```
from shioaji import QuoteSTKv1, Exchange

def quote_callback(exchange: Exchange, quote:QuoteSTKv1):
 print(f"Exchange: {exchange}, Quote: {quote}")

api.quote.set_on_quote_stk_v1_callback(quote_callback)
```

### Cat

```
Exchange: TSE, Quote: Quote(code='2330', datetime=datetime.datetime(2022, 7, 1, 10, 43, 15, 430329), open=Decimal('471.5'), avg_price=Decimal('467.91'), close=Decimal('461'), high=Decimal('474'), low=Decimal('461'), amount=Decimal('461000'), total_amount=Decimal('11834476000'), volume=1, total_volume=25292, tick_type=2, chg_type=4, price_chg=Decimal('-15'), pct_chg=Decimal('-.15'), bid_side_total_vol=9350, ask_side_total_vol=15942, bid_side_total_cnt=2730, ask_side_total_cnt=2847, closing_oddlot_shares=0, closing_ oddlot_close=Decimal('0.0'), closing_ oddlot_amount=Decimal('0'), closing_ oddlot_bid_price=Decimal('0.0'), closing_ oddlot_ask_price=Decimal('0.0'), fixed_trade_vol=0, fixed_trade_amount=Decimal('0'), bid_price=Decimal('461'), Decimal('460.5'), Decimal('460'), Decimal('459.5'), Decimal('459')), bid_volume=[220, 140, 994, 63, 132], diff_bid_vol=[-1, 0, 0, 0, 0], ask_price=[Decimal('461.5'), Decimal('462'), Decimal('462.5'), Decimal('463'), Decimal('463.5')], ask_volume=[115, 101, 103, 147, 91], diff_ask_vol=[0, 0, 0, 0, 0], avail_borrowing=9579699, suspend=0, simtrade=0)
```

- 盤中零股與一般證券共用 callback函式。
- 更進階的callback使用可以參見[綁訂報價模式](#)。

## 期貨

利用訂閱商品檔的方式去取得即時行情。

### Subscribe

```
api.quote.subscribe?

Signature:
 api.quote.subscribe(
 contract:shioaji.contracts.Contract,
 quote_type:shioaji.constant.QuoteType=<QuoteType.Tick: 'tick'>,
 intraday_odd:bool=False,
 version: shioaji.constant.QuoteVersion=v0: 'v0'>,
)
Docstring: <no docstring>
Type: method
```

### Quote Parameters:

```
quote_type: 訂閱類型 {'tick', 'bidask', 'quote'}
intraday_odd: 盤中零股 {True, False}
version: 行情版本 {'v1', 'v0'}
```

## TICK

範例



```
api.quote.subscribe(
 api.Contracts.Futures.TXF['TXF202107'],
 quote_type = sj.constant.QuoteType.Tick,
 version = sj.constant.QuoteVersion.v1,
)
```



### QuoteVersion.v1    QuoteVersion.v0

Response Code: 200 | Event Code: 16 | Info: TIC/v1/FOP/\*/TFE/TXFG1 | Event: Subscribe or Unsubscribe ok

```
Exchange.TAIFEX
Tick(
 code = 'TXFG1',
 datetime = datetime.datetime(2021, 7, 1, 10, 42, 29, 757000),
 open = Decimal('17678'),
 underlying_price = Decimal('17849.57'),
 bid_side_total_vol= 32210,
 ask_side_total_vol= 33218,
 avg_price = Decimal('17704.663999'),
 close = Decimal('17753'),
 high = Decimal('17774'),
 low = Decimal('17655'),
 amount = Decimal('17753'),
 total_amount = Decimal('913790823'),
 volume = 1,
 total_volume = 51613,
 tick_type = 0,
 chg_type = 2,
 price_chg = Decimal('41'),
 pct_chg = Decimal('0.231481'),
 simtrade = 0
)
```

Response Code: 200 | Event Code: 16 | Info: L/\*/TXFG1 | Event: Subscribe or Unsubscribe ok

```
L/TFE/TXFG1
{
 'Amount': [17754.0],
 'AmountSum': [913027415.0],
 'AvgPrice': [17704.623134],
 'Close': [17754.0],
 'Code': 'TXFG1',
 'Date': '2021/07/01',
 'DiffPrice': [42.0],
 'DiffRate': [0.237127],
 'DiffType': [2],
 'High': [17774.0],
 'Low': [17655.0],
 'Open': 17678.0,
 'TargetKindPrice': 17849.57,
 'TickType': [2],
 'Time': '10:42:25.552000',
 'TradeAskVolSum': 33198,
 'TradeBidVolSum': 32180,
 'VolSum': [51570],
 'Volume': [1]
}
```

## 屬性

## Tick

**QuoteVersion.v1      QuoteVersion.v0**

```

code (str): 商品代碼
datetime (datetime.datetime): 日期
open (Decimal): 開盤價
underlying_price (Decimal): 標的物價格
bid_side_total_vol(int): 買盤成交總量 (lot)
ask_side_total_vol(int): 賣盤成交總量 (lot)
avg_price (Decimal): 均價
close (Decimal): 成交價
high (Decimal): 最高價(自開盤)
low (Decimal): 最低價(自開盤)
amount (Decimal): 成交額 (NTD)
total_amount (Decimal): 總成交額 (NTD)
volume (int): 成交量 (lot)
total_volume (int): 總成交量 (lot)
tick_type (int): 內外盤別{1: 外盤, 2: 內盤, 0: 無法判定}
chg_type (int): 漲跌註記{1: 漲停, 2: 漲, 3: 平盤, 4: 跌, 5: 跌停}
price_chg (Decimal): 漲跌
pct_chg (Decimal): 漲跌幅 (%)
simtrade (int): 試撮

Amount (list of float): 成交額 (成交價)
AmountSum (list of float): 總成交額 (總成交價)
AvgPrice (list of float): 均價
Close (list of float): 成交價
Code (str): 商品代碼
Date (str): 日期 (yyyy/MM/dd)
DiffPrice (list of float): 漲跌
DiffRate (list of float): 漲跌幅 (%)
DiffType (list of int): 漲跌註記{1: 漲停, 2: 漲, 3: 平盤, 4: 跌, 5: 跌停}
High (list of float): 最高價(自開盤)
Low (list of float): 最低價(自開盤)
Open (float): 開盤價
TargetKindPrice float: 標的物價格
TickType (:List:int): 內外盤別{1: 外盤, 2: 內盤, 0: 無法判定}
Time (str): 時間 (HH:mm:ss.fffff)
TradeAskVolSum (int): 賣盤成交總量 (lot)
TradeBidVolSum (int): 買盤成交總量 (lot)
VolSum (list of int): 總成交量 (lot)
Volume (list of int): 成交量 (lot)

```

## BIDASK

## 範例

```

api.quote.subscribe(
 api.Contracts.Futures.TXF['TXF202107'],
 quote_type = sj.constant.QuoteType.BidAsk,
 version = sj.constant.QuoteVersion.v1
)

```



### QuoteVersion.v1    QuoteVersion.v0

Response Code: 200 | Event Code: 16 | Info: QU0/v1/FOP/\*/TFE/TXFG1 | Event: Subscribe or Unsubscribe ok

Exchange.TAIFEX

```
BidAsk(
 code = 'TXFG1',
 datetime = datetime.datetime(2021, 7, 1, 10, 51, 31, 999000),
 bid_total_vol = 66,
 ask_total_vol = 101,
 bid_price = [Decimal('17746'), Decimal('17745'), Decimal('17744'), Decimal('17743'), Decimal('17742')],
 bid_volume = [1, 14, 19, 17, 15],
 diff_bid_vol = [0, 1, 0, 0, 0],
 ask_price = [Decimal('17747'), Decimal('17748'), Decimal('17749'), Decimal('17750'), Decimal('17751')],
 ask_volume = [6, 22, 25, 32, 16],
 diff_ask_vol = [0, 0, 0, 0],
 first_derived_bid_price = Decimal('17743'),
 first_derived_ask_price = Decimal('17751'),
 first_derived_bid_vol = 1,
 first_derived_ask_vol = 1,
 underlying_price = Decimal('17827.94'),
 simtrade = 0
)
```

Response Code: 200 | Event Code: 16 | Info: Q/\*/TXFG1 | Event: Subscribe or Unsubscribe ok

Q/TFE/TXFG1

```
{
 'AskPrice': [17747.0, 17748.0, 17749.0, 17750.0, 17751.0],
 'AskVolSum': 99,
 'AskVolume': [6, 22, 25, 31, 15],
 'BidPrice': [17746.0, 17745.0, 17744.0, 17743.0, 17742.0],
 'BidVolSum': 81,
 'BidVolume': [1, 12, 23, 25, 20],
 'Code': 'TXFG1',
 'Date': '2021/07/01',
 'DiffaskVol': [0, 0, 0, 0, 0],
 'DiffAskVolSum': 0,
 'DiffbidVol': [0, 0, 2, 0, 0],
 'DiffBidVolSum': 0,
 'FirstDerivedAskPrice': 17751.0,
 'FirstDerivedAskVolume': 1,
 'FirstDerivedBidPrice': 17743.0,
 'FirstDerivedBidVolume': 1,
 'TargetKindPrice': 17828.46,
 'Time': '10:51:29.999000'
}
```

## 屬性

 BidAsk

## QuoteVersion.v1    QuoteVersion.v0

```

code (str): 商品代碼
datetime (datetime.datetime): 時間
bid_total_vol (int): 委買量總計 (lot)
ask_total_vol (int): 委賣量總計 (lot)
bid_price (:List:decimal): 委買價
bid_volume (:List:int): 委買量 (lot)
diff_bid_vol (:List:int): 委買價增減量 (lot)
ask_price (:List:decimal): 委賣價
ask_volume (:List:int): 委賣量 (lot)
diff_ask_vol (:List:int): 委賣價增減量 (lot)
first_derived_bid_price (decimal): 衍生一檔委買價
first_derived_ask_price (decimal): 衍生一檔委賣價
first_derived_bid_vol (int): 衍生一檔委買量
first_derived_ask_vol (int): 衍生一檔委賣量
underlying_price (decimal): 標的物價格
simtrade (int): 試撮

AskPrice (:List:float): 委賣價
AskVolSum (int): 委賣量總計(lot)
AskVolume (:List:int): 委賣量
BidPrice (:List:float): 委買價
BidVolSum (int): 委買量總計(lot)
BidVolume (:List:int): 委買量
Code (str): 商品代碼
Date (str): 日期 (yyyy/MM/dd)
DiffAskVol (:List:int): 買價增減量(lot)
DiffAskVolSum (int):
DiffBidVol (:List:int): 買價增減量(lot)
DiffBidVolSum (int):
FirstDerivedAskPrice (float): 衍生一檔委賣價
FirstDerivedAskVolume (int): 衍生一檔委賣量
FirstDerivedBidPrice (float): 衍生一檔委買價
FirstDerivedBidVolume (int): 衍生一檔委買量
TargetKindPrice (float): 標的物價格
Time (str): 時間 (HH:mm:ss.fffff)

```

## QUOTE

## 範例



```

api.quote.subscribe(
 api.Contracts.Futures.TXF['TXF202512'],
 quote_type = sj.constant.QuoteType.Quote,
 version = sj.constant.QuoteVersion.v1,
)

```

## ✓ Out

```
Response Code: 200 | Event Code: 16 | Info: QU0/v2/FOP/*/TFE/TXFL5 | Event: Subscribe or Unsubscribe ok

Exchange.TAIFEX
Quote(
 code = 'TXFL5',
 datetime = datetime.datetime(2025, 12, 11, 29, 46, 247000),
 open = Decimal('28266'),
 avg_price = Decimal('28284.374007'),
 close = Decimal('28220'),
 high = Decimal('28342'),
 low = Decimal('28192'),
 amount = Decimal('28220'),
 total_amount = Decimal('946904273'),
 volume = 0,
 total_volume = 33478,
 tick_type = 1,
 chg_type = 2,
 price_chg = Decimal('119'),
 pct_chg = Decimal('0.423472'),
 bid_side_total_vol = 15254,
 ask_side_total_vol = 17295,
 bid_side_total_cnt = 22855,
 ask_side_total_cnt = 21882,
 bid_price = [Decimal('28218'), Decimal('28217'), Decimal('28216'), Decimal('28215'), Decimal('28214')],
 bid_volume = [5, 6, 11, 6, 10],
 diff_bid_vol = [0, 0, 0, 0, 0],
 ask_price = [Decimal('28220'), Decimal('28221'), Decimal('28222'), Decimal('28223'), Decimal('28224')],
 ask_volume = [2, 8, 14, 9, 9],
 diff_ask_vol = [0, -1, 1, 0, 0],
 first_derived_bid_price = Decimal('28215'),
 first_derived_ask_price = Decimal('28222'),
 first_derived_bid_vol = 1,
 first_derived_ask_vol = 1,
 underlying_price = Decimal('28185.62'),
 simtrade = False
)
```

屬性

## \_quote

```
code (str): 商品代碼
datetime (datetime.datetime): 時間
open (Decimal): 開盤價
avg_price (Decimal): 均價
close (Decimal): 成交價
high (Decimal): 最高價(自開盤)
low (Decimal): 最低價(自開盤)
amount (Decimal): 成交額 (NTD)
total_amount (Decimal): 總成交額 (NTD)
volume (int): 成交量 (lot)
total_volume (int): 總成交量 (lot)
tick_type (int): 外內盤別[1: 外盤, 2: 內盤, 0: 無法判定]
chg_type (int): 漲跌註記[1: 漲停, 2: 漲, 3: 平盤, 4: 跌, 5: 跌停]
price_chg (Decimal): 漲跌
pct_chg (Decimal): 漲跌幅 (%)
bid_side_total_vol (int): 買盤成交總量 (lot)
ask_side_total_vol (int): 賣盤成交總量 (lot)
bid_side_total_cnt (int): 買盤成交筆數
ask_side_total_cnt (int): 賣盤成交筆數
bid_price (:List:Decimal): 委買價
bid_volume (:List:int): 委買量 (lot)
diff_bid_vol (:List:int): 委買價減量 (lot)
ask_price (:List:Decimal): 委賣價
ask_volume (:List:int): 委賣量 (lot)
diff_ask_vol (:List:int): 委賣價減量 (lot)
first_derived_bid_price (Decimal): 衍生一檔委買價
first_derived_ask_price (Decimal): 衍生一檔委賣價
first_derived_bid_vol (int): 衍生一檔委買量
first_derived_ask_vol (int): 衍生一檔委賣量
underlying_price (Decimal): 標的物價格
simtrade (bool): 試撮
```

## CALLBACK

預設狀況下我們將即時行情使用 print 的方式呈現。可根據個人需求修改函數。請避免在函數內進行運算。

## Tick

decorator方式

## QuoteVersion.v1    QuoteVersion.v0

```
from shioaji import TickFOPv1, Exchange

@api.on_tick_fop_v1()
def quote_callback(exchange:Exchange, tick:TickFOPv1):
 print(f"Exchange: {exchange}, Tick: {tick}")

@api.quote.on_quote
def quote_callback(topic: str, quote: dict):
 print(f"Topic: {topic}, Quote: {quote}")
```

傳統方式

## QuoteVersion.v1    QuoteVersion.v0

```
from shioaji import TickFOPv1, Exchange

def quote_callback(exchange:Exchange, tick:TickFOPv1):
 print(f"Exchange: {exchange}, Tick: {tick}")

api.quote.set_on_tick_fop_v1_callback(quote_callback)

def quote_callback(topic: str, tick: dict):
 print(f"Topic: {topic}, Tick: {tick}")

api.quote.set_quote_callback(quote_callback)
```

Out

## QuoteVersion.v1    QuoteVersion.v0

```
Exchange: Exchange.TAIFEX, Tick: Tick(code='TXFG1', datetime=datetime.datetime(2021, 7, 2, 13, 17, 22, 784000), open=Decimal('17651'), underlying_price=Decimal('17727.12'), trade_bid_total_vol=61550, trade_ask_volume=60914, avg_price=Decimal('17657.999752'), close=Decimal('17653'), high=Decimal('17724'), low=Decimal('17588'), amount=Decimal('35306'), total_amount=Decimal('1683421593'), volume=2, total_volume=95335, tick_type=1, chg_type=2, price_chg=Decimal('7'), pct_chg=Decimal('0.039669'), simtrade=0)
```

```
Topic: L/TFE/TXFG1, Quote: {'Amount': [17654.0], 'AmountSum': [1682856730.0], 'AvgPrice': [17657.961764], 'Close': [17654.0], 'Code': 'TXFG1', 'Date': '2021/07/02', 'DiffPrice': [8.0], 'DiffRate': [0.045336], 'DiffType': [2], 'High': [17724.0], 'Low': [17588.0], 'Open': 17651.0, 'TargetKindPrice': 17725.14, 'TickType': [1], 'Time': '13:17:16.533000', 'TradeAskVolSum': 60890, 'TradeBidVolSum': 61520, 'VolSum': [95303], 'Volume': [1]}
```

## BidAsk

decorator方式

## QuoteVersion.v1    QuoteVersion.v0

```
from shioaji import BidAskFOPv1, Exchange

@api.on_bidask_fop_v1()
def quote_callback(exchange:Exchange, bidask:BidAskFOPv1):
 print(f"Exchange: {exchange}, BidAsk: {bidask}")

@api.quote.on_quote
def quote_callback(topic: str, quote: dict):
 print(f"Topic: {topic}, Quote: {quote}")
```

### 傳統方式

#### QuoteVersion.v1    QuoteVersion.v0

```
from shioaji import BidAskFOPv1, Exchange

def quote_callback(exchange:Exchange, bidask:BidAskFOPv1):
 print(f"Exchange: {exchange}, BidAsk: {bidask}")

api.quote.set_on_bidask_fop_v1_callback(quote_callback)

def quote_callback(topic: str, quote: dict):
 print(f"Topic: {topic}, Quote: {quote}")

api.quote.set_quote_callback(quote_callback)
```

### Cat

#### QuoteVersion.v1    QuoteVersion.v0

```
Exchange: Exchange.TAIFEX, BidAsk: BidAsk(code='TXFG1', datetime=datetime.datetime(2021, 7, 2, 13, 18, 0, 684000), bid_total_vol=69, ask_total_vol=94, bid_price=Decimal('17651'), Decimal('17650'), Decimal('17649'), Decimal('17648'), Decimal('17647')), bid_volume=[10, 12, 18, 18, 11], diff_bid_vol=[0, 0, 0, 0, 0], ask_price=[Decimal('17653'), Decimal('17654'), Decimal('17655'), Decimal('17656'), Decimal('17657')], ask_volume=[6, 17, 29, 22, 20], diff_ask_vol=[0, 0, 0, 0, 0], first_derived_bid_price=Decimal('17647'), first_derived_ask_price=Decimal('17657'), first_derived_bid_vol=2, first_derived_ask_vol=3, underlying_price=Decimal('17725.5'), simtrade=0)
```

```
Topic: Q/TFE/TXFG1, Quote: {'AskPrice': [17653.0, 17654.0, 17655.0, 17656.0, 17657.0], 'AskVolSum': 85, 'AskVolume': [3, 16, 24, 22, 20], 'BidPrice': [17651.0, 17650.0, 17649.0, 17648.0, 17647.0], 'BidVolSum': 67, 'BidVolume': [10, 10, 18, 18, 11], 'Code': 'TXFG1', 'Date': '2021/07/02', 'DiffAskVol': [-4, -2, 0, 0, 0], 'DiffAskVolSum': 0, 'DiffBidVol': [1, 0, 2, 0, 0], 'DiffBidVolSum': 0, 'FirstDerivedAskPrice': 17657.0, 'FirstDerivedAskVolume': 3, 'FirstDerivedBidPrice': 17647.0, 'FirstDerivedBidVolume': 2, 'TargetKindPrice': 17716.19, 'Time': '13:17:57.809000'}
```

### Quote

### decorator方式

```
from shioaji import QuoteFOPv1, Exchange

@api.on_quote_fop_v1()
def quote_callback(exchange:Exchange, quote:QuoteFOPv1):
 print(f"Exchange: {exchange}, Quote: {quote}")

api.quote.set_on_quote_fop_v1_callback(quote_callback)
```

### 傳統方式

```
from shioaji import QuoteFOPv1, Exchange

def quote_callback(exchange:Exchange, quote:QuoteFOPv1):
 print(f"Exchange: {exchange}, Quote: {quote}")

api.quote.set_on_quote_fop_v1_callback(quote_callback)
```

### Cat

```
Exchange: Exchange.TAIFEX, Quote: Quote(code='TXFL5', datetime=datetime.datetime(2025, 12, 12, 11, 29, 46, 247000), open=Decimal('28266'), avg_price=Decimal('28284.374007'), close=Decimal('28220'), high=Decimal('28342'), low=Decimal('28192'), amount=Decimal('28220'), total_amount=Decimal('946904273'), volume=0, total_volume=33478, tick_type=1, chg_type=2, price_chg=Decimal('119'), pct_chg=Decimal('0.423472'), bid_side_total_vol=15254, ask_side_total_vol=17295, bid_side_total_cnt=22855, ask_side_total_cnt=21882, bid_price=[Decimal('28218'), Decimal('28217'), Decimal('28216'), Decimal('28215'), Decimal('28214')], bid_volume=[5, 6, 11, 6, 10], diff_bid_vol=[0, 0, 0, 0, 0], ask_price=[Decimal('28220'), Decimal('28221'), Decimal('28222'), Decimal('28223'), Decimal('28224')], ask_volume=[2, 8, 14, 9, 9], diff_ask_vol=[0, -1, 1, 0, 0], first_derived_bid_price=Decimal('28215'), first_derived_ask_price=Decimal('28222'), first_derived_bid_vol=1, first_derived_ask_vol=1, underlying_price=Decimal('28185.62'), simtrade=False)
```

- 更進階的callback使用可以參見綁訂報價模式。

## 5.4.2 歷史行情

### Ticks

取得方式可以以一整天、某時間區段或是某天的最後幾筆。預設為商品最近交易日的Ticks。

#### Ticks

```
api.ticks?

Signature:
api.ticks(
 contract: shioaji.contracts.BaseContract,
 date: str = '2022-12-26',
 query_type: shioaji.constant.TicksQueryType = <TicksQueryType.AllDay: 'AllDay'>,
 time_start: Union[str, datetime.time] = None,
 time_end: Union[str, datetime.time] = None,
 last_cnt: int = 0,
 timeout: int = 30000,
 cb: Callable[[shioaji.data.Ticks], NoneType] = None,
) -> shioaji.data.Ticks

Docstring:
get contract tick column
```

### 取得特定日期 TICKS

#### Info

```
ticks = api.ticks(
 contract=api.Contracts.Stocks["2330"],
 date="2023-01-16"
)
ticks
```

#### Out

```
Ticks(
 ts=[1673859600113699000, 1673859600228800000, 1673859600244294000, 1673859600308595000],
 close=[506.0, 505.0, 506.0, 506.0],
 volume=[3340, 1, 17, 2],
 bid_price=[505.0, 505.0, 506.0, 506.0],
 bid_volume=[122, 320, 60, 58],
 ask_price=[506.0, 506.0, 507.0, 507.0],
 ask_volume=[13, 22, 702, 702],
 tick_type=[1, 2, 1, 2]
)
```

#### Ticks

```
ts (int): timestamp
close (float): 成交價
volume (int): 成交量
bid_price (float): 委買價
bid_volume (int): 委買量
ask_price (float): 委賣價
ask_volume (int): 委賣量
tick_type (int): 外內盤別{1: 外盤, 2: 內盤, 0: 無法判定}
```

