永豐金API及競賽策略格式說明

陸明德

Shioaji

1. 開戶並簽署文件
2. 新增API KEY
3. 請注意API KEY權限只勾選行情/資料，並且勾選無限制IP
4. 競賽前策略要交給助教，此API KEY也會，所以
   a. 為了安全（只勾選行情/資料）
   b. 為了方便（勾選無限制IP）
5. 請把API之simulation設為True

Shioaji（期貨即時行情BidAsk）

1. 期貨即時最佳五檔
2. 用來判斷下單成交價格
3. 依照右圖設定，取得即時報價

[參考]

```python
from collections import defaultdict, deque

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'].append(tick)

# In order to use context, set bind=True
@api.on_bidask_fop_v1(bind=True)
def quote_callback(self, exchange:Exchange, bidask:BidAskFOPv1):
    # append quote to message queue
    self['bidask'].append(bidask)
```

競賽策略限制說明

標的:台指期(台股期貨)

策略:當沖(當日沖銷)

手續費:來回100

限制:一次下一口, 同一時間只能持有一單

pysimulation說明

1. 使用pip install pysimulation 安裝
2. 用來模擬交易並更新交易紀錄
3. 每一次下單皆會更新交易紀錄寫入CSV檔
4. 創建order object, 並且填入學號

```
import pysimulation
order = pysimulation.order('111062693')
```

place_order說明

1. 使用place_order下單，傳入bidask，買賣，倉別
2. 買賣
   a. ‘Buy’
   b. ’Sell’
3. 倉別
   a. 'New'
   b. ‘Cover’

```
order.place_order(bidask, 'Buy', 'New')
```

```
Order(
code:    TXFG1
action:  Buy
octype:  New
price:   17747
fee:50
tax:71
ts:  2023-02-20
01:25:26
)
```

`list_trades()`說明

1. 回傳當天交易紀錄, list格式
2. 按照時間排序
3.

```
[{'code': 'TXFG1', 'action': 'Buy', 'octype': 'New',
'price': 17747, 'fee': 50, 'tax': 71, 'ts': '2023-02-20
01:25:26'}, {'code': 'TXFG1', 'action': 'Sell', 'octype':
'Cover', 'price': 17780, 'fee': 50, 'tax': 71, 'ts':
'2023-02-20 01:25:28'}]
```

當沖策略範例(冠軍操作法ORB)

1. 使用30分鐘K線圖
2. 第二根K棒開於第一根之高低點之內
3. 突破第一根K棒高點則做多
4. 跌破第一根K棒低點則放空
5. 停損設為2%
6. 收盤前15分鐘市價出場

當沖策略範例(冠軍操作法ORB)

```python
import shioaji as sj
import pysimulation

api = sj.Shioaji(simulation=True)
accounts = api.login(
    api_key="",      # 請修改此處
    secret_key=""    # 請修改此處
)

order = pysimulation.order('111062693') # 請改成自己的學號
```

```python
# 近月臺股期貨商品檔
contract = min(
    [
        x for x in api.Contracts.Futures.TXF
        if x.code[-2:] not in ["R1", "R2"]
    ],
    key=lambda x: x.delivery_date
)

from shioaji import TickFOPv1, BidAskFOPv1, Exchange

from collections import defaultdict, deque

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'].append(tick)

# In order to use context, set bind=True
@api.on_bidask_fop_v1(bind=True)
def quote_callback(self, exchange:Exchange, bidask:BidAskFOPv1):
    # append quote to message queue
    self['bidask'].append(bidask)

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

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

1. 台指期近月TXFR1無法取得行情資料
2. 改成圖片程式碼取得近月商品檔代號
3. 訂閱即時TICK及BIDASK資料

當沖策略範例(冠軍操作法ORB)

```
while True:
    if datetime.datetime.now(pytz.timezone('ROC')).time() >= datetime.time(9, 15, 0):
        high = msg_queue['tick'][-1]['high']
        low = msg_queue['tick'][-1]['low']
        # 開盤30分K，最高價最低價用於判斷多空
        break
    else:
        continue
```

1. 時區設為台灣
2. 開盤30分後(9:15)，取得最高最低價用來判斷多空進場

當沖策略範例(冠軍操作法ORB)

```python
while True:
    if msg_queue['tick'][-1]['close'] > high:
        # 成交價 > 最高價，新倉買進
        order.place_order(msg_queue['bidask'][-1], 'Buy', 'New')
        break
    elif msg_queue['tick'][-1]['close'] < low:
        # 成交價 < 最低價，新倉賣出
        order.place_order(msg_queue['bidask'][-1], 'Sell', 'New')
        break
    elif datetime.datetime.now(pytz.timezone('ROC')).time() >= datetime.time(13, 30, 0):
        break
```

1. 突破第一根K棒高點則做多
2. 跌破第一根K棒低點則放空
3. 2%停損
4. 超過時間就跳出判斷

```python
while True:
    # 2%停損
    if order.list_trades()[-1]['action'] == 'Buy':
        if msg_queue['tick'][-1]['close'] < order.list_trades()[-1]['price'] * 0.98:
            # 新倉買進2%停損，賣出平倉
            order.place_order(msg_queue['bidask'][-1], 'Sell', 'Cover')
            break
    elif order.list_trades()[-1]['action'] == 'Sell':
        if msg_queue['tick'][-1]['close'] > order.list_trades()[-1]['price'] * 1.02:
            # 新倉賣出2%停損，買進平倉
            order.place_order(msg_queue['bidask'][-1], 'Buy', 'Cover')
            break
    elif datetime.datetime.now(pytz.timezone('ROC')).time() >= datetime.time(13, 30, 0):
        break
```

當沖策略範例(冠軍操作法ORB)

