et press

程式交易的理論與實踐

深入淺出的程式交易工具書

蔡寨民（Calvin）著

程式交易的理論與實踐

〈推薦序 1〉

蔡嘉民先生（Calvin）出版大作《程式交易的理論與實踐》，邀請區區作序，深感榮幸，還記得初遇 Calvin 時，他還是香港大學本科生。吾友馮達生博士及吳嘉儀博士伉儷從紐約回港，特意在他們的研討會茶聚時介紹 Calvin，並提到他與數位同學成立團隊，參加算法投資比賽。Calvin 當時已經對各種算法有堅固認識，滿腔熱血，誓要在比賽中闖出名堂。他們的團隊到最後不負眾望，贏得 2016-2017 年度算法投資比賽季軍。Calvin 並於畢業後，立即開展算法投資事業，並取得卓越成就，實在可喜可賀.

本指南以 Calvin 的實戰經驗為基礎，涵蓋各大範疇：從大戶、散戶的心態、動作，以及各種投資算法，硬件和軟件設施，皆有仔細討論，必令讀者對程式交易有深入了解。

深望此書令大眾對程式交易有科學化的認識，以致活學活用：不單帶來穩健財富，還於各種市況下保持身心康泰。

黃寶誠

史丹福大學統計學博士

香港大學統計及精算學系名譽教授

香港中文大學統計學系兼任教授

〈推薦序 2〉

收到嘉民的寫序邀請，再細閱此書後，驚覺才短短幾年，嘉民已自成一家、成長為獨當一面的投資專家。書中對程式演算交易的各方各面都有涉及而且條理分明，對初接觸此類交易的人幫助很大。書中首先分析散戶之對手類型：是莊家大戶，基金定其他。因這對於能否雙贏或是零和遊戲至關重要。不同對手用不同策略應對，否則後果堪虞。至於策略種類，書中詳細分為趨勢性交易，相反方向交易，市場中性交易，波動率交易及市場莊家交易各類型……當中有些常為散戶忽略。

在不同時區中不同的交易市場，內裡不同的資產如債券、股票、外滙等各有特點。衍生工具如期指、期權扮演不同角色，應如何選擇，作者亦一一簡說。再者人手交易與程式交易之間，有何優劣。怎樣除去散戶陋習。執行程式交易時，各式費用紛呈，面對滑價時又應怎樣處理。書中都有提到。

書內後半部集中解釋如何為交易程式揀選策略。回溯測試中有不同的選擇指標，何者重要。找到交易策略後應怎樣優化內裡的參數。注碼的分配，風險的管理及何時應放棄程式而改由人手干預。作者都羅列詳盡，分析清楚。

環視坊間有關書籍，詳細述説程式交易的不多，所以本書實在難能可貴，應能幫投資者打開程式交易大門。兼且書中多處流露個人投資心得，可感受到作者並非單單是個投資者，而是有一套完整交易邏輯的思想投資家。

Dr. S. P. Yung
Department of Mathematics
The University of Hong Kong

〈推薦序 3〉

身為作者前上司，很開心被邀請為他的新書寫推薦序。認識作者的第一天，作者給我的感覺是對金融行業充滿熱誠，憑著他天賦的記憶力、觀察和分析能力，加上最重要的勤奮，促使他不斷努力吸收知識和學問。

17 世紀荷蘭東印度公司在阿姆斯特丹成立全世界首家證券市場，市場交易隨著科技不斷進步，由公開叫價到螢幕交易，到近年發展的程式交易——透過電腦有系統、紀律地執行交易策略已是市場交易發展趨向。我本人很榮幸可以見證這個歷史性發展。在剛剛入行的時候，公開叫價已是在轉到初步螢幕交易。那時候我的前輩須要努力學習用電腦，現在到了我要學習程式交易了。

這本書提供不少金融資訊，既包括程式交易系統開發不同交易策略，利用歷史或即市數據作模擬買賣，來預測其交易策略的回報，還有對不同金融產品的新註解。作者分享投資實戰經驗，讓讀者感受成功交易的興奮與交易挫敗的迷惘。最重要的是可以讓每一位讀者都可以深深地體會到現在程式交易是如何重要。

我深信讀者從這本書中能夠得到啟發，值得推薦給所有對金融市場有興趣的讀者。

Judy Chan
Founder and Managing Director
Smart Wise Financial Holding

〈推薦序 4〉

我認識作者已是一年多前在一場中港學界程式交易比賽上，當天在場聽他講解參賽的交易策略時，已察覺到他對市場動態的敏銳觸覺和對交易者動機的透徹洞悉力，令其團隊研發的交易策略成功地揉合了人性表現和數據分析的特性，他們最後亦不負眾望，勇奪「最佳程式交易策略」獎項，確是實至名歸。

但真正令在下佩服的並不單是作者在比賽當天的卓越表現，而是跟他認識後，在交易知識上的慷慨分享，這書的面世便是最佳證明。

倘若你是一個慣用傳統（技術或基本）分析去作交易決定的投機（資）者，但又正準備開發一些另類的交易方式，這書倡議的方法必定能令你擴闊視野，獲益良多。

常道「魔鬼都在細節裡」，作者在這書裡能以深入淺出的方法，有系統地去解釋作為一個稱職的程式交易員必須留意的細節，從交易策略、回溯測試、策略優化到風險管理，都一絲不苟，鉅細靡遺地清楚交代。

這本書絕對不是那些魚目混珠的甚麼投資必勝法，有經驗的交易員應該明白這類所謂「天書」在世上並不存在！

若你有志成為（或轉型為）一位程式交易員，但卻不知道從何起步，這書可以給你一個可靠的路線圖，提升你能成為一個出色程式交易員的機會率；儘管你已從事程式交易一段時間，這書亦可助你溫故知新，從而改善你現有的表現。

當然交易策略成功與否就全靠讀者自身的創意和經驗了！

Walter Cheung
基金經理
CONTRENDIAN

〈推薦序 5〉

Ever since meeting the author at an inter-university trading competition, I have enjoyed discussing novel trade ideas and market observations during our regular lunch gatherings. Through our discussions, I became increasingly impressed with his passion for program trading, and the unique insights that can only come from a true enthusiast.

While there are numerous trading books on the market, this one sets itself apart as it offers a refreshing dose of common sense. The reader will not find excessive hype or unrealistic promises of winning trading formulae, but practical explanations about who the market participants are, what drives market prices, and why certain trading strategies may work. In an era where common sense is uncommon, reading a book that is well grounded in the first principles of trading is indeed enlightening.

For the motivated reader, the author offers a systematic process for program trading. He explains in detail all aspects of the complete trading process — hypothesis, algorithm, back-testing, operations, evaluation, enhancement, sizing and risk management. These processes are what professional traders

practice every day to stay ahead of the game. He emphasizes the dangers of relying on human judgment, while highlighting the importance of applying critical thinking to evaluate trading algorithms. He also introduces the concept of diversification across asset class, geography and trading instrument. Experienced and novice traders alike will find these concepts invaluable.

The book is a practical guide for the trader who wants to tilt the odds in one's favor over the long run. It is not a recipe book for the occasional cook, but a fundamental manual on how to train oneself to become a master chef — in the world of program trading.

Frank Ho
Portfolio Manager
CONTRENDIAN

〈推薦序 6〉

認識蔡嘉民（Calvin）起碼有幾年時間，大家都是程式交易的同好，當時我提議 Calvin 寫一本關於程式交易的書，正是大家現在手上這本。

程式交易在香港似乎是小眾玩意，對比起鄰近地區例如台灣、中國大陸等，香港在這面起步得非常遲。後來到 Alpha Go 出現，間接引起了大眾對程式交易的興趣，程式交易才算是在香港引起了注意。

可是網上能找到關於香港市場的程式交易資訊非常有限，而且香港亦從未有財經書籍以程式交易為主題，一般沒有編程經驗的投資者難以找到學習的方向，猶如老鼠拉龜，拉牛上樹，於是乎我和 Calvin 合力創辦了「香港程式交易研究中心」，一間以程式交易為主題的教學機構，專注程式交易工具開發、解決方案及教授程式交易的課程。

我和 Calvin 都並非修讀電腦出身，Calvin 畢業於港大醫學院，我畢業於港大商學院，大家所讀的科目均與程式風馬牛不相及，只是大家抱著對交易的一股熱誠，驅使我們去自學程式，驅使我們去運用程式交易。當初我們自學時遇到的困難，在市場上交「學費」的情景，依然歷歷在目。一路走來的經驗可說是全部以真

金白銀換取，自然明白初學者所面對的困窘，尤其是非主修電腦的初學者所困惑的，我們都完全明白。

程式語言不是外星語言，程式交易亦非想像中的困難，大家所欠缺的，往往是一條清晰的學習階梯，以及有經驗人士的指引，希望大家可以籍著 Calvin 這本書了解到程式交易的真實面貌，期望香港愈來愈多人投入程式交易，亦祝願 Calvin 此書一紙風行。

歐陽一心

香港程式交易研究中心董事

《期權投機客》系列作者

目錄

第 4 章 MultiCharts 系統

第 5 章 基本交易程式編寫

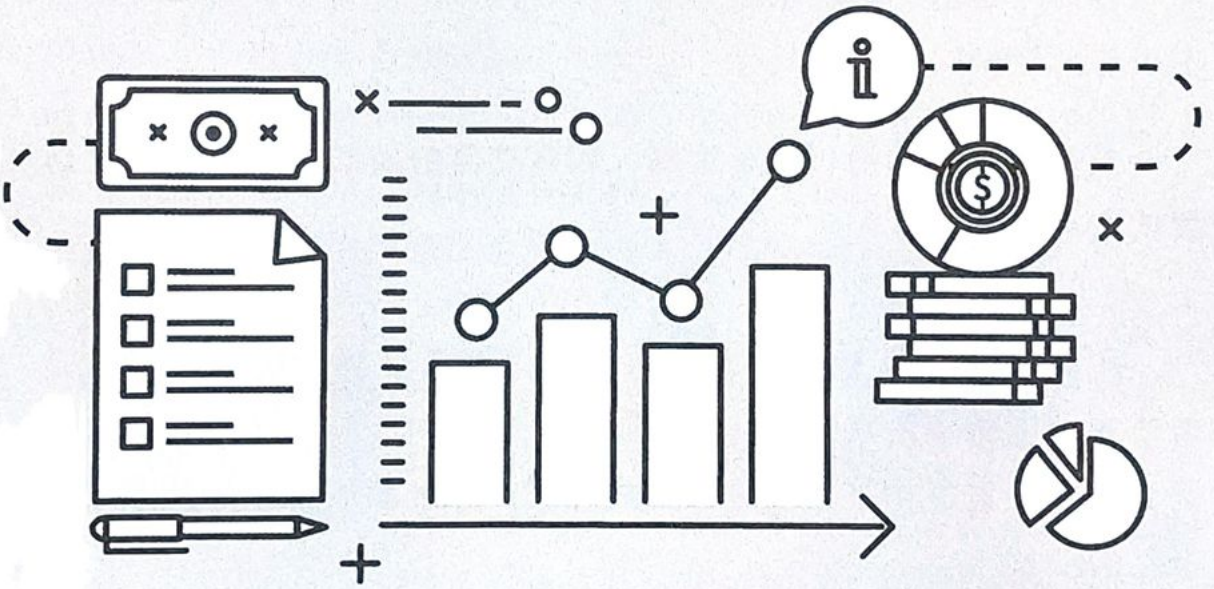
目錄

第 6 章　策略表現判斷

第 7 章　交易的數學

第 1 章

為甚麼你要學程式交易



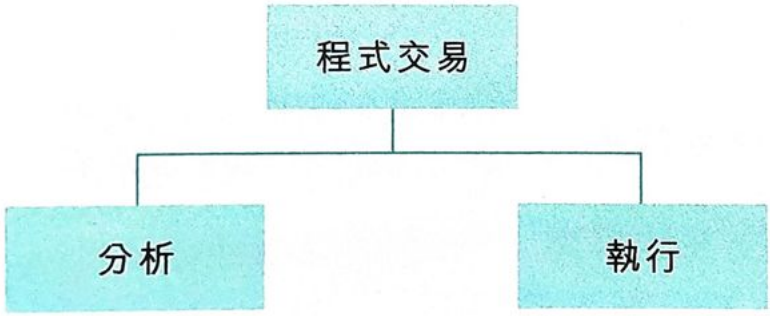
- 程式交易的定義
- 程式交易的好處
- 程式交易的種類
- 常見誤解

〈 1.1 〉

程式交易的定義

凡是運用電腦進行分析策略、執行交易的操盤方法都可稱之為程式交易。

程式交易已不是新奇的事物，自從電腦發明以來，不少投資者都嘗試透過運用電腦去計算出對自己有利的交易方法，簡單如運用 Excel 計算恒指平均線突破的命中率，已經屬於程式交易中「分析」的部分。



然而，對於不懂寫程式的散戶，分析部分遠比執行簡單。這是由於執行部分需要較高的程式水平，才能將交易完全自動化。因此散戶通常會按照電腦分析的結果手動下單，這種操作手法我們稱之

為半自動（Semi-auto），雖然如此，但也算是最接近程式交易的操盤方法。

程式交易的定義非常廣闊，簡單如移動平均線的交叉策略、技術指標的突破、價格型態的辨認、連跌幾日的博反彈部署都可以運用程式交易進行；而層次較高的策略，例如雙股票的配對交易、期權莊家的報價、債券之間的套戥等，就絕對屬於程式交易的範疇；甚至乎運用語言辨識技術分析央行行長的說話是利好或利淡、分辨社交媒體對股市的樂觀或悲觀情緒等，同樣是程式交易。

當然，高層次的程式交易方式是大戶的專利，但對散戶而言，即使只運用電腦分析技術指標、價格訊號等傳統交易方法，仍然有巨大的獲利空間，而關於獲利空間的課題，將會在本書內一一探討。

〈 1.2 〉

程式交易的好處

"People have a strong tendency to go along with the status quo or default option."

- Richard H. Thaler（2017 年諾貝爾經濟學獎得主）

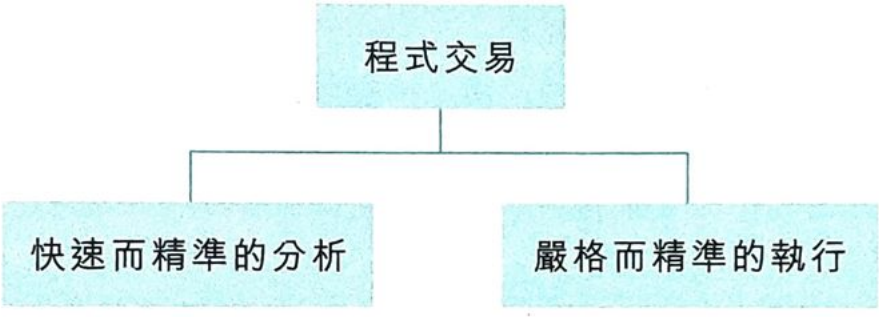
程式交易和人手交易幾乎是二元對立，其原因非常簡單，因為人類不是理性的生物。

人類進行交易活動時，存在大量的心理偏差。最著名的 Loss Aversion 曾指出，投資者在虧損和獲利時對風險的厭惡程度是不成正比的，例如，當投資出現虧損，部分投資者總是傾向承受更大的風險，結果輸錢後再失控地胡亂交易，令情況更加惡化；但在獲微利時，他們的心理卻變得厭惡風險，由於太早止賺，錯失賺更多的機會。

類似的心理偏差多不勝數，例如稟賦效應、現狀偏差、熱手謬誤等，在此不一一詳解。重點是人類在進行手動交易時，往往會受心理偏差影響，導致出現高買低賣情況，許多投資者都有難受的交

易體驗，因抵受不了心理壓力而下錯決定，事後便認為自己愚蠢地犯下的交易錯誤，這些都是手動交易的常見問題。

因此，程式交易的好處顯然而見，除了剛才提及的分析外，更重要的是執行部分，電腦可以冷冰冰地執行你的交易指令，完全不會受情緒和心理因素影響，所以剛才的圖例可以改寫成下圖。



散戶要由手動交易過度到程式交易，是一個脫胎換骨的過程，彷彿由凡間晉升到另一個境界，不單是知識的進步，更會大幅改變對交易的看法。此時，你再回看那些只懂手動交易的朋友，就會發覺自己的境界難而言喻，其他人真是「夏蟲不可語冰」。

〈 1.3 〉

程式交易的種類

程式交易中，各種門派功夫可說是大相逕庭，所以我們很難概括地描述程式交易，而且其分類方法也有很多，為了讓讀者容易上手，筆者在此以最經典的分類作例，但要注意它只是眾多分類方法的其中一種，待讀者掌握基本功後，可再了解其他分類方法。