### 轉成DataFrame

#### Info

```
import pandas as pd
df = pd.DataFrame(**ticks)
df.ts = pd.to_datetime(df.ts)
df.head()
```



| ts                            | ask_price | close | bid_volume | volume | ask_volume | tick_type | bid_price |
|-------------------------------|-----------|-------|------------|--------|------------|-----------|-----------|
| 2023-01-16<br>09:00:00.113699 | 506       | 506   | 122        | 3340   | 13         | 1         | 505       |
| 2023-01-16<br>09:00:00.228800 | 506       | 505   | 320        | 1      | 22         | 2         | 505       |
| 2023-01-16<br>09:00:00.244294 | 507       | 506   | 60         | 17     | 702        | 1         | 506       |
| 2023-01-16<br>09:00:00.308595 | 507       | 506   | 58         | 2      | 702        | 2         | 506       |

取得特定時間區段 TICKS



```
ticks = api.ticks(
 contract=api.Contracts.Stocks["2330"],
 date="2023-01-16",
 query_type=sj.constant.TicksQueryType.RangeTime,
 time_start="09:00:00",
 time_end="09:20:01"
)
ticks
```



```
Ticks(
 ts=[1673859600113699000, 1673859600228800000, 1673859600244294000, 1673859600308595000],
 close=[506.0, 505.0, 506.0, 506.0],
 volume=[3340, 1, 17, 2],
 bid_price=[505.0, 505.0, 506.0, 506.0],
 bid_volume=[122, 320, 60, 58],
 ask_price=[506.0, 506.0, 507.0, 507.0],
 ask_volume=[13, 22, 702, 702],
 tick_type=[1, 2, 1, 2]
)
```

取得最後數筆 TICKS



```
ticks = api.ticks(
 contract=api.Contracts.Stocks["2330"],
 date="2023-01-16",
 query_type=sj.constant.TicksQueryType.LastCount,
 last_cnt=4,
)
ticks
```



```
Ticks(
 ts=[1673859600113699000, 1673859600228800000, 1673859600244294000, 1673859600308595000],
 close=[506.0, 505.0, 506.0, 506.0],
 volume=[3340, 1, 17, 2],
 bid_price=[505.0, 505.0, 506.0, 506.0],
 bid_volume=[122, 320, 60, 58],
 ask_price=[506.0, 506.0, 507.0, 507.0],
 ask_volume=[13, 22, 702, 702],
 tick_type=[1, 2, 1, 2]
)
```

**KBar** **Kbars**

```
api.kbars?

Signature:
api.kbars(
 contract: shioaji.contracts.BaseContract,
 start: str = '2023-01-15',
 end: str = '2023-01-16',
 timeout: int = 30000,
 cb: Callable[[shioaji.data.Kbars], NoneType] = None,
) -> shioaji.data.Kbars
Docstring:
get Kbar
```

```
kbars = api.kbars(
 contract=api.Contracts.Stocks["2330"],
 start="2023-01-15",
 end="2023-01-16",
)
kbars
```

**Out**

```
Kbars(
 ts=[1673859660000000000, 1673859720000000000, 1673859780000000000, 1673859840000000000],
 Open=[506.0, 505.0, 505.0, 504.0],
 High=[508.0, 506.0, 506.0, 505.0],
 Low=[505.0, 505.0, 504.0, 504.0],
 Close=[505.0, 505.0, 504.0, 504.0],
 Volume=[5308, 1018, 543, 209]
)
```

**Kbars**

```
ts (int): timestamp
Open (float): open price
High (float): the highest price
Low: (float): the lowest price
Close (float): close price
Volume (int): volume
```

轉成DataFrame

```
import pandas as pd
df = pd.DataFrame(**kbars)
df.ts = pd.to_datetime(df.ts)
df.head()
```

 **Out**

| <b>Close</b> | <b>Amount</b> | <b>Low</b> | <b>Volume</b> | <b>ts</b>           | <b>Open</b> | <b>High</b> |
|--------------|---------------|------------|---------------|---------------------|-------------|-------------|
| 505          | 2.68731e+09   | 505        | 5308          | 2023-01-16 09:01:00 | 506         | 508         |
| 505          | 5.14132e+08   | 505        | 1018          | 2023-01-16 09:02:00 | 505         | 506         |
| 504          | 2.74112e+08   | 504        | 543           | 2023-01-16 09:03:00 | 505         | 506         |
| 504          | 1.0542e+08    | 504        | 209           | 2023-01-16 09:04:00 | 504         | 505         |

**資料歷史期間** **Historical Periods**

|         | <b>Start Date</b> | <b>End Date</b> |
|---------|-------------------|-----------------|
| Index   | 2020-03-02        | Today           |
| Stock   | 2020-03-02        | Today           |
| Futures | 2020-03-22        | Today           |

**連續期貨合約**

期貨合約一旦到期，合約即不再有效，亦即他將不會出現在您的 `api.Contracts` 裡。為了取得到期的期貨合約歷史資料，我們提供連續期貨合約。`R1`, `R2` 是近月及次月的連續期貨合約，他們會自動在結算日更換新的合約。您可以使用 `R1`, `R2` 合約來取得歷史資料，例如 `api.Contracts.Futures.TXF.TXFR1`。以下顯示如何使用 `R1`, `R2` 合約取得到期期貨的歷史 Ticks 及 Kbars。

**Ticks** **Ticks**

```
ticks = api.ticks(
 contract=api.Contracts.Futures.TXF.TXFR1,
 date="2020-03-22"
)
ticks
```

 **Out**

```
Ticks(
 ts=[1616166000030000000, 1616166000140000000, 1616166000140000000, 1616166000190000000],
 close=[16011.0, 16013.0, 16014.0, 16011.0],
 volume=[49, 2, 2, 1],
 bid_price=[0.0, 16011.0, 16011.0, 16011.0],
 bid_volume=[0, 1, 1, 1],
 ask_price=[0.0, 16013.0, 16013.0, 16013.0],
 ask_volume=[0, 1, 1, 1]
)
ticks
```

**Kbars**

### Kbars

```
kbars = api.kbars(
 contract=api.Contracts.Futures.TXF.TXFR1,
 start="2023-01-15",
 end="2023-01-16",
)
kbars
```

### Out

```
Kbars(
 ts=[16164027600000000000, 16164028200000000000, 16164028800000000000, 16164029400000000000],
 Open=[16018.0, 16018.0, 16000.0, 15992.0],
 High=[16022.0, 16020.0, 16005.0, 15999.0],
 Low=[16004.0, 16000.0, 15975.0, 15989.0],
 Close=[16019.0, 16002.0, 15992.0, 15994.0],
 Volume=[1791, 864, 1183, 342]
)
```

### 5.4.3 市場快照

市場快照為證券、期貨及選擇權當下資訊。內容包含開盤價、最高價、最低價、收盤價、變動價、均價、成交量、總成交量、委買價、委買量、委賣價、委賣量和昨量。



市場快照每次最多500檔商品。

#### 市場快照

```
>> api.snapshots?

Signature:
api.snapshots(
 contracts: List[Union[shioaji.contracts.Option, shioaji.contracts.Future, shioaji.contracts.Stock, shioaji.contracts.Index]],
 timeout: int = 30000,
 cb: Callable[[shioaji.data.Snapshot], NoneType] = None,
) -> List[shioaji.data.Snapshot]
Docstring:
get contract snapshot info
```

#### 範例



```
contracts = [api.Contracts.Stocks['2330'],api.Contracts.Stocks['2317']]
snapshots = api.snapshots(contracts)
snapshots
```



```
[
 Snapshot(
 ts=1673620200000000000,
 code='2330',
 exchange='TSE',
 open=507.0,
 high=509.0,
 low=499.0,
 close=500.0,
 tick_type=TickType.Sell: 'Sell',
 change_price=13.5,
 change_rate=2.77,
 change_type=ChangeType.Up: 'Up',
 average_price=502.42,
 volume=48,
 total_volume=77606,
 amount=24000000,
 total_amount=38990557755,
 yesterday_volume=20963.0,
 buy_price=500.0,
 buy_volume=122.0,
 sell_price=501.0,
 sell_volume=1067,
 volume_ratio=3.7
),
 Snapshot(
 ts=1673620200000000000,
 code='2317',
 exchange='TSE',
 open=99.0,
 high=99.5,
 low=98.6,
 close=98.6,
 tick_type=TickType.Sell: 'Sell',
 change_price=0.0,
 change_rate=0.0,
 change_type=ChangeType.Unchanged: 'Unchanged',
 average_price=98.96,
 volume=63,
 total_volume=17809,
 amount=6211800,
 total_amount=1762344817,
 yesterday_volume=18537.0,
 buy_price=98.6,
 buy_volume=607.0,
 sell_price=98.7,
 sell_volume=4,
 volume_ratio=0.96
)
]
```

轉成Dataframe



```
df = pd.DataFrame(s.__dict__ for s in snapshots)
df.ts = pd.to_datetime(df.ts)
df
```



| ts                     | code | exchange | open | high | low  | close | tick_type | change_price |
|------------------------|------|----------|------|------|------|-------|-----------|--------------|
| 2023-01-13<br>14:30:00 | 2330 | TSE      | 507  | 509  | 499  | 500   | Sell      | 13.5         |
| 2023-01-13<br>14:30:00 | 2317 | TSE      | 99   | 99.5 | 98.6 | 98.6  | Sell      | 0            |

## 屬性

### Snapshot

```
ts (int): 取得資訊時間戳記
code (str): 商品代碼
exchange (Exchange): 交易所
open (float): 開盤價
high (float): 最高價
low (float): 最低價
close (float): 收盤價
tick_type (TickType): 收盤買賣別 {None, Buy, Sell}
change_price (float): 漲跌
change_rate (float): 漲跌幅
change_type (ChangeType): 漲跌 {LimitUp, Up, Unchanged, Down, LimitDown}
average_price (float): 均價
volume (int): 單量
total_volume (int): 成交量
amount (int): 單量成交金額
total_amount (int): 成交金額
yesterday_volume (float): 昨量
buy_price (float): 委買價
buy_volume (float): 委買量
sell_price (float): 實出價
sell_volume (int): 委賣量
volume_ratio (float): 昨量比
```

## 5.4.4 或有券源

### 或有券源

```
>> api.short_stock_sources?

Signature:
api.short_stock_sources(
 contracts: List[shioaji.contracts.Stock],
 timeout: int = 5000,
 cb: Callable[[shioaji.data.ShortStockSource], NoneType] = None,
) -> List[shioaji.data.ShortStockSource]
```

## 範例



```
contracts = [
 api.Contracts.Stocks['2330'],
 api.Contracts.Stocks['2317']
]
short_stock_sources = api.short_stock_sources(contracts)
short_stock_sources
```



```
[
 ShortStockSource(code='2330', short_stock_source=58260, ts=1673943433000000000),
 ShortStockSource(code='2317', short_stock_source=75049, ts=1673943433000000000)
]
```

轉成 DataFrame



```
df = pd.DataFrame(s.__dict__ for s in short_stock_sources)
df.ts = pd.to_datetime(df.ts)
df
```



| code | short_stock_source | ts                  |
|------|--------------------|---------------------|
| 2330 | 58260              | 2023-01-17 08:17:13 |
| 2317 | 75049              | 2023-01-17 08:17:13 |

## 屬性

### ShortStockSource

```
code (str): 商品代碼
short_stock_source (float): 或有券源
ts (int): 時間戳記
```

## 5.4.5 資券餘額

### Credit Enquiries

```
>> api.credit_enquires?

Signature:
api.credit_enquires(
 contracts: List[shioaji.contracts.Stock],
 timeout: int = 30000,
 cb: Callable[[shioaji.data.CreditEnquiry], NoneType] = None,
) -> List[shioaji.data.CreditEnquiry]
```

### 範例



```
contracts = [
 api.Contracts.Stocks['2330'],
 api.Contracts.Stocks['2890']
]
credit_enquires = api.credit_enquires(contracts)
credit_enquires
```



```
[
 CreditEnquiry(update_time='2020-12-11 13:30:13', system='HE', stock_id='2330', margin_unit=1381),
 CreditEnquiry(update_time='2020-12-11 13:30:02', system='HC', stock_id='2330', margin_unit=1371),
 CreditEnquiry(update_time='2020-12-11 13:30:05', system='HN', stock_id='2330', margin_unit=1357),
 CreditEnquiry(update_time='2020-12-11 13:30:03', system='HF', stock_id='2330', margin_unit=1314),
 CreditEnquiry(update_time='2020-12-09 10:56:05', system='HE', stock_id='2890'),
 CreditEnquiry(update_time='2020-12-11 09:33:04', system='HN', stock_id='2890'),
 CreditEnquiry(update_time='2020-12-02 09:01:03', system='HF', stock_id='2890')
]
```

### 轉成DataFrame



```
df = pd.DataFrame(c.__dict__ for c in credit_enquires)
df.update_time = pd.to_datetime(df.update_time)
df
```

ut

|   | <b>margin_unit</b> | <b>short_unit</b> | <b>stock_id</b> | <b>system</b> | <b>update_time</b>     |
|---|--------------------|-------------------|-----------------|---------------|------------------------|
| 0 | 1381               | 0                 | 2330            | HE            | 2020-12-11<br>13:30:13 |
| 1 | 1371               | 0                 | 2330            | HC            | 2020-12-11<br>13:30:02 |
| 2 | 1357               | 0                 | 2330            | HN            | 2020-12-11<br>14:31:19 |
| 3 | 1314               | 0                 | 2330            | HF            | 2020-12-11<br>14:31:19 |
| 4 | 0                  | 0                 | 2890            | HE            | 2020-12-09<br>10:56:05 |
| 5 | 0                  | 0                 | 2890            | HN            | 2020-12-11<br>09:33:04 |
| 6 | 0                  | 0                 | 2890            | HF            | 2020-12-02<br>09:01:03 |

## 屬性

CreditEnquiry

update\_time ([str](#)): 更新時間  
 system ([str](#)): 類別  
 stock\_id ([str](#)): 商品代碼  
 margin\_unit ([int](#)): 資餘額  
 short\_unit ([int](#)): 務餘額

## 5.4.6 排行

包含漲跌幅、漲跌、高低價差、成交量及成交金額排行。Scanners 利用 scanner\_type 去取得不同類型的排行。

### Scanners

```
>> api.scanners?

Signature:
api.scanners(
 scanner_type: shioaji.constant.ScannerType,
 ascending: bool = True,
 date: str = None,
 count: shioaji.shioaji.ConstrainedIntValue = 100, # 0 <= count <= 200
 timeout: int = 30000,
 cb: Callable[[List[shioaji.data.ChangePercentRank]], NoneType] = None,
) -> List[shioaji.data.ChangePercentRank]
```

排名預設為由大到小排序，ascending 預設值為 True。若要由小到大排序請將 ascending 設為 False。count 為排行數量。

### 支援的排行類別

ChangePercentRank: 依價格漲跌幅排序  
 ChangePriceRank: 依價格漲跌排序  
 DayRangeRank: 依高低價差排序  
 VolumeRank: 依成交量排序  
 AmountRank: 依成交金額排序

## 範例

### 依價格漲跌幅排序

```
scanners = api.scanners(
 scanner_type=sj.constant.ScannerType.ChangePercentRank,
 count=1
)
scanners
```

### Cat

```
[
 ChangePercentRank(
 date='2021-04-09',
 code='5211',
 name='蒙恬',
 ts=1617978600000000000,
 open=16.4,
 high=17.6,
 low=16.35,
 close=17.6,
 price_range=1.25,
 tick_type=1,
 change_price=1.6,
 change_type=1,
 average_price=17.45,
 volume=7,
 total_volume=1742,
 amount=123200,
 total_amount=30397496,
 yesterday_volume=514,
 volume_ratio=3.39,
 buy_price=17.6,
 buy_volume=723,
 sell_price=0.0,
 sell_volume=0,
 bid_orders=237,
 bid_volumes=82,
 ask_orders=33,
 ask_volumes=64
)
]
```

## 轉成 DataFrame



```
scanners = api.scanners(
 scanner_type=sj.constant.ScannerType.ChangePercentRank,
 count=5
)
df = pd.DataFrame(s.__dict__ for s in scanners)
df.ts = pd.to_datetime(df.ts)
df
```



| <b>date</b> | <b>code</b> | <b>name</b> | <b>ts</b>                     | <b>open</b> | <b>high</b> | <b>low</b> | <b>close</b> | <b>price_range</b> |
|-------------|-------------|-------------|-------------------------------|-------------|-------------|------------|--------------|--------------------|
| 2023-01-17  | 6259        | 百徽          | 2023-01-17<br>11:11:41.030000 | 22.8        | 23.75       | 22.45      | 23.75        | 1.3                |
| 2023-01-17  | 6788        | 華景電         | 2023-01-17<br>11:19:01.924000 | 107         | 116         | 107        | 116          | 9                  |
| 2023-01-17  | 2540        | 愛山林         | 2023-01-17<br>11:17:39.435000 | 85.2        | 85.2        | 83         | 85.2         | 2.2                |
| 2023-01-17  | 8478        | 東哥遊艇        | 2023-01-17<br>11:18:33.702000 | 350.5       | 378         | 347        | 378          | 31                 |
| 2023-01-17  | 6612        | 奈米醫材        | 2023-01-17<br>11:15:32.752000 | 102         | 109         | 102        | 109          | 7                  |

## 屬性

## ChangePercentRank

**date (str):** 交易日  
**code (str):** 股票代號  
**name (str):** 股票名稱  
**ts (int):** 時間戳記  
**open (float):** 開盤價  
**high (float):** 最高價  
**low (float):** 最低價  
**close (float):** 收盤價  
**price\_range (float):** 價格區間(最高價-最低價)  
**tick\_type (int):** 內外盤別 {1: 內盤, 2: 外盤, 0: 無法判定}  
**change\_price (float):** 價格漲跌  
**change\_type (int):** 漲跌  
 {LimitUp, Up, Unchanged, Down, LimitDown}  
**average\_price (float):** 均價  
**volume (int):** 成交量  
**total\_volume (int):** 總成交量  
**amount (int):** 成交金額  
**total\_amount (int):** 總成交金額  
**yesterday\_volume (int):** 昨日總成交量  
**volume\_ratio (float):** 總成交量/昨日總成交量  
**buy\_price (float):** 委買價  
**buy\_volume (int):** 委買量  
**sell\_price (float):** 委賣價  
**sell\_volume (int):** 委賣量  
**bid\_orders (int):** 內盤總成交單量  
**bid\_volumes (int):** 內盤總成交量  
**ask\_orders (int):** 外盤總成交單量  
**ask\_volumes (int):** 外盤總成交量

## 5.4.7 處置及注意股

### 處置股

#### Disposition Stocks

```
>> api.punish?

Signature:
api.punish(
 timeout: int = 5000,
 cb: Callable[[shioaji.data.Punish], NoneType] = None,
) -> shioaji.data.Punish
Docstring: get punish information
```

### 範例



```
punish = api.punish()
punish
```



```
Punish(
 code=['2349', '2408', ...],
 start_date=[datetime.date(2025, 12, 17), datetime.date(2025, 12, 8), ...],
 end_date=[datetime.date(2025, 12, 31), datetime.date(2025, 12, 19), ...],
 updated_at=[datetime.datetime(2025, 12, 16, 18, 18, 20), datetime.datetime(2025, 12, 16, 18, 18, 20), ...],
 interval=['5分鐘', '5分鐘', ...],
 unit_limit=[10.0, 10.0, ...],
 total_limit=[30.0, 30.0, ...],
 description=['...', '...', ...],
 announced_date=[datetime.date(2025, 12, 16), datetime.date(2025, 12, 5), ...]
)
```

### 轉成DataFrame



```
df = pd.DataFrame(punish.__dict__)
df
```



| code | start_date | end_date   | updated_at             | interval | unit_limit | total_limit | description |
|------|------------|------------|------------------------|----------|------------|-------------|-------------|
| 2349 | 2025-12-17 | 2025-12-31 | 2025-12-16<br>18:18:20 | 5分鐘      | 10.0       | 30.0        | ...         |
| 2408 | 2025-12-08 | 2025-12-19 | 2025-12-16<br>18:18:20 | 5分鐘      | 10.0       | 30.0        | ...         |

## 屬性

 Punish

```
code (list[str]): 商品代碼
start_date (list[date]): 處置開始日期
end_date (list[date]): 處置結束日期
updated_at (list[datetime]): 更新時間
interval (list[str]): 撥合間隔
unit_limit (list[float]): 單筆委託上限
total_limit (list[float]): 單日委託上限
description (list[str]): 處置內容
announced_date (list[date]): 公告日期
```

## 注意股

 Attention Stocks

```
>> api.notice?

Signature:
api.notice(
 timeout: int = 5000,
 cb: Callable[[shioaji.data.Notice], NoneType] = None,
) -> shioaji.data.Notice
Docstring: get notice information
```

## 範例



```
notice = api.notice()
notice
```

 Out

```
Notice(
 code=['089508', '2349', ...],
 updated_at=[datetime.datetime(2025, 12, 16, 18, 18, 19), datetime.datetime(2025, 12, 16, 18, 18, 19), ...],
 close=[9.85, 16.2, ...],
 reason=['最近六個營業日累積收盤價漲幅達39.34% (第一款)', '最近六個營業日累積收盤價漲幅達41.59% (第一款) ...'],
 announced_date=[datetime.date(2025, 12, 16), datetime.date(2025, 12, 16), ...]
)
```

## 轉成DataFrame



```
df = pd.DataFrame(notice.__dict__)
df
```

 **Out**

| <b>code</b> | <b>updated_at</b>      | <b>close</b> | <b>reason</b>                         | <b>announced_date</b> |
|-------------|------------------------|--------------|---------------------------------------|-----------------------|
| 089508      | 2025-12-16<br>18:18:19 | 9.85         | 最近六個營業日累積收盤價漲幅達<br>39.34% ( 第一款 )。    | 2025-12-16            |
| 2349        | 2025-12-16<br>18:18:19 | 16.2         | 最近六個營業日累積收盤價漲幅達<br>41.59% ( 第一款 ) ... | 2025-12-16            |

屬性

 **Notice**

```
code (list[str]): 商品代碼
updated_at (list[datetime]): 更新時間
close (list[float]): 收盤價
reason (list[str]): 注意交易資訊
announced_date (list[date]): 公告日期
```

## 5.5 下單

### 5.5.1 證券



下單前必須先[登入](#)及啟用[憑證](#)。

證券委託單

#### 證券委託單

**version>=1.0      version<1.0**

```
price (float or int): 價格
quantity (int): 委託數量
action (str): {Buy: 買, Sell: 賣}
price_type (str): {LMT: 限價, MKT: 市價, MKP: 範圍市價}
order_type (str): 委託類別 {ROD, IOC, FOK}
order_cond (str): {Cash:現股, MarginTrading:融資, ShortSelling:融券}
order_lot (str): {
 Common:整股,
 Fixing:定盤,
 Odd:盤後零股,
 IntradayOdd:盤中零股
}
daytrade_short (bool): 先賣後買
custom_field (str): 備註，只允許輸入大小寫英文字母及數字，且長度最長為 6
account (:obj:Account): 下單帳號
ca (binary): 憑證

price (float or int): 價格
quantity (int): 委託數量
action (str): {Buy: 買, Sell: 賣}
price_type (str): {LMT: 限價, MKT: 市價, MKP: 範圍市價}
order_type (str): 委託類別 {ROD, IOC, FOK}
order_cond (str): {Cash:現股, MarginTrading:融資, ShortSelling:融券}
order_lot (str): {
 Common:整股,
 Fixing:定盤,
 Odd:盤後零股,
 IntradayOdd:盤中零股
}
first_sell (str): 先賣後買 {true, false}
custom_field (str): 備註，只允許輸入大小寫英文字母及數字，且長度最長為 6
account (:obj:Account): 下單帳號
ca (binary): 憑證
```

下單

下單時必須提供商品資訊 `contract` 及下單資訊 `order`。

#### 下單

```
api.place_order?

Signature:
 api.place_order(
 contract: shioaji.contracts.Contract,
 order: shioaji.order.Order,
 timeout: int = 5000,
 cb: Callable[[shioaji.order.Trade], NoneType] = None,
) -> shioaji.order.Trade
Docstring:
 placing order
```