```
if datetime.datetime.now(pytz.timezone('ROC')).time() >= datetime.time(13, 30, 0):
    if order.list_trades()[-1]['octype'] == 'New':
        if order.list_trades()[-1]['action'] == 'Buy':
            # 當沖13:30強制沖銷，賣出平倉
            order.place_order(msg_queue['bidask'][-1], 'Sell', 'Cover')
        elif order.list_trades()[-1]['action'] == 'Sell':
            # 當沖13:30強制沖銷，買進平倉
            order.place_order(msg_queue['bidask'][-1], 'Buy', 'Cover')
    break
```

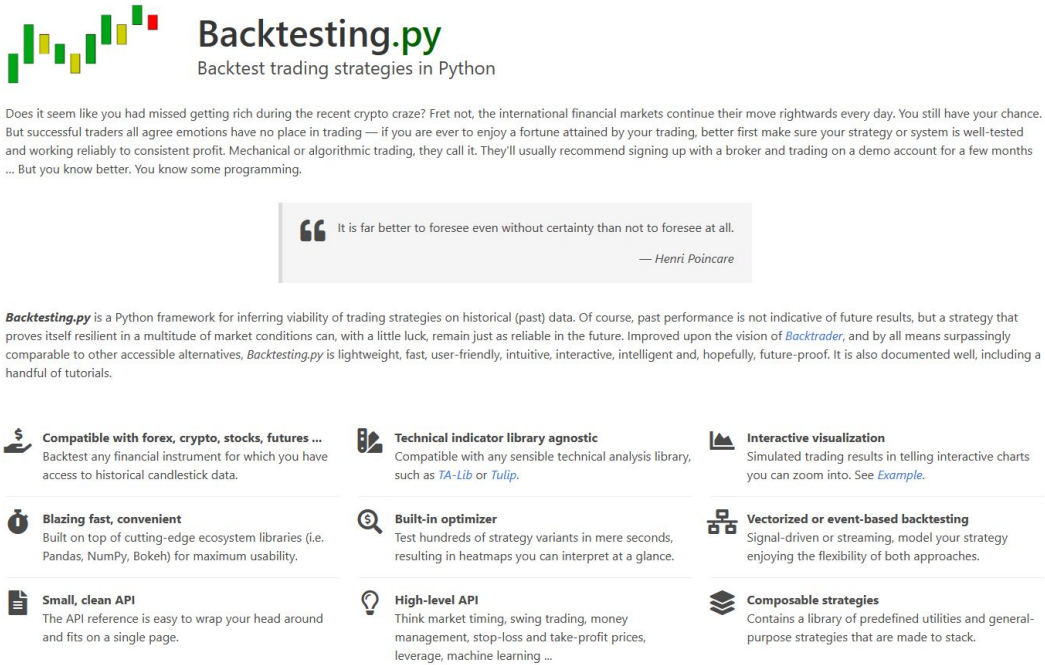
重要！必須包含

當沖一定是當日沖銷，會在13:30強制沖銷

假設今天是實單會自動沖銷

To do

了解永豐金API、認識使用技術指標(ta lib)、發想策略、回測策略



Backtesting.py

Backtest trading strategies in Python

Does it seem like you had missed getting rich during the recent crypto craze? Fret not, the international financial markets continue their move rightwards every day. You still have your chance. But successful traders all agree emotions have no place in trading — if you are ever to enjoy a fortune attained by your trading, better first make sure your strategy or system is well-tested and working reliably to consistent profit. Mechanical or algorithmic trading, they call it. They'll usually recommend signing up with a broker and trading on a demo account for a few months ... But you know better. You know some programming.

> It is far better to foresee even without certainty than not to foresee at all.
>
> — *Henri Poincare*

Backtesting.py is a Python framework for inferring viability of trading strategies on historical (past) data. Of course, past performance is not indicative of future results, but a strategy that proves itself resilient in a multitude of market conditions can, with a little luck, remain just as reliable in the future. Improved upon the vision of *Backtrader*, and by all means surpassingly comparable to other accessible alternatives, *Backtesting.py* is lightweight, fast, user-friendly, intuitive, interactive, intelligent and, hopefully, future-proof. It is also documented well, including a handful of tutorials.

Compatible with forex, crypto, stocks, futures ...
Backtest any financial instrument for which you have access to historical candlestick data.

Technical indicator library agnostic
Compatible with any sensible technical analysis library, such as *TA-Lib* or *Tulip*.

Interactive visualization
Simulated trading results in telling interactive charts you can zoom into. See *Example*.

Blazing fast, convenient
Built on top of cutting-edge ecosystem libraries (i.e. Pandas, NumPy, Bokeh) for maximum usability.

Built-in optimizer
Test hundreds of strategy variants in mere seconds, resulting in heatmaps you can interpret at a glance.

Vectorized or event-based backtesting
Signal-driven or streaming, model your strategy enjoying the flexibility of both approaches.

Small, clean API
The API reference is easy to wrap your head around and fits on a single page.

High-level API
Think market timing, swing trading, money management, stop-loss and take-profit prices, leverage, machine learning ...

Composable strategies
Contains a library of predefined utilities and general-purpose strategies that are made to stack.

沒有想法？

TradingView有很多技術指標以及交易策略可供參考