程式交易一般可分為以下 4 類：

1. 方向性交易
2. 市場中性交易
3. 波動率交易
4. 市場莊家交易

（1）方向性交易 Delta Trading

根據筆者的非正式統計，幾乎所有初學者都是由方向性交易開始的，因為其餘 3 種交易方法都難以用手動方式執行，所以程式執行方向性交易是最容易的。

易上手不代表無用，反而其優點相當明顯，這就是制定策略的難度較低。一般來說，制定方向性策略的重點是市況波幅的判斷機制，趨勢式訊號會在大升市或大跌市中食盡大浪，而區間式策略就會在牛皮市中「密食當三番」。因此，對於方向性交易，開倉訊號本身的命中率其實是次要，重要的是趨勢市或牛皮市的判斷機制，而制定這機制的難度卻稍為比起判斷方向容易，只要配合程式交易嚴格執行的特點，獲利不是難事。對於程式交易新手，這絕對是一個好消息，所以本書的實例多從方向性策略出發，讀者可以這交易法作為學習程式交易的切入點。

既然說難度較低，為何會有其他的交易方法出現？在交易世界中，除了利潤之外，另一個必須關注的因素是風險。方向性交易方法最難控制的是風險水平，尤其是需要按金維持的期貨倉位，若果倉位下行風險太大，有機會觸及按金不足的水平，萬一被斬倉，即使該策略理論上之後能重新獲利，但已沒有足夠的資金繼續執行。

籠統來說，比較其他程式交易策略，方向性策略的風險是最難控制的因素，所以投資者才會設計出市場中性、莊家交易、波動率交易等策略。那些策略通常比方向性策略有較低風險，而利潤風險比例則較高。對機構投資者（如對沖基金等）來說，這些比例數據是尤其重要的，因為他們須經常向客戶交待，亦要面對資金隨時存取的因素，倉位變化不能太大，這解釋了為何散戶在運用程式交易時較多執行方向性交易，而機構投資者則較多參與市場中性交易、波動率交易等。

順帶一提，方向性交易的英文為 Delta Trading，其中 Delta 的意思是交易者要 Take Delta，即是操盤時要決定倉位的 Delta 是正數還是負數，正數是好倉，負數則是淡倉。由此可見，交易者要對資產價格升跌的方向有看法才能 Take Delta，而 Delta 一向用來描述衍生工具與相關資產的價格關係，例如揸期指的 Delta 是 +1，沽期指的 Delta 是 -1，這就是 Delta Trading 英文名稱的由來。

（2）市場中性交易 Market Neutral Trading

著名香港期權書作家歐陽一心大作《期權投機客市場中性操盤攻略》中提到，市場中性策略其實是「不論大市升跌方向都能賺錢」的策略。最典型的例子莫過於配對交易（Pair Trading）。例如，在美國加息潮中，投資者認為銀行股是最受惠的，商品股卻受制於商品價格受加息所壓，因此認為滙控（0005.HK）比起中海油（0883.HK）強勢，於是買入滙控股票，同時沽空同等價值的中海油股票，就算兩隻股票同時升或同時跌，只要滙控的升幅大於中海油，或者滙控跌幅較小，都能夠從整體組合中獲利。

上述例子純粹是事件驅動式的炒法，真正的配對交易方法還有很多種。例如是相同板塊的建設銀行（0939.HK）及工商銀行（1398.HK），中長線內兩者的同步率很高，若果短時間內兩者的價格差距拉闊，可以用配對交易方法買入較平者，同時沽空較貴者，期望兩者價格中長線回歸平均。這種手法涉及更多統計學的計算，人手幾乎難以實行。

當然，除了配對交易外，市場中性策略還可以涉及各種衍生工具，例如是期權、可換股債、信貸違約掉期（CDS）、跨品種套戥等。這些策略的共通點是執行速度對策略成效有莫大的影響。試想像，當你正進行配對交易時，明明想買入滙控同時沽空中海油，但兩個交易之間足足相隔數分鐘之久，期間兩者價格有機會大幅改變，嚴重減少本可賺取的利潤，這種情況正是執行速度太慢的問題，也是手動交易永遠不能解決的。程式交易的執行速度優勢就能彌補這個缺口，令市場中性交易策略變得可行。

（3）波動率交易 Volatility Trading

波動率通常分為歷史波幅（Historical Volatility，簡稱 HV）及引伸波幅（Implied Volatility，簡稱 IV）。這兩個專有名詞常見於期權交易上，當中後者正是期權計價公式中的一個重要因素，所以波動率交易大都涉及期權交易，是程式交易種類中較困難的範疇。

考慮到不是每位讀者都有期權知識，這裡只作簡單描述。波動率的最經典顯示形式，莫過於美國的波動率指數（CBOE © Volatility Index，簡稱 VIX，俗稱恐懼指數）。根據著名香港期權書作家歐陽一心大作《期權投機客市場 VIX 獲利絕技》，VIX 是根據美國標普 500 指數期權的引伸波幅編製而成，是全球最直接能反映股市引伸波幅的指數，而由於 VIX 本身亦有眾多衍生工具，例如

是 VIX 的期貨、期權、ETN 等，投資者可透過交易 VIX 的衍生工具作波動率交易。

雖然散戶很少接觸波動率交易，更遑論相關的程式交易，但其實散戶也可間接受惠於大戶的波動率程式交易。舉例説，美國是一個市場深度極高的市場，各種 ETF、VIX 的衍生工具都有大戶不停運用程式進行套戥、投機、對沖等動作，大大增加這些產品的交投量，令到買賣差價收窄，交易亦更加容易。一般投資者想在美國市場上買入 VIX 期貨作對沖亦非常容易，毋須承擔太闊的買賣差價；相反，觀乎俗稱「香港 VIX」的恒指波幅指數（VHSI）期貨，由於缺乏大戶參與，買賣差價太闊，多年來的成交量都遠低於美國的同類產品。

去到波動率交易的層面，已經是散戶難以接觸的技術水平，而接著要介紹的市場莊家交易更是大戶專利。不過，作為一本入門書籍，即使未有機會操作，當中的原理仍然是不可不知的。

（4）市場莊家交易 Market Maker Trading

老實説，這種獲利方式能否稱為「交易」都成疑問，這裡以交易稱呼亦屬約定俗成。

所謂市場莊家，是指一些為欠缺流動性的資產提供買賣報價者，即是流動性提供者，最典型的例子是期權莊家（Options

Market Maker，行內簡稱為 OMM）。由於香港有很多場內期權都欠缺投資者參與，在缺乏流動性的情況下，產品也難以吸引投資者，形成惡性循環。故此，交易所引入市場莊家機制，以極低甚至免手續費的方式吸引市場莊家為期權提供買入價（Bid Price）和賣出價（Ask Price），莊家藉此賺取兩者之間的差價（Bid-Ask Spread），同時提供市場流動性，讓其他投資者可以根據莊家報價進行期權交易，一舉數得。

一般而言，投資者買入或沽出期權莊家出價的期權合約，不是與期權莊家直接對賭。原因是期權莊家只想賺取 Bid-Ask Spread，並無意進行對賭，所以他們在接受投資者的盤後，會在其他市場進行對沖，例如，投資者沽出大量價內認購期權，莊家便等於買入了大量價內認購期權，當面對股價下跌的風險，莊家可以因應手上期權倉位的風險參數，沽空適當數量的正股，藉以對沖股價下跌的風險。由此可見，接盤和對沖的動作幾乎要同步執行，若果期間有任何延遲，股價下跌的因素很容易就抵銷莊家原先所賺的 Bid-Ask Spread。因此，莊家必須使用程式進行相關的操作，這自然在程式交易的範疇之內。

介紹了 4 種程式交易種類後，相信大家已明白程式交易的範圍相當之廣，非三言兩語可以概括。在接著下來的章節中，我們主要探討方向性交易的操盤方法，作為學習程式交易的切入點。

〈 1.4 〉

常見誤解

誤解 1：程式交易的命中率極高。

解說： 程式交易與高命中率沒有必然關係。一個成功交易程式的最基本條件是長線獲利，命中率只是其中一個要考慮的因素。舉例說，某策略的命中率高達 80%，然而每次止賺只能獲利 10 點期指，但若然要止蝕，損失卻高達 200 點。雖然止蝕的機會只有 20%，但整體策略的期望值卻必然是負數，絕不適合用來交易。

換個角度，即使策略的命中率低至 20%，但每次命中都能獲利 200 點，而每次止蝕則要損失 10 點，這便是一個值得執行的策略。

誤解 2：程式交易是難以學習的。

解說： 許多投資者以為，進行程式交易必須具備深厚的數學根基，所以擔心自己數學水平不夠。其實，這是一個誤解，

尤其是當操盤的金額愈少，程式交易的難度就愈低；只有當操盤金額達到某水平後，程式交易的難度才會以幾何級數提高。以香港市場為例，遇到明顯難度的樽頸位大約是 1,000 萬至 2,000 萬港元左右，這資金水平遇到的滑價問題（Slippage，後文會深入探討）會特別明顯，必須透過大量分散的方式去克服。當散戶可操盤的資金少於這金額水平，難度不會太高。

一般散戶的程式交易都不需要高深的數學，基本入門級的統計學知識已經足以應付。特別注意是方程式、理論等東西都不是重點，基本的邏輯思維才是關鍵。此外，因為程式是電腦的語言，而電腦對程式的理解是非黑即白的，所以編寫交易策略的程式必須是絕對的客觀，對買賣訊號的定義不能有半點含糊，否則電腦便無法理解。

另外，由於程式交易牽涉的是一環接一環的程式和電腦設定，程式交易員必須細心處理，否則一個細微的錯誤都足以改寫策略的盈虧。

誤解 3：程式交易是搞來搞去「得個吉」。

解說： 這是筆者觀察到許多半途而廢者的感言。學習程式交易所需要的苦功遠超單純的買賣股票，然而當你開展了程式交易的路，就會發現所遇到的問題是以前從來未想過的，許

多看似細微的決定，其實是經過一大輪邏輯推論的結果。回頭一看，許多普通散戶的「成功」可能有大量的僥倖成份。故此，學習程式交易不只是改變交易工具這麼簡單，更是改變你對世界看法的一道大門。

誤解 4：散戶進行程式交易沒有優勢。

解說： 程式交易常常被誤會為大戶專利。這是由於過往的電腦硬件、程式知識仍未普及，只有大戶才擁有足夠財力。但時至今日，不少程式員出身的散戶早就踏入程式交易的世界，至於沒有程式根底的散戶，所欠缺的就只是編寫程式的技能而已。換個角度，大戶也是被迫依靠程式的力量去增加自己的優勢，因為其金額太大，難以承受人手交易所帶來的風險，唯有透過這種軍備競賽提升競爭力。然而正如剛才所說，資金愈多，程式交易的難度就愈高，散戶看似勢孤力弱，但資金少反而成為散戶的優勢。

散戶的另一項優勢是夠靈活。試想像某細價股在平日的成交量只有數十萬元，即使是散戶也可以掃起股價，到平倉的時候就必須分段沽出，靈活度較低。反之，這種情況對大戶來說可算是家常便飯，而其實大戶要面對的問題比起散戶更多。因此，散戶也不應妄自菲薄，只要好好運用靈活的優勢，一樣可以成功。

誤解 5：程式交易是完全客觀的。

解說： 雖然程式交易可以防止執行策略時會被情緒影響判斷，可以進行機械式操盤，但在制定策略的階段，其實仍有不少人為決定，當中包括歷史數據的採樣。

歷史數據不是愈多愈好，過舊的數據對未來操盤根本沒有統計意義。而在制定策略時，決定採用的歷史數據長度會受目標資產的特質、市場政策、地區經濟環境等外在因素影響，沒有一條公式可以簡單地解決這問題，所以這要依靠程式交易員的主觀決定。

所以說程式交易不是科學，亦非完全客觀，只是將主觀因素減至可接受程度。例如，在判斷香港恒指期貨的歷史數據採樣時，程式交易員可判斷 2003 年之前的數據可以忽視不理，因為當時的恒指成份股仍未包含大量內銀股、內險股等現時佔港股半壁江山的藍籌股。這種以主觀判斷數據採樣的範圍所導致的偏差和失誤都較少，比起完全手動交易已經有天大的差別。

誤解 6：根據有效市場假說，程式交易的優勢早已耗盡。

解說： 有效市場假說（Efficient Market Hypothesis）指出，所有資產價格早已反映其各樣因素，所以無法運用數據再發

掘未被發掘的機會。但是，金融市場始終是由人組成的，人的存在代表人類的心理會重複地以相同方式影響資產價格。例如，每當處於熊市恐慌階段，人的恐懼心理會蓋過理性，所以相同的恐慌性拋售會重複地出現；若果只用手動交易，交易者的心理有很大機會表現出同樣特性，但運用程式就沒有這問題。這種「歷史會重複」的假設是基於人類心理和不會改變的生物本能，所以其重複性會相當之高。

當然，「歷史會重複」不等於「歷史會簡單重複」。每個熊市出現的暴跌都不可能有相同的點數或百分比，所以程式交易員要以聰明的方法去尋找重複的模式或機會，這就是他們必須掌握的知識之一。

誤解 7：家用的電腦無法進行程式交易。

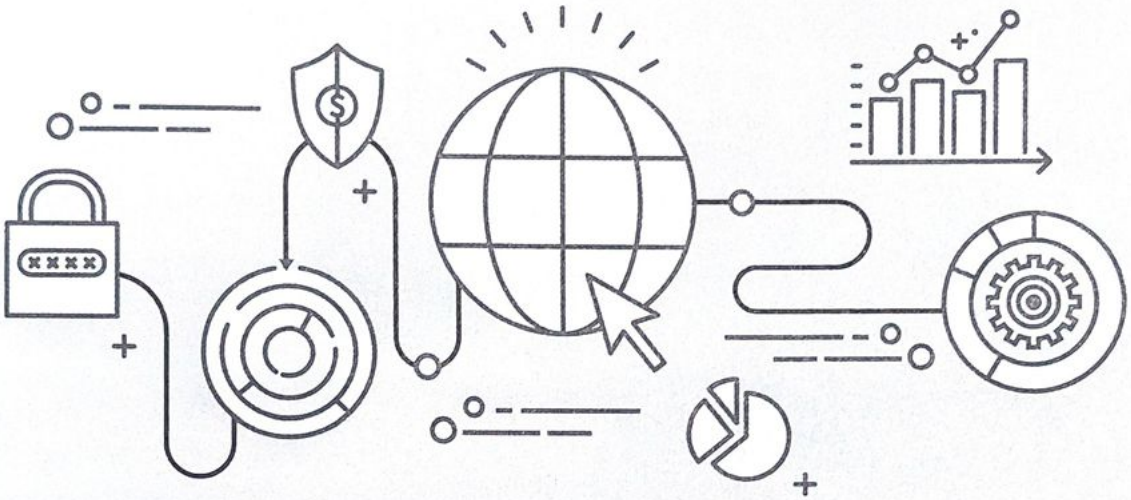
解說： 大多數散戶以為中央處理器（CPU）才是硬件中的樽頸，但根據筆者的經驗，絕大多數散戶撰寫的策略程式都毋須強大的 CPU，反之，網路的反應速度（ping）、記憶體（RAM）的多少更為重要，原因是前者影響市價盤會否以較好的價格成交，後者則影響回測和優化的效率。當然，更快的 CPU 可以減少回測和優化所需的時間，然而影響並不明顯。以相同系列的 Coffee Lake intel i7-8700 與 i5-8400 為例，兩者的優化時間最多只有一兩成的差距，對比程式本身所帶來的影響，可說是小巫見大巫。若果

程式寫得簡潔，少用一兩個 for loop，隨時可以倍數計地優化所需的時間，可見 CPU 的重要性遠不及寫得一手好 code。

當然，所謂「工欲善其事必先利其器」，好的硬件可以帶來更大的方便。現時連 MultiCharts 等第三方程式交易軟件都已經支援多核心的處理器，大幅減少優化所需的時間。因此，若你仍然使用只有 4 核心的 i3 處理器，速度顯然不及擁有 6 核心的 i5 處理器，制定策略便需要更長時間等待結果。

至於交易速度方面，樽頸多數是網路反應速度（ping），現時香港的光纖寬頻一般已可將 ping 壓至 50ms（ms = 毫秒，即千分之一秒）以下，即使是運用第三方軟件的程式交易員，要達到十分之一秒的數據存取和執行速度都是沒有問題的，當然在這種高速下，能否找到交易的優勢就是另一個問題，重點是香港的光纖網速絕對足夠在家中執行程式交易，所以大家不用擔心。