#### 商品檔

```
contract = api.Contracts.Stocks.TSE.TSE2890
```

## 委託單

**version>=1.0      version<1.0**

```
order = api.Order(
 price=17,
 quantity=3,
 action=sj.constant.Action.Buy,
 price_type=sj.constant.StockPriceType.LMT,
 order_type=sj.constant.OrderType.ROD,
 order_lot=sj.constant.StockOrderLot.Common,
 # daytrade_short=False,
 custom_field="test",
 account=api.stock_account
)

order = api.Order(
 price=17,
 quantity=3,
 action=sj.constant.Action.Buy,
 price_type=sj.constant.TFTStockPriceType.LMT,
 order_type=sj.constant.TFTOrderType.ROD,
 order_lot=sj.constant.TFTStockOrderLot.Common,
 # first_sell=sj.constant.StockFirstSell.No,
 custom_field="test",
 account=api.stock_account
)
```

## 下單

```
trade = api.place_order(contract, order)
trade
```

## 完成

```
Trade(
 contract=Stock(
 exchange=<Exchange.TSE: 'TSE'>,
 code='2890',
 symbol='TSE2890',
 name='永豐金',
 category='17',
 unit=1000,
 limit_up=19.05,
 limit_down=15.65,
 reference=17.35,
 update_date='2023/01/12',
 day_trade=DayTrade.Yes: 'Yes'
),
 order=Order(
 action=<Action.Buy: 'Buy'>,
 price=17,
 quantity=3,
 id='531e27af',
 seqno='000002',
 ordno='000001',
 account=Account(
 account_type=<AccountType.Stock: 'S'>,
 person_id='A123456789',
 broker_id='9A95',
 account_id='1234567',
 signed=True
),
 custom_field='test',
 price_type=<StockPriceType.LMT: 'LMT'>,
 order_type=<OrderType.ROD: 'ROD'>,
 daytrade_short=False
),
 status=OrderStatus(
 id='531e27af',
 status=<Status.PendingSubmit: 'PendingSubmit'>,
 status_code='00',
 order_datetime=datetime.datetime(2023, 1, 12, 11, 18, 3, 867490),
 deals=[]
)
)
```

下單完同時也會收到從交易所傳回來的資料，詳情內容可詳見[下單回報](#)。

您需要執行 `update_status` 已更新 `trade` 物件的狀態。

### 更新委託狀態(成交後)

```
api.update_status(api.stock_account)
trade
```

### Cut

```
Trade(
 contract=Stock(
 exchange=<Exchange.TSE: 'TSE'>,
 code='2890',
 symbol='TSE2890',
 name='永豐金',
 category='17',
 unit=1000,
 limit_up=19.05,
 limit_down=15.65,
 reference=17.35,
 update_date='2023/01/12',
 day_trade=<DayTrade.Yes: 'Yes'>
),
 order=Order(
 action=Action.Buy: 'Buy',
 price=17,
 quantity=3,
 id='531e27af',
 seqno='000002',
 ordno='000001',
 account=Account(
 account_type=<AccountType.Stock: 'S'>,
 person_id='A123456789',
 broker_id='9495',
 account_id='1234567',
 signed=True
),
 custom_field='test',
 price_type=<StockPriceType.LMT: 'LMT'>,
 order_type=<OrderType.ROD: 'ROD'>,
 daytrade_short=False
),
 status=OrderStatus(
 id='531e27af',
 status=<Status.Submitted: 'Submitted'>,
 status_code='00',
 order_datetime=datetime(2023, 1, 12, 11, 18, 3, 867490),
 order_quantity=3,
 deals=[]
)
)
```

### 委託單狀態

- PendingSubmit : 傳送中
- PreSubmitted : 預約單
- Submitted : 傳送成功
- Failed : 失敗
- Cancelled : 已刪除
- Filled : 完全成交
- PartFilled : 部分成交

改單



```
api.update_order?

Signature:
api.update_order(
 trade: shioaji.order.Trade,
 price: Union[pydantic.types.StrictInt, float] = None,
 qty: int = None,
 timeout: int = 5000,
 cb: Callable[[shioaji.order.Trade], NoneType] = None,
) -> shioaji.order.Trade
Docstring: update the order price or qty
```

改價



```
api.update_order(trade=trade, price=17.5)
api.update_status(api.stock_account)
trade
```



```
Trade(
 contract=Stock(
 exchange=<Exchange.TSE: 'TSE'>,
 code='2890',
 symbol='TSE2890',
 name='永豐金',
 category='17',
 unit=1000,
 limit_up=19.05,
 limit_down=15.65,
 reference=17.35,
 update_date='2023/01/12',
 day_trade=<DayTrade.Yes: 'Yes'>
),
 order=Order(
 action=Action.Buy: 'Buy',
 price=17,
 quantity=3,
 id='531e27af',
 seqno='000002',
 ordno='000001',
 account=Account(
 account_type=<AccountType.Stock: 'S'>,
 person_id='A123456789',
 broker_id='9A95',
 account_id='1234567',
 signed=True
),
 custom_field='test',
 price_type=<StockPriceType.LMT: 'LMT'>,
 order_type=<OrderType.ROD: 'ROD'>,
 daytrade_short=False
),
 status=OrderStatus(
 id='531e27af',
 status=<Status.Submitted: 'Submitted'>,
 status_code='00',
 order_datetime=datetime(2023, 1, 12, 11, 18, 3, 867490),
 modified_price=17.5,
 order_quantity=3,
 deals=[]
)
)
```

改量(減量)

update\_order 只能用來減少原委託單的委託數量。

### 增量(減量)

```
api.update_order(trade=trade, qty=1)
api.update_status(api.stock_account)
trade
```

### Out

```
Trade(
 contract=Stock(
 exchange=<Exchange.TSE: 'TSE'>,
 code='2890',
 symbol='TSE2890',
 name='永豐金',
 category='17',
 unit=1000,
 limit_up=19.05,
 limit_down=15.65,
 reference=17.35,
 update_date='2023/01/12',
 day_trade=<DayTrade.Yes: 'Yes'>
),
 order=Order(
 action=<Action.Buy: 'Buy'>,
 price=17,
 quantity=3,
 id='531e27af',
 seqno='000002',
 ordno='000001',
 account=Account(
 account_type=<AccountType.Stock: 'S'>,
 person_id='A123456789',
 broker_id='9A95',
 account_id='1234567',
 signed=True
),
 custom_field='test',
 price_type=<StockPriceType.LMT: 'LMT'>,
 order_type=<OrderType.ROD: 'ROD'>,
 daytrade_short=False
),
 status=OrderStatus(
 id='531e27af',
 status=<Status.Submitted: 'Submitted'>,
 status_code='00',
 order_datetime=datetime(2023, 1, 12, 11, 18, 3, 867490),
 order_quantity=3,
 cancel_quantity=1,
 deals=[]
)
)
```

## 刪單

### 刪單

```
api.cancel_order(trade)
api.update_status(api.stock_account)
trade
```

 Out

```

Trade(
 contract=Stock(
 exchange=<Exchange.TSE: 'TSE'>,
 code='2890',
 symbol='TSE2890',
 name='永豐金',
 category='17',
 unit=1000,
 limit_up=19.05,
 limit_down=15.65,
 reference=17.35,
 update_date='2023/01/12',
 day_trade=<DayTrade.Yes: 'Yes'>
),
 order=Order(
 action=<Action.Buy: 'Buy'>,
 price=17,
 quantity=3,
 id='531e27af',
 seqno='000002',
 ordno='000001',
 account=Account(
 account_type=<AccountType.Stock: 'S'>,
 person_id='A123456789',
 broker_id='9A95',
 account_id='1234567',
 signed=True
),
 custom_field='test',
 price_type=<StockPriceType.LMT: 'LNT'>,
 order_type=<OrderType.ROD: 'ROD'>,
 daytrade_short=False
),
 status=OrderStatus(
 id='531e27af',
 status=<Status.Cancelled: 'Cancelled'>,
 status_code='00',
 order_datetime=datetime(2023, 1, 12, 11, 18, 3, 867490),
 order_quantity=3,
 cancel_quantity=3,
 deals=[]
)
)

```

成交

 更新委託狀態

```

api.update_status(api.stock_account)
trade

```



```
Trade(
 contract=Stock(
 exchange=<Exchange.TSE: 'TSE'>,
 code='2890',
 symbol='TSE2890',
 name='永豐金',
 category='17',
 unit=1000,
 limit_up=19.05,
 limit_down=15.65,
 reference=17.35,
 update_date='2023/01/12',
 day_trade=<DayTrade.Yes: 'Yes'>
),
 order=Order(
 action=<Action.Buy: 'Buy'>,
 price=17,
 quantity=3,
 id='531e27af',
 seqno='000002',
 ordno='000001',
 account=Account(
 account_type=<AccountType.Stock: 'S'>,
 person_id='A123456789',
 broker_id='9A95',
 account_id='1234567',
 signed=True
),
 custom_field='test',
 price_type=<StockPriceType.LMT: 'LNT'>,
 order_type=<OrderType.ROD: 'ROD'>,
 daytrade_short=False
),
 status=OrderStatus(
 id='531e27af',
 status=<Status.Filled: 'Filled'>,
 status_code='00',
 order_datetime=datetime(2023, 1, 12, 11, 18, 3, 867490),
 order_quantity=3,
 deals=[
 Deal(seq='000001', price=17, quantity=3, ts=1673501631.62918)
]
)
)
```

## 範例

### 證券下單範例 ( jupyter )

買賣別



```
order = api.Order(
 price=12,
 quantity=1,
 action=sj.constant.Action.Buy,
 price_type=sj.constant.StockPriceType.LMT,
 order_type=sj.constant.OrderType.ROD,
 order_lot=sj.constant.StockOrderLot.Common,
 custom_field="test",
 account=api.stock_account
)
```



```
order = api.Order(
 price=12,
 quantity=1,
 action=sj.constant.Action.Sell,
 price_type=sj.constant.StockPriceType.LMT,
 order_type=sj.constant.OrderType.ROD,
 order_lot=sj.constant.StockOrderLot.Common,
 custom_field="test",
 account=api.stock_account
)
```

## Daytrade Short

**version>=1.0    version<1.0**

```
order = api.Order(
 price=12,
 quantity=1,
 action=sj.constant.Action.SELL,
 price_type=sj.constant.StockPriceType.LMT,
 order_type=sj.constant.OrderType.ROD,
 order_lot=sj.constant.StockOrderLot.Common,
 daytrade_short=True,
 custom_field="test",
 account=api.stock_account
)

order = api.Order(
 price=12,
 quantity=1,
 action=sj.constant.Action.SELL,
 price_type=sj.constant.TFTStockPriceType.LMT,
 order_type=sj.constant.TFTOrderType.ROD,
 order_lot=sj.constant.TFTStockOrderLot.Common,
 first_sell=sj.constant.StockFirstSell.Yes,
 custom_field="test",
 account=api.stock_account
)
```

## ROD + LMT

## ROD + LMT

**version>=1.0    version<1.0**

```
order = api.Order(
 price=12,
 quantity=1,
 action=sj.constant.Action.SELL,
 price_type=sj.constant.StockPriceType.LMT,
 order_type=sj.constant.OrderType.ROD,
 order_lot=sj.constant.StockOrderLot.Common,
 custom_field="test",
 account=api.stock_account
)

order = api.Order(
 price=12,
 quantity=1,
 action=sj.constant.Action.SELL,
 price_type=sj.constant.TFTStockPriceType.LMT,
 order_type=sj.constant.TFTOrderType.ROD,
 order_lot=sj.constant.TFTStockOrderLot.Common,
 custom_field="test",
 account=api.stock_account
)
```

## 5.5.2 期貨選擇權



下單前必須先[登入](#)及啟用[憑證](#)。

### 期貨委託單

```
price (float or int): 價格
quantity (int): 委託數量
action (str): {Buy: 買, Sell: 賣}
price_type (str): {LMT: 限價, MKT: 市價, MKP: 範圍市價}
order_type (str): 委託類別 {ROD, IOC, FOK}
octype (str): {Auto: 自動, New: 新倉, Cover: 平倉, DayTrade: 當沖}
account (:obj:Account): 下單帳號
ca (binary): 憑證
```

### 下單

下單時必須提供商品資訊 contract 及下單資訊 order。



```
api.place_order?

Signature:
 api.place_order(
 contract: shioaji.contracts.Contract,
 order: shioaji.order.Order,
 timeout: int = 500,
 cb: Callable[[shioaji.order.Trade], NoneType] = None,
) -> shioaji.order.Trade
Docstring:
 placing order
```



```
contract = api.Contracts.Futures.TXF.TXF202301
```



### version>=1.0      version<1.0

```
order = api.Order(
 action=sj.constant.Action.Buy,
 price=14400,
 quantity=3,
 price_type=sj.constant.FuturesPriceType.LMT,
 order_type=sj.constant.OrderType.ROD,
 octype=sj.constant.FuturesOCType.Auto,
 account=api.futopt_account
)

order = api.Order(
 action=sj.constant.Action.Buy,
 price=14400,
 quantity=3,
 price_type=sj.constant.FuturesPriceType.LMT,
 order_type=sj.constant.FuturesOrderType.ROD,
 octype=sj.constant.FuturesOCType.Auto,
 account=api.futopt_account
)
```



```
trade = api.place_order(contract, order)
trade
```



```
Trade(
 contract=Future(
 code='TXXA3',
 symbol='TF202301',
 name='臺股期貨01',
 category='TXX',
 delivery_month='202301',
 delivery_date='2023/01/30',
 underlying_kind='I',
 unit=1,
 limit_up=16270.0,
 limit_down=13312.0,
 reference=14791.0,
 update_date='2023/01/12'
),
 order=Order(
 action<Action.Buy: 'Buy'>,
 price=14400,
 quantity=3,
 id='5efffd1',
 seqno='000004',
 ordno='000003',
 account=Account(
 account_type=<AccountType.Future: 'F'>,
 person_id='A123456789',
 broker_id='F002000',
 account_id='1234567',
 signed=True
),
 price_type=<StockPriceType.LMT: 'LMT'>,
 order_type=<OrderType.ROD: 'ROD'>
),
 status=OrderStatus(
 id='5efffd1',
 status=<Status.PendingSubmit: 'PendingSubmit'>,
 status_code='00',
 order_datetime=datetime.datetime(2023, 1, 12, 14, 56, 13, 995651),
 deals=[]
)
)
```

下單完同時也會收到從交易所傳回來的資料，詳情內容可詳見[下單回報](#)。

您需要執行 `update_status` 已更新 trade 物件的狀態。



```
api.update_status(api.futopt_account)
trade
```

## ✓ 單

```
Trade(
 contract=Future(
 code='TXFA3',
 symbol='TF202301',
 name='臺股期貨01',
 category='TTF',
 delivery_month='202301',
 delivery_date='2023/01/30',
 underlying_kind='I',
 unit=1,
 limit_up=16270.0,
 limit_down=13312.0,
 reference=14791.0,
 update_date='2023/01/12'
),
 order=Order(
 action=Action.Buy: 'Buy',
 price=14400,
 quantity=3,
 id='5efffd1',
 seqno='000004',
 ordno='000003',
 account=Account(
 account_type=<AccountType.Future: 'F'>,
 person_id='A123456789',
 broker_id='F002000',
 account_id='1234567',
 signed=True
),
 price_type=<StockPriceType.LMT: 'LNT'>,
 order_type=<OrderType.ROD: 'ROD'>
),
 status=OrderStatus(
 id='5efffd1',
 status=<Status.Submitted: 'Submitted'>,
 status_code='00',
 order_datetime=datetime(2023, 1, 12, 14, 56, 13, 995651),
 deals=[]
)
)
```

## ↑ 委託單狀態

- PendingSubmit : 傳送中
- PreSubmitted : 預約單
- Submitted : 傳送成功
- Failed : 失敗
- Cancelled : 已刪除
- Filled : 完全成交
- PartFilled : 部分成交

改善

## ↑ 改單

```
api.update_order?

Signature:
 api.update_order(
 trade: shioaji.order.Trade,
 price: Union[pydantic.types.StrictInt, float] = None,
 qty: int = None,
 timeout: int = 5000,
 cb: Callable[[shioaji.order.Trade], NoneType] = None,
) -> shioaji.order.Trade
Docstring: update the order price or qty
```

## 改價



```
api.update_order(trade=trade, price=14450)
api.update_status(api.futopt_account)
trade
```



```
Trade(
 contract=Future(
 code='TXFA3',
 symbol='TF202301',
 name='臺股期貨01',
 category='TF',
 delivery_month='202301',
 delivery_date='2023/01/30',
 underlying_kind='I',
 unit=1,
 limit_up=16270.0,
 limit_down=13312.0,
 reference=14791.0,
 update_date='2023/01/12'
),
 order=Order(
 action=Action.Buy: 'Buy',
 price=14400,
 quantity=3,
 id='5efffde1',
 seqno='000004',
 ordno='000003',
 account=Account(
 account_type=<AccountType.Future: 'F'>,
 person_id='A123456789',
 broker_id='F002000',
 account_id='1234567',
 signed=True
),
 price_type=<StockPriceType.LMT: 'LMT'>,
 order_type=<OrderType.ROD: 'ROD'>
),
 status=OrderStatus(
 id='5efffde1',
 status=<Status.Submitted: 'Submitted'>,
 status_code='00',
 order_datetime=datetime(2023, 1, 12, 14, 56, 13, 995651),
 modified_price=14450,
 order_quantity=3,
 deals=[]
)
)
```

## 改量(減量)

`update_order` 只能用來減少原委託單的委託數量。



```
api.update_order(trade=trade, qty=1)
api.update_status(api.futopt_account)
trade
```



```

Trade(
 contract=Future(
 code='TXFA3',
 symbol='TXF202301',
 name='臺股期貨01',
 category='TXX',
 delivery_month='202301',
 delivery_date='2023/01/30',
 underlying_kind='I',
 unit=1,
 limit_up=16270.0,
 limit_down=13312.0,
 reference=14791.0,
 update_date='2023/01/12'
),
 order=Order(
 action=<Action.Buy: 'Buy'>,
 price=14400,
 quantity=3,
 id='5efffd1',
 seqno='000004',
 ordno='000003',
 account=Account(
 account_type=<AccountType.Future: 'F'>,
 person_id='A123456789',
 broker_id='F002000',
 account_id='1234567',
 signed=True
),
 price_type=<StockPriceType.LMT: 'LNT'>,
 order_type=<OrderType.ROD: 'ROD'>
),
 status=OrderStatus(
 id='5efffd1',
 status=<Status.Submitted: 'Submitted'>,
 status_code='00',
 order_datetime=datetime(2023, 1, 12, 14, 56, 13, 995651),
 order_quantity=3,
 cancel_quantity=1,
 deals=[]
)
)
)

```

刪單



```

api.cancel_order(trade)
api.update_status(api.futopt_account)
trade

```



```

Trade(
 contract=Future(
 code='TXFA3',
 symbol='TF202301',
 name='臺股期貨01',
 category='TXX',
 delivery_month='202301',
 delivery_date='2023/01/30',
 underlying_kind='I',
 unit=1,
 limit_up=16270.0,
 limit_down=13312.0,
 reference=14791.0,
 update_date='2023/01/12'
),
 order=Order(
 action=<Action.Buy: 'Buy'>,
 price=14400,
 quantity=3,
 id='5efffd1',
 seqno='000004',
 ordno='000003',
 account=Account(
 account_type=<AccountType.Future: 'F'>,
 person_id='A123456789',
 broker_id='F002000',
 account_id='1234567',
 signed=True
),
 price_type=<StockPriceType.LMT: 'LMT'>,
 order_type=<OrderType.ROD: 'ROD'>
),
 status=OrderStatus(
 id='5efffd1',
 status=<Status.Cancelled: 'Cancelled'>,
 status_code='00',
 order_datetime=datetime(2023, 1, 12, 14, 56, 13, 995651),
 order_quantity=3,
 cancel_quantity=3,
 deals=[]
)
)
)

```

成交

**更新委託狀態(成交後)**

```
api.update_status(api.futopt_account)
trade
```



```
Trade(
 contract=Future(
 code='TXFA3',
 symbol='TF202301',
 name='臺股期貨01',
 category='TTF',
 delivery_month='202301',
 delivery_date='2023/01/30',
 underlying_kind='I',
 unit=1,
 limit_up=16270.0,
 limit_down=13312.0,
 reference=14791.0,
 update_date='2023/01/12'
),
 order=Order(
 action=<Action.Buy: 'Buy'>,
 price=14400,
 quantity=3,
 id='5efffd1',
 seqno='000004',
 ordno='000003',
 account=Account(
 account_type=<AccountType.Future: 'F'>,
 person_id='A123456789',
 broker_id='F002000',
 account_id='1234567',
 signed=True
),
 price_type=<StockPriceType.LMT: 'LNT'>,
 order_type=<OrderType.ROD: 'ROD'>
),
 status=OrderStatus(
 id='5efffd1',
 status=<Status.Filled: 'Filled'>,
 status_code='00',
 order_datetime=datetime(2023, 1, 12, 14, 56, 13, 995651),
 order_quantity=3,
 deals=[
 Deal(seq='000001', price=14400, quantity=3, ts=1673501631.62918)
]
)
)
```

## 範例

### 期權下單範例 ( jupyter )

買賣別



```
order = api.Order(
 action=sj.constant.Action.Buy,
 price=14400,
 quantity=2,
 price_type=sj.constant.FuturesPriceType.LMT,
 order_type=sj.constant.OrderType.ROD,
 octype=sj.constant.FuturesOCTYPE.Auto,
 account=api.futopt_account
)
```



```
order = api.Order(
 action=sj.constant.Action.Sell,
 price=14400,
 quantity=2,
 price_type=sj.constant.FuturesPriceType.LMT,
 order_type=sj.constant.OrderType.ROD,
 octype=sj.constant.FuturesOCTYPE.Auto,
 account=api.futopt_account
)
```

**ROD + LMT****ROD + LMT**

```
order = api.Order(
 action=sj.constant.Action.Sell,
 price=1400,
 quantity=2,
 price_type=sj.constant.FuturesPriceType.LNT,
 order_type=sj.constant.OrderType.ROD,
 octype=sj.constant.FuturesOCType.Auto,
 account=api.futopt_account
)
```

### 5.5.3 零股

零股下單範例 ( jupyter )



下單前必須先登入及啟用憑證。

下單



```
contract = api.Contracts.Stocks.TSE.TSE0050
order = api.Order(
 price=90,
 quantity=10,
 action=sj.constant.Action.Buy,
 price_type=sj.constant.StockPriceType.LMT,
 order_type=sj.constant.OrderType.ROD,
 order_lot=sj.constant.StockOrderLot.IntradayOdd,
 account=api.stock_account,
)
trade = api.place_order(contract, order)
trade
```



```
Trade(
 contract=Stock(
 exchange=<Exchange.TSE: 'TSE'>,
 code='0050',
 symbol='TSE0050',
 name='元大台灣50',
 category='00',
 limit_up=115.8,
 limit_down=94.8,
 reference=105.3,
 update_date='2020/09/21',
 margin_trading_balance=15390,
 short_selling_balance=2,
 day_trade=<DayTrade.Yes: 'Yes'>
),
 order=Order(
 action=<Action.Buy: 'Buy'>,
 price=90.0,
 quantity=10,
 id='38e68afe',
 seqno='482283',
 ordno='W4313',
 account=Account(
 account_type=<AccountType.Stock: 'S'>,
 person_id='YOUR_PERSON_ID',
 broker_id='9495',
 account_id='0506112',
 signed=True
),
 price_type=<StockPriceType.LMT: 'LMT'>,
 order_type=<OrderType.ROD: 'ROD'>,
 order_lot=<StockOrderLot.IntradayOdd: 'IntradayOdd'>
),
 status=OrderStatus(
 id='38e68afe',
 status=<Status.Submitted: 'Submitted'>,
 status_code='00',
 order_datetime=datetime.datetime(2020, 9, 21, 14, 38, 51),
 deals=[]
)
)
```

## 改單



**零股不能進行改價**

update\_order 只能用來減少原委託單的委託數量。



```
api.update_order(trade=trade, qty=2)
api.update_status(api.stock_account)
trade
```



```
Trade(
 contract=Stock(
 exchange=<Exchange.TSE: 'TSE'>,
 code='0050',
 symbol='TSE0050',
 name='元大台灣50',
 category='00',
 limit_up=115.8,
 limit_down=94.8,
 reference=105.3,
 update_date='2020/09/21',
 margin_trading_balance=15390,
 short_selling_balance=2,
 day_trade=<DayTrade.Yes: 'Yes'>
),
 order=Order(
 action=<Action.Buy: 'Buy'>,
 price=90.0,
 quantity=10,
 id='9b44c3b2',
 seqno='482293',
 ordno='WA328',
 account=Account(
 account_type=<AccountType.Stock: 'S'>,
 person_id='YOUR_PERSON_ID',
 broker_id='9A95',
 account_id='0506112',
 signed=True
),
 price_type=<StockPriceType.LMT: 'LMT'>,
 order_type=<OrderType.ROD: 'ROD'>,
 order_lot=<StockOrderLot.IntradayOdd: 'IntradayOdd'>),
 status=OrderStatus(
 id='9b44c3b2',
 status=<Status.Submitted: 'Submitted'>,
 status_code='00',
 order_datetime=datetime(2020, 9, 21, 14, 54, 36),
 cancel_quantity=2,
 deals=[]
)
)
```

## 刪單



```
api.cancel_order(trade)
api.update_status(api.stock_account)
trade
```



```

Trade(
 contract=Stock(
 exchange=<Exchange.TSE: 'TSE'>,
 code='0050',
 symbol='TSE0050',
 name='元大台灣50',
 category='00',
 limit_up=115.8,
 limit_down=94.8,
 reference=105.3,
 update_date='2020/09/21',
 margin_trading_balance=15390,
 short_selling_balance=2,
 day_trade=<DayTrade.Yes: 'Yes'>
),
 order=Order(
 action=<Action.Buy: 'Buy'>,
 price=90.0,
 quantity=10,
 id='9b44c3b2',
 seqno='482293',
 ordno='WA328',
 account=Account(
 account_type=<AccountType.Stock: 'S'>,
 person_id='YOUR_PERSON_ID',
 broker_id='9A95',
 account_id='0506112',
 signed=True
),
 price_type=<StockPriceType.LMT: 'LNT'>,
 order_type=<OrderType.ROD: 'ROD'>,
 order_lot=<StockOrderLot.IntradayOdd: 'IntradayOdd'>
),
 status=OrderStatus(
 id='9b44c3b2',
 status=<Status.Cancelled: 'Cancelled'>,
 status_code='00',
 order_datetime=datetime(2020, 9, 21, 14, 54, 36),
 cancel_quantity=10,
 deals=[]
)
)

```

## 5.5.4 組合單



下單前必須先[登入](#)及啟用[憑證](#)。

### 下單

組合單提供類型包括:**價格價差、時間價差、跨式、勒式、轉換以及逆轉**。組合規則詳見期交所[文件](#)。



```
api.place_comboorder?

Signature:
 api.place_comboorder(
 combo_contract: shioaji.contracts.ComboContract,
 order: shioaji.order.ComboOrder,
 timeout: int = 5000,
 cb: Callable[[shioaji.order.ComboTrade], NoneType] = None,
)
Docstring:
 placing combo order
```

下單時必須提供商品資訊 contract 及下單資訊 order。商品資訊無關前後順序，只需提供認可的組合。



```
contract_1 = api.Contracts.Options.TX4.TX420211017850C
contract_2 = api.Contracts.Options.TX4.TX420211017850P
combo_contract = sj.contracts.ComboContract(
 legs=[
 sj.contracts.ComboBase(action="Sell", **contract_1.dict()),
 sj.contracts.ComboBase(action="Sell", **contract_2.dict()),
]
)
```



```
order = api.ComboOrder(
 price_type="LMT",
 price=1,
 quantity=1,
 order_type="IOC",
 octype=sj.constant.FuturesOCType.New,
)
```



```
trade = api.place_comboorder(combo_contract, order)
```

### 刪單

trade 為要刪的單，可從[查詢](#)取得。



```
api.cancel_comboorder(trade)
```

## 查詢狀態

如同 `list_trades` 及 `update_status` 的概念。在取得組合單狀態前，必須利用 `update_combostatus` 更新狀態。



```
api.update_combostatus()
api.list_combotrades()
```



```
[
 ComboTrade(
 contract=ComboContract(
 legs=[
 ComboBase(
 security_type=<SecurityType.Option: 'OPT'>,
 exchange=<Exchange.TAIFEX: 'TAIFEX'>,
 code='TX16000L1',
 symbol='TX5202112016000C',
 name='臺指選擇權12W5月 16000C',
 category='TX5',
 delivery_month='202112',
 delivery_date='2021/12/29',
 strike_price=16000.0,
 option_right=<OptionRight.Call: 'C'>,
 underlying_kind='I',
 unit=1,
 limit_up=3630.0,
 limit_down=68.0,
 reference=1850.0,
 update_date='2021/12/23',
 action=<Action.Sell: 'Sell'>),
 ComboBase(
 security_type=<SecurityType.Option: 'OPT'>,
 exchange=<Exchange.TAIFEX: 'TAIFEX'>,
 code='TX16000XL1',
 symbol='TX5202112016000P',
 name='臺指選擇權12W5月 16000P',
 category='TX5',
 delivery_month='202112',
 delivery_date='2021/12/29',
 strike_price=16000.0,
 option_right=<OptionRight.Put: 'P'>,
 underlying_kind='I',
 unit=1,
 limit_up=1780.0,
 limit_down=0.1,
 reference=0.9,
 update_date='2021/12/23',
 action=<Action.Sell: 'Sell'>)
],
),
 order=Order(
 action=<Action.Sell: 'Sell'>,
 price=1.0,
 quantity=1,
 id='46989de8',
 seqno='743595',
 ordno='000000',
 account=Account(
 account_type=<AccountType.Future: 'F'>,
 person_id='YOUR_PERSON_ID',
 broker_id='FO02000',
 account_id='1234567',
 signed=True
),
 price_type=<StockPriceType.LMT: 'LMT'>,
 order_type=<OrderType.IOC: 'IOC'>,
 octype=<FuturesOCType.New: 'New'>
),
 status=ComboStatus(
 id='46989de8',
 status=<Status.Failed: 'Failed'>,
 status_code='9909',
 order_datetime=datetime.datetime(2021, 12, 23, 8, 46, 47),
 msg='可委託金額不足',
 modified_price=1.0,
 deals={}
)
)
]
```

## 5.5.5 預收券款

當現貨觸發一些交易異常條件，需先預收券款。異常條件包括：注意股票、警示股票、處置股票及管理股票。



- 必須先[登入](#)及啟用[憑證](#)。
- 服務時間為交易日8:00~14:30。

查詢圈券狀態



```
reserve_summary_resp = api.stock_reserve_summary(account)
```



```
ReserveStocksSummaryResponse(
 response=ReserveStocksSummary(
 stocks=[
 ReserveStockSummary(
 contract=Contract(
 security_type=<SecurityType.Stock: 'STK'>,
 exchange=<Exchange.TSE: 'TSE'>,
 code='2890',
 name='永豐金'
),
 available_share=5000,
 reserved_share=0
)
],
 account=StockAccount(
 person_id='X123456789',
 broker_id='9A95',
 account_id='12345678',
 signed=True
)
)
)
```

借券圈券申請



```
contract = api.Contracts.Stocks["2890"]
resp = api.reserve_stock(account, contract, 1000)
```



```
ReserveStockResponse(
 response=ReserveOrderResp(
 contract=Stock(
 exchange=<Exchange.TSE: 'TSE'>,
 code='2890',
 symbol='TSE2890',
 name='永豐金'
),
 account=StockAccount(
 person_id='X123456789',
 broker_id='9495',
 account_id='12345678',
 signed=True),
 share=1000,
 status=True,
 info=''
)
)
```

## 查詢圈券明細



```
resp = api.stock_reserve_detail(account)
```



```
ReserveStocksDetailResponse(
 response=ReserveStocksDetail(stocks=[
 ReserveStockDetail(
 contract=Contract(
 security_type=<SecurityType.Stock: 'STK'>,
 exchange=<Exchange.TSE: 'TSE'>,
 code='6153',
 name='嘉聯益'
),
 share=1000,
 order_ts=1638253253,
 status=True,
 info='已完成'
)
],
 account=StockAccount(
 person_id='X123456789',
 broker_id='9495',
 account_id='12345678',
 signed=True)
)
```

## 預收款項申請



```
contract = api.Contracts.Stocks["2890"]
resp = api.reserve_earmarking(account, contract, 1000, 15.15)
```



```
ReserveEarmarkingResponse(
 response=EarmarkingOrderResp(
 contract=Stock(
 exchange=<Exchange.TSE: 'TSE'>,
 code='2890',
 symbol='TSE2890',
 name='永豐金',
),
 account=StockAccount(
 person_id='X123456789',
 broker_id='9A95',
 account_id='12345678',
 signed=True
),
 share=1000,
 price=15.15,
 status=True,
 info='OK'
)
)
```

## 查詢預收款項



```
api.earmarking_detail(account)
```



```
EarmarkStocksDetailResponse(
 response=EarmarkStocksDetail(
 stocks:[
 EarmarkStockDetail(
 contract=Contract(
 security_type=<SecurityType.Stock: 'STK'>,
 exchange=<Exchange.TSE: 'TSE'>,
 code='2890',
 name='永豐金'
),
 share=1000,
 price=15.15,
 amount=15171,
 order_ts=1638416488,
 status=False,
 info='扣款失敗'),
 EarmarkStockDetail(
 contract=Contract(
 security_type=<SecurityType.Stock: 'STK'>,
 exchange=<Exchange.TSE: 'TSE'>,
 code='2890',
 name='永豐金'
),
 share=1000,
 price=15.15,
 amount=15171,
 order_ts=1638415662, status=True,
 info='')
],
 account=StockAccount(
 person_id='X123456789',
 broker_id='9A95',
 account_id='12345678',
 signed=True
)
)
)
```

## 範例

## 查詢所有名下帳號的圈券狀態



```
import shioaji as sj

api = sj.Shioaji()
accounts = api.login("YOUR_PERSON_ID", "YOUR_PASSWORD", contracts_timeout=10000)
api.activate_ca(
 ca_path="/c/your/ca/path/Sinopac.pfx",
 ca_passwd="YOUR_CA_PASSWORD",
 person_id="Person of this Ca",
)
for account in accounts:
 if account.account_type == AccountType.Stock:
 reserve_summary_resp = api.stock_reserve_summary(account)
 for reserve_stock_summary in reserve_summary_resp.response.stocks:
 if reserve_stock_summary.available_share:
 resp = api.reserve_stock(
 account,
 reserve_stock_summary.contract,
 reserve_stock_summary.available_share
)
```

## 5.5.6 查詢狀態



必須先[登入](#)及啟用[憑證](#)。

在取得 Trade 狀態前，必須先利用 update\_status 進行更新。如果無法成功刪單或改單，你可以對特定 trade 物件進行更新，並確認在 trade 中的 OrderStatus，是否為可刪改狀態。update\_status 預設查詢為名下所有帳號。若想查詢特定帳號，將帳號帶入 account。

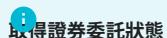


api.update\_status?



Signature:  
api.update\_status(  
 account: shioaji.account.Account = None,  
 trade: shioaji.order.Trade = None,  
 timeout: int = 5000,  
 cb: Callable[[List[shioaji.order.Trade]], NoneType] = None,  
)  
Docstring: update status of [all](#) trades you have

取得證券委託狀態



api.update\_status(api.stock\_account)  
api.list\_trades()



```
[
 Trade(
 contract=Stock(
 exchange=<Exchange.TSE: 'TSE'>,
 code='2890',
 symbol='TSE2890',
 name='永豐金',
 category='17',
 unit=1000,
 limit_up=19.05,
 limit_down=15.65,
 reference=17.35,
 update_date='2023/01/12',
 day_trade=<DayTrade.YesNo: 'Yes'>
),
 order=Order(
 action=<Action.Buy: 'Buy'>,
 price=17,
 quantity=3,
 id='531e27af',
 seqno='000002',
 ordno='000001',
 account=Account(
 account_type=<AccountType.Stock: 'S'>,
 person_id='A123456789',
 broker_id='9A95',
 account_id='1234567',
 signed=True
),
 custom_field='test',
 price_type=<StockPriceType.LMT: 'LMT'>,
 order_type=<OrderType.ROD: 'ROD'>,
 daytrade_short=True
),
 status=OrderStatus(
 id='531e27af',
 status=<Status.Filled: 'Filled'>,
 status_code='00',
 order_datetime=datetime.datetime(2023, 1, 12, 11, 18, 3, 867490),
 order_quantity=3,
 deals=[
 Deal(seq='000001', price=17, quantity=3, ts=1673501631.62918)
]
)
)
]
]
```

#### 取得期貨委託狀態



```
api.update_status(api.futopt_account)
api.list_trades()
```



```
[
 Trade(
 contract=Future(
 code='TFX3',
 symbol='TFX202301',
 name='臺股期貨01',
 category='TFX',
 delivery_month='202301',
 delivery_date='2023/01/30',
 underlying_kind='I',
 unit=1,
 limit_up=16270.0,
 limit_down=13312.0,
 reference=14791.0,
 update_date='2023/01/12'
),
 order=Order(
 action=<Action.Buy: 'Buy'>,
 price=14400,
 quantity=3,
 id='5efffd1',
 seqno='000004',
 ordno='000003',
 account=Account(
 account_type=<AccountType.Future: 'F'>,
 person_id='A123456789',
 broker_id='F002000',
 account_id='1234567',
 signed=True
),
 price_type=<StockPriceType.LMT: 'LMT'>,
 order_type=<OrderType.ROD: 'ROD'>
),
 status=OrderStatus(
 id='5efffd1',
 status=<Status.Filled: 'Filled'>,
 status_code='00',
 order_datetime=datetime(2023, 1, 12, 14, 56, 13, 995651),
 order_quantity=3,
 deals=[
 Deal(seq='000001', price=14400, quantity=3, ts=1673501631.62918)
]
)
)
]
```

## 更新特定交易狀態

### 更新特定交易狀態

```
you can get trade from place_order
trade = api.place_order(contract, order)

or get from api.list_trades()
trade = api.list_trades()[0]

api.update_status(trade=trade)
```

## 委託及成交狀態屬性

### 委託狀態屬性

```
id (str): 關聯Order物件編碼
status (:obj:Status): {
 Cancelled: 已刪除,
 Filled: 完全成交,
 PartFilled: 部分成交,
 Failed: 失敗,
 PendingSubmit: 傳送中,
 PreSubmitted: 預約單,
 Submitted: 傳送成功
}
status_code (str): 狀態碼
order_datetime (datetime): 委託時間
order_quantity (int): 委託數量
modified_price (float): 改價金額
cancel_quantity (int): 取消委託數量
deals (:List:Deal): 成交資訊
```

### 成交屬性

```
seq (str): 成交序號
price (int or float): 成交價
quantity (int): 成交數量
ts (float): 成交時間戳
```

## 5.5.7 回報資訊

### 證券

#### 委託回報

當證交所收到委託將會回傳回報。在回報中分為四部分，包括operation、order、status及contract。以下我們會在進行詳細的說明。

## 委託回報

**version>=1.0      version<1.0**

```
OrderState.StockOrder {
 'operation': {
 'op_type': 'New',
 'op_code': '00',
 'op_msg': ''
 },
 'order': {
 'id': '97b63e2f',
 'seqno': '267677',
 'ordno': 'IM394',
 'account': {
 'account_type': 'S',
 'person_id': '',
 'broker_id': '9A95',
 'account_id': '1234567',
 'signed': True
 },
 'action': 'Buy',
 'price': 16.0,
 'quantity': 1,
 'order_type': 'ROD',
 'price_type': 'LMT',
 'order_cond': 'Cash',
 'order_lot': 'Common',
 'custom_field': 'test'
 },
 'status': {
 'id': '97b63e2f',
 'exchange_ts': 1673576134.038,
 'modified_price': 0.0,
 'cancel_quantity': 0,
 'order_quantity': 1,
 'web_id': '137'
 },
 'contract': {
 'security_type': 'STK',
 'exchange': 'TSE',
 'code': '2890',
 'symbol': '',
 'name': '',
 'currency': 'TWD'
 }
}

OrderState.TFTOrder {
 'operation': {
 'op_type': 'New',
 'op_code': '00',
 'op_msg': ''
 },
 'order': {
 'id': '97b63e2f',
 'seqno': '267677',
 'ordno': 'IM394',
 'account': {
 'account_type': 'S',
 'person_id': '',
 'broker_id': '9A95',
 'account_id': '1234567',
 'signed': True
 },
 'action': 'Buy',
 'price': 16.0,
 'quantity': 1,
 'order_type': 'ROD',
 'price_type': 'LMT',
 'order_cond': 'Cash',
 'order_lot': 'Common',
 'custom_field': 'test'
 },
 'status': {
 'id': '97b63e2f',
 'exchange_ts': 1673576134.038,
 'modified_price': 0.0,
 'cancel_quantity': 0,
 'order_quantity': 1,
 'web_id': '137'
 },
 'contract': {
 'security_type': 'STK',
 'exchange': 'TSE',
 'code': '2890',
 'symbol': '',
 'name': '',
 'currency': 'TWD'
 }
}
```

## 委託回報資訊

### operation

```
op_type (str): {
 "New": 新單,
 "Cancel": 刪單,
 "UpdatePrice": 改價,
 "UpdateQty": 改量
}
op_code (str): {"00": 成功, others: 失敗}
op_msg (str): 錯誤訊息
```

### order

```
id (str): 與成交回報的trade_id相同
seqno (str): 平台單號
ordno (str): 委託單號
account (dict): 帳戶資訊
action (str): 買賣別 {Buy, Sell}
price (float or int): 委託價格
quantity (int): 委託數量
order_type (str): 委託類別 {ODD, IOC, FOK}
price_type (str): {LMT: 限價, MKT: 市價, MKP: 範圍市價}
order_cond (str): {
 Cash: 現股,
 MarginTrading: 融資,
 ShortSelling: 融券
}
order_lot (str): {
 Common: 整股,
 Fixing: 定盤,
 Odd: 盤後零股,
 IntradayOdd: 盤中零股
}
custom_field (str): 自訂欄位
```

### status

```
id (str): 與成交回報的trade_id相同
exchange_ts (int): 交易所時間
modified_price (float or int): 改價
cancel_quantity (int): 取消數量
order_quantity (int): 委託數量
web_id (str): 下單平台代碼
```

### contract

```
security_type (str): 商品類別
exchange (str): 交易所
code (str): 商品代碼
symbol (str): 符號
name (str): 商品名稱
currency (str): 幣別
```

## 成交回報

當搓合成功，證交所會傳送成交回報告知。搓合成功包含部分成交以及完全成交，可以從委託回報中的 id 去對應成交回報中的 trade\_id 去確認是否為同一筆委託單。

## 成交回報

**version>=1.0      version<1.0**

```
OrderState.StockDeal {
 'trade_id': '9c6ae2eb',
 'seqno': '269866',
 'ordno': 'IN497',
 'exchange_seq': '669915',
 'broker_id': '9A95',
 'account_id': '1234567',
 'action': 'Buy',
 'code': '2890',
 'order_cond': 'Cash',
 'order_lot': 'IntradayOdd',
 'price': 267.5,
 'quantity': 3,
 'web_id': '137',
 'custom_field': 'test',
 'ts': 1673577256.354
}

OrderState.TFTDeal {
 'trade_id': '9c6ae2eb',
 'seqno': '269866',
 'ordno': 'IN497',
 'exchange_seq': '669915',
 'broker_id': '9A95',
 'account_id': '1234567',
 'action': 'Buy',
 'code': '2890',
 'order_cond': 'Cash',
 'order_lot': 'IntradayOdd',
 'price': 267.5,
 'quantity': 3,
 'web_id': '137',
 'custom_field': 'test',
 'ts': 1673577256.354
}
```

## 成交回報

**trade\_id (str):** 與委託回報id相同  
**seqno (str):** 平台單號  
**ordno (str):** 前五碼為同委託回報委託單號，後三碼為同筆委託成交交易序號。  
**exchange\_seq (str):** 回報序號  
**broker\_id (str):** 分行代碼  
**account\_id (str):** 帳號  
**action (str):** 買賣別 {Buy, Sell}  
**code (str):** 商品代碼  
**order\_cond (str):** {  
 Cash: 現股,  
 MarginTrading: 融資,  
 ShortSelling: 融券  
}  
**order\_lot (str):** {  
 Common: 整股,  
 Fixing: 定盤,  
 Odd: 盤後零股,  
 IntradayOdd: 盤中零股  
}  
**price (float or int):** 成交價  
**quantity (int):** 成交量  
**web\_id (str):** 平台代碼  
**custom\_field (str):** 自訂欄位  
**ts (int):** 成交時間戳

## 注意

交易所回傳訊息優先順序成就好報大於委託回報，所以當委託立即成交可能會先收到成就好報。

### 回報處理

欲處理委託、成就好報，詳細可參見[Callback](#)。

## 期貨

### 委託回報

當期交所收到委託將會回傳回報。在回報中分為四部分，包括operation、order、status及contract。以下我們會在進行詳細的說明。

 委託回報

**version>=1.0    version<1.0**

```

OrderState.FuturesOrder {
 'operation': {
 'op_type': 'New',
 'op_code': '00',
 'op_msg': ''
 },
 'order': {
 'id': 'fcb42a6e',
 'seqno': '585886',
 'ordno': '00',
 'account': {
 'account_type': 'F',
 'person_id': '',
 'broker_id': 'F002000',
 'account_id': '1234567',
 'signed': True
 },
 'action': 'Buy',
 'price': 27000.0,
 'quantity': 1,
 'order_type': 'ROD',
 'price_type': 'LNT',
 'market_type': 'Night',
 'oc_type': 'New',
 'subaccount': '',
 'combo': False
 },
 'status': {
 'id': 'fcb42a6e',
 'exchange_ts': 1764731566.0,
 'modified_price': 0.0,
 'cancel_quantity': 0,
 'order_quantity': 1,
 'web_id': 'Z'
 },
 'contract': {
 'security_type': 'FUT',
 'code': 'TXF',
 'full_code': 'TXFL5',
 'exchange': 'T1M',
 'delivery_month': '202512',
 'delivery_date': '',
 'strike_price': 0.0,
 'option_right': 'Future'
 }
}

OrderState.Order {
 'operation': {
 'op_type': 'New',
 'op_code': '00',
 'op_msg': ''
 },
 'order': {
 'id': 'fcb42a6e',
 'seqno': '585886',
 'ordno': '00',
 'account': {
 'account_type': 'F',
 'person_id': '',
 'broker_id': 'F002000',
 'account_id': '1234567',
 'signed': True
 },
 'action': 'Buy',
 'price': 27000.0,
 'quantity': 1,
 'order_type': 'ROD',
 'price_type': 'LNT',
 'market_type': 'Night',
 'oc_type': 'New',
 'subaccount': '',
 'combo': False
 },
 'status': {
 'id': 'fcb42a6e',
 'exchange_ts': 1764731566.0,
 'modified_price': 0.0,
 'cancel_quantity': 0,
 'order_quantity': 1,
 'web_id': 'Z'
 },
 'contract': {
 'security_type': 'FUT',
 'code': 'TXF',
 'full_code': 'TXFL5',
 'exchange': 'T1M',
 'delivery_month': '202301',
 'delivery_date': '',
 'strike_price': 0.0,
 'option_right': 'Future'
 }
}

```

## 委託回報資訊

### operation

```
op_type (str): {
 "New": 新單,
 "Cancel": 刪單,
 "UpdatePrice": 改價,
 "UpdateQty": 改量
}
op_code (str): {"00": 成功, others: 失敗}
op_msg (str): 錯誤訊息
```