第2章
如何開始著手程式交易

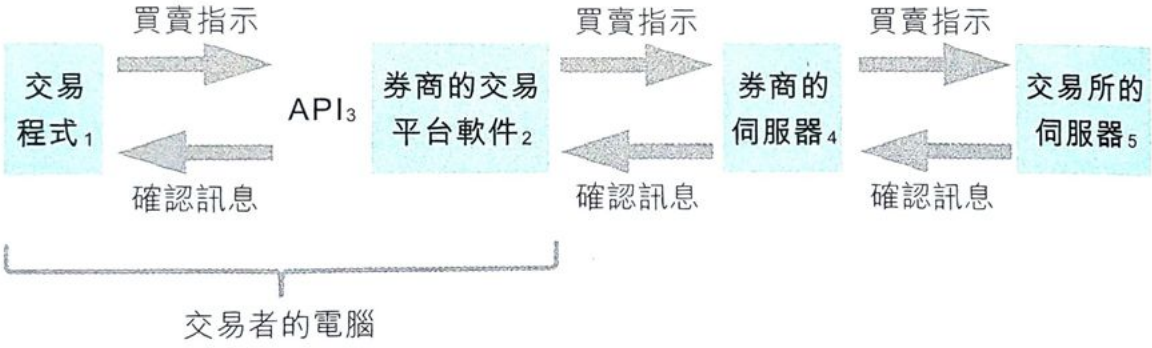


- 交易網路的結構
- 程式語言的選擇
- 第三方程式交易軟件
- 雲端程式交易平台
- 小結

〈 2.1 〉

交易網路的結構

學習程式交易的第一步，是要了解程式和網路的運行結構，先看看一個簡化版的圖表。



名詞說明

(1) **交易程式**：你編寫的策略程式，包含著買賣指令。

(2) **券商的交易平台軟件**：手動交易時所運用的平台，例如，在香港十分流行的 Sharp Point Trader（SP Trader）、Interactive

Broker（IB）專用的 Trader Work Station 等。你可使用鍵盤、滑鼠等輸入工具傳送買賣指示。

（3）**API**：這是應用編程介面（Application Programming Interface），意思是運用程式控制交易平台軟件的協定，例如，TWS 的 Python API 下單指示碼是 placeOrder()（注意大小楷），這是由設計券商交易平台的程式員所設定的，讓其他程式交易者可在自己的程式中，利用 placeOrder() 代替人手操作的滑鼠和鍵盤。因此，自動運作的程式可直接發出交易指示，毋須人手親自輸入。

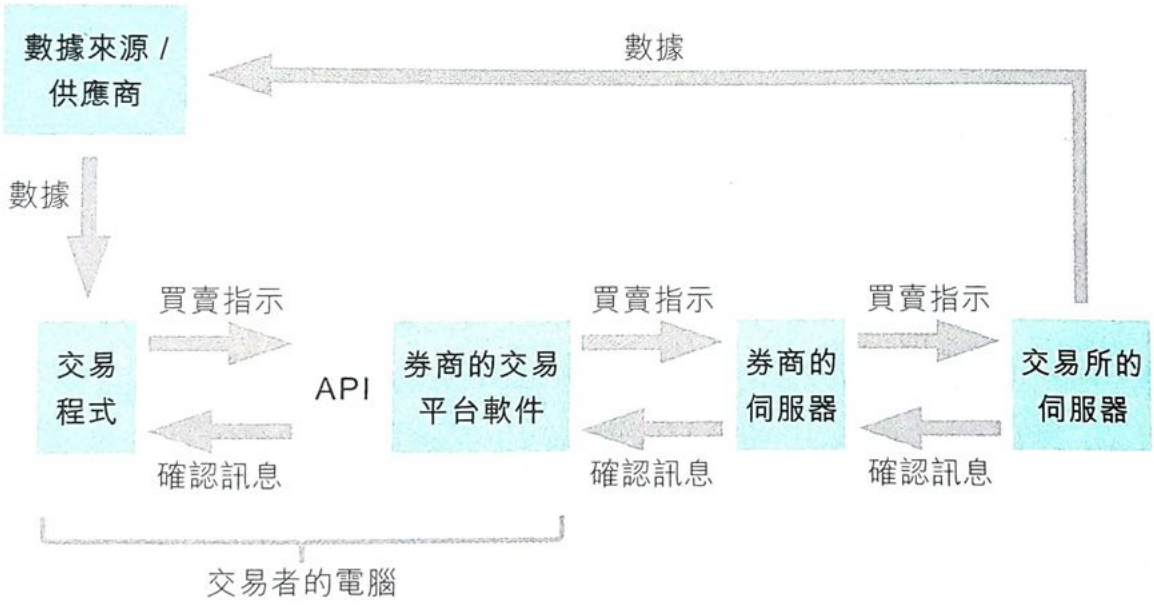
注意：本港大部分券商都沒有開放旗下交易平台的 API，主要是基於監管和安全理由。因此，程式交易者必須選擇有開放交易平台 API 的券商，才能執行程式交易。

（4）**券商的伺服器**：券商是交易者和交易所的中介，主要從事把關工作，例如，檢查交易者是否有足夠的資金進行其交易指示，假設戶口只有 10 萬元，交易指示卻要求買入 10 張恒指期貨，由於沒有足夠按金執行，券商會拒絕這個指示，而該指示亦不會被發送至交易所。

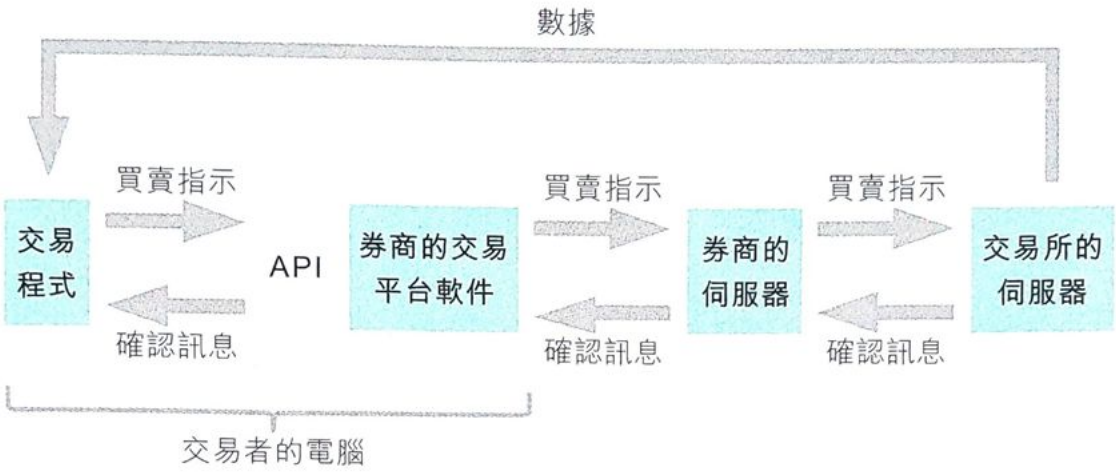
（5）**交易所的伺服器**：這裡就是真正進行交易的地方，除了有來自不同券商的買賣指示，亦會涉及該交易所的各種資產價格。一旦符合條件，交易就會完成，交易所的伺服器會發出

確認訊息到券商的伺服器，再由券商的伺服器送到交易者的電腦，經交易平台軟件傳送到交易者的程式。

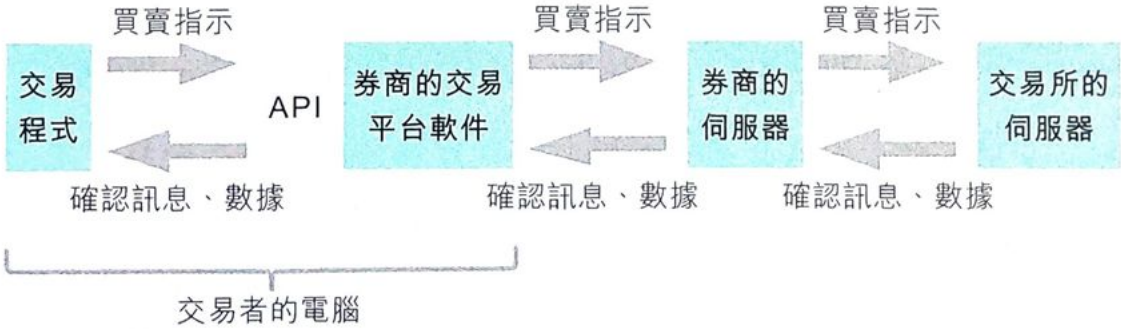
不過，除非交易策略極度簡單（例如，每朝 10 點不問價地買入 1 張期指，這就只需要本機時鐘），否則絕大部分的程式交易都需要採用實時的資產價格來源，因此數據來源（Data Source）在程式交易中就佔據著一大角色，就如下圖：



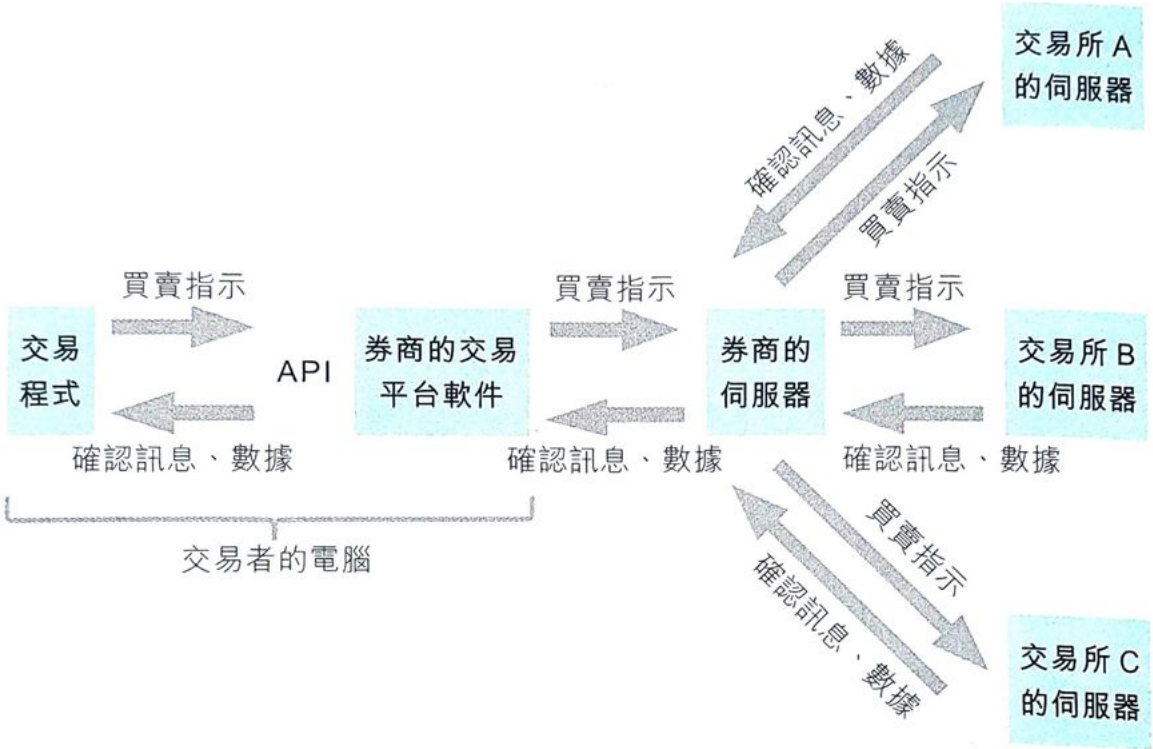
當然，如果你負擔得起交易所的數據專線，上圖的數據箭咀就會直接由交易所指向你的程式，就如下圖：



但是，交易所專線和數據供應商的收費都比較高昂，以港交所 OMD 為例，光是證券報價的牌照費和接駁費已經超過 10 萬元，不是一般散戶能負擔，因此很多散戶都會使用券商提供的數據，而圖表又要改成：



有些交易者不只活躍於一個市場，亦可能同時間在不同交易所進行交易。這個圖表就會改成：



由此可見，程式交易者要掌握的工具主要有 3 個：交易程式、交易平台軟件、數據源。世界上實在有太多種交易平台、太多交易所、太多數據供應商，而本書是為了向初學者展示程式交易的操作，所以本書會從一般香港散戶的角度出發，以最常見的交易軟件 MultiCharts 和少數會開放 API 的證券行 Interactive Broker（下稱 IB）作例子示範，有程式根底的讀者可以尋找其他相對應的證券行作交易。關於 MultiCharts 和 IB 的資訊，會在稍後的篇章中講解。

〈 2.2 〉
程式語言的選擇

對於不懂寫程式的新手，在開始學習時須選擇一種程式語言來學習。程式語言就好比外語，以法文和英文為例，兩者的文法、用字、句子結構等大有不同，彼此不能共通，所以在選擇前應先了解它們之間的分別。

以下是來自 IB 官方網站提供的 API 支援方式列表。

InteractiveBrokers ENGLISH SEARCH LOG IN
HOME WHY IBYR PRICING PRODUCTS TECHNOLOGY EDUCATION

Use the following table to see which API technology best fits your needs.

Access Technology	Performance	Platform
Java	Very robust and reliable; high performance.	Platform-independent
C++ (POSIX-compliant)	Very robust and reliable; high performance.	Platform-independent
Python	Very robust and reliable; high performance.	Platform-independent
.NET (C#)	Very robust and reliable; high performance.	Windows only
C++ (MFC)[1]	Very robust and reliable; high performance.	Windows only
ActiveX[2]	Somewhat robust and reliable (ActiveX can lose events); fairly high performance.	Windows only
DDE	Limited; uses obsolete technologies; lower perfomrace.	Windows only

基本上這個列表可以分為兩類：

第一類：Java、C++、Python、C#

它們都是程式語言，有獨立執行的能力，例如，我們可以編寫一個完整獨立執行的 Java 程式，去控制 IB 的交易平台 Trader Workstation（下稱 TWS）。每種程式語言都是獨當一面，有其特點和優缺點，即使是職業程式員亦甚少可以懂得全部，精通一至兩種已經非常足夠。

第二類：ActiveX、DDE

它們不是程式語言，而是某特定平台上的技術協定，須依賴另一個平台去運作，例如，ActiveX 是微軟的產物，當初設計是嵌入旗下的 Internet Explorer（下稱 IE）瀏覽器內，讓用戶可透過 IE 瀏覽器控制其他程式，所以 ActiveX 必須配合瀏覽器使用。至於 DDE，它是微軟 Excel 動態資料交換技術（Dynamic Data Exchange）的簡稱，用戶可藉此透過 Excel 控制其他程式。

基本上，我們不會使用第二類的技術，因為其限制太大，可靠性亦較低，這一點連 IB 的官方網站都有這樣的說明。因此，值得考慮的是第一類的程式語言。以下是各種程式語言的介紹：

(1) C++

它可說是程式語言的王者，但也是許多初學者的惡夢。筆者的朋友不信邪走去學 C++，10 個有 7 個會半途而廢，其餘 3 個就轉學了 Python。

python

```
print ("hello world")
```

java

```
public class HelloWorld
{
public static void main (String [ ] args)
  {
  System.out.println("Hello,World");
   }
}
```

C++

```
#include<iostream>
using namespace std;
int main()
{
cout<<"hello world"<<endl;
return 0;
}
```

```
Assemply:
global _main
   extern _printf
   section .text
_main:
   push   message
   call    _printf
   add    esp, 4
   ret
message:
   db 'Hello, World', 10, 0
```

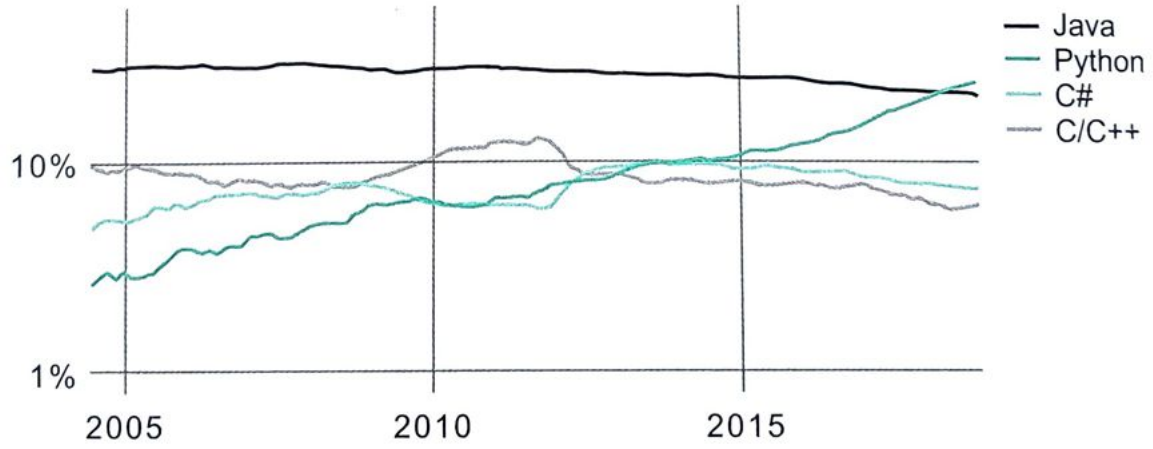
這是由於 C++ 是上述程式語言中最接近底層的語言，可控制的層次亦是最深和最多的，其好處是執行的效率和速度都比較高，所以大多數機構投資者和大戶都會以 C++ 作為其程式交易的語言，然而高速的代價是過於複雜。舉例說，對於一個相同的簡單指令，Python 只須編寫 1 行代碼（Code），Java 須要 3 行，C++ 就須要 5 行；別看輕這幾行代碼的差距，當設計以萬行計的大型程式時，C++ 的複雜性是可以幾何級數地增長，所以會遠遠超越其他語言！