### order

```
id (str): 與成交回報的trade_id相同
seqno (str): 平台單號
ordno (str): 委託單號
account (dict): 帳號資訊
action (str): 買賣別
price (float or int): 委託價
quantity (int): 委託量
order_cond (str): {
 Cash: 現股,
 MarginTrading: 融資,
 ShortSelling: 融券
}
order_type (str): 委託類別 {ROD, IOC, FOK}
price_type (str): {LMT: 限價, MKT: 市價, MKP: 範圍市價}
market_type (str): 市場別 {Day: 日盤, Night: 夜盤}
oc_type(str): {New: 新倉, Cover: 平倉, Auto: 自動}
subaccount(str): 子帳號
combo (bool): 是否為組合單
```

### status

```
id (str): 與成交回報的trade_id相同
exchange_ts (int): 交易所時間
modified_price (float or int): 改價
cancel_quantity (int): 取消數量
order_quantity (int): 委託數量
web_id (str): 下單平台代碼
```

### contract

```
security_type (str): 商品類別
code (str): 商品代碼
full_code (str): 商品代碼(含交割月份)
exchange (str): 交易所
delivery_month (str): 交割月份
delivery_date (str): 交割日期
strike_price (float): 履約價
option_right (str): {Future, OptionCall, OptionPut}
```

### 成交回報

當搓合成功，期交所會傳送成交回報告知。搓合成功包含部分成交以及完全成交，可以從委託回報中的 id 去對應成交回報中的 trade\_id 去確認是否為同一筆委託單。

## 成交回報

version>=1.0      version<1.0

```
OrderState.FuturesDeal {
 'trade_id': '4e6df0f6',
 'seqno': '458545',
 'ordno': 'tA0deX10',
 'exchange_seq': 'j5006396',
 'broker_id': 'F002000',
 'account_id': '1234567',
 'action': 'Sell',
 'code': 'TX1',
 'full_code': 'TX127900L5',
 'price': 25.0,
 'quantity': 1,
 'subaccount': '',
 'security_type': 'OPT',
 'delivery_month': '202512',
 'strike_price': 27900.0,
 'option_right': 'OptionCall',
 'market_type': 'Day',
 'combo': False,
 'ts': 1764685425.0
}

OrderState.FDeal {
 'trade_id': '4e6df0f6',
 'seqno': '458545',
 'ordno': 'tA0deX10',
 'exchange_seq': 'j5006396',
 'broker_id': 'F002000',
 'account_id': '1234567',
 'action': 'Sell',
 'code': 'TX1',
 'full_code': 'TX127900L5',
 'price': 25.0,
 'quantity': 1,
 'subaccount': '',
 'security_type': 'OPT',
 'delivery_month': '202512',
 'strike_price': 27900.0,
 'option_right': 'OptionCall',
 'market_type': 'Day',
 'combo': False,
 'ts': 1764685425.0
}
```

## 成交回報

trade\_id (**str**): 與委託回報id相同  
 seqno (**str**): 平台單號  
 ordno (**str**): 前五碼為同委託回報委託單號，後三碼為同筆委託成交交易序號。  
 exchange\_seq (**str**): 回報序號  
 broker\_id (**str**): 分行代碼  
 account\_id (**str**): 帳號  
 action (**str**): 買賣別  
 code (**str**): 商品代碼  
 full\_code (**str**): 商品代碼(含交割月份)  
 price (**float or int**): 成交價  
 quantity (**int**): 成交量  
 subaccount (**str**): 子帳號  
 security\_type (**str**): 商品類別  
 delivery\_month (**str**): 交割月份  
 strike\_price (**float**): 履約價  
 option\_right (**str**): {Future, OptionCall, OptionPut}  
 market\_type (**str**): {Day, Night}  
 ts (**int**): 成交時間戳

## 注意

交易所回傳訊息優先順序成交回報大於委託回報，所以當委託立即成交可能會先收到成交回報。

## 回報處理

欲處理委託、成交回報，詳細可參見[Callback](#)。

## 5.6 CallBack

### 5.6.1 委託回報

每次您使用 `place_order`、`update_order` 或者 `cancel_order` 時，預設皆會收到來自交易所的委託或成交回報。如果您不想收到任何回報通知，您可以參考[委託回報](#)將其關閉。我們亦提供了處理委託及成交回報的介面。如果您正在建立自己的交易系統，這會非常有幫助。

#### 處理委託及成交回報

您可以使用 `set_order_callback` 來處理委託及成交回報。以下範例顯示，自製的委託回報函數(`order_cb`)將先 `print my_order_callback` 然後才 `print` 委託及成交回報。

#### 設定委託回報函式

```
def order_cb(stat, msg):
 print('my_order_callback')
 print(stat, msg)

api.set_order_callback(order_cb)
```

#### 下單

```
version>=1.0 version<1.0

contract = api.Contracts.Stocks.TSE.TSE2890
order = api.Order(
 price=16,
 quantity=1,
 action=sj.constant.Action.Buy,
 price_type=sj.constant.StockPriceType.LMT,
 order_type=sj.constant.OrderType.ROD,
 order_lot=sj.constant.StockOrderLot.Common,
 custom_field="test",
 account=api.stock_account
)
trade = api.place_order(contract, order)

contract = api.Contracts.Stocks.TSE.TSE2890
order = api.Order(
 price=16,
 quantity=1,
 action=sj.constant.Action.Buy,
 price_type=sj.constant.TFTStockPriceType.LMT,
 order_type=sj.constant.TFTOrderType.ROD,
 order_lot=sj.constant.TFTStockOrderLot.Common,
 custom_field="test",
 account=api.stock_account
)
trade = api.place_order(contract, order)
```

委託回報

 委託回報

**version>=1.0      version<1.0**

```
my_order_callback
OrderState.StockOrder {
 'operation': {
 'op_type': 'New',
 'op_code': '00',
 'op_msg': ''
 },
 'order': {
 'id': '97b63e2f',
 'seqno': '267677',
 'ordno': 'IM394',
 'account': {
 'account_type': 'S',
 'person_id': '',
 'broker_id': '9A95',
 'account_id': '1234567',
 'signed': True
 },
 'action': 'Buy',
 'price': 16.0,
 'quantity': 1,
 'order_type': 'ROD',
 'price_type': 'LNT',
 'order_cond': 'Cash',
 'order_lot': 'Common',
 'custom_field': 'test'
 },
 'status': {
 'id': '97b63e2f',
 'exchange_ts': 1673576134.038,
 'modified_price': 0.0,
 'cancel_quantity': 0,
 'order_quantity': 1,
 'web_id': '137'
 },
 'contract': {
 'security_type': 'STK',
 'exchange': 'TSE',
 'code': '2890',
 'symbol': '',
 'name': '',
 'currency': 'TWD'
 }
}

my_order_callback
OrderState.TFTOrder {
 'operation': {
 'op_type': 'New',
 'op_code': '00',
 'op_msg': ''
 },
 'order': {
 'id': '97b63e2f',
 'seqno': '267677',
 'ordno': 'IM394',
 'account': {
 'account_type': 'S',
 'person_id': '',
 'broker_id': '9A95',
 'account_id': '1234567',
 'signed': True
 },
 'action': 'Buy',
 'price': 16.0,
 'quantity': 1,
 'order_type': 'ROD',
 'price_type': 'LNT',
 'order_cond': 'Cash',
 'order_lot': 'Common',
 'custom_field': 'test'
 },
 'status': {
 'id': '97b63e2f',
 'exchange_ts': 1673576134.038,
 'modified_price': 0.0,
 'cancel_quantity': 0,
 'order_quantity': 1,
 'web_id': '137'
 },
 'contract': {
 'security_type': 'STK',
 'exchange': 'TSE',
 'code': '2890',
 'symbol': '',
 'name': '',
 'currency': 'TWD'
 }
}
```

## 成交回報

成交回報

version>=1.0      version<1.0

```
my_order_callback
OrderState.StockDeal {
 'trade_id': '9c6ae2eb',
 'seqno': '269866',
 'ordno': 'IN497',
 'exchange_seq': '669915',
 'broker_id': '9A95',
 'account_id': '1234567',
 'action': 'Buy',
 'code': '2890',
 'order_cond': 'Cash',
 'order_lot': 'IntradayOdd',
 'price': 267.5,
 'quantity': 3,
 'web_id': '137',
 'custom_field': 'test',
 'ts': 1673577256.354
}

my_order_callback
OrderState.TFTDeal {
 'trade_id': '9c6ae2eb',
 'seqno': '269866',
 'ordno': 'IN497',
 'exchange_seq': '669915',
 'broker_id': '9A95',
 'account_id': '1234567',
 'action': 'Buy',
 'code': '2890',
 'order_cond': 'Cash',
 'order_lot': 'IntradayOdd',
 'price': 267.5,
 'quantity': 3,
 'web_id': '137',
 'custom_field': 'test',
 'ts': 1673577256.354
}
```

## 5.6.2 事件

我們使用solace作為mesh broker。事件可視為你與solace的連接狀態。如果你沒有相關網路經驗，可以略過此部分。不用擔心在不用任何的設定下，我們將重連預設為50次。只需要請你確保你的網絡連接狀態正常。



```
@api.quote.on_event
def event_callback(resp_code: int, event_code: int, info: str, event: str):
 print(f'Event code: {event_code} | Event: {event}'')
```



Event code: 16 | Event: Subscribe or Unsubscribe ok

如同報價callback，你可以利用兩種方式設定事件callback。



```
api.quote.set_event_callback?
```



Signature:  
api.quote.set\_event\_callback(func:Callable[[int, int, str, str], NoneType]) -> None  
Docstring: <no docstring>  
Type: method

事件代碼

| <b>Event Code</b> | <b>Event Code Enumerator</b>                        | <b>Description</b>                                                                                                                                                                                    |
|-------------------|-----------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 0                 | SOLCLIENT_SESSION_EVENT_UP_NOTICE                   | The Session is established.                                                                                                                                                                           |
| 1                 | SOLCLIENT_SESSION_EVENT_DOWN_ERROR                  | The Session was established and then went down.                                                                                                                                                       |
| 2                 | SOLCLIENT_SESSION_EVENT_CONNECT_FAILED_ERROR        | The Session attempted to connect but was unsuccessful.                                                                                                                                                |
| 3                 | SOLCLIENT_SESSION_EVENT_REJECTED_MSG_ERROR          | The appliance rejected a published message.                                                                                                                                                           |
| 4                 | SOLCLIENT_SESSION_EVENT_SUBSCRIPTION_ERROR          | The appliance rejected a subscription (add or remove).                                                                                                                                                |
| 5                 | SOLCLIENT_SESSION_EVENT_RX_MSG_TOO_BIG_ERROR        | The API discarded a received message that exceeded the Session buffer size.                                                                                                                           |
| 6                 | SOLCLIENT_SESSION_EVENT_ACKNOWLEDGEMENT             | The oldest transmitted Persistent/Non-Persistent message that has been acknowledged.                                                                                                                  |
| 7                 | SOLCLIENT_SESSION_EVENT_ASSURED_PUBLISHING_UP       | Deprecated -- see notes in solClient_session_startAssuredPublishing. The AD Handshake (that is, Guaranteed Delivery handshake) has completed; the publisher and Guaranteed messages can be sent.      |
| 8                 | SOLCLIENT_SESSION_EVENT_ASSURED_CONNECT_FAILED      | Deprecated -- see notes in solClient_session_startAssuredPublishing. The appliance rejected the AD Handshake to start Guaranteed publishing. Use SOLCLIENT_SESSION_EVENT_ASSURED_DELIVERY_UP instead. |
| 8                 | SOLCLIENT_SESSION_EVENT_ASSURED_DELIVERY_DOWN       | Guaranteed Delivery publishing is not available. The guaranteed delivery capability on the session has been disabled by some action on the appliance.                                                 |
| 9                 | SOLCLIENT_SESSION_EVENT_TE_UNSUBSCRIBE_ERROR        | The Topic Endpoint unsubscribe command failed.                                                                                                                                                        |
| 9                 | SOLCLIENT_SESSION_EVENT_DTE_UNSUBSCRIBE_ERROR       | Deprecated name; SOLCLIENT_SESSION_EVENT_TE_UNSUBSCRIBE_ERROR is preferred.                                                                                                                           |
| 10                | SOLCLIENT_SESSION_EVENT_TE_UNSUBSCRIBE_OK           | The Topic Endpoint unsubscribe completed.                                                                                                                                                             |
| 10                | SOLCLIENT_SESSION_EVENT_DTE_UNSUBSCRIBE_OK          | Deprecated name; SOLCLIENT_SESSION_EVENT_TE_UNSUBSCRIBE_OK is preferred.                                                                                                                              |
| 11                | SOLCLIENT_SESSION_EVENT_CAN_SEND                    | The send is no longer blocked.                                                                                                                                                                        |
| 12                | SOLCLIENT_SESSION_EVENT_RECONNECTING_NOTICE         | The Session has gone down, and an automatic reconnect attempt is in progress.                                                                                                                         |
| 13                | SOLCLIENT_SESSION_EVENT_RECONNECTED_NOTICE          | The automatic reconnect of the Session was successful; the Session was established again.                                                                                                             |
| 14                | SOLCLIENT_SESSION_EVENT_PROVISION_ERROR             | The endpoint create/delete command failed.                                                                                                                                                            |
| 15                | SOLCLIENT_SESSION_EVENT_PROVISION_OK                | The endpoint create/delete command completed.                                                                                                                                                         |
| 16                | SOLCLIENT_SESSION_EVENT_SUBSCRIPTION_OK             | The subscribe or unsubscribe operation has succeeded.                                                                                                                                                 |
| 17                | SOLCLIENT_SESSION_EVENT_VIRTUAL_ROUTER_NAME_CHANGED | The appliance's Virtual Router Name changed during a reconnect operation. This could render existing queue temporary topics invalid.                                                                  |
| 18                | SOLCLIENT_SESSION_EVENT_MODIFYPROP_OK               | The session property modification completed.                                                                                                                                                          |
| 19                | SOLCLIENT_SESSION_EVENT_MODIFYPROP_FAIL             | The session property modification failed.                                                                                                                                                             |

| <b>Event<br/>Code</b> | <b>Event Code Enumerator</b>                       | <b>Description</b>                                                                                                                                        |
|-----------------------|----------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------|
| 20                    | SOLCLIENT_SESSION_EVENT_REPUBLISH_UNACKED_MESSAGES | After successfully reconnecting a disconnected session, the SDK received an unknown publisher flow name response when reconnecting the GD publisher flow. |

## 5.7 帳務

### 5.7.1 銀行餘額

用於查詢現貨交割帳戶餘額，需要先登入。



```
api.account_balance?
```



Signature:  
api.account\_balance(  
 timeout: int = 5000,  
 cb: Callable[[shioaji.position.AccountBalance], NoneType] = None,  
)  
Docstring: query stock account balance



```
api.account_balance()
```



```
AccountBalance(
 status=<FetchStatus.Fetched: 'Fetched'>,
 acc_balance=100000.0,
 date='2023-01-06 13:30:00.000000',
 errmsg=''
)
```



#### AccountBalance

status (FetchStatus): 資料回傳狀態  
acc\_balance (float): 餘額  
date (str): 查詢日期  
errmsg (str): 錯誤訊息

## 5.7.2 交易額度



查詢時間為交易日 8:30~15:00。

用於查詢證券帳戶交易額度，需要先登入。



```
api.trading_limits?
```



**Signature:**  
`api.trading_limits(  
 account: shioaji.account.Account = None,  
 timeout: int = 5000,  
 cb: Callable[[shioaji.position.TradingLimits], NoneType] = None,  
) -> shioaji.position.TradingLimits`  
**Docstring:** query stock account trading limits



```
api.trading_limits(api.stock_account)
```



```
TradingLimits(
 status=<FetchStatus.Fetched: 'Fetched'>,
 trading_limit=1000000,
 trading_used=0,
 trading_available=1000000,
 margin_limit=0,
 margin_used=0,
 margin_available=0,
 short_limit=0,
 short_used=0,
 short_available=0
)
```



**status (FetchStatus):** 資料回傳狀態  
**trading\_limit (int):** 電子交易總額度  
**trading\_used (int):** 電子交易已用額度  
**trading\_available (int):** 電子交易可用額度  
**margin\_limit (int):** 融資額度上限  
**margin\_used (int):** 融資已用額度  
**margin\_available (int):** 融資可用額度  
**short\_limit (int):** 融券額度上限  
**short\_used (int):** 融券已用額度  
**short\_available (int):** 融券可用額度

### 5.7.3 保證金

用於查詢期貨帳戶的保證金，需先[登入](#)。



```
api.margin?
```



```
Signature:
api.margin(
 account: shioaji.account.Account = None,
 timeout: int = 5000,
 cb: Callable[[shioaji.position.Margin], NoneType] = None,
) -> shioaji.position.Margin
Docstring: query future account of margin
```



```
api.margin(api.futopt_account)
```



```
Margin(
 status=<FetchStatus.Fetched: 'Fetched'>,
 yesterday_balance=6000.0,
 today_balance=6000.0,
 deposit_withdrawal=0.0,
 fee=0.0,
 tax=0.0,
 initial_margin=0.0,
 maintenance_margin=0.0,
 margin_call=0.0,
 risk_indicator=999.0,
 royalty_revenue_expenditure=0.0,
 equity=6000.0,
 equity_amount=6000.0,
 option_openbuy_market_value=0.0,
 option_opensell_market_value=0.0,
 option_open_position=0.0,
 option_settle_profitloss=0.0,
 future_open_position=0.0,
 today_future_open_position=0.0,
 future_settle_profitloss=0.0,
 available_margin=6000.0,
 plus_margin=0.0,
 plus_margin_indicator=0.0,
 security_collateral_amount=0.0,
 order_margin_premium=0.0,
 collateral_amount=0.0
)
```

## Margin

status (FetchStatus): 資料回傳狀態  
yesterday\_balance (float): 前日餘額  
today\_balance (float): 今日餘額  
deposit\_withdrawal (float): 存提  
fee (float): 手續費  
tax (float): 期交稅  
initial\_margin (float): 原始保證金  
maintenance\_margin (float): 維持保證金  
margin\_call (float): 追繳保證金  
risk\_indicator (float): 風險指標  
royalty\_revenue\_expenditure (float): 權利金收入與支出  
equity (float): 權益數  
equity\_amount (float): 權益總值  
option\_openbuy\_market\_value (float): 未沖銷買方選擇權市值  
option\_opensell\_market\_value (float): 未沖銷賣方選擇權市值  
option\_open\_position (float): 參考未平倉選擇權損益  
option\_settle\_profitloss (float): 參考選擇權平倉損益  
future\_open\_position (float): 未沖銷期貨浮動損益  
today\_future\_open\_position (float): 參考當日未沖銷期貨浮動損益  
future\_settle\_profitloss (float): 期貨平倉損益  
available\_margin (float): 可動用(出金)保證金  
plus\_margin (float): 依「加收保證金指標」所加收之保證金  
plus\_margin\_indicator (float): 加收保證金指標  
security\_collateral\_amount (float): 有價證券抵繳總額  
order\_margin\_premium (float): 委託保證金及委託權利金  
collateral\_amount (float): 有價品額

## 5.7.4 未實現損益

用於查詢帳戶未實現損益，需要先[登入](#)。

### 未實現損益



```
api.list_positions?
```



**Signature:**  
`api.list_positions(  
 account: shioaji.account.Account = None,  
 unit: shioaji.constant.Unit = <Unit.Common: 'Common'>,  
 timeout: int = 5000,  
 cb: Callable[[List[Union[shioaji.position.StockPosition, shioaji.position.FuturePosition]]], NoneType] = None,  
) -> List[Union[shioaji.position.StockPosition, shioaji.position.FuturePosition]]`

**Docstring:**  
`query account of unrealized gain or loss`

**Args:**

- `account (:obj:`Account`):`  
`choice the account from listing account (Default: stock account)`

證券

整股部位



```
api.list_positions(api.stock_account)
```



```
[
 StockPosition(
 id=0,
 code='2890',
 direction=<Action.Buy: 'Buy'>,
 quantity=12,
 price=2.79,
 last_price=16.95,
 pnl=169171.0,
 yd_quantity=12,
 margin_purchase_amount=0,
 collateral=0,
 short_sale_margin=0,
 interest=0
)
)
```

轉成DataFrame



```
positions = api.list_positions(api.stock_account)
df = pd.DataFrame(s._dict_ for s in positions)
df
```



Out

| <b>id</b> | <b>code</b> | <b>direction</b> | <b>quantity</b> | <b>price</b> | <b>last_price</b> | <b>pnl</b> | <b>yd_quantity</b> | <b>cond</b> | <b>margin</b> |
|-----------|-------------|------------------|-----------------|--------------|-------------------|------------|--------------------|-------------|---------------|
| 0         | 0           | 2890             | Buy             | 12           | 2.79              | 16.95      | 169172             | 12          |               |



### StockPosition

```

id (int): 部位代碼
code (str): 商品代碼
direction (Action): {Buy: 買, Sell: 賣}
quantity (int): 數量
price (float): 平均價格
last_price (float): 目前股價
pnl (float): 損益
yd_quantity (int): 昨日庫存數量
cond (StockOrderCond): {
 Cash: 現股/預設值,
 Netting: 餘額交割,
 MarginTrading: 融資,
 ShortSelling: 融券,
 Emerging: 興櫃
}
margin_purchase_amount (int): 融資金額
collateral (int): 擔保品
short_sale_margin (int): 保證金
interest (int): 利息

```

零股部位

單位為股數



```

api.list_positions(
 api.stock_account,
 unit=sj.constant.Unit.Share
)

```



Out

```

[
 StockPosition(
 id=0,
 code='2890',
 direction=Action.Buy: 'Buy',
 quantity=10000,
 price=10.1,
 last_price=12.0,
 pnl=1234.0,
 yd_quantity=10000,
 margin_purchase_amount=0,
 collateral=0,
 short_sale_margin=0,
 interest=0
)
]

```

期貨選擇權

account 預設為證券帳號，若欲查詢期權內容需帶入期權帳號。



```

api.list_positions(api.futopt_account)

```



```
[
 FuturePosition(
 id=0,
 code='TX201370J2',
 direction=<Action.Buy: 'Buy'>,
 quantity=3,
 price=131.0000,
 last_price=126.0,
 pnl=-750.0
)
]
```

轉成DataFrame



```
positions = api.list_positions(api.futopt_account)
df = pd.DataFrame(p._dict_ for p in positions)
df
```



| <b>id</b> | <b>code</b> | <b>direction</b> | <b>quantity</b> | <b>price</b> | <b>last_price</b> | <b>pnl</b> |
|-----------|-------------|------------------|-----------------|--------------|-------------------|------------|
| 0         | TXFA3       | Buy              | 4               | 14181        | 14375             | 155200     |



### FuturePosition

**id** (**int**): 部位代碼  
**code** (**str**): 商品代碼  
**direction** (**Action**): {Buy: 買, Sell: 賣}  
**quantity** (**int**): 數量  
**price** (**float**): 平均價格  
**last\_price** (**float**): 目前價格  
**pnl** (**float**): 損益

## 未實現損益 - 明細

可從針對 `list_positions` 得到的結果，將 `id` 帶入 `detail_id` 查詢該筆明細。

證券



```
api.list_position_detail?
```



**Signature:**  
`api.list_position_detail(  
 account: shioaji.account.Account = None,  
 detail_id: int = 0,  
 timeout: int = 5000,  
 cb: Callable[[List[Union[shioaji.position.StockPositionDetail, shioaji.position.FuturePositionDetail]]], NoneType] = None,  
) -> List[Union[shioaji.position.StockPositionDetail, shioaji.position.FuturePositionDetail]]`  
**Docstring:**  
`query account of position detail`

**Args:**

- `account` (:obj:`Account`):  
`choose the account from listing account (Default: stock account)`
- `detail_id` (**int**): the `id` is from `Position object`, Position is from `list_positions`



```
position_detail = api.list_position_detail(api.stock_account, 1)
position_detail
```



```
[StockPositionDetail(
 date='2023-02-22',
 code='3558',
 quantity=0,
 price=1461.0,
 last_price=1470.0,
 dseq='WA371',
 direction=<Action.Buy: 'Buy'>,
 pnl=9.0,
 currency=<Currency.TWD: 'TWD'>,
 fee=1.0
)
]
```

轉成DataFrame



```
df = pd.DataFrame(pnl.__dict__ for pnl in position_detail)
df
```



| date       | code | quantity | price  | last_price | direction  | pnl  | currency     | fee |
|------------|------|----------|--------|------------|------------|------|--------------|-----|
| 2023-02-22 | 3558 | 0        | 1461.0 | WA371      | Action.Buy | 11.0 | Currency.TWD | 1.0 |



date (str): 交易日期  
code (str): 商品代碼  
quantity (int): 數量  
price (float): 付出成本  
last\_price (float): 現值  
dseq (str): 委託書號  
direction (Action): (Buy: 買, Sell: 賣)  
pnl (decimal): 損益  
currency (string): 幣別 {NTD, USD, HKD, EUR, CAD, BAS}  
fee (decimal): 交易手續費  
cond (StockOrderCond): {  
 Cash: 現股(預設值),  
 Netting: 餘額交割,  
 MarginTrading: 融資,  
 ShortSelling: 融券,  
 Emerging: 興櫃  
}  
ex\_dividends(int): 除息金額  
interest (int): 除息  
margintrading\_amt(int): 融資金額  
collateral (int): 擔保品

期貨選擇權



```
position_detail = api.list_position_detail(api.futopt_account, 0)
position_detail
```

Cut

```
[
 FuturePositionDetail(
 date='2023-02-14',
 code='MXFC3',
 quantity=1,
 price=15611.0,
 last_price=15541.0,
 dseq='tA0n8',
 direction=Action.Buy: 'Buy'>,
 pnl=-3500.0,
 currency<Currency.TWD: 'TWD'>,
 entry_quantity=1
)
]
```

轉成DataFrame

```
df = pd.DataFrame(pnl.__dict__ for pnl in position_detail)
df
```

Cut

| date       | code  | quantity | price   | last_price | dseq  | direction  | pnl     | curr      |
|------------|-------|----------|---------|------------|-------|------------|---------|-----------|
| 2023-02-14 | MXFC3 | 1        | 15611.0 | 15541.0    | tA0n8 | Action.Buy | -3500.0 | Currency. |

屬性

code (**str**): 商品代碼  
 date (**str**): 交易日期  
 quantity (**int**): 數量  
 price (**float**): 價格  
 last\_price (**float**): 目前股價  
 dseq (**str**): 委託書號  
 direction (**Action**): {Buy: 買, Sell: 賣}  
 pnl (**float**): 損益  
 currency (**str**): 幣別 {NTD, USD, HKD, EUR, CAD, BAS}  
 fee (**float or int**): 交易手續費  
 entry\_quantity(**int**): 新倉數量

## 5.7.5 已實現損益

需要先[登錄](#)。

### 已實現損益



```
api.list_profit_loss?
```



**Signature:**

```
api.list_profit_loss(
 account: shioaji.account.Account = None,
 begin_date: str = '',
 end_date: str = '',
 unit: shioaji.constant.Unit = <Unit.Common: 'Common'>,
 timeout: int = 5000,
 cb: Callable[[List[shioaji.position.ProfitLoss]], NoneType] = None,
) -> List[shioaji.position.ProfitLoss]
```

**Docstring:**

```
query account of profit loss
```

**Args:**

- account (:obj:`Account`): choice the account from listing account (Default: stock account)
- begin\_date (**str**): the start date of query profit loss (Default: today)
- end\_date (**str**): the end date of query profit loss (Default: today)

帶入想查詢的時間區間。begin\_date 為起始時間，end\_date 為結束時間。unit 為數量單位，Common 為整股，Share 為零股。



```
profitloss = api.list_profit_loss(api.stock_account, '2020-05-05', '2020-05-30')
profitloss
```



```
[StockProfitLoss(
 id=0,
 code='2890',
 seqno='14816',
 dseq='ID111',
 quantity=1,
 price=10.1,
 pnl=1234.0,
 pr_ratio=0.1237,
 cond='Cash',
 date='2020-05-22'
)]
```

轉成DataFrame



```
df = pd.DataFrame(pnl._dict_ for pnl in profitloss)
df
```

## Out

| <b>id</b> | <b>code</b> | <b>cond</b>         | <b>date</b> | <b>pnl</b> | <b>pr_ratio</b> | <b>price</b> | <b>quantity</b> | <b>seqno</b> |
|-----------|-------------|---------------------|-------------|------------|-----------------|--------------|-----------------|--------------|
| 0         | 2890        | StockOrderCond.Cash | 2020-05-22  | 1000.0     | 0.1237          | 10.1         | 1               | 14816        |

## StockProfitLoss

```

id (int): 可利用此id查詢明細
code (str): 商品代碼
seqno (str): seqno no.
dseq (str): seqno no.
quantity (int): 數量
price (float): 價格
pnl (float): 損益
pr_ratio (float): 損益比
cond (StockOrderCond): {
 Cash: 現股(預設值),
 Netting: 餘額交割,
 MarginTrading: 融資,
 ShortSelling: 融券,
 Emerging: 興櫃
}
date (str): 交易日期

```

## FutureProfitLoss

```

id (int): 可利用此id查詢明細
code (str): 商品代碼
quantity (int): 數量
pnl (float): 損益
date (str): 交易日期
direction (Action): 買賣別 {Buy, Sell}
entry_price (int): 進倉價格
cover_price (int): 平倉價格
tax (int): 交易稅
fee (int): 交易手續費

```

## 已實現損益 - 明細

可從針對 `list_profit_loss` 得到的結果，將 `id` 帶入 `detail_id` 查詢該筆明細。`unit` 為數量單位，`Common` 為整股，`Share` 為零股。

## api.list\_profit\_loss\_detail?

```

Signature:
api.list_profit_loss_detail(
 account: shioaji.account.Account = None,
 detail_id: int = 0,
 unit: shioaji.constant.Unit = <Unit.Common: 'Common'>,
 timeout: int = 5000,
 cb: Callable[[List[Union[shioaji.position.StockProfitDetail, shioaji.position.FutureProfitDetail]]], NoneType] = None,
) -> List[Union[shioaji.position.StockProfitDetail, shioaji.position.FutureProfitDetail]]
Docstring:
query account of profit loss detail

Args:
 account (:obj:Account):
 choose the account from listing account (Default: stock account)
 detail_id (int): the id is from ProfitLoss object, ProfitLoss is from list_profit_loss

```



```
profitloss_detail = api.list_profit_loss_detail(api.stock_account, 2)
profitloss_detail
```



```
[StockProfitDetail(
 date='2020-01-01',
 code='2890',
 quantity=1,
 dseq='IX000',
 fee=20,
 tax=0,
 currency='TWD',
 price=10.8,
 cost=10820,
 rep_margintrading_amt=0,
 rep_collateral=0,
 rep_margin=0,
 shortselling_fee=0,
 ex_dividend_amt=0,
 interest=0
)
]
```

轉成DataFrame



```
df = pd.DataFrame(pnl.__dict__ for pnl in profitloss_detail)
df
```



| date       | code | quantity | dseq  | fee | tax | currency | price | cost  | rep_margintrading_amt |
|------------|------|----------|-------|-----|-----|----------|-------|-------|-----------------------|
| 2020-01-01 | 2890 | 1        | IX000 | 20  | 0   | TWD      | 10.8  | 10820 |                       |



### StockProfitDetail

```
date (str): 交易日期
code (str): 商品代碼
quantity (int): 數量
dseq (str): 委託書號
fee (int): 交易手續費
tax (int): 交易稅
currency (str): 幣別 {NTD, USD, HKD, EUR, CAD, BAS}
price (float): 成交單價
cost (int): 付出成本
rep_margintrading_amt (int): 債還融資金額
rep_collateral (int): 債還擔保品
rep_margin (int): 債還保證金
shortselling_fee (int): 融券手續費
ex_dividend_amt: 除息金額
interest (int): 利息
trade_type (TradeType): {Common, DayTrade}
cond (StockOrderCond): {
 Cash: 現股(預設值),
 Netting: 餘額交割,
 MarginTrading: 融資,
 ShortSelling: 融券,
 Emerging: 興櫃
}
```

## FutureProfitDetail

```
date (str): 交易日期
code (str): 商品代碼
quantity (int): 數量
dseq (str): 委託書號
fee (int): 交易手續費
tax (int): 交易稅
currency (str): 幣別 {NTD, USD, HKD, EUR, CAD, BAS}
direction (Action): 買賣別 {Buy, Sell}
entry_date (str): 進倉日期
entry_price (int): 進倉價格
cover_price (int): 平倉價格
pnl (int): 損益
```

### 已實現損益 - 彙總

用於查詢一段時間內的損益彙總。



```
api.list_profit_loss_summary?
```



Signature:  
`api.list_profit_loss_summary(  
 account: shioaji.account.Account = None,  
 begin_date: str = '',  
 end_date: str = '',  
 timeout: int = 5000,  
 cb: Callable[[ProfitLossSummaryTotal], NoneType] = None,  
) -> ProfitLossSummaryTotal`

Docstring:  
`query summary profit loss of a period time`

Args:

`account (:obj:`Account`):`  
`choice the account from listing account (Default: stock account)`

`begin_date (str):` the start date of query profit loss (Default: today)

`end_date (str):` the end date of query profit loss (Default: today)

帶入想查詢的時間區間。begin\_date 為起始時間，end\_date 為結束時間。



```
profitloss_summary = api.list_profit_loss_summary(api.stock_account, '2020-05-05', '2020-05-30')
profitloss_summary
```



```
ProfitLossSummaryTotal(
 status=<FetchStatus.Fetched>,
 profitloss_summary:[
 StockProfitLossSummary(
 code='2890',
 quantity=2000,
 entry_price=17,
 cover_price=10,
 pnl=-11585.0,
 currency='NTD',
 entry_cost=34550,
 cover_cost=21600,
 buy_cost=33112,
 sell_cost=21527,
 pr_ratio=-34.99
)
],
 total=ProfitLossTotal(
 quantity=2000,
 buy_cost=33112,
 sell_cost=21527,
 pnl=-11585.0,
 pr_ratio=-34.99
)
)
```

轉成DataFrame



```
df = pd.DataFrame(pnl.__dict__ for pnl in profitloss_summary.profitloss_summary)
df
```



| code | quantity | entry_price | cover_price | pnl    | currency | entry_cost | cover_cost |
|------|----------|-------------|-------------|--------|----------|------------|------------|
| 2890 | 2000     | 17          | 10          | -11585 | NTD      | 34550      | 21600      |



### StockProfitLossSummary

```
code (str): 商品代碼
quantity (int): 數量
entry_price (int): 進倉價格
cover_price (int): 平倉價格
pnl (float): 損益
currency (str): 幣別
entry_cost (int): 進倉金額(不含手續費及交易稅)
cover_cost (int): 平倉金額(不含手續費及交易稅)
buy_cost (int): 付出成本
sell_cost (int): 賣出收入
pr_ratio (float): 損益比
cond (StockOrderCond): {
 Cash: 現股(預設值),
 Netting: 餘額交割,
 MarginTrading: 融資,
 ShortSelling: 融券,
 Emerging: 興櫃
}
```



### FutureProfitLossSummary

```
code (str): 商品代碼
quantity (int): 數量
entry_price (int): 進倉價格
cover_price (int): 平倉價格
pnl (float): 損益
currency (str): 幣別
direction (Action): 買賣別 {Buy, Sell}
tax (int): 交易稅
fee (int): 交易手續費
```

## 5.7.6 結算

用於查詢交割款，需要先[登錄](#)。

### Settlements



```
api.settlements?
```



Signature:  
`api.settlements(  
 account: shioaji.account.Account = None,  
 timeout: int = 5000,  
 cb: Callable[[List[shioaji.position.SettlementV1]], NoneType] = None,  
) -> List[shioaji.position.SettlementV1]`  
 Docstring: query stock account of settlements



```
settlements = api.settlements(api.stock_account)
settlements
```



```
[
 SettlementV1(date=datetime.date(2022, 10, 13), amount=0.0, T=0),
 SettlementV1(date=datetime.date(2022, 10, 14), amount=0.0, T=1),
 SettlementV1(date=datetime.date(2022, 10, 17), amount=0.0, T=2)
,]
```

轉成DataFrame



```
df = pd.DataFrame([s.__dict__ for s in settlements]).set_index("T")
df
```



| T | date       | amount |
|---|------------|--------|
| 0 | 2022-10-13 | 0      |
| 1 | 2022-10-14 | 0      |
| 2 | 2022-10-17 | 0      |

### SettlementV1

```
date (datetime.date): 交割日期
amount (float): 交割金額
T (int): Tday
```

## 5.8 模擬模式

使用者能先在模擬環境熟悉我們所提供的服務，可避免在正式環境操作失誤造成財物的損失。以下會詳細說明在測試環境所提供的功能。

### 模擬環境

```
import shioaji as sj
api = sj.Shioaji(simulation=True)
```

### 5.8.1 可使用的APIs

#### 行情資料

1. quote.subscribe
2. quote.unsubscribe
3. ticks
4. kbars
5. snapshots
6. short\_stock\_sources
7. credit\_enquires
8. scanners

#### 下單

1. place\_order
2. update\_order
3. cancel\_order
4. update\_status
5. list\_trades

#### 帳務

1. list\_positions
2. list\_profit\_loss

## 5.9 進階指南

### 5.9.1 綁訂報價模式

Shioaji 提供綁訂報價模式，可以用來將報價儲存於訊息佇列，將報價推送至Redis Stream，或者實現觸價委託單。我們提供以下範例，讓您可以更了解綁訂報價模式如何運作。

#### 範例

綁訂報價至訊息佇列

#### pythonic way by using decorator

```
from collections import defaultdict, deque
from shioaji import TickFOPv1, Exchange

set context
msg_queue = defaultdict(deque)
api.set_context(msg_queue)

In order to use context, set bind=True
@api.on_tick_fop_v1(bind=True)
def quote_callback(self, exchange:Exchange, tick:TickFOPv1):
 # append quote to message queue
 self[tick.code].append(tick)

subscribe
api.quote.subscribe(
 api.Contracts.Futures.TXF['TXF202107'],
 quote_type = sj.constant.QuoteType.Tick,
 version = sj.constant.QuoteVersion.v1
)
```

#### traditional way

```
def quote_callback(self, exchange:Exchange, tick:TickFOPv1):
 # append tick to context
 self[tick.code].append(tick)

In order to use context, set bind=True
api.quote.set_on_tick_fop_v1_callback(quote_callback, bind=True)
```

#### Cut

```
after subscribe and wait for a few seconds ...
print(msg_queue)
defaultdict(collections.deque,
{
 'TXFG1': [
 Tick(code='TXFG1', datetime=datetime.datetime(2021, 7, 5, 10, 0, 21, 220000), open=Decimal('17755'), underlying_price=Decimal('17851.88'), bid_side_total_vol=34824,
 ask_side_total_vol=36212, avg_price=Decimal('17837.053112'), close=Decimal('17833'), high=Decimal('17900'), low=Decimal('17742'), amount=Decimal('17833'), total_amount=Decimal('981323314'),
 volume=1, total_volume=55016, tick_type=1, chg_type=2, price_chg=Decimal('184'), pct_chg=Decimal('1.042552'), simtrade=0),
 Tick(code='TXFG1', datetime=datetime.datetime(2021, 7, 5, 10, 0, 21, 781000), open=Decimal('17755'), underlying_price=Decimal('17851.88'), bid_side_total_vol=34825,
 ask_side_total_vol=36213, avg_price=Decimal('17837.053056'), close=Decimal('17834'), high=Decimal('17900'), low=Decimal('17742'), amount=Decimal('17834'), total_amount=Decimal('981341148'),
 volume=1, total_volume=55017, tick_type=1, chg_type=2, price_chg=Decimal('185'), pct_chg=Decimal('1.048218'), simtrade=0)
]
})
```

將報價推送至REDIS STREAM

在開始之前，請先安裝redis。



```

import redis
import json
from shioaji import TickFOPv1, Exchange

redis setting
r = redis.Redis(host='localhost', port=6379, db=0, decode_responses=True)

set up context
api.set_context(r)

In order to use context, set bind=True
@api.on_tick_fop_v1(bind=True)
def quote_callback(self, exchange:Exchange, tick:TickFOPv1):
 # push them to redis stream
 channel = 'Q:' + tick.code # ='Q:TXFG1' in this example
 self.xadd(channel, {'tick':json.dumps(tick.to_dict(raw=True))})

```



```

after subscribe and wait for a few seconds ...
r.xread({'Q:TXFG1':'0-0'})
[
 ['Q:TXFG1',
 [
 ('1625454940107-0',
 {'tick':
 {'code': 'TXFG1', 'datetime': '2021-07-05T11:15:49.066000', 'open': '17755', 'underlying_price': '17904.03', 'bid_side_total_vol': 49698, 'ask_side_total_vol': 51490,
 'avg_price': '17851.312322', 'close': '17889', 'high': '17918', 'low': '17742', 'amount': '268335', 'total_amount': '1399310819', 'volume': 15, 'total_volume': 78387, 'tick_type': 2, 'chg_type': 2, 'price_chg': '240', 'pct_chg': '1.35985', 'simtrade': 0}
 }
),
 ('1625454941854-0',
 {'tick':
 {'code': 'TXFG1', 'datetime': '2021-07-05T11:15:50.815000', 'open': '17755', 'underlying_price': '17902.58', 'bid_side_total_vol': 49702, 'ask_side_total_vol': 51478,
 'avg_price': '17851.313258', 'close': '17888', 'high': '17918', 'low': '17742', 'amount': '35776', 'total_amount': '1399346595', 'volume': 2, 'total_volume': 78389, 'tick_type': 2, 'chg_type': 2, 'price_chg': '239', 'pct_chg': '1.354184', 'simtrade': 0}
 }
)
]
]
]

parse redis stream
[json.loads(x[-1]['tick']) for x in r.xread({'Q:TXFG1':'0-0'})[0][-1]]
[
 {
 'code': 'TXFG1',
 'datetime': '2021-07-05T11:15:49.066000',
 'open': '17755',
 'underlying_price': '17904.03',
 'bid_side_total_vol': 49698,
 'ask_side_total_vol': 51490,
 'avg_price': '17851.312322',
 'close': '17889',
 'high': '17918',
 'low': '17742',
 'amount': '268335',
 'total_amount': '1399310819',
 'volume': 15,
 'total_volume': 78387,
 'tick_type': 2,
 'chg_type': 2,
 'price_chg': '240',
 'pct_chg': '1.35985',
 'simtrade': 0
 },
 {
 'code': 'TXFG1',
 'datetime': '2021-07-05T11:15:50.815000',
 'open': '17755',
 'underlying_price': '17902.58',
 'bid_side_total_vol': 49702,
 'ask_side_total_vol': 51478,
 'avg_price': '17851.313258',
 'close': '17888',
 'high': '17918',
 'low': '17742',
 'amount': '35776',
 'total_amount': '1399346595',
 'volume': 2,
 'total_volume': 78389,
 'tick_type': 2,
 'chg_type': 2,
 'price_chg': '239',
 'pct_chg': '1.354184',
 'simtrade': 0
 }
]

```

觸價委託單

觸價委託單，在市場價格觸及委託單上所設定之價位時，委託單立刻轉為限價單或市價單。

以下僅為範例，請小心使用並自行承擔風險

### Example: stop order

```

import time
from typing import Union

import shioaji as sj

class StopOrderExecutor:
 def __init__(self, api: sj.Shioaji) -> None:
 self.api = api
 self._stop_orders = {}

 def on_quote(
 self, quote: Union[sj.BidAskFOPv1, sj.BidAskSTKv1, sj.TickFOPv1, sj.TickSTKv1]
) -> None:
 code = quote.code
 if code in self._stop_orders:
 for stop_order in self._stop_orders[code]:
 if stop_order['executed']:
 continue
 if hasattr(quote, "ask_price"):
 price = 0.5 * float(
 quote.bid_price[0] + quote.ask_price[0]
) # mid price
 else:
 price = float(quote.close) # Tick
 is_execute = False
 if stop_order["stop_price"] >= stop_order["ref_price"]:
 if price >= stop_order["stop_price"]:
 is_execute = True
 elif stop_order["stop_price"] < stop_order["ref_price"]:
 if price <= stop_order["stop_price"]:
 is_execute = True
 if is_execute:
 self.api.place_order(stop_order["contract"], stop_order["pending_order"])
 stop_order['executed'] = True
 stop_order['ts_executed'] = time.time()
 print(f"execute stop order: {stop_order}")
 else:
 self._stop_orders[code]

 def add_stop_order(
 self,
 contract: sj.contracts.Contract,
 stop_price: float,
 order: sj.order.Order,
) -> None:
 code = contract.code
 snap = self.api.snapshots([contract])[0]
 # use mid price as current price to avoid illiquidity
 ref_price = 0.5 * (snap.buy_price + snap.sell_price)
 stop_order = {
 "code": contract.code,
 "stop_price": stop_price,
 "ref_price": ref_price,
 "contract": contract,
 "pending_order": order,
 "ts_create": time.time(),
 "executed": False,
 "ts_executed": 0.0
 }
 if code not in self._stop_orders:
 self._stop_orders[code] = []
 self._stop_orders[code].append(stop_order)
 print(f"add stop order: {stop_order}")

 def get_stop_orders(self) -> dict:
 return self._stop_orders

 def cancel_stop_order_by_code(self, code: str) -> None:
 if code in self._stop_orders:
 _ = self._stop_orders.pop(code)

 def cancel_stop_order(self, stop_order: dict) -> None:
 code = stop_order["code"]
 if code in self._stop_orders:
 self._stop_orders[code].remove(stop_order)
 if len(self._stop_orders[code]) == 0:
 self._stop_orders.pop(code)

 def cancel_all_stop_orders(self) -> None:
 self._stop_orders.clear()

```

- 使用snapshots的中價作為參考價格，以區分觸價委託單的方向。

基本上，委託單會在您的電腦上待命，只有在商品價格觸擊所設定價格時，觸價委託單才會被送出，以下範例顯示如何提交限價觸價委託單(Stop-Limit Order)。

### Set up a stop order

```
shioaji order
contract = api.Contracts.Futures.TXF['TXF202301']
order = api.Order(
 action='Buy',
 price=14800,
 quantity=1,
 price_type='LMT',
 order_type='ROD',
 octype=sj.constant.FuturesOCType.Auto,
 account=api.futopt_account
)

Stop Order Executor
soe = StopOrderExecutor(api)
soe.add_stop_order(contract=contract, stop_price=14805, order=order)
```

### Cut

```
add stop order: {
 'code': 'TXFA3',
 'stop_price': 14805,
 'ref_price': 14790,
 'contract': Future(
 code='TXFA3',
 symbol='TXF202301',
 name='臺股期貨01',
 category='TXF',
 delivery_month='202301',
 delivery_date='2023/01/30',
 underlying_kind='I',
 unit=1,
 limit_up=16241.0,
 limit_down=13289.0,
 reference=14765.0,
 update_date='2023/01/10'
),
 'pending_order': Order(
 action=<Action.Buy: 'Buy'>,
 price=14800,
 quantity=1,
 account=FutureAccount(person_id='A123456789', broker_id='F002000', account_id='1234567', signed=True, username='PAIUSER'),
 price_type=<StockPriceType.LMT: 'LMT'>,
 order_type=<OrderType.ROD: 'ROD'>
),
 'ts_create': 1673329115.1056178,
 'executed': False,
 'ts_executed': 0.0
}
```