依筆者所見，絕大多數沒有編寫程式經驗的朋友，都嚴重低估學習程式語言的難度，一聽到 C++ 是最高效率的，就衝著去學習，結果不但浪費時間，更令自信心受損，唯一學到的可能是對程式員的尊重。

（2）Java

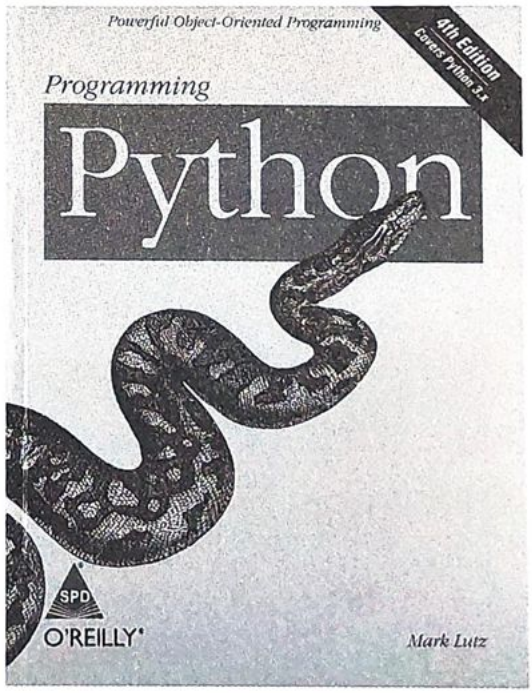
Java 的難度低於 C++，算是 IB API 的入門選擇。筆者的首套自建交易系統都是以 Java 寫成，而選擇 Java 的原因是 IB 在當時仍未支援 Python，若果想程式能夠在 Windows 以外的平台運行，除了 Java 外，就只剩下 C++ 可選擇，但捨難取易的原則下，筆者已二話不說地排除了 C++ 和 C#。

PYPL Popularity of Programming Language



根據 Google 搜尋的統計，截至 2018 年 5 月，Java 一直長時間盤踞著全球最受歡迎程式語言的寶座，直至 2018 年中才被 Python 打破這局面。當一種程式語言極受歡迎，其學習資源也會非常豐富，因為你在編程時遇到的問題，可能已在地球上的某個角落被解決了，只要 Google 一下就能輕易找到答案。由此可見，程式員會強調程式 Community，是因為其豐富的學習資源是其他程式語言難以比擬的。

（3）Python



隨著大數據和人工智能的興起，Python 一躍成為全球程式員的寵兒，很多知名巨企，如 Facebook、Google、Oracle 等，都有大量工作是以 Python 編寫的程式，可見 Python 的出現改寫了整個行業的生態，亦令不少初學程式交易的投資者受惠。

Python 的特點是語法簡潔，把許多累贅的部分去除或自動化，讓程式員專注於程式本身的邏輯。就如早前提過，C++ 要 5 行才能完成的指示，Python 只要 1 行便可，所以若運用 C++

編寫同樣的人工智能發展項目，推遲 4 至 5 年也不足為奇！由於開發效率較高，很多大數據和人工智能的項目都使用 Python。

不過，Python 的運行速度比 C++ 低，但對散戶程式交易者來說，這可能只是百分之一秒的差距。因此，除非要進行高頻交易，否則採用 Python 執行一般頻率的交易，可說是完全沒有延誤問題。

Python 的另一個優點，就是有海量的開源免費擴充程式庫，例如，數學工具 Numpy、資料結構工具 Pandas、製作圖表工具 Matplotlib、甚至深度學習工具 Tensorflow 等，數之不盡，包羅萬有，進一步省卻開發程式的時間。

集合眾多優點，可以預見 Python 在未來依然是全世界最受歡迎的程式語言，難怪 IB 自 2017 年開始都正式支援 Python API，跟上了時代。筆者的期權交易程式是採用 Python 的，這是因為期權的交易程式遠比股票或期貨程式複雜，有很多獨有的變數，如到期日、行使價、Call Put、Long Short，甚至是期權組合等，所以選擇以 Python 作開發語言，可大大提升開發的效率。

(4) C#

與 C++ 一樣，同屬 C 語言家族的一份子，所以語法上有不少相似的地方。但筆者沒有仔細深入研究，因為 C# 有一個非常嚴重的問題，就是只支援 Windows ！

由於 C# 是微軟 .NET framework 下的產物，只支援 Windows 作業系統。雖然有方法在 Linux 和 Mac 系統上執行以 C# 編寫的程式，但這不及其他語言方便。試想想，踏入 Windows 10 後，系統連自動更新仍無法停止，強制於運作中更新及重新開機，是所有電腦用戶的惡夢。因此，筆者絕不考慮用 C# 作為學習程式交易的起步點。

〈 2.3 〉

第三方程式交易軟件

程式語言是一門需要花大量時間才能掌握的學問，即使是被職業程式員認為「最容易上手」的 Python，對毫無編程經驗的初學者來說，依然是門檻非常高的一種語言，而且程式交易牽涉大量與網路有關的技術，光是連接券商 API 已是一大挑戰。粗略估計，初學者要由零開始到完全掌握程式語言及 API 相關的知識，需時起碼 200 小時以上，並未包括實際交易策略的研發，對有正職的業餘交易者來說，這些時間成本可說是非常高。

由於程式語言和 API 知識的門檻實在太高，坊間不少專為程式交易而設的第三方程式交易軟件應運而生。這些第三方軟件通常有以下特點：

(1) API 連接標準化

這些軟件通常內置了不少常用券商的 API 連接設定，例如是 Port Number、IP 地址等，用戶可以輕易設置連線設定。另外，不同券商的數據存取和下盤的 API 會有不同的設定，第三方軟件的內置設定可以讓用戶省略設定的繁瑣功夫。

（2）數據處理標準化

實時數據處理是令程式員相當頭痛的問題，因為真實的實時數據更新頻率是無法預計的，用淺白的語言來說，永遠沒有人能知道下一單的交易是以甚麼價格和在甚麼時間成交，因此程式員必須設法用某個頻率將數據重新採樣，例如把每單交易的 Tick Data（單一成交數據）組合成 1 分鐘的 OHLC（Open High Low Close），再進行分析和運算，這個繁複的工序，第三方軟件會為你代勞。

（3）簡單的程式語言

第三方軟件通常會預留一個讓用戶自由編寫程式的空間。雖然要使用開發商自家設計的程式語言編寫，聽起來好像與傳統程式語言的編輯器沒有分別，但由於第三方軟件已解決了連線、數據處理、回測優化引擎等最花時間的龐大工程，用戶只須集中編寫與交易策略有關的程式。這些程式比起傳統語言簡單得多。舉例說，一個完整可執行的平均線交叉交易策略，第三方平台的語言所需的額外程式行數頂是數十行，但若果用傳統語言編寫，起碼要過千行代碼才能順利執行。

（4）內置回測優化引擎

回測和優化是制定交易策略時必須進行的標準工序，著眼點離不開對風險和回報的統計數字。第三方軟件內置有回測和優化引擎，能自動計算各種策略的風險和回報，並以圖表方式展現給用戶，若果只用傳統程式語言制作回測和優化工

具，將會是相當龐大的工程，絕非 3 數天之內可以完成的工作。因此，第三方軟件的內置回測和優化引擎可以減少開發時間，讓用戶集中精力制定策略。

其實，第三方軟件是由一整隊職業程式員所開發，初學者要自製出比第三方軟件更強的工具是不可能的，因此使用第三方軟件可節省的時間是以年計的，也絕不誇張。

然而第三方軟件始終有其缺點和局限，使用前必須先注意：

（1） 自由度的限制

自由度是指用戶可以控制的空間，要是第三方軟件沒有設定的功能，用戶就難以控制，又或者軟件本身的某些功能有強制的設定，用戶便無法更改。這些問題其實是不嚴重的，但若果用戶使用時沒有正確了解，可能會導致交易上的損失。

（2） 欠缺期權相容性

除了期貨之外，幾乎其他所有衍生工具都難以利用第三方軟件進行程式交易，這是由於第三方軟件多數以股票、期貨的角度設計，但期權會涉及多個行使價和到期月份的選擇，甚至期權組合須用上多個期權倉位作組合，難以使用第三方軟件。因此，期權交易者只能以傳統程式語言配合券商的 API 作期權程式交易。

(3) 執行效率略低

比較以傳統程式語言編寫的程式，第三方軟件的執行效率永遠有所不及，這是由於第三方軟件始終是商業軟件，單一程式內會包含大部分用戶所需要的功能，如果要使用上自家設計的程式語言，更要經過多一重編譯，速度自然比不上完全自製的程式。幸好現時很多第三方軟件已經在執行效率上進行優化，散戶使用起來完全沒有問題；但對高階的用戶來說，若果要執行複雜和高速的交易，尤其是在幾毫秒內決勝負的高頻交易領域，就不可能利用第三方軟件了。

市面上流行的第三方軟件

現時香港比較流行的第三方軟件有 MultiCharts、Amibroker、MetaTrader，其實除了這幾款外，還有很多其他選擇，例如即將會提到的 TradeStation，可惜程式交易在香港仍未算流行，所以主流的第三方軟件一般只有這幾種。

(1) MultiCharts

MultiCharts 可說是香港程式交易界中最主流的第三方軟件。不少香港的程式交易員會以 MultiCharts 進行期貨程式交易。這套軟件在 2005 年面世，當時全

球領先的第三方程式交易軟件為 TradeStation， MultiCharts 的創辦人本想為 TradeStation 加入更多功能，但發現與其不斷修改 TradeStation，不如打造一個全新的軟件。

MultiCharts 是一個收費軟件，終身費用為 1,497 美元，略貴於其他第三方軟件，但 MultiCharts 的 Power Language 程式語言結構與 TradeStation 的 Easy Language 有超過 95% 相同，所以 MultiCharts 承繼了 TradeStation 的龐大用戶群。此外，當用戶遇到任何問題，幾乎都可以在 Google 找到答案，這是對用戶的極大優點。

香港有不少券商會向客戶提供免費的中國版 MultiCharts，雖然版本較國際版落後最少兩代（執筆時國際版為 MultiCharts 12，中國版則為 MultiCharts 9），而且中國版只有簡體中文介面，沒有英文和繁體中文，但兩者的核心功能大同小異，所以初學者可藉此節省一筆軟件費用。但正所謂「羊毛出自羊身上」，券商所提供的中國版 MultiCharts 必須透過該券商進行交易，而交易佣金一般會較高，所以對交易頻率較高的用戶而言，國際版會比較划算。其實，國際版每年於感恩節、聖誕節、復活節都有減價優惠，有興趣的讀者可以留意。

(2) Amibroker

要數 MultiCharts 的最大宿敵，當然非 Amirboker 莫屬。這

兩款第三方軟件經常被拿來比較。基本上，MultiCharts 有的功能，Amibroker 都有，你有 walk forward testing ？我跟！你出 3D 分析圖？我又跟！你出 Monte Carlos Stimulation ？我照跟！你竟可支援多核心優化？我唯有立即跟！

所以，兩者的比較其實只有兩點。第一是價錢，Amibroker 比較便宜，最平的 Standard Edition 只需 279 美元，最貴的 Ultimate Pack Pro 亦只是 499 美元，相比感恩節的特價 MultiCharts 還要平一截。筆者當年還是窮苦學生，第一套購買的第三方軟件就是 Amibroker Ultimate Pack Pro，折實不用 4,000 港元呢。

第二是用戶群。MultiCharts 的優勢始終是其龐大的用戶群，這一點在華文地區尤其明顯。早前，MultiCharts 在中國大陸和台灣的代理商進行了大力宣傳和教育，網上又有不少以中文撰寫的網友分享，方便懶得閱讀英文的用戶；相比之下，Amibroker 的社群雖然也不小，但仍難以與 MultiCharts 的社群匹敵，用戶可能要花更多時間在網上尋找資料，可見平價費用的背後，是「魚與熊掌不可兼得」的道理。

(3) MetaTrader

有別於主要用於期貨的 MultiCharts 和 Amibroker，MetaTrader 主要用於外滙和

商品交易，這款第三方軟件費用全免，所以大部分外滙交易商都很樂意向客戶提供支援。

MetaTrader 內置有一個特別的訊號訂閱功能，可以讓用戶訂閱其他用戶的交易訊號，甚至透過「Copy Trading」的功能，自動按其他用戶的交易訊號執行交易，這些訊號有的會免費發放，亦有的需要收費；反過來說，用戶亦可以申請成為訊號提供者，向其他用戶收取訂閱費。不過大家要注意，選擇訂閱訊號是一門大學問，而且 Copy Trading 在香港屬於證監會監管範圍的灰色地帶，運用上有一定的風險。

現時，MetaTrader 有 MT4 和 MT5 兩個主流版本。雖然 MT5 是最新版本，但由於相比 MT4 的改動太大，不少用戶仍堅守 MT4 陣型，所以新舊版本的支持者數量不相伯仲。但隨著硬件的發展，不支援 64bit 版本的 MT4 會逐漸被淘汰。

程式語言方面，MetaTrader 使用的 MQL5 語法接近 C++，難度遠高於 MultiCharts 的 Power Language 及 Amibroker 的 AFL，一個以 MQL5 寫成的策略，其程式行數過千行都不足為奇，所以不可稱之為簡易，而花在學習 MQL5 的時間已足夠學懂 Python，可見其學習難度之大。

〈 2.4 〉
雲端程式交易平台

剛才介紹過幾種主流程式語言，大家可能心想要用 Python、C# 等語言編寫交易程式，必須完全靠自己建立系統，其實除了直接經 API 接駁券商之外，還可以利用雲端程式交易平台。這種平台的自由度和編寫難度介乎直接自建系統與第三方軟件之間，對一些已有一定編程經驗的人士來說，它是一個不錯的選擇。

顧名思義，雲端程式交易平台所提供的開發環境類似第三方交易軟件，只是由於雲端平台技術更高，需要使用傳統的程式語言，如 Python、C# 等。此外，透過雲端伺服器亦有優勢，包括為用戶帶來高速、高穩定的開發環境。因此，不少有編程知識的初學者都會從雲端平台開始上手。而隨著技術不停開發，以前的平台只提供回測和優化的運算力服務，現時已演變到可以直接接駁證券行的 API 作真錢的自動交易，功能上可說是大包圍。

雲端程式交易平台的特點和好處：

（1）高速回測及優化

雲端平台的伺服器處理速度遠超一般桌上電腦，運用雲端系統可以大幅減少優化所需的時間。

（2）24 小時運作

除了速度之外，穩定性也是雲端伺服器的強項之一。許多桌上電腦可能遇到的問題，如寬頻斷線、停電、Windows 強制更新等，都不會在雲端平台遇到。即使用來連接雲端的桌上電腦關機，雲端平台都可以繼續運作。

（3）強大社群

最受歡迎的雲端平台（如 Quantopian、QuantConnect 等）早已累積一大班用戶，不少熱心的用戶設計了許多實用的程式和模組，讓新加入的用戶可以快速上手。

（4）免費 / 便宜數據

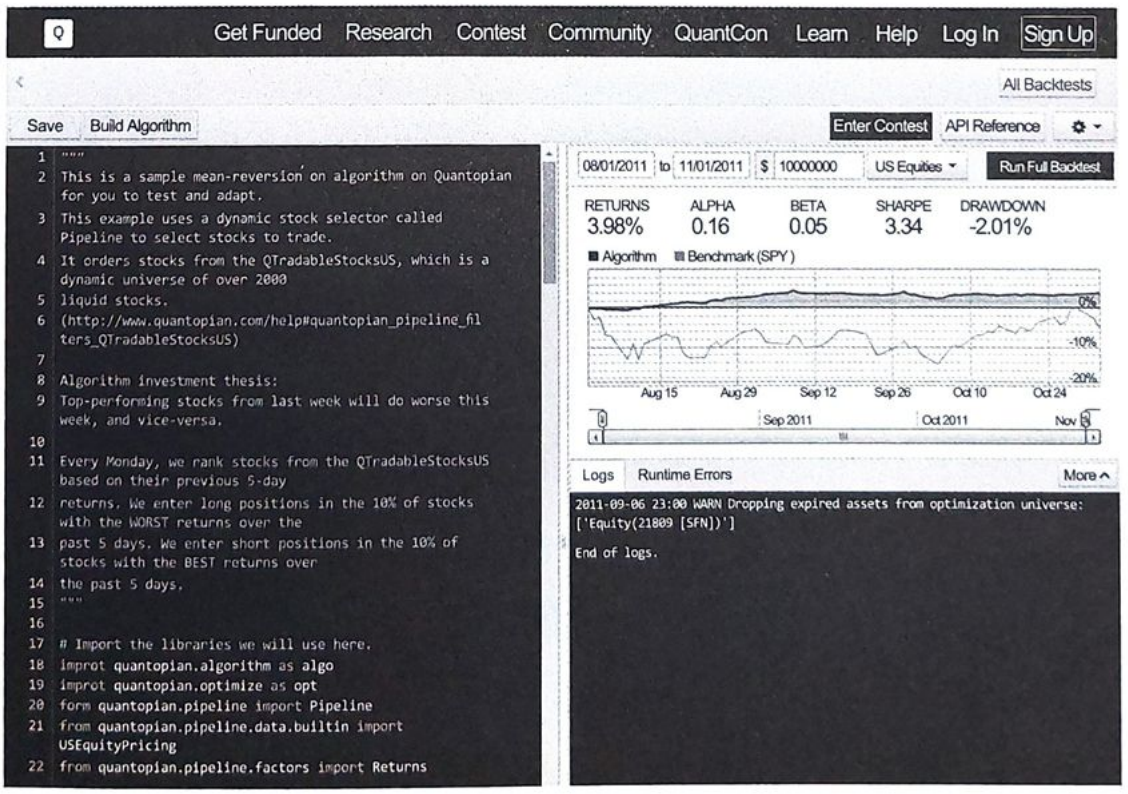
為了照顧交易者的需要，雲端平台會提供非常便宜的歷史數據、即市數據作回測和交易之用。因此，用戶可省卻找第三方數據供應商的麻煩。