- 市價觸價委託單(Stop-Market Order): `price_type = 'MKT'`

最後，我們將 `StopOrderExecutor` 綁訂在報價上。請注意，您必須訂略商品報價，觸價委託單才會執行。

### Set up context and callback function

```
from shioaji import TickFOPv1, Exchange

set up context
api.set_context(soe)

In order to use context, set bind=True
@api.on_tick_fop_v1(bind=True)
def quote_callback(self, exchange:Exchange, tick:TickFOPv1):
 # pass tick object to Stop Order Executor
 self.on_quote(tick)

subscribe
api.quote.subscribe(
 contract,
 quote_type = sj.constant.QuoteType.Tick,
 version = sj.constant.QuoteVersion.v1
)
```

 **Out: Once close/mid price hit stop price**

```
execute stop order: {
 'code': 'TXFA3',
 'stop_price': 14805,
 'ref_price': 14790,
 'contract': Future(
 code='TXFA3',
 symbol='TXF202301',
 name='臺股期貨01',
 category='TXX',
 delivery_month='202301',
 delivery_date='2023/01/30',
 underlying_kind='I',
 unit=1,
 limit_up=16241.0,
 limit_down=13289.0,
 reference=14765.0,
 update_date='2023/01/10'
),
 'pending_order': Order(
 action=<Action.Buy: 'Buy'>,
 price=14800,
 quantity=1,
 account=FutureAccount(person_id='A123456789', broker_id='F002000', account_id='1234567', signed=True, username='PAIUSER'),
 price_type=<StockPriceType.LMT: 'LMT'>,
 order_type=<OrderType.ROD: 'ROD'>
),
 'ts_create': 1673329115.1056178,
 'executed': True,
 'ts_executed': 1673329161.3224185
}
```

## 5.9.2 非阻塞模式範例

阻塞(Blocking)模式為函數必須等待某事完成。每個函數都是等待的，不管是在做 I/O 還是在做 CPU 任務。舉例來說，如果函數試圖從資料庫中獲取數據，那麼它需要停下來等待回傳結果，收到回傳結果後，才會繼續處理接下來的任務。相反地，非阻塞(non-blocking)模式，不會等待操作完成。如果您嘗試在短時間內發送批量操作，則非阻塞模式非常有用。我們提供以下範例讓您更了解之間的區別。

切換阻塞/非阻塞模式為利用參數 `timeout`。將API參數 `timeout` 設置為 0 為非阻塞模式。`timeout` 預設值為 5000 (毫秒)，表示該函數最多等待 5 秒。

非阻塞模式下單

將 `place_order` 函數中設置 `timeout = 0`。



```
contract = api.Contracts.Futures.TXF['TXF202301']
order = api.Order(
 action=sj.constant.Action.Sell,
 price=14000,
 quantity=1,
 price_type=sj.constant.FuturePriceType.LMT,
 order_type=sj.constant.OrderType.ROD,
 octype=sj.constant.FutureOCType.Auto,
 account=api.futopt_account
)
trade = api.place_order(contract, order, timeout=0)
trade
```



```
Trade(
 contract=Future(
 code='TXFA3',
 symbol='TXF202301',
 name='臺股期貨01',
 category='TXF',
 delivery_month='202301',
 delivery_date='2023/01/30',
 underlying_kind='I',
 unit=1,
 limit_up=16241.0,
 limit_down=13289.0,
 reference=14765.0,
 update_date='2023/01/10'
),
 order=Order(
 action=<Action.Sell: 'Sell'>,
 price=14000,
 quantity=1,
 account=FutureAccount(
 person_id='123456789',
 broker_id='F002000',
 account_id='1234567',
 signed=True,
 username='PAPIUSER'
),
 price_type=<StockPriceType.LMT: 'LMT'>,
 order_type=<OrderType.ROD: 'ROD'>
),
 status=OrderStatus(status=<Status.Inactive: 'Inactive'>)
)
```

在非阻塞模式中取得的 `Trade` 物件，因為委託單仍在傳輸中還未送至交易所，所以會缺少一些資訊。在 `Order` 物件中沒有 `id` 和 `seqno`，`OrderStatus` 物件中沒有 `id`、`status_code`、`order_datetime` 和 `deals`，`status` 顯示為 `Inactive`。在非阻塞模式中要取得上述提到的資訊可利用 委託回報 和 非阻塞模式下單回調 兩種方式。

## 委託回報



```
OrderState.FuturesOrder {
 'operation': {
 'op_type': 'New',
 'op_code': '00',
 'op_msg': ''
 },
 'order': {
 'id': 'de616839',
 'seqno': '500009',
 'ordno': '000009',
 'action': 'Sell',
 'price': 14000,
 'quantity': 1,
 'order_type': 'R0D',
 'price_type': 'LNT',
 'oc_type': 'Auto',
 'custom_field': ''
 },
 'status': {
 'id': 'de616839',
 'exchange_ts': 1673334371.492948,
 'order_quantity': 1,
 'modified_price': 0,
 'cancel_quantity': 0,
 'web_id': 'Z'
 },
 'contract': {
 'security_type': 'FUT',
 'exchange': 'TAIFEX',
 'code': 'TXFA3'
 }
}
```

## 非阻塞模式下單回調



```
from shioaji.order import Trade

def non_blocking_cb(trade:Trade):
 print('_my_callback_')
 print(trade)

trade = api.place_order(
 contract,
 order,
 timeout=0,
 cb=non_blocking_cb # only work in non-blocking mode
)
```

### Out: place order callback

```

_my_callback_
contract=Future(
 code='TXFA3',
 symbol='TXF202301',
 name='臺股期貨01',
 category='TXF',
 delivery_month='202301',
 delivery_date='2023/01/30',
 underlying_kind='I',
 unit=1,
 limit_up=16241.0,
 limit_down=13289.0,
 reference=14765.0,
 update_date='2023/01/10'
),
order=Order(
 action=Action.Sell: 'Sell'>,
 price=14000,
 quantity=1,
 id='40fd85d6',
 seqno='958433',
 ordno='ky01g',
 account=FutureAccount(
 person_id='F123456789',
 broker_id='F002000',
 account_id='1234567',
 signed=True,
 username='PAP1USER'
),
 price_type=<StockPriceType.LMT: 'LMT'>,
 order_type=<OrderType.ROD: 'ROD'>
),
status=OrderStatus(
 id='40fd85d6',
 status=<Status.Submitted: 'Submitted'>,
 status_code=' ',
 order_datetime=datetime.datetime(2023, 01, 10, 15, 14, 32),
 deals=[]
)
)

```

#### 比較兩者模式

在非阻塞模式下，執行 `place_order` 大約需要 0.01 秒，這比阻塞模式下的執行時間快 12 倍。雖然非阻塞模式下單效率更高，需等待交易所收到委託後，委託單才會生效。

### contract and order

```

contract = api.Contracts.Futures.TXF['TXF202301']
order = api.Order(
 action='Sell',
 price=14000,
 quantity=1,
 price_type='LMT',
 order_type='ROD',
 octype=sj.constant.FuturesOCType.Auto,
 account=api.futopt_account
)

```

### Blocking

```

start_time = time.time()
api.place_order(contract, order) # block and wait for the order response
print(time.time() - start_time)
0.136578369140625 <- may be different

```

### Non-Blocking

```

start_time = time.time()
api.place_order(contract, order, timeout=0) # non-block, the order is in transmission (inactive).
print(time.time() - start_time)
0.011670351028442383 <- may be different

```

### 支援非等待模式的函數

```
- place_order
- update_order
- cancel_order
- update_status
- list_positions
- list_position_detail
- list_profit_loss
- list_profit_loss_detail
- list_profit_loss_summary
- settlements
- margin
- ticks
- kbars
```

### 5.9.3 觸價委託範例

#### 觸價委託範例

這是一個簡單的範例，說明如何實作價格監控以及觸價委託。

```
from pydantic import BaseModel

class TouchOrderCond(BaseModel):
 contract: Contract
 order: Order
 order: Order
 touch_price: float

class TouchOrder:
 def __init__(self, api: sj.Shioaji, condition: TouchOrderCond):
 self.flag = False
 self.api = api
 self.order = condition.order
 self.contract = condition.contract
 self.touch_price = condition.touch_price
 self.api.quote.subscribe(self.contract)
 self.api.quote.set_quote_callback(self.touch)

 def touch(self, topic, quote):
 price = quote["Close"][0]
 if price == self.touch_price and not self.flag:
 self.flag = True
 self.api.place_order(self.contract, self.order)
 self.api.quote.unsubscribe(self.contract)
```

完整程式碼詳見 [TouchPrice Order Extention](#)

## 5.9.4 行情管理

本篇教學完整專案的程式碼可以參考 [sj-trading](#)，完整使用範例 jupyter notebook 可以參考 [quote\\_manager\\_usage](#)。

本專案是使用 uv 建立的，如果還不熟悉如何使用 uv 建立專案並使用 uv 管理依賴，建議回到 [環境設定](#) 章節從頭學習起。

在開始進行行情管理器的編寫前，我們會使用 Polars 這個套件來處理行情資料，所以需要將它加入專案的依賴中，同時本篇教學中會有如何用 Polars 快速對多商品計算技術指標的範例，所以也需要將 polars\_talib 這個套件加入專案的依這個。

### 新增 Polars 依賴

```
uv add polars polars_talib
```

如果你對 Polars 不熟悉，可以參考 [Polars 官方文件](#) 來了解該如何使用他。

polars\_talib 是一個 Polars 的擴充套件，它提供了 polars expression 版本的 ta-lib 完整功能，讓我們可以很方便的用 Polars 進行技術指標的計算，他是由 shioaji 作者開發的，詳細使用可以參考 [polars\\_ta\\_extension](#)。

Polars 是一個高效的 DataFrame 套件，適合用來處理大量資料，並且不需要任何額外的設定，就可以使用多核心來加速資料處理。這篇範例中我們可以看到如何使用 Shioaji 的行情管理器來取得行情資料，並且使用 Polars 來做並行化運算，同時將商品的 ticks 進行分 K 轉換，並且做平行化的多商品技術指標計算。

### 新增 quote.py

在 src/sj\_trading/ 新增 quote.py 檔案，並且新增以下程式碼

```
import shioaji as sj
from typing import List

class QuoteManager:
 def __init__(self, api: sj.Shioaji):
 self.api = api
 self.api.quote.set_on_tick_stk_v1_callback(self.on_stk_v1_tick_handler)
 self.api.quote.set_on_tick_fop_v1_callback(self.on_fop_v1_tick_handler)
 self.ticks_stk_v1: List[sj.TickSTKv1] = []
 self.ticks_fop_v1: List[sj.TickFOPv1] = []

 def on_stk_v1_tick_handler(self, _exchange: sj.Exchange, tick: sj.TickSTKv1):
 self.ticks_stk_v1.append(tick)

 def on_fop_v1_tick_handler(self, _exchange: sj.Exchange, tick: sj.TickFOPv1):
 self.ticks_fop_v1.append(tick)
```

這個部分比較單純，讓收到行情的 handle func 盡可能地做最少的事，我們定義了一個 QuoteManager 類別，並且在初始化時設定了註冊兩個回調函數，分別是 on\_stk\_v1\_tick\_handler 和 on\_fop\_v1\_tick\_handler，這兩個函數會在接收到行情資料時被呼叫，並且將行情資料存入 ticks\_stk\_v1 和 ticks\_fop\_v1 中。

### 增加 QuoteManager 訂閱與取消訂閱的方法

```
def __init__(self, api: sj.Shioaji):
 # skip
 self.subscribed_stk_tick: Set[str] = set()

def subscribe_stk_tick(self, codes: List[str], recover: bool = False):
 for code in codes:
 contract = self.api.Contracts.Stocks[code]
 if contract is not None and code not in self.subscribed_stk_tick:
 self.api.quote.subscribe(contract, "tick")
 self.subscribed_stk_tick.add(code)

def unsubscribe_stk_tick(self, codes: List[str]):
 for code in codes:
 contract = self.api.Contracts.Stocks[code]
 if contract is not None and code in self.subscribed_stk_tick:
 self.api.quote.unsubscribe(contract, "tick")
 self.subscribed_stk_tick.remove(code)

def unsubscribe_all_stk_tick(self):
 for code in self.subscribed_stk_tick:
 contract = self.api.Contracts.Stocks[code]
 if contract is not None:
 self.api.quote.unsubscribe(contract, "tick")
 self.subscribed_stk_tick.clear()
```

上面我們增加了 `subscribe_stk_tick` 方法，這個方法會將傳入的商品代碼列表中的商品代碼加入到 `subscribed_stk_tick` 中，並且呼叫 Shioaji 的 `subscribe` 方法來訂閱行情，`subscribed_stk_tick` 是一個 `Set`，用來存放已經訂閱的商品代碼，避免重複訂閱以及方便後續將所有訂閱商品取消訂閱。

### 增加 QuoteManager 拿出訂閱的 ticks 的方法

```
def __init__(self, api: sj.Shioaji):
 # skip
 self.df_stk: pl.DataFrame = pl.DataFrame(
 [],
 schema=[
 ("datetime", pl.Datetime),
 ("code", pl.Utf8),
 ("price", pl.Float64),
 ("volume", pl.Int64),
 ("tick_type", pl.Int8),
],
)

def get_df_stk(self) -> pl.DataFrame:
 poped_ticks, self.ticks_stk_v1 = self.ticks_stk_v1, []
 if poped_ticks:
 df = pl.DataFrame([tick.to_dict() for tick in poped_ticks]).select(
 pl.col("datetime", "code"),
 pl.col("close").cast(pl.Float64).alias("price"),
 pl.col("volume").cast(pl.Int64),
 pl.col("tick_type").cast(pl.Int8),
)
 self.df_stk = self.df_stk.vstack(df)
 return self.df_stk
```

`__init__` 中我們定義了一個 `df_stk` 的 Polars DataFrame，用來存放所有訂閱的台股 tick 資料，`get_df_stk` 方法會將 `ticks_stk_v1` 中的資料轉換成 Polars DataFrame，並且回傳，到這邊我們就已經可以初步看到可以拿出來的 DataFrame 了。

`df_stk`

### 增加 QuoteManager 將 ticks 轉換成 K 線的方法

```
def get_df_stk_kbar(
 self, unit: str = "1m", exprs: List[pl.Expr] = []
) -> pl.DataFrame:
 df = self.get_df_stk()
 df = df.groupby(
 pl.col("datetime").dt.truncate(unit),
 pl.col("code"),
 maintain_order=True,
).agg(
 pl.col("price").first().alias("open"),
 pl.col("price").max().alias("high"),
 pl.col("price").min().alias("low"),
 pl.col("price").last().alias("close"),
 pl.col("volume").sum().alias("volume"),
)
 if exprs:
 df = df.with_columns(exprs)
 return df
```

在 `get_df_stk_kbar` 方法中，我們將 `get_df_stk` 拿到 Ticks 的 DataFrame 根據 code 和 truncate 後的 datetime 進行分組，並且對每個分組進行聚合開高低收量，最後回傳一個新的 DataFrame，這個 DataFrame 就是我們所需要的 K 線資料了，並且這邊保留了 `exprs` 參數，讓使用者可以傳入一些額外的運算式，來進行更多的運算。在這邊 `truncate` 的單位我們使用 `1m` 來表示 1 分鐘，如果想要拿到 5 分鐘的 K 線，可以將單位改成 `5m`，1 小時 K 可以將單位改成 `1h`，如果想要更多不同的單位可以參考 `truncate` 的 API 文件。

### 自定義技術指標計算

```
import polars as pl
import polars_ta as pta

quote_manager.get_df_stk_kbar("5m", [
 pl.col("close").ta.ema(5).over("code").fill_nan(None).alias("ema5"),
 pta.macd(pl.col("close"), 12, 26, 9).over("code").struct.field("macd").fill_nan(None),
])
```

在這邊使用 `polars_ta` 的 expression 來計算技術指標，並且將計算出來的指標加入到 K 線資料中，這邊我們計算了 `ema` 和 `macd` 兩種指標，更多指標可以參考 [polars\\_ta\\_extension 支援指標列表](#)。

在這個 `polars_ta` 的 expression 中，使用 `over("code")` 來將指標計算結果根據商品代碼進行分組做每個商品獨立的運算，所以即使所有的商品都在同一個 DataFrame 中，計算出來的結果還是每個商品獨立的，並且這個 `over` 的 partition 是會自動平行運算的，所以即使有大量的商品，也可以很快的計算出來，使用 `alias` 來將計算結果的欄位名稱設置為 `ema5`，在 `macd` 指標中回傳的是多個欄位的 struct，這邊取出 struct 中的 `macd` 欄位。

因為這邊傳入的只是表達式非常輕量，可以根據你需要的任何表達式進行新增就可以看到你需要的各種技術指標了，當然如果你要使用 `polars` expression 做出自己的指標，也是可以的，這邊只是提供一個可以做運算的接口以及簡單的使用範例。

### 回補錯過的行情

```
def fetch_ticks(self, contract: BaseContract) -> pl.DataFrame:
 code = contract.code
 ticks = self.api.ticks(contract)
 df = pl.DataFrame(ticks.dict()).select(
 pl.from_epoch("ts", time_unit="ns").dt.cast_time_unit("us").alias("datetime"),
 pl.lit(code).alias("code"),
 pl.col("close").alias("price"),
 pl.col("volume").cast(pl.Int64),
 pl.col("tick_type").cast(pl.Int8),
)
 return df

def subscribe_stk_tick(self, codes: List[str], recover: bool = False):
 for code in codes:
 # skip
 if recover:
 df = self.fetch_ticks(contract)
 if not df.is_empty():
 code_ticks = [t for t in self.ticks_stk_v1 if t.code == code]
 if code_ticks:
 t_first = code_ticks[0].datetime
 df = df.filter(pl.col("datetime") < t_first)
 self.df_stk = self.df_stk.vstack(df)
 else:
 self.df_stk = self.df_stk.vstack(df)
```

在訂閱的時候我們可能會超過當天開盤的時間，這時候訂閱即時資料將會缺乏錯過的資料，所以這邊我們實作使用 api 回補歷史 tick 的資料，這邊我們使用 `fetch_ticks` 方法來取得歷史 tick 的資料，並且將取得資料加入到 `df_stk` 中。

以上我們已經完成了一個可以訂閱行情、回補錯過行情、計算技術指標的行情管理器了，這邊我們將所有程式碼整合起來，並且在 jupyter lab 中使用。

完整的 QuoteManager 可以參考 [quote.py](#)。

完整使用範例 jupyter notebook 可以參考 [quote\\_manager\\_usage](#)。

## 6. 升版指南

1.0 為主要版本，本文檔幫助用戶遷移到版本 1.0。

### 6.1 Shioaji 物件

移除參數 backend



```
import shioaji as sj
sj.Shioaji?
```

**version>=1.0    version<1.0**

```
Init signature:
sj.Shioaji(
 simulation: bool = False,
 proxies: Dict[str, str] = {},
 currency: str = 'NTD',
)
Docstring:
shioaji api

Functions:
 login
 logout
 activate_ca
 list_accounts
 set_default_account
 get_account_margin
 get_account_openposition
 get_account_settle_profitloss
 get_stock_account_funds
 get_stock_account_unreal_profitloss
 get_stock_account_real_profitloss
 place_order
 update_order
 update_status
 list_trades

Objects:
 Quote
 Contracts
 Order

Init docstring:
initialize Shioaji to start trading

Args:
 simulation (bool):
 - False: to trading on real market (just use your Sinopac account to start trading)
 - True: become simulation account(need to contract as to open simulation account)
 proxies (dict):
 specific the proxies of your https
 ex: {'https': 'your-proxy-url'}
 currency (str):
 {NTX, USX, NTD, USD, HKD, EUR, JPY, GBP}
 set the default currency for display
```

```
Init signature:
sj.Shioaji(
 backend: str = 'http',
 simulation: bool = False,
 proxies: Dict[str, str] = {},
 currency: str = 'NTD',
)
Docstring:
shioaji api

Functions:
 login
 activate_ca
 list_accounts
 set_default_account
 get_account_margin
 get_account_openposition
 get_account_settle_profitloss
 get_stock_account_funds
 get_stock_account_unreal_profitloss
 get_stock_account_real_profitloss
 place_order
 update_order
 update_status
 list_trades
```

```
Objects:
 Quote
 Contracts
 Order

Init docstring:
initialize Shioaji to start trading
```

```
Args:
 backend (str): {http, socket}
 use http or socket as backend currently only support http, async socket backend coming soon.
 simulation (bool):
 - False: to trading on real market (just use your Sinopac account to start trading)
 - True: become simulation account(need to contract as to open simulation account)
 proxies (dict):
 specific the proxies of your https
 ex: {'https': 'your-proxy-url'}
 currency (str):
 {NTX, USX, NTD, USD, HKD, EUR, JPY, GBP}
 set the default currency for display
```

## 6.2 登入

登入參數 person\_id 及 passwd 變更為 api\_key 及 secret\_key。您可以在 [Token](#) 深入了解如何取得 API Key。



**version>=1.0      version<1.0**

```
import shioaji as sj
api = sj.Shioaji()
api.login(
 api_key="YOUR_API_KEY",
 secret_key="YOUR_SECRET_KEY"
)

import shioaji as sj
api = sj.Shioaji()
api.login(
 person_id="YOUR_PERSON_ID",
 passwd="YOUR_PASSWORD",
)
```



```
[FutureAccount(person_id='', broker_id='', account_id='', signed=True, username=''),
 StockAccount(person_id='', broker_id='', account_id='', signed=True, username='')
]
```

## 6.3 證券下單

TFTStockOrder 更改為 StockOrder

## StockOrder

**version>=1.0    version<1.0**

```
>> sj.order.StockOrder?

Init signature:
sj.order.StockOrder(
 *,
 action: shioaji.constant.Action,
 price: Union[pydantic.types.StrictInt, float],
 quantity: shioaji.order.ConstrainedIntValue,
 id: str = '',
 seqno: str = '',
 ordno: str = '',
 account: shioaji.account.Account = None,
 custom_field: shioaji.order.ConstrainedStrValue = '',
 ca: str = '',
 price_type: shioaji.constant.StockPriceType,
 order_type: shioaji.constant.OrderType,
 order_lot: shioaji.constant.StockOrderLot = <StockOrderLot.Common: 'Common'>,
 order_cond: shioaji.constant.StockOrderCond = <StockOrderCond.Cash: 'Cash'>,
 daytrade_short: bool = False,
) -> None

>> sj.order.TFTStockOrder?

Init signature:
sj.order.TFTStockOrder(
 *,
 action: shioaji.constant.Action,
 price: Union[pydantic.types.StrictInt, float],
 quantity: shioaji.order.ConstrainedIntValue,
 id: str = '',
 seqno: str = '',
 ordno: str = '',
 account: shioaji.account.Account = None,
 custom_field: shioaji.order.ConstrainedStrValue = '',
 ca: str = '',
 price_type: shioaji.constant.TFTStockPriceType,
 order_type: shioaji.constant.TFTOrderType,
 order_lot: shioaji.constant.TFTStockOrderLot = <TFTStockOrderLot.Common: 'Common'>,
 order_cond: shioaji.constant.StockOrderCond = <StockOrderCond.Cash: 'Cash'>,
 first_sell: shioaji.constant.StockFirstSell = <StockFirstSell.No: 'false'>,
) -> None
```

### 6.3.1 下單

- TFTStockPriceType 更改為 StockPriceType
- TFTOrderType 更改為 OrderType
- TFTStockOrderLot 更改為 StockOrderLot
- first\_sell 更改為 daytrade\_short，型態更改為 Bool

## Order

**version>=1.0    version<1.0**

```
order = api.Order(
 price=12,
 quantity=1,
 action=sj.constant.Action.Sell,
 price_type=sj.constant.StockPriceType.LMT,
 order_type=sj.constant.OrderType.ROD,
 order_lot=sj.constant.StockOrderLot.Common,
 daytrade_short=True,
 custom_field="test",
 account=api.stock_account
)

order = api.Order(
 price=12,
 quantity=1,
 action=sj.constant.Action.Sell,
 price_type=sj.constant.TFTStockPriceType.LMT,
 order_type=sj.constant.TFTOrderType.ROD,
 order_lot=sj.constant.TFTStockOrderLot.Common,
 first_sell=sj.constant.StockFirstSell.Yes,
 custom_field="test",
 account=api.stock_account
)
```