唯一缺點：欠缺香港市場數據。

在全世界的程式交易界別中，香港市場就好像一個孤島，美國、中國市場都已有很多開源、免費的數據包、模組等工具，但香港市場就仍然缺乏。雲端平台的情況也是一樣，用戶只有外國的股票、期貨等數據，而偏偏外國市場有最多高手參與，香港人要取得交易優勢的難度亦較高。

平台介紹

(1) Quantopian



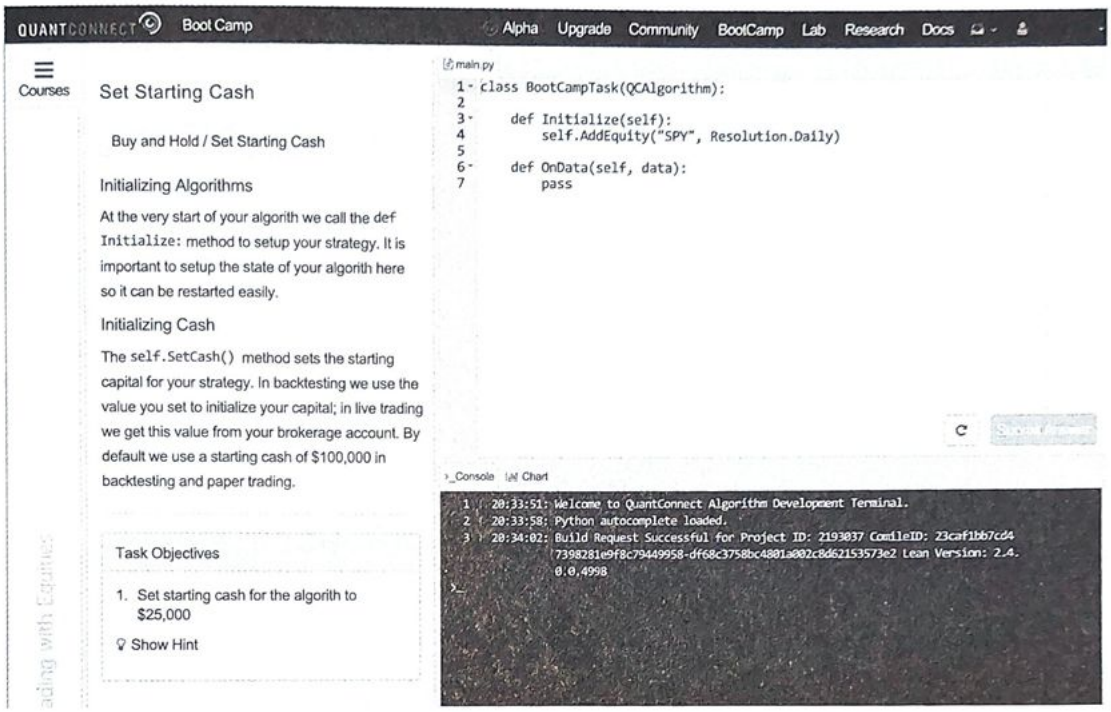
所有雲端程式交易平台中，最著名的就是 Quantopian。它早在 2011 年創立，可說是雲端程式交易平台的鼻祖，在 2015 年世紀經濟論壇中，Quantopian 被認為是「金融投資管理功能中的重大金融創新之一」，創辦人 John Fawcett 和 Jean Bredeche 早期於專為大量對沖基金開發軟件的公司工作，因此 Quantopian 系統內幾乎都是專業級的功能，技術上處於行內頂尖的位置。

Quantopian 有 3 大優勢，包括龐大的用戶群，大部分免費數據，以及定期舉辦名為 QuanCon 的程式交易比賽。重點是第三點，因為勝出者可以獲得 Quantopian 出資資助，讓該程式可以真錢作交易，產生的利潤更會和勝出者分成，所以它吸引了全球不少高手踴躍參與。最著名的成功例子是一名在德州的 21 歲大三學生 Spencer Singleton，贏得了 Quantopian 的 10 萬美元獎金後，搖身一變成為了量化基金經理。

唯一可惜的是，自 2017 年開始，Quantopian 全面停止支援自動交易，用戶不能夠透過 Quantopian 連接證券行的 API。這算是 Quantopian 平台的最大缺點。雖然官方沒有明言將來會否重啟自動交易，但不少人相信因為 QuanCon 辦得太出色，所以現時 Quantopian 只全力發掘有潛質的用戶和策略，自動交易的重啟則暫時沒有時間表。

順提一提，Quantopian 所使用的程式語言是 Python。

(2) QuantConnect



接著出場的是 QuantConnect。比起 Quantopian，它的名氣較小，但算是雲端程式交易平台的二哥。基本上，QuantConnect 都有 Quantopian 的功能。最大不同的是 QuantConnect 沒有舉辦像 QuanCon 的比賽。不過，QuantConnect 設立了名為 Alpha Streams 的策略交易所，用戶可以直接提交自己的策略，被看中的優秀策略會供其他使用者使用，再與提供策略的用戶分成利潤。

另外，QuantConnect 支援的程式語言比 Quantopian 多，現時有 Python、C#、F# 等，讓熟悉 .net framework 的程式員都可以輕鬆加入。至於支援的證券行，則有 IB、FXCM、Oanda、GDAX 等。自從 Quantopian 停止支援自動交易後，不少用戶都過檔到 QuantConnect，使其人氣在近年一時無兩，成為雲端程式交易平台的熱門選擇之一。

〈小結〉

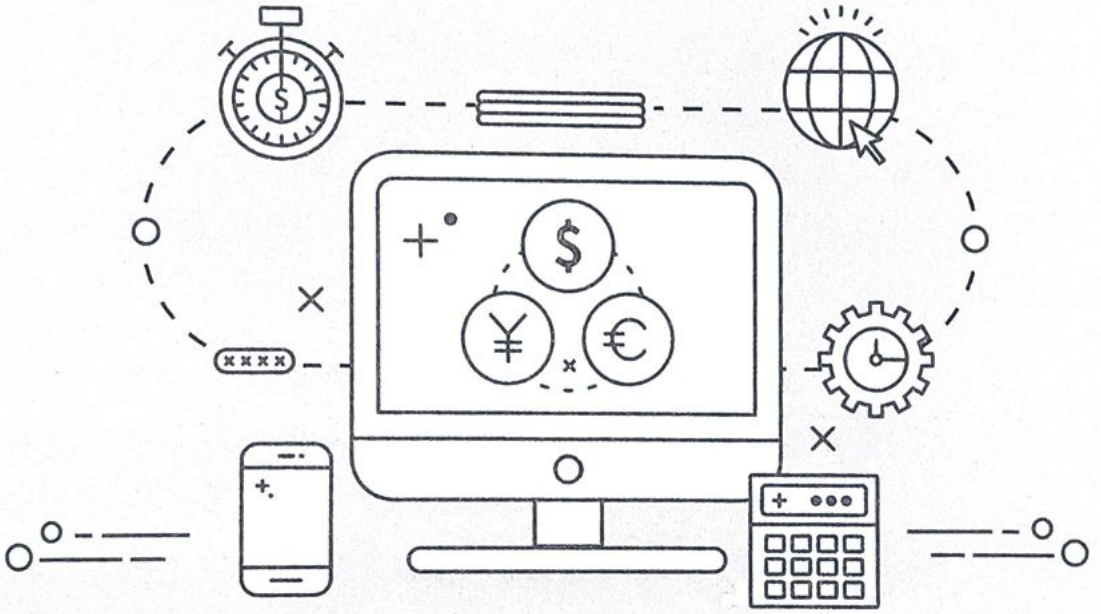
在本章裡，筆者介紹了程式交易的基本網路結構、程式語言的種類、第三方程式交易軟件和雲端程式交易平台。讀者應該能掌握到程式交易網路的基本運作原理、API 的角色、以及程式的角色等。

對已經懂得編寫程式的讀者而言，可以先閱讀券商的 API 文件，看看該文件的難度會否太高，又或者嘗試由最簡單的連線設定開始，感受親自編寫的實際難度，再決定是否編寫自製的交易程式，或是使用雲端平台作起步點。

至於沒有編程經驗的讀者，筆者強烈建議初學者由第三方程式交易軟件開始學習。筆者見過太多人嚴重低估編寫程式的難度，只是在學習傳統程式語言的路上就很快放棄。其實，在學習第三方軟件的程式語言時，初學者都可以建立編寫程式的基本知識，並對程式架構有更深入的認識，所以第三方軟件是一個相當不錯的踏腳石。直到第三方軟件再難以滿足你的資金量或操盤方式時，才去學習傳統程式語言都未遲。

在接下來的篇章裡，筆者會以 MultiCharts 為例子示範，但當中的交易觀念不只限於一個平台之內，有興趣的讀者可以其他程式語言或第三方軟件作測試。

第 3 章
系統化交易步驟



- 交易標準步驟
- 回溯測試
- 優化
- 小結

〈 3.1 〉

交易標準步驟

先別急著立即寫 Code，學習程式交易之前，必須先對交易有根本的認知，否則即使你能寫得一手好 Code，都無法掌握程式交易真正的優勢。

以下是程式交易的標準步驟：

(1) 構思交易策略 → (2) 回測和優化 → (3) 以小量資金 Forward Run → (4) 執行 → (5) 定期檢視執行結果

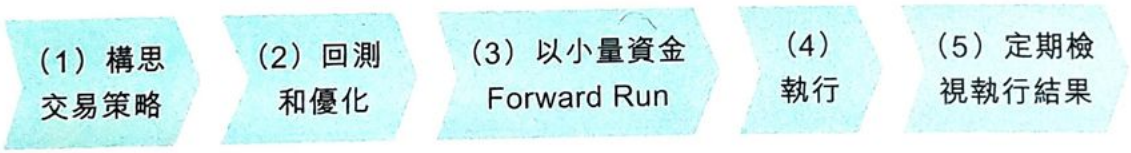
(1) 構思交易策略

構思交易策略的方式有很多，但出發點離不開「觀察」二字，例如，透過觀察股價的變化、不同版塊股票價格之間的關係、某時段的股價特性、技術指標的表現等。不少人會為此窮一生的精力，可見構思交易策略是整個步驟中最難、也是最初的一環。

〈 3.1 〉
交易標準步驟

先別急著立即寫 Code，學習程式交易之前，必須先對交易有根本的認知，否則即使你能寫得一手好 Code，都無法掌握程式交易真正的優勢。

以下是程式交易的標準步驟：



(1) 構思交易策略

構思交易策略的方式有很多，但出發點離不開「觀察」二字，例如，透過觀察股價的變化、不同版塊股票價格之間的關係、某時段的股價特性、技術指標的表現等。不少人會為此窮一生的精力，可見構思交易策略是整個步驟中最難、也是最初的一環。

當然，觀察不能只靠肉眼，但肉眼觀察所帶出的靈感往往有不少啟發性，因為人類的腦部擁有強大的模式辨認能力，不少人手交易時用到的方式，如 Momentum、Price Action 等，都可輕易憑肉眼辨認，並為制定策略帶來靈感。

如果交易者有其他分析工具協助，可以觀察到更多細節，從而令策略有更大的變化。即使是運用簡單的 Excel 工具，也可以尋找到價格變化的季節性、波幅的特性、與其他股票的互動關係等；針對這些特性，再制定出對應的策略，例如波幅操作、配對交易等。

總而言之，構思交易策略是指從觀察中想出的合適的交易方法。在未進行（2）回測和優化之前，任何策略都值得考慮，直到證實回測和優化行不通，才須重新思考，即是構思、證實失敗、再構思、再證實失敗，這過程會佔據整個步驟的大部分時間。而這步驟的難度在於沒有既定方法保證能令你找到交易靈感，因為每個人的觀察力、悟性、智商都有不同，你觀察到的別人不一定觀察得到。在金融市場，真正致勝的一步往往就是這一步，而往後的回測、優化、執行等步驟則是 Hard Knowledge，只要付出努力就總會學懂，但交易點子的誕生是最難捉摸的，所以也是最珍貴的。

（2）回測和優化

當你想出一個交易點子後，便要進行回測（Backtest），例如，當你觀察到某股票的價格出現超賣反彈，你想知道 RSI 是否一

個評估超賣的有效指標，於是你可以找出所有 RSI 跌穿 30 的價格紀錄，並計算該水平博反彈的贏面有多大，然後比較命中時的獲利有多少，以及失敗時的損失有多少；如果預期的利潤大於預期的虧損（就如下方的不等式），這策略就基本上是可行的。

命中率 × 命中時利潤 > 失敗率 × 失敗時虧損

透過回測，你會發現命中率不是衡量策略優劣的最關鍵因素。某些策略的命中率很高，但失敗所產生的虧損卻可以輸掉之前的所有利潤；某些策略命中時的獲利叫人驚訝，但失敗率卻奇高。這些情況都要透過回測才能發現。另外，策略的止賺和止蝕亦會直接影響以上不等式中的 4 個因素，所以止賺和止蝕的方法可以改寫策略的選擇。

一個可行的策略不一定有良好的回測表現，但起碼要展現出有改進的可能性，出現嚴重虧損的策略就必須排除。若果交易頻率不高，回測結果卻顯示嚴重虧損，可測試一下方向相反的策略。例如，原本的策略是在某股票的 RSI 跌穿 30 時博反彈，而方向相反的策略就是 RSI 跌穿 30 追沽獲利，若這策略能通過回測，就能發現對於該股票，RSI 屬於順勢（Trend Following）的訊號，這也是一個不錯的發現。

優化（Optimization）是把訊號的參數改變為不同數值再作多次回測，而這裡的「多次」可能高達百萬次，視乎參數組合的數量。每個參數組合的回測結果都會被紀錄，藉以找出成效最好的參數組合。以剛才 RSI 為例，RSI 的數值範圍由 0 到 100 都可以做回測，而 RSI 所使用的數據量預設為 14，亦可以由 10 至 50 再作測試，更重要的是止賺和止蝕的數值，究竟應該止賺多於止蝕，還是止蝕多於止賺？這些答案都可以透過優化找到，而成功的優化結果除了能增加獲利外，更可以減低執行中所承受的風險。

回測和優化絕對是程式交易中的重要課題，尤其是優化中的 Curve Fitting 問題，更須加倍注意。我們將在往後的篇章詳細講解。

（3）以小量資金 Forward Run

即使交易點子的回測結果是正面，而優化的結果又能令表現大幅改進，大家亦千萬別急著 All-in，因為在程式交易中，我們須運用到許多在生活中少見的邏輯思維，其中一樣就是「未知的未知」（Unknown Unknowns）。

「未知的未知」是前美國國防部長拉姆斯菲爾德在 2002 年回應記者提問時的名言。當時，由於美國未能拿出伊拉克政府擁有大殺傷力武器的證據，這句言論廣受批評，但實際上，這個名詞也有其意義。從下圖的矩陣中可見，世上所有事物可分為 4 種，分別是

「已知的已知」、「未知的已知」，「已知的未知」和「未知的未知」，前兩項都本質都是「已知」，而後兩項的本質就是「未知」。從風險管理的角度來看，風險的本質是不確定性，亦即是「未知」，所以我們的著眼點應該是後兩項。

	known	unknown
known	known knowns	known unknowns
unknown	unknown knowns	unknown unknowns

「已知的未知」是指一些已知會變化的因素。滑價就是其中的例子，在開始交易前，我們已知滑價會隨著市場的流動性而變化，雖然我們無法在事前知道會滑價多少，但總可以估計滑價的範圍，從而及早作出對應方法，因此屬於「已知的未知」。

至於「未知的未知」，是指一些在事前無法預計的因素，通常是程式碼在運行回測與真錢交易存在分別所致。但是，我們就連這項「未知」是甚麼都不知道，要直至真正交易開始後，那些因素才會浮現。筆者在初學程式交易時就遇過，一行程式碼可設定作收市前平倉，以為可用於真錢的 DayTrade 交易，豈料在真錢運作時，卻發現這行程式碼不單沒有在收市前平倉，反而把倉位留到第二朝

開市才平倉，仔細研究後才發現該程式碼原來只能用作回測，不能用於真錢交易，害我損失了不少。

對於這種無法在事前預計的風險，我們只能用小量資金作測試，其用意是先透過測試把未知的損失控制在較小數目內，一旦發現問題就立即修正，直到運行順暢後，才逐步把資金增加至計劃中的水平。

（4）及（5）執行及定期檢視執行結果

經過了前 3 步，你的程式已算順利跑起來了，但這不代表之後可以放任不理，始終程式操控著你的血汗錢呢！因此，當資金增加到原先計劃的水平後，仍必須定期檢視執行結果，尤其是當交易策略是建基於某些假設（Assumption）之上，就必須注意在逆境時的表現。

舉例說，大部分程式交易的邏輯是建基於順勢和逆勢（Swing Trading）之上，即使程式中有判斷股價將走單邊或走區間，這判斷設定總會建基於某些假設之上，所以一旦這些假設不再成立，程式的表現可能會一落千丈。可怕的是，當程式的利潤下滑，我們亦未必能分辨出這是正常的回撤（Drawdown），或是假設的失效，因為要判斷假設是否失效，我們往往要等待虧損擴大才能證實。因此，我們必須為策略定下應變計劃，在未能確認假設失效時，定下整套策略的終極止蝕位，這一點將會在風險管理的篇章中作詳細解釋。

這步驟的重點是，在程式的實際運作中，你總會遇到很多在回測時未能遇到的情況，這是由於歷史不是簡單地重複的。因此，我們必須在事前早作心理準備，以及制定應變策略。特別是程式運行了一段時間後，出現某些日子的表現與平日有較大出入，在這情況下，判斷程式的去或留就是交易員必須面對的決策問題。

別看輕這套交易標準步驟，許多投資者花費多年時間，都未必會有系統地思考交易，而程式交易者就因為要編程的關係，必須把所有的步驟系統化。由此可見，要成為程式交易員不單要學寫程式，連帶看待事物的方式也要改變，這對交易員來說絕對是一個好事。

〈 3.2 〉
回溯測試

回溯測試（Backtesting，簡稱回測），指運用電腦模擬測試某策略在過去某段時間內應用於某金融產品的表現。我們只須用程式定義策略的各項條件，設定需要測試的金融產品和時間範圍，電腦就能為我們計算出各項風險和回報的數據。

在進行回測時，我們須注意以下 3 點：

（1） 策略有客觀的定義

客觀定義策略是程式化的關鍵，很多技術指標和 Price Action 的策略都符合這個條件。不過，坊間有很多難以客觀定義的策略，它們就難以化為程式。例如，數浪、分辨型態和背馳等以肉眼辨別的訊號都是過於主觀，難以編程式的例子。

（2） 回測的時間範圍

如果時間範圍太短，所得結果缺乏代表性，例如，只採用過去兩年的數據，若果股市在該兩年間處於牛市，做好倉的策略自然表現優異，但這不代表該策略在未來都會一直領先。

但盲目地加長時間範圍亦不一定更好，舉個誇張的例子，若你採用過去 30 年的恒生指數期貨數據，當中最舊的數據肯定與現時的價格表現相差甚遠，因為在這 30 年間，市場環境、投資者組成、恒指成份股等因素早已面目全非，加入那些舊數據反而有害。因此，選取的時間範圍是一個中間落墨的學問，太短或太長都不合適。

(3) 數據運用的邏輯性

由於回測的過程是運用金融產品的歷史數據，在寫程式存取數據時，部分初學者可能不自覺地運用了「未來」的數據，嚴重違反當中的邏輯。例如，策略是於每天交易日結束時，存取了第二天的開市價，交易條件設置為「明天開市價高於今天收市價便買入」，這樣寫的話恐怕不可能蝕錢吧，但在現實中，我們怎可能存取明天的開市價？這種邏輯陷阱看似無稽，但當策略愈來愈複雜時，即使是老手都有可能不小心犯上這個錯誤。

回溯測試例子

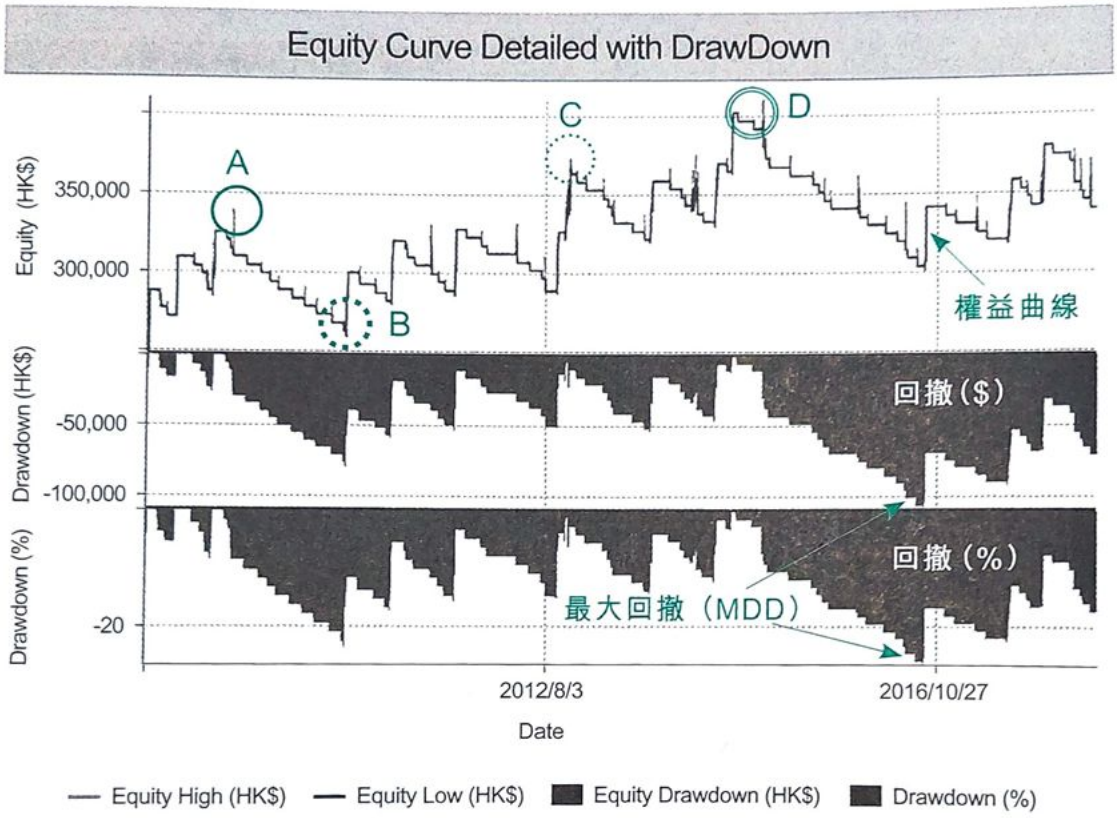
舉個實際例子，筆者用 Power Language 編寫了一個簡單的 MACD 策略，用上了傳統的 12-26-9 參數，並設置了 100 點止蝕、750 點止賺，程式碼如下：

```
1  input: FastLen(12),SLowLen(26),MACDLen(9),SL(100),SPT(750);
2  var: macd_var(0),macd_xaverage(0),diff(0);
3
4  macd_var = MACD( Close, FastLen, SlowLen ) ;
5  macd_xaverage = XAverage( macd_var, MACDLen ) ;
6  diff = macd_var - macd_xaverage ;
7
8  condition1 = diff cross over 0;
9
10 if condition1 then
11      buy next bar at market;
12
13 setstoploss(SL*bigpointvalue);
14 setprofittarget(SPT*bigpointvalue);
   Col 60    Ch 60  SAVED  COMPILED
```

可以看出，這段程式碼相當簡單，前段設置了所需要用到的參數，中段設置了計算 MACD 的方式，最後段設定符合條件時的買入恒指期貨動作，以及止賺止蝕的設定。

一個交易策略是否必須設定止賺止蝕？未必，若果策略中有反手的設定，於某條件觸發時直接由好倉變成淡倉，再由淡倉變成好倉，就能做到不斷交易的效果。但若果沒有反手機制，又沒有止賺止蝕，該策略便是一個長期持倉的策略，就如上面的 MACD 策略，如果沒有設定止賺止蝕，在第一個動作出現後，就會永遠持倉下去，變成 Buy and Hold 了。

運用 2008 年 6 月至 2018 年 6 月的數據，得出的結果如下：



這是程式交易者最常看到的回測結果，畫面顯示有 3 個部分，最上方的圖表是權益曲線（Equity Curve），反映倉位的資產水平，所以權益曲線向上代表策略在過去賺錢，向下則代表策略蝕錢。左邊的 Y 軸顯示筆者設定的起步點為 $250,000，你可因應自己的需要自行設定起步的金額。

中間和下方部分同樣代表風險的回撤（Drawdown），分別只是中間部分以銀碼作單位，而下方部分則以百分比作單位。回撤的

定義有點複雜，是「到達該時間點時，曾經出現過的權益曲線最高位，與現時權益水平的差距」，例如上圖的時間範圍是 2008 年 6 月至 2018 年 6 月，當時間線（X 軸）至 2009 年 5 月，權益曲線創了當時的高位 $330,000（A 點），然後開始拾級而下，直到 2012 年 11 月才再創新高（C 點）。因此，由 2009 年 5 月至 2012 年 11 月，回撤都會以 2009 年 5 月的高位計算，即 $330,000。舉例說，在 2010 年 7 月（B 點），當時的權益水平是 $262,000，所以回撤（$）和回撤（%）分別是 $68,000 和 20%。

每當權益曲線創新高，回撤的計算標準都會改變，例如，由 C 點到 D 點的範圍，回撤都會以 C 點的權益計算，而從 D 點開始，回撤就會以 D 點的權益計算。因此，隨著時間推進和權益曲線創新高，相同的回撤百分比未必代表相同的回撤金額，例如，當權益曲線的前高位是 $330,000，回撤 10% 所代表的金額只是 $33,000，而當前高位推高至 $660,000，只需 5% 便可代表同樣金額的 $33,000。由於基數的改變會導致回撤百分比與金額出現差異，使用時必須清楚了解當中的計算方法。

這個簡單的回測結果可展現所有交易策略中最重要的兩個元素：回報和風險。回報可由權益曲線代表，曲綫愈是向上斜，代表策略的賺錢能力愈高；而風險則由回撤代表，回撤愈少代表倉位所承受的風險愈低。

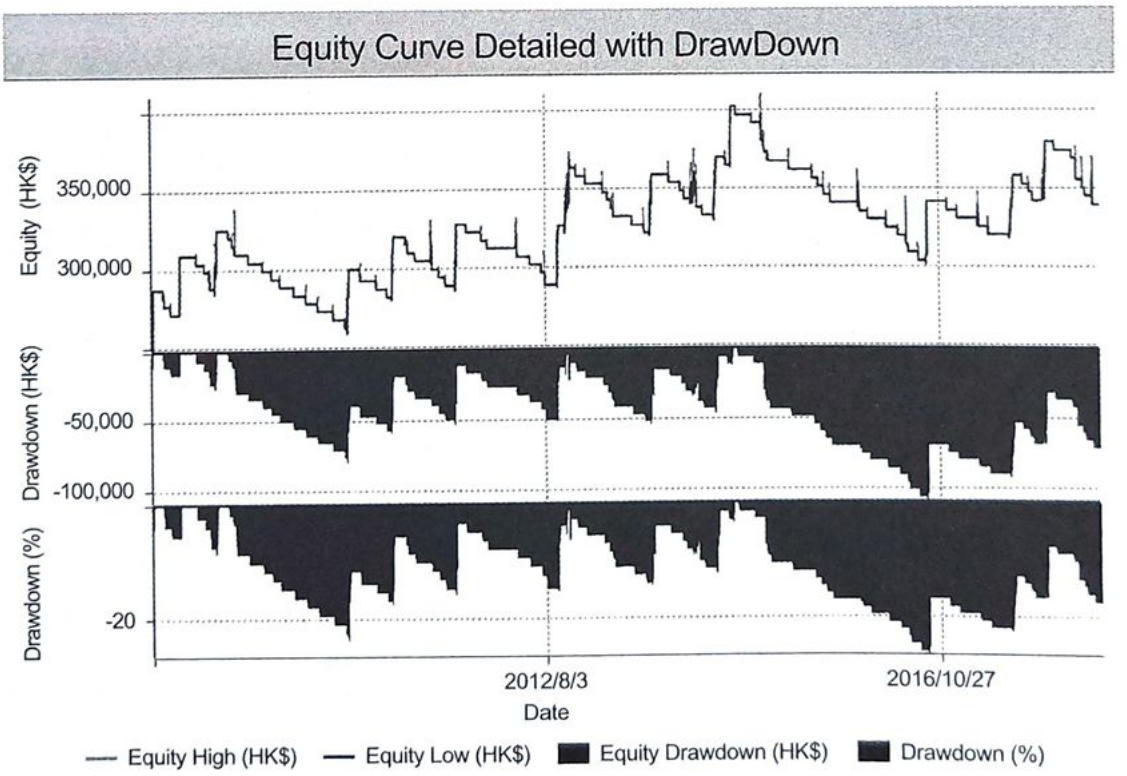
當然，這裡的回報和風險是以歷史數據計算，與未來的表現沒有絕對的必然性。這一點非常重要，而且單憑權益曲線和回撤亦難

以代表整體情況。因此，交易者發明了許多深入分析風險和回報的方法和工具，筆者會在稍後的篇章作深入講解。讀者在現階段只須了解回測的目標和形式就可以。

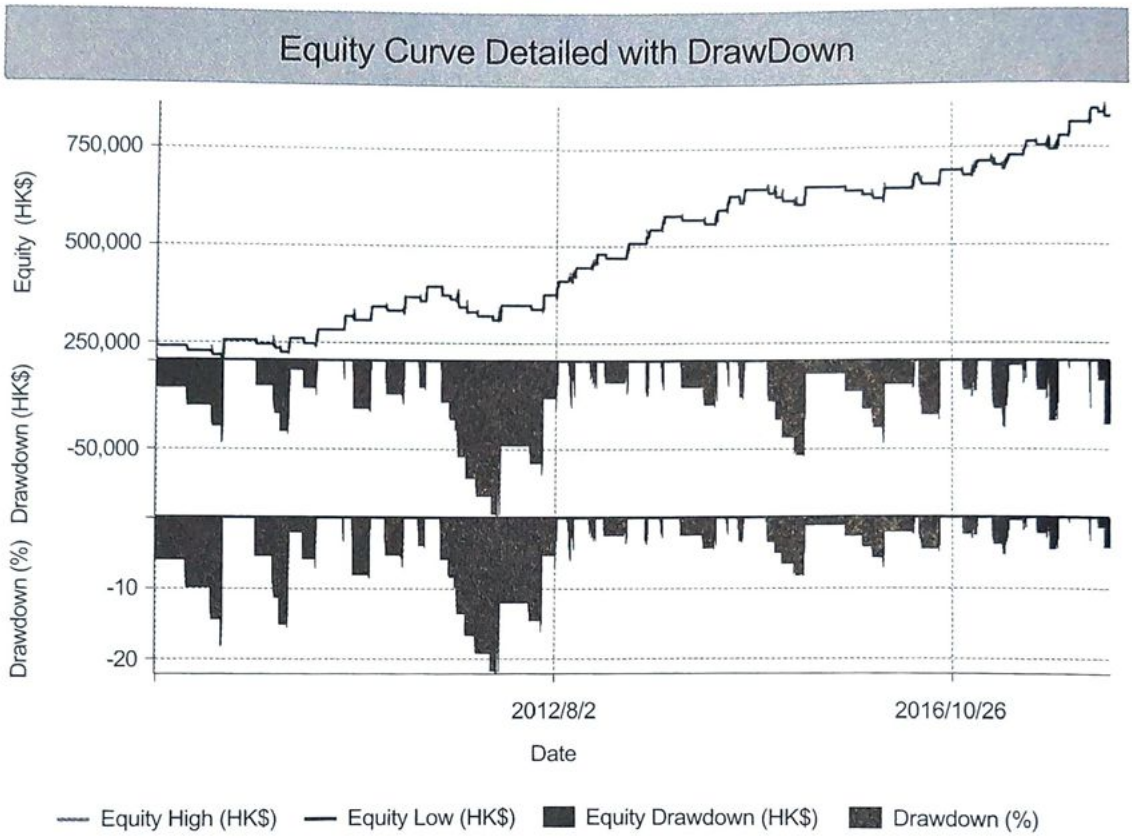
〈 3.3 〉
優化

優化是指用不同的參數組合進行多次回測，可以説是回測的強化版。若以編程的語言來説，就是回測加上 For Loop 而已。透過測試不同的參數組合，我們可以得知哪組參數可以令策略有最佳的表現。

優化前



優化後



以先前的 10 年恒指期貨 MACD 策略為例，如果採用的是傳統的 12-26-9 參數，10 年的總利潤只有 45%，若以複利效應計算，平均每年只有約 4%，回報表現只屬一般。而在優化後，將 MACD 的參數和止賺止蝕的點數都改變，則可以找到一個表現更佳的組合，令策略的回報大幅提高。