### 6.3.2 委託回報

TFTOrder 更改為 StockOrder

**Order Callback**

**version>=1.0    version<1.0**

```

OrderState.StockOrder {
 'operation': {
 'op_type': 'New',
 'op_code': '00',
 'op_msg': ''
 },
 'order': {
 'id': 'c21b876d',
 'seqno': '429832',
 'ordno': 'W2892',
 'action': 'Buy',
 'price': 12.0,
 'quantity': 10,
 'order_cond': 'Cash',
 'order_lot': 'Common',
 'custom_field': 'test',
 'order_type': 'ROD',
 'price_type': 'LNT'
 },
 'status': {
 'id': 'c21b876d',
 'exchange_ts': 1583828972,
 'modified_price': 0,
 'cancel_quantity': 0,
 'web_id': '137'
 },
 'contract': {
 'security_type': 'STK',
 'exchange': 'TSE',
 'code': '2890',
 'symbol': '',
 'name': '',
 'currency': 'TWD'
 }
}

OrderState.TFTOrder {
 'operation': {
 'op_type': 'New',
 'op_code': '00',
 'op_msg': ''
 },
 'order': {
 'id': 'c21b876d',
 'seqno': '429832',
 'ordno': 'W2892',
 'action': 'Buy',
 'price': 12.0,
 'quantity': 10,
 'order_cond': 'Cash',
 'order_lot': 'Common',
 'custom_field': 'test',
 'order_type': 'ROD',
 'price_type': 'LNT'
 },
 'status': {
 'id': 'c21b876d',
 'exchange_ts': 1583828972,
 'modified_price': 0,
 'cancel_quantity': 0,
 'web_id': '137'
 },
 'contract': {
 'security_type': 'STK',
 'exchange': 'TSE',
 'code': '2890',
 'symbol': '',
 'name': '',
 'currency': 'TWD'
 }
}

```

### 6.3.3 成交回報

TFTDeal 更改為 StockDeal

## Deal Callback

**version>=1.0      version<1.0**

```
OrderState.StockDeal {
 'trade_id': '12ab3456',
 'exchange_seq': '123456',
 'broker_id': 'your_broker_id',
 'account_id': 'your_account_id',
 'action': <Action.Buy: 'Buy'>,
 'code': '2890',
 'order_cond': <StockOrderCond.Cash: 'Cash'>,
 'order_lot': <TFTStockOrderLot.Common: 'Common'>,
 'price': 12,
 'quantity': 10,
 'web_id': '137',
 'custom_field': 'test',
 'ts': 1583828972
}

OrderState.TFTDeal {
 'trade_id': '12ab3456',
 'exchange_seq': '123456',
 'broker_id': 'your_broker_id',
 'account_id': 'your_account_id',
 'action': <Action.Buy: 'Buy'>,
 'code': '2890',
 'order_cond': <StockOrderCond.Cash: 'Cash'>,
 'order_lot': <TFTStockOrderLot.Common: 'Common'>,
 'price': 12,
 'quantity': 10,
 'web_id': '137',
 'custom_field': 'test',
 'ts': 1583828972
}
```

## 6.4 期貨下單

### FuturesOrder

**verion>=1.0      verion<1.0**

```
>> sj.order.FuturesOrder?

Init signature:
sj.order.FuturesOrder(
 ,
 action: shioaji.constant.Action,
 price: Union[pydantic.types.StrictInt, float],
 quantity: shioaji.order.ConstrainedIntValue,
 id: str = '',
 seqno: str = '',
 ordno: str = '',
 account: shioaji.account.Account = None,
 custom_field: shioaji.order.ConstrainedStrValue = '',
 ca: str = '',
 price_type: shioaji.constant.FuturesPriceType,
 order_type: shioaji.constant.OrderType,
 octype: shioaji.constant.FuturesOCType = <FuturesOCType.Auto: 'Auto'>,
) -> None

>> sj.order.FuturesOrder?

Init signature:
sj.order.FuturesOrder(
 ,
 action: shioaji.constant.Action,
 price: Union[pydantic.types.StrictInt, float],
 quantity: shioaji.order.ConstrainedIntValue,
 id: str = '',
 seqno: str = '',
 ordno: str = '',
 account: shioaji.account.Account = None,
 custom_field: shioaji.order.ConstrainedStrValue = '',
 ca: str = '',
 price_type: shioaji.constant.FuturesPriceType,
 order_type: shioaji.constant.FuturesOrderType,
 octype: shioaji.constant.FuturesOCType = <FuturesOCType.Auto: 'Auto'>,
) -> None
```

### 6.4.1 下單

FuturesOrderType 更改為 OrderType

```
Order
verion>=1.0 verion<1.0

order = api.Order(
 action=sj.constant.Action.Buy,
 price=100,
 quantity=1,
 price_type=sj.constant.FuturesPriceType.LMT,
 order_type=sj.constant.OrderType.ROD,
 octype=sj.constant.FuturesOCType.Auto,
 account=api.futopt_account
)

order = api.Order(
 action=sj.constant.Action.Buy,
 price=100,
 quantity=1,
 price_type=sj.constant.FuturesPriceType.LMT,
 order_type=sj.constant.FuturesOrderType.ROD,
 octype=sj.constant.FuturesOCType.Auto,
 account=api.futopt_account
)
```

### 6.4.2 委託回報

FOrder 更改為 FuturesOrder

## Order Event

**version>=1.0      version<1.0**

```
OrderState.FuturesOrder {
 'operation': {
 'op_type': 'New',
 'op_code': '00',
 'op_msg': ''
 },
 'order': {
 'id': '02c347f7',
 'seqno': '956201',
 'ordno': 'KV00H',
 'action': 'Sell',
 'price': 17760.0,
 'quantity': 1,
 'order_cond': None,
 'order_type': 'ROD',
 'price_type': 'LMT',
 'market_type': 'Night',
 'oc_type': 'New',
 'subaccount': ''
 },
 'status': {
 'id': '02c347f7',
 'exchange_ts': 1625729890,
 'modified_price': 0.0,
 'cancel_quantity': 0,
 "web_id": "P"
 },
 'contract': {
 'security_type': 'FUT',
 'code': 'TXF',
 'exchange': 'TIM',
 'delivery_month': '202107',
 'strike_price': 0.0,
 'option_right': 'Future'
 }
}

OrderState.FOrder {
 'operation': {
 'op_type': 'New',
 'op_code': '00',
 'op_msg': ''
 },
 'order': {
 'id': '02c347f7',
 'seqno': '956201',
 'ordno': 'KV00H',
 'action': 'Sell',
 'price': 17760.0,
 'quantity': 1,
 'order_cond': None,
 'order_type': 'ROD',
 'price_type': 'LMT',
 'market_type': 'Night',
 'oc_type': 'New',
 'subaccount': ''
 },
 'status': {
 'id': '02c347f7',
 'exchange_ts': 1625729890,
 'modified_price': 0.0,
 'cancel_quantity': 0,
 "web_id": "P"
 },
 'contract': {
 'security_type': 'FUT',
 'code': 'TXF',
 'exchange': 'TIM',
 'delivery_month': '202107',
 'strike_price': 0.0,
 'option_right': 'Future'
 }
}
```

## 6.4.3 成交回報

FDeal 更改為 FuturesDeal

### Deal Event

**version>=1.0      version<1.0**

```
OrderState.FuturesDeal {
 "trade_id": "02c347f7",
 "seqno": "956344",
 "ordno": "ky00N110",
 "exchange_seq": "a0000060",
 "broker_id": "F002000",
 "account_id": "9104000",
 "action": "Sell",
 "code": "TXF",
 "price": 17650.0,
 "quantity": 4,
 "subaccount": "",
 "security_type": "FUT",
 "delivery_month": "202107",
 "strike_price": 0.0,
 "option_right": "Future",
 "market_type": "Day",
 "ts": 1625800369
}
```

```
OrderState.FDeal {
 "trade_id": "02c347f7",
 "seqno": "956344",
 "ordno": "ky00N110",
 "exchange_seq": "a0000060",
 "broker_id": "F002000",
 "account_id": "9104000",
 "action": "Sell",
 "code": "TXF",
 "price": 17650.0,
 "quantity": 4,
 "subaccount": "",
 "security_type": "FUT",
 "delivery_month": "202107",
 "strike_price": 0.0,
 "option_right": "Future",
 "market_type": "Day",
 "ts": 1625800369
}
```

## 6.5 行情資料



版本>=1.1將不再提供**QuoteVersion.v0**，請更改至**QuoteVersion.v1**。

### 6.5.1 Callback

#### Tick

### decorator方式

**QuoteVersion.v1      QuoteVersion.v0**

```
from shioaji import TickSTKv1, Exchange

@api.on_tick_stk_v1()
def quote_callback(exchange: Exchange, tick: TickSTKv1):
 print(f"Exchange: {exchange}, Tick: {tick}")

@api.quote.on_quote
def quote_callback(topic: str, quote: dict):
 print(f"Topic: {topic}, Quote: {quote}")
```

### 传统方式

#### QuoteVersion.v1    QuoteVersion.v0

```
from shioaji import TickSTKv1, Exchange

def quote_callback(exchange: Exchange, tick:TickSTKv1):
 print(f"Exchange: {exchange}, Tick: {tick}")

api.quote.set_on_tick_stk_v1_callback(quote_callback)

def quote_callback(topic: str, quote: dict):
 print(f"Topic: {topic}, Quote: {quote}")

api.quote.set_quote_callback(quote_callback)
```

### Cut

#### QuoteVersion.v1    QuoteVersion.v0

```
Exchange: Exchange.TSE, Tick: Tick(code='2330', datetime=datetime.datetime(2021, 7, 2, 13, 16, 35, 92970), open=Decimal('590'), avg_price=Decimal('589.05'), close=Decimal('590'), high=Decimal('593'), low=Decimal('587'), amount=Decimal('590000'), total_amount=Decimal('8540101000'), volume=1, total_volume=14498, tick_type=1, chg_type=4, price_chg=Decimal('-3'), pct_chg=Decimal('-0.505902'), trade_bid_volume=6638, ask_side_total_vol=7860, bid_side_total_cnt=2694, ask_side_total_cnt=2705, closing_ordinal_shares=0, fixed_trade_vol=0, suspend=0, simtrade=0, intraday_odd=0)
```

```
Topic: MKT/*TSE/2330, Quote: {'AmountSum': [4739351000.0], 'Close': [596.0], 'Date': '2021/03/30', 'TickType': [2], 'Time': '10:01:33.349431', 'VolSum': [7932], 'Volume': [1]}
```

### BidAsk

### decorator方式

#### QuoteVersion.v1    QuoteVersion.v0

```
from shioaji import BidAskSTKv1, Exchange

@api.on_bidask_stk_v1()
def quote_callback(exchange: Exchange, bidask:BidAskSTKv1):
 print(f"Exchange: {exchange}, BidAsk: {bidask}")

api.quote.on_quote
def quote_callback(topic: str, quote: dict):
 print(f"Topic: {topic}, Quote: {quote}")

api.quote.set_quote_callback(quote_callback)
```

### 传统方式

#### QuoteVersion.v1    QuoteVersion.v0

```
from shioaji import BidAskSTKv1, Exchange

def quote_callback(exchange: Exchange, bidask:BidAskSTKv1):
 print(f"Exchange: {exchange}, BidAsk: {bidask}")

api.quote.set_on_bidask_stk_v1_callback(quote_callback)

def quote_callback(topic: str, quote: dict):
 print(f"Topic: {topic}, Quote: {quote}")

api.quote.set_quote_callback(quote_callback)
```

```

QuoteVersion.v1 QuoteVersion.v0

Exchange: Exchange.TSE, BidAsk: BidAsk(code='2330', datetime=datetime.datetime(2021, 7, 2, 13, 17, 29, 726428), bid_price=[Decimal('589'), Decimal('588'), Decimal('587'), Decimal('586'), Decimal('585')], bid_volume=[223, 761, 1003, 809, 1274], diff_bid_vol=[0, 0, 0, 0, 0], ask_price=[Decimal('590'), Decimal('591'), Decimal('592'), Decimal('593'), Decimal('594')], ask_volume=[304, 232, 183, 242, 131], diff_ask_vol=[1, 0, 0, 0, 0], suspend=0, simtrade=0, intraday_odd=0)

Topic: QUT/idcdmzpr01/TSE/2330, Quote: {'AskPrice': [590.0, 591.0, 592.0, 593.0, 594.0], 'AskVolume': [303, 232, 183, 242, 131], 'BidPrice': [589.0, 588.0, 587.0, 586.0, 585.0], 'BidVolume': [224, 762, 1003, 809, 1274], 'Date': '2021/07/02', 'Time': '13:17:26.391840'}

```

## 6.6 期貨帳務資訊

移除以下API

1. get\_account\_margin
2. get\_account\_openposition
3. get\_account\_settle\_profitloss

取而代之

1. margin
2. list\_positions( api.futopt\_account )
3. list\_profit\_loss( api.futopt\_account )
4. list\_profit\_loss\_detail( api.futopt\_account )
5. list\_profit\_loss\_summary( api.futopt\_account )

欲瞭解更多期貨帳務API，請參見。

最後在[GITHUB](#)上給我們支持與鼓勵吧

The screenshot shows the GitHub repository page for `Shioaji`. The repository is public and has 6 stars, 5 watchers, and 0 forks. It contains 1 branch, 68 tags, and 83 commits. The repository is described as a "new cross platform api for trading (跨平台證券交易API)". It includes links to `sinotrade.github.io` and various tags like `python`, `docker`, `cross-platform`, `trading`, `trading-api`, `market-data`, `trading-platform`, `trade`, `trading-simulator`, `quote`, `streaming-data`, and `pythonic-api`.

## 7. 問與答

### 7.0.1 下單

#### 如何下市價單(MKT)、範圍市價單(MKP)

```
order = api.Order(
 action=sj.constant.Action.Buy,
 price=0, # MKT, MKP will not use price parameter
 quantity=1,
 price_type='MKP', # change to MKT or MKP
 order_type='IOC', # MKT, MKP only accept IOC order
 octype=sj.constant.FuturesOctype.Auto,
 account=api.futopt_account
)
```

#### 如何掛漲(跌)停限價ROD單

First, we need to know the `limit up(limit down)` price of the security. Just take a look at the `api.Contracts`, you will find the information you want.

#### i

```
api.Contracts.Stocks.TSE['TSE2330']
```

#### Out

```
Stock(
 exchange=<Exchange.TSE: 'TSE'>,
 code='2330',
 symbol='TSE2330',
 name='台積電',
 category='24',
 unit=1000,
 limit_up=653.0,
 limit_down=535.0,
 reference=594.0,
 update_date='2021/08/27',
 margin_trading_balance=6565,
 short_selling_balance=365,
 day_trade=<DayTrade.Yes: 'Yes'>
)
```

Example place LMT and ROD order at limit up price.

#### i

```
contract = api.Contracts.Stocks.TSE['TSE2330']
price = contract.limit_up
order = api.Order(
 action=sj.constant.Action.Buy,
 price=price,
 quantity=1,
 price_type='LMT',
 order_type='ROD',
 order_lot=sj.constant.StockOrderLot.Common,
 account=api.stock_account
)
```

## 7.0.2 行情

### 為什麼行情只能收幾行就斷掉了

If your code something like this, and possibly run code on cmd/terminal with `python stream.py`. Then you definitely won't get any additional ticks, since the python program has already terminated.

**version>=1.0      version<1.0**

```
import shioaji as sj

api = sj.Shioaji(simulation=True)
api.login('YOUR_API_KEY', 'YOUR_SECRET_KEY')
api.quote.subscribe(
 api.Contracts.Stocks["2330"],
 quote_type = sj.constant.QuoteType.Tick
)

stream.py
import shioaji as sj

api = sj.Shioaji(simulation=True)
api.login('YOUR_PERSON_ID', '2222')
api.quote.subscribe(
 api.Contracts.Stocks["2330"],
 quote_type = sj.constant.QuoteType.Tick
)
```

If you wish your python program to survive, please modify you python script as below.

**version>=1.0      version<1.0**

```
stream.py
import shioaji as sj
from threading import Event

api = sj.Shioaji(simulation=True)
api.login('YOUR_API_KEY', 'YOUR_SECRET_KEY')
api.quote.subscribe(
 api.Contracts.Stocks["2330"],
 quote_type = sj.constant.QuoteType.Tick
)

Event().wait()

stream.py
import shioaji as sj
from threading import Event

api = sj.Shioaji(simulation=True)
api.login('YOUR_PERSON_ID', '2222')
api.quote.subscribe(
 api.Contracts.Stocks["2330"],
 quote_type = sj.constant.QuoteType.Tick
)

Event().wait()
```

## 7.0.3 其他

### 出現 Account not acceptable，可能原因如下

- 未完成[簽署]([https://sinotrade.github.io/zh\\_TW/tutor/prepare/terms/#\\_1](https://sinotrade.github.io/zh_TW/tutor/prepare/terms/#_1))及[API測試]([https://sinotrade.github.io/zh\\_TW/tutor/prepare/terms/#api](https://sinotrade.github.io/zh_TW/tutor/prepare/terms/#api))。
- ['update\_status'](./tutor/order/UpdateStatus)預設查詢為名下所有帳號，若想使用預設查詢方式，請確認所有帳號皆有完成簽署及測試。

### 如何更改shioaji.log

Please add environment variable before import shioaji. (version >= 0.3.3.dev0)

linux or Mac OS:

```
export SJ_LOG_PATH=/path/to/shioaji.log
```

windows:

```
set SJ_LOG_PATH=C:\path\to\shioaji.log
```

### 如何更改contracts下載路徑

Please add environment variable before import shioaji. (version >= 0.3.4.dev2)

linux or Mac OS:

```
export SJ_CONTRACTS_PATH=MY_PATH
```

windows:

```
set SJ_CONTRACTS_PATH=MY_PATH
```

python:

```
os.environ["SJ_CONTRACTS_PATH"] = MY_PATH
```

### 輸入密碼錯誤3次怎麼辦

線上解鎖

\*\* Note that you only have 2 chances to unlock your account online in a day. \*\*

\*\* We've migrate QA site to [Shioaji Forum](#) \*\*

## 8. 發佈版本

---

### 8.1 version: 1.3.1 (2025-12-29)

---

- fix: race condition in contracts

⌚ commit\_id: ee2be75c

⚡  
release\_at: 2025-12-29 16:00:00.000

### 8.2 version: 1.3.0 (2025-12-17)

---

- feat: futures quote
- feat: trading\_limits api

⌚ commit\_id: d570ac7b

⚡  
release\_at: 2025-12-17 16:00:00.000

### 8.3 version: 1.2.9 (2025-10-29)

---

- feat: support py3.14
- fix: profit\_loss callback error

⌚ commit\_id: de2133bf

⚡  
release\_at: 2025-10-29 16:00:00.000

### 8.4 version: 1.2.8 (2025-09-10)

---

- feat: punish api
- feat: notice api

⌚ commit\_id: 534d1ab5

⚡  
release\_at: 2025-09-10 16:00:00.000

### 8.5 version: 1.2.7 (2025-08-13)

---

- refactor: futures profit loss

⌚ commit\_id: e417ffcc

⚡  
release\_at: 2025-08-13 16:00:00.000

### 8.6 version: 1.2.6 (2025-06-16)

---

- feat: support python 3.13

- feat: support linux aarch64
- fix: mac import link error
- feat: contract download event
- feat: pysolace upgrade 0.9.51 (solclient 7.33.0.3)
- chore: drop support for python 3.6

 commit\_id: cf4b448d

⚡  
release\_at: 2025-06-16 16:00:00.000

## 8.7 version: 1.2.5 (2024-10-01)

---

- feat: refactor expire time of CA

 commit\_id: 6621685a

⚡  
release\_at: 2024-10-01 02:25:01.723

## 8.8 version: 1.2.4 (2024-08-28)

---

- feat: support py3.12

 commit\_id: a287f56c

⚡  
release\_at: 2024-08-28 16:00:00.000

## 8.9 version: 1.2.3 (2024-03-06)

---

- feat: change default site to bc
- feat: pysolace upgrade 0.9.40(solclient 7.28.0.4)
- feat: support apple silicon chip

 commit\_id: 8096bbac

⚡  
release\_at: 2024-03-06 16:00:00.000

## 8.10 version: 1.2.2 (2024-01-09)

---

- fix: remove column of profitloss in future

 commit\_id: ca973a81

⚡  
release\_at: 2024-01-09 02:27:31.383

## 8.11 version: 1.2.1 (2023-12-22)

---

- fix: windows inject dll issue

 commit\_id: 2a413848

↳  
release\_at: 2023-12-22 01:19:17.043

## 8.12 version: 1.2.0 (2023-12-20)

---

- feat: vpn
- feat: rust version ca
- refactor: test report flow

⌚ commit\_id: 856f39ea

↳  
release\_at: 2023-12-20 16:00:00.000

## 8.13 version: 1.1.13 (2023-11-01)

---

feat: impl ca.get\_sign on Darwin

⌚ commit\_id: 729f058e

↳  
release\_at: 2023-11-01 05:36:29.553

## 8.14 version: 1.1.12 (2023-08-22)

---

- feat: usage add limit and available byte info

⌚ commit\_id: cf5e4628

↳  
release\_at: 2023-08-22 16:00:00.000

## 8.15 version: 1.1.11 (2023-08-04)

---

- fix: custom\_field in validator for only support number and alphabet
- fix: pydantic v2 trade issue
- fix: pydantic v2 contracts cache issue

⌚ commit\_id: cc1da47e

↳  
release\_at: 2023-08-04 08:00:37.000

## 9. 使用限制

為避免影響其他使用者連線，請遵守以下使用規範



### • 現貨：

#### 近 30 日使用 API 成交金額      每日流量限制

|        |       |
|--------|-------|
| 0      | 500MB |
| 1 - 1億 | 2GB   |
| > 1億   | 10GB  |

### • 期貨：

#### 近 30 日使用 API 成交金額      每日流量限制

|                       |       |
|-----------------------|-------|
| 0                     | 500MB |
| 1 - 大台1000口 / 小台4000口 | 2GB   |
| > 大台1000口 / 小台4000口   | 10G   |



### 流量及連線數查詢

```
api.usage()
```



```
UsageStatus(connections=1, bytes=41343260, limit_bytes=2147483648, remaining_bytes=2106140388)
```

connection: 連線數量  
bytes: 已使用流量  
limit\_bytes: 每日流量限制  
remaining\_bytes: 剩餘可使用流量

 **數**

## • 行情：

```
credit_enquire, short_stock_sources, snapshots, ticks, kbars
```

## • 以上查詢總次數 5 秒上限 50 次

## • 盤中查詢 ticks 次數不得超過 10 次

## • 盤中查詢 kbars 次數不得超過 270 次

## • 帳務：

```
list_profit_loss_detail, account_balance, list_settlements, list_profit_loss, list_positions, margin
```

以上查詢總次數 5 秒上限 25 次

## • 委託：

```
place_order, update_status, update_qty, update_price, cancel_order
```

以上查詢總次數 10 秒上限 250 次

## • 訂閱數：

`api.subscribe()` 數量為200個

## • 連線：

同一永豐金證券 `person_id`，僅可使用最多5個連線。注意：`api.login()` 即建立一個連線

## • 登入：

`api.login()` 一天上限1000次

 **Warn**

## • 若流量超過限制，行情(ticks、snapshots、kbars)類查詢將回傳空值，其他功能不受影響

## • 若使用量超過限制，將暫停服務一分鐘

## • 若當日連續多次超過限制，本公司將暫停該 IP 及 ID 使用權

## • 若 ID 被暫停使用，請洽 Shioaji 管理人員