優化看似簡單，只要明白回測的原理就自然明白優化是甚麼一回事。但在實際使用上，大家仍然有以下 3 點要注意。

(1) 過度優化

優化不是神兵利器，反而可能是程式交易者的糖衣陷阱。程式交易者最怕是優化後的策略只是巧合而成的美好幻象，這問題可稱為過度優化（Overfitting / Curve Fitting）。這現象是因為某些參數在歷史數據中巧合地捕捉到最低位買和最高位沽的時機，所以獲利只屬巧合，並不是歸功於策略的交易邏輯。而由於歷史不會簡單的重複，這種巧合而成的參數在未來不一定能捕捉到相同的價格特性，導致實際交易和回測結果出現重大的落差。

(2) 優化的所需時間

初學者通常希望把所有參數都加入優化，以為參數愈多愈好。這想法不單會帶來剛才提及的過度優化問題，也會大幅增加優化所需的時間。原因是隨着參數的數量增加，可測試的範圍也會擴大。

Set Optimizable Inputs

☑	Signal Name	Input Name	Current Value	Strat Value	End Value	Step	Step Count
☑	maca_expamle	FastLen	12	11	20	1	10
☑	maca_expamle	SLowLen	26	21	30	1	10
☑	maca_expamle	MACDLen	9	11	20	1	10

# of combinations: 1,000

Set Optimizable Inputs

☑	Signal Name	Input Name	Current Value	Strat Value	End Value	Step	Step Count
☑	maca_expamle	FastLen	12	11	30	1	20
☑	maca_expamle	SLowLen	26	21	40	1	20
☑	maca_expamle	MACDLen	9	11	30	1	20

# of combinations: 8,000

以上圖的優化為例，假設需要優化的參數有 3 個，包括 MACD 中的快線長度、慢線長度和 MACD 線長度，快線由 11 試到 20，慢線由 21 試到 30，MACD 線由 11 試到 20。每條線都需要試 10 次，總共組合數量就是 1,000（=10×10×10）。若果將每個參數的範圍增加 10，每個參數需要試 20 次，組合數量就會暴增至 8,000（=20×20×20），電腦的處理時間就會增加 7 倍。如果優化 1,000 個組合要 1 小時，8,000 個組合就要 8 小時。由於參數組合數量的倍數效應是以幾何級數增長，會嚴重影響工作效率。

解決方法是盡量在不影響交易邏輯的前題下，減少需要優化的參數數量；此外，也可以從減少參數在優化範圍內的遞增級數着手，例如，當長線由 11 試到 30 時，一般的遞增級數是 1，即是要 11、12、13、……、29、30 逐一去試，如果把遞增級數改為 2，即是需測試的的數字是 11、13、15、17……29，由於參數組合的數量減少一半，優化的時間亦可以大幅減少。

(3) 優化結果的看待

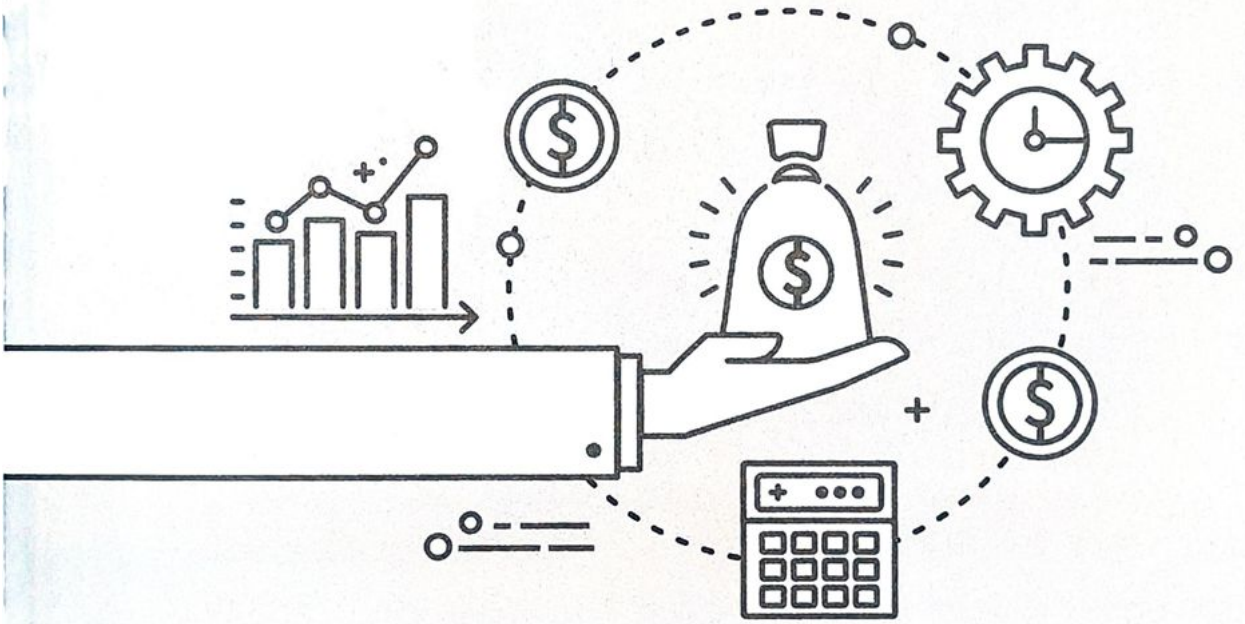
很多人以為優化的結果是回報愈高愈好。這想法除了有過度優化的問題外，也忽略了風險的重要性。假設在優化出來的參數組合中，A 組的回報有 50%，B 組的回報則只有 40%，若果只看回報，A 組的確較為優勝。不過，若果把回撤加入考慮，A 組回報的最大回撤有 30%，而 B 組的回撤則只有 10%，那很明顯是 B 組優勝。由此可見，回報只是優化結果的一部分，我們還要考慮其他因素，例如，交易頻率、回報的標準差等，我們會在稍後的篇章中探討各種不同因素。

〈 小結 〉

本章旨在分享程式交易的標準步驟，當中包括前期最重要的回測和優化。讀者應該發覺，程式交易中每一個步驟都是環環緊扣、缺一不可的。初學者須清楚了解回測和優化的意義和限制，亦要明白為何程式交易者須花最長時間於這步驟。

在接下來的篇章，我們會探討如何解讀回測和優化的結果，要謹記所有數字本身都是冷冰冰的，只有人類的解讀才能賜予它們生命。正確的解讀方法可以讓你了解策略的實際風險和回報，但錯誤的解讀就會害你輸錢，當中的風險是初學者難以估計的。

第4章
MultiCharts 系統



- 平台概覽
- 接駁證券行
- 數據導入（歷史數據）
- 數據導入（即市數據）
- 回測設定
- 自動交易設定

〈 4.1 〉

平台概覽

MultiCharts（下稱 MC）作為全球最為大眾化的第三方程式交易軟件，可以說是程式交易新手的最佳起步點。MC 的功能算是齊整，除了高階的程式交易策略（如波幅交易、配對交易、期權交易等策略）外，基本上絕大部分方向性的交易方法都可以運用 MC 進行。

首先，大家要明白的是，MC 和大部分第三方軟件一樣，都是一個 Bar-based System，即是程式的邏輯是建基於 Bar，而當中最典型的當然是陰陽燭（Candle Stick）。通常在一般的報價軟件內，你都可以定義每一支 Bar 代表的時間，如 5 分鐘、1 小時、1 日、1 星期等。我們亦可以在 MC 中設定 Bar 的大小，以程式交易的術語來說，我們會稱為「Resolution」，例如，一個 5 分鐘 Resolution 的圖表就代表每支 Bar 是 5 分鐘。

Bar-based system

▲ MC 的圖表會顯示所有策略的買入和賣出訊號，向上箭咀是買入訊號，向下箭咀是沽出訊號。

另外，圖表是 MC 系統重要的一環。無論做回測或是真實的自動交易，我們都須將預先寫好的交易策略設置於圖表上，系統會根據交易策略的邏輯計算出何時買入和何時沽出，並紀錄開倉的價位，以便計算回測的結果，以及準備執行止賺和止蝕的動作。如上圖所示，向上和向下箭咀指著的分別是以往應該買入和沽出的 Bar；如果執行真實自動交易的模式，新的即市數據會在圖表最右邊不斷產生新的 Bar，以 1 分鐘 Resolution 圖表為例，圖表的最右邊會每 1 分鐘產生一支新 bar，如果這支新 Bar 的價格觸發到策略的交易訊號，系統會自動執行該交易指示，並且以向上或向下箭

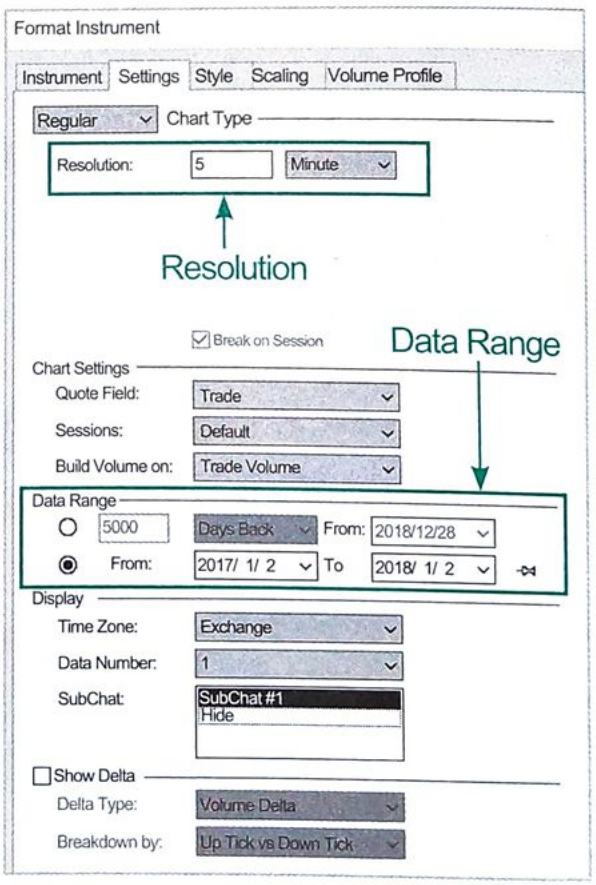
咀對該 Bar 作出標示。這個動作會重複執行，形成真實的自動交易的系統。

整個 MC 系統，尤其是程式編寫和自動交易的部分，都會以 Bar-based System 的邏輯運行，所以我們必須預先設定 Resolution，好讓 MC 可以進行回測、優化和自動交易。如果設定了錯誤的 Resolution，MC 仍然會執行，但效果可能與預期相差很遠。因此，在編寫程式前，調整正確的設定是非常重要的。

Format Instrument

在 MC 系統中，策略本身會以程式編寫，而策略以外的所有設定都會在圖表設定，而設定的地方就是 Format Instrument。基本上我們多數都會使用 Instrument 和 Settings 兩個 Tab。

Settings 裡面有兩個最重要的設定，首先是剛才所提到的 Resolution，另外就是 Data Range。



Resolution ： 設定每支 Bar 的時間 / 大小

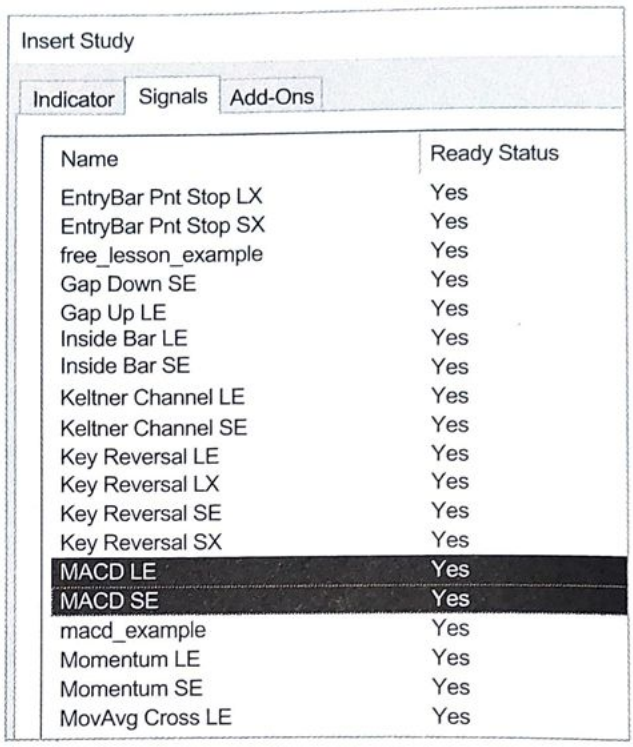
Data Range ： 設定圖表的數據範圍，Days Back 和指定範圍的選取方法都同樣常用。

注意，Data Range 是一個非常重要的設定，代表著圖表的數據範圍，即是由開始日期至結束日期，而回測和優化的結果亦只限於這段時間範圍。當進行回測和優化時，如果設定範圍太短，由於所包含的數據太少，結果會缺乏代表性；但如果設定範圍太長，數據則會包含許多過時和不會對未來帶來任何啟示的資訊。因此，Data Range 必須是一個合理的範圍。

設定回測範圍是一項大學問，一般由數年至十幾年都可以是合理範圍，而定義的邏輯會涉及交易者的主觀判斷，沒有公式可言。例如，自 2003 年開始，大量內地重磅股到香港上市，並改寫了恒生指數成份股的格局，所以 2003 年是一個重要的分水嶺。此外，滬港通開通的 2014 年亦適合用作起步點，因為北水在當年開始參與港股交易，數據可作為短線交易的參考。這些說法都比較符合邏輯，算是一個較簡易的制定回測範圍方法。由此可見，交易者在交易前最好對該資產有深入的認識，否則會很容易選擇不適合的 Data Range。

Insert Study

接著要了解的，是設置交易策略的地方。正如剛才所說，我們要先將交易策略設置於圖表上，系統才會根據其邏輯計算買入和沽出訊號的出現時間，從而得出回測結果。

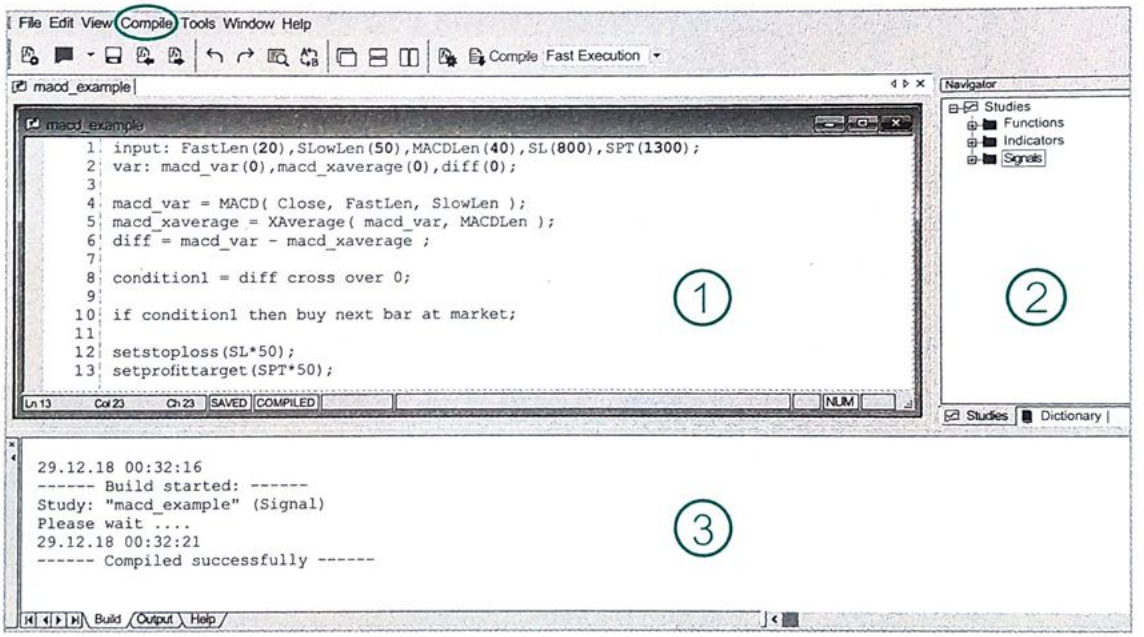


Insert Study 有 3 個 tab，分別是 Indicator、Signals 和 Add-Ons。當中最重要的是 Signals。因為在 MC 的系統內，Signal 本身就是交易策略，只有 Signal 中的交易指示能夠進行回測和真實交易，所以上述的「把交易策略設置於圖表」就是將 Signal 於圖表中啟用。

MC 內置了不少策略，除了 RSI、MACD 等技術指標，還有型態分析，如區間突破等。不過， MC 內置的策略絕對不適宜直接使用，因為這些策略只可作示範用途，就連完整的止賺止蝕、注碼控制都欠奉，而這些額外的部分就需要自行編寫。因此，當交易者在使用傳統指標時，也應該自行建立新策略，首先將內置策略的程式碼 Copy and Paste 至自己的策略中，然後再加上其餘部分。

此外，大家還要留意 Insert Study 右側顯示的 Ready Status。狀態 "Yes" 代表該策略已經完成編譯，可以直接執行，但如果狀態為 "No" 或 "undefined"，則代表該策略的程式碼已經儲存好但未進行編譯，又或是已經編譯好但有新改動未儲存。總之，凡是 "Yes" 以外的狀態都不能使用。

Power Language Editor



MC 內置的 Power Language Editor（下稱 PLE）用於編寫交易策略的程式碼，可以理解為編寫傳統程式語言時所使用的 IDE（Integrated Development Environment）。程式交易者會花大量時間在這裡設計和編寫交易策略，並來回觀看回測的結果。

PLE 主要分為 3 大部分：

（1） 編輯器

編寫程式的地方，而關於程式編寫的部分會在稍後的篇章介紹。

（2） Netvigator

這部分有兩個 Tab，分別是 Studies 和 Dictionary。後者沒有實質的功能，只是方便用戶在此作查詢之用。

至於 Studies，裡面有 3 個 Folders，分別是 Functions、Indicators 和 Signals。就如上文所說，Signal 本身就是策略，編寫交易程式就即是編寫 Signals，所以是程式交易的靈魂。Functions 的中文是函數，是計算數值時用到的程式碼。例如，交易者可將計算 RSI 的所有程式碼放在一起，當 Signal 需要計算 RSI 時，只須寫入 Function 的名稱便可完成計算。

Indicator 是畫圖用的指標。例如，交易者可以此把移動平均線置於圖表上。Indicator 與 Signal 的最大分別是前者只能用作畫圖，不能作出任何交易指示，而後者則可以有交易指示。因此，Indicator 的作用其實不多。

這 3 個 Folders 都會存放著各自的檔案，任何在 PLE 編寫的程式都會歸類於這 3 個 Folders 之中，所以在這裡可以選取以往編寫過的策略。

(3) Output Bar

Output Bar 有 3 個 Tab，分別是 Build、Output 和 Help。首先是 Build，它會顯示關於程式碼編譯的信息，如圖中顯示的成功編譯信息，代表程式碼沒有錯誤或問題；而如果程式碼有出錯，Build 就會顯示錯誤信息，交易者可按此作修正。

Output 所顯示的是關鍵字 Print 所包括的變數和數字。交易者在編寫程式時，如果希望檢視某一個變數在某一個時間點的數值，就會使用 Print 這個關鍵字。

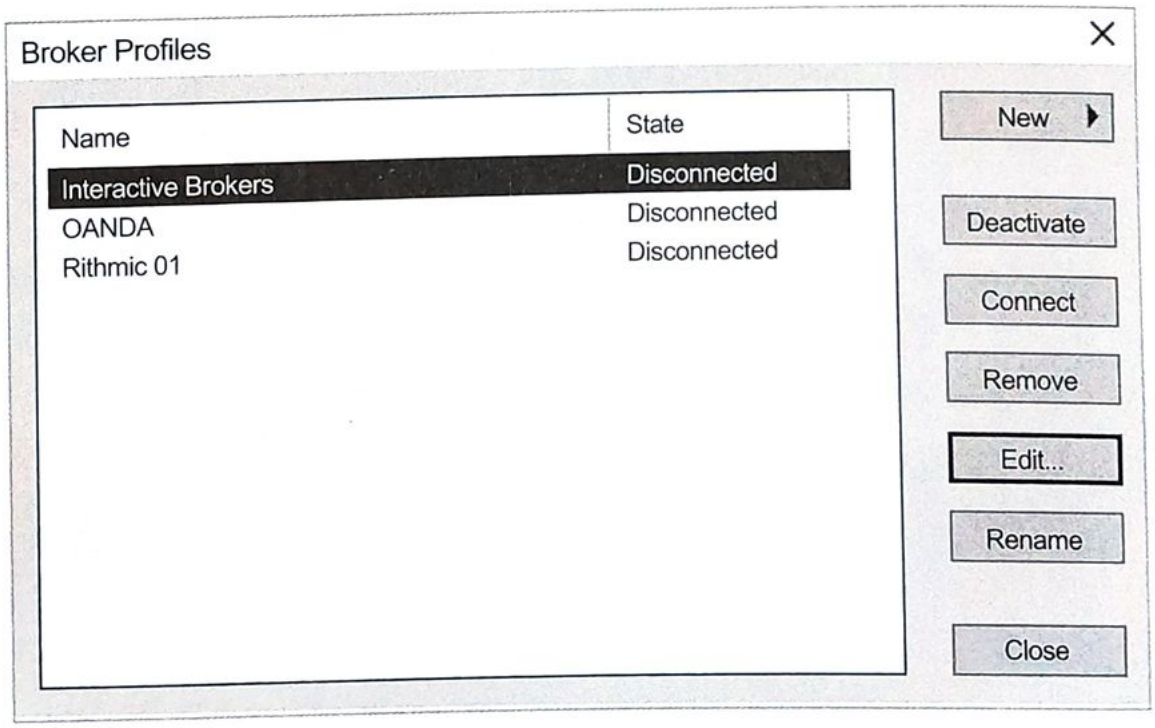
至於 Help，Google 都可能比它有用，所以交易者大都極少使用。

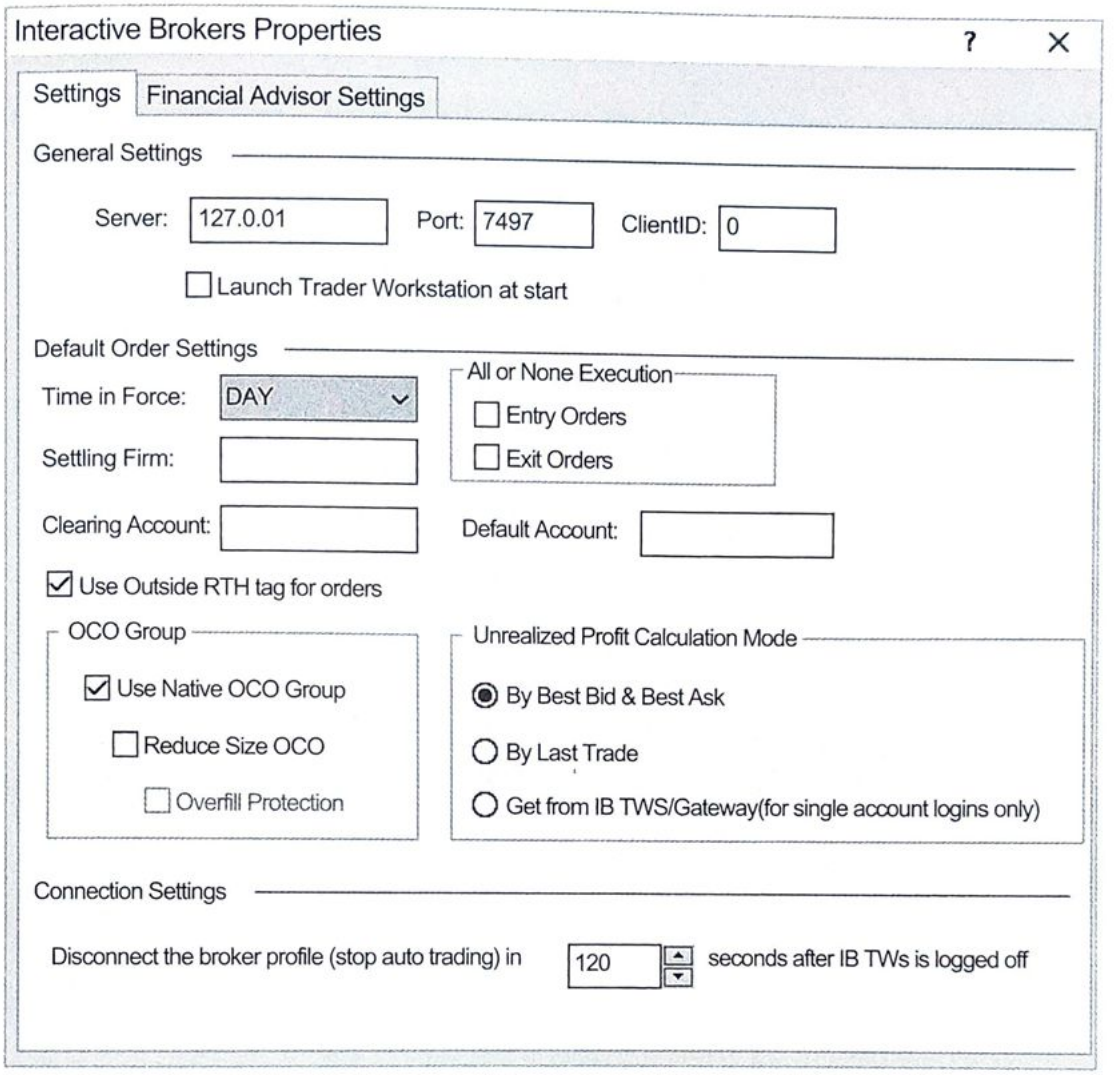
最後要注意的是圈中的 Compile，即是編譯。當交易者在程式編輯器完作編程後，必須按 Compile 才可以使用。所以，大家應養成良好習慣，記得在寫好程式後按一下 Compile 鍵，如果程式碼有錯，就要留意 Output Bar 內的提示信息。

〈 4.2 〉

接駁證券行

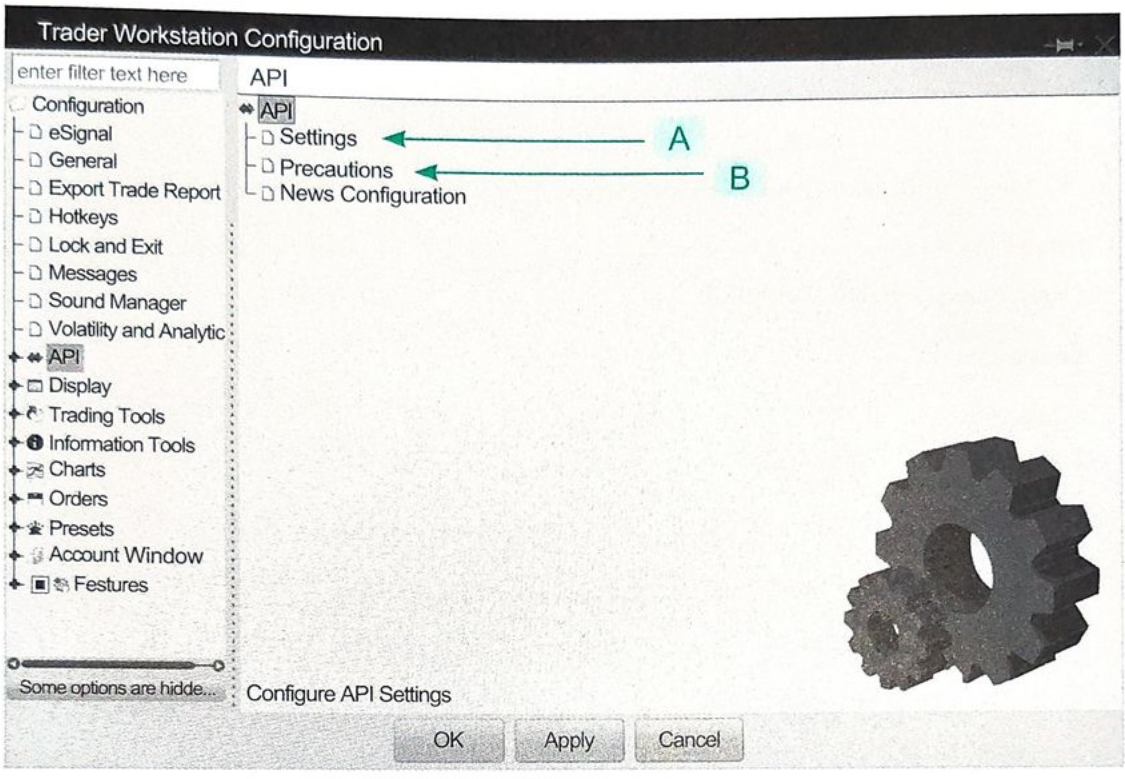
接下來要探討接駁證券行的設定。本書會以 Interactive Broker（下稱 IB）作為示範，因為 MC 已經內置了許多 IB 的設定，交易者只須作簡單的設置便可以連接。此外，IB 的平台功能非常強大，旗下的 Trader Workstation（下稱 TWS）預設開放 API 給所有用戶，而且佣金收費幾乎平絕所有證券行，所以絕大部分進行程式交易的散戶都以 IB 進行交易。



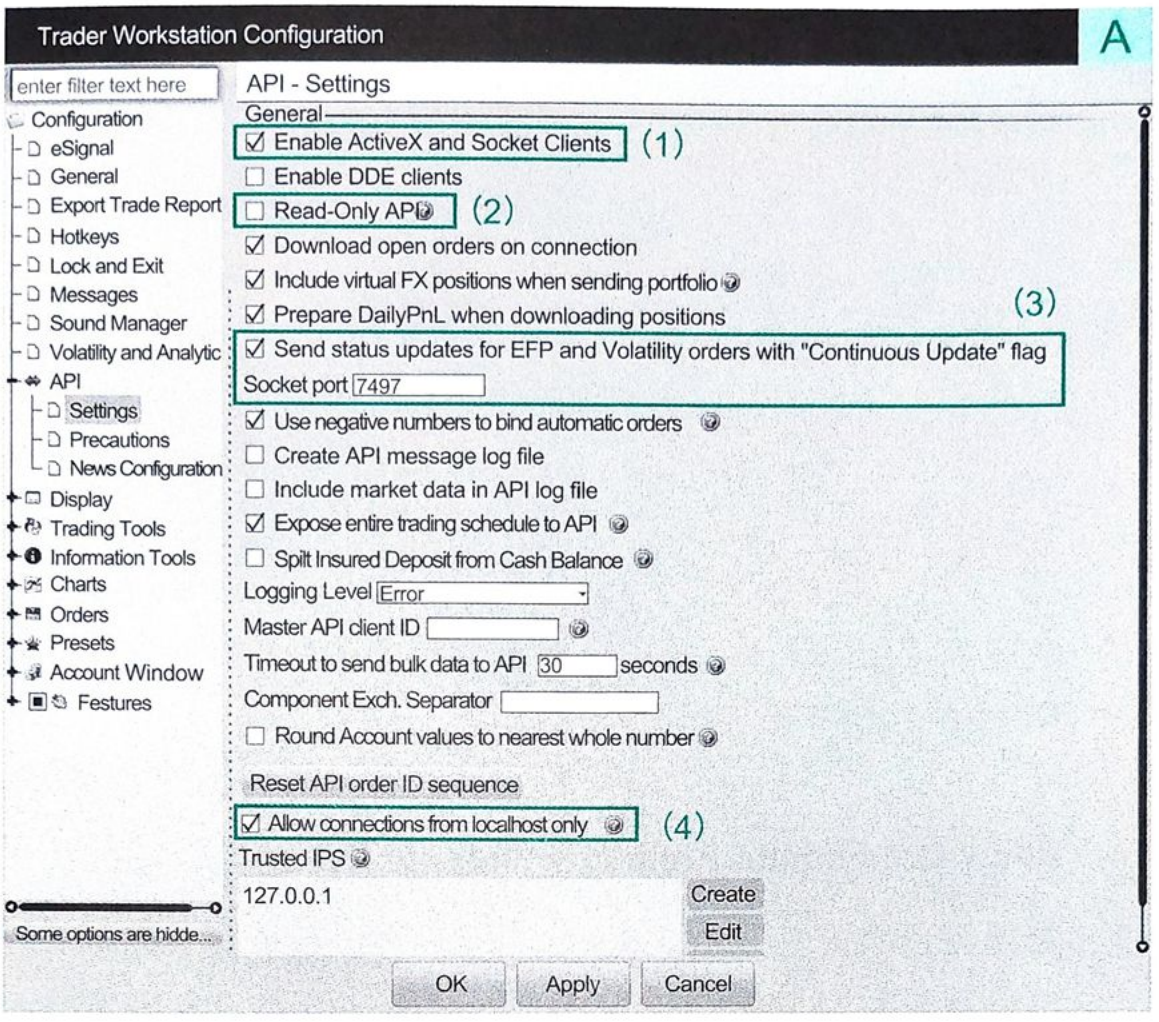


MC 裡面的 Broker Profile 內置了 3 個證券行連接設定，第一個就是 IB。在其 Properties 中，已內置 IB TWS 所需的 Server IP 和 Port number。交易者可使用預設的設定，毋須作更改。不過，我們仍須更改 TWS 內關於連線的設定，讓 TWS 和 IB 兩者能夠透過 API 互相溝通，連接數據和傳達交易指示。

TWS 視窗



在 Trader Worksatation Configuration 中，須更改設定的地方主要是 Settings 和 Precaution。



先看看 Settings A 部分，當中有 4 項值得留意：

（1）Enable ActiveX and Socket Clients ：

必須選「✓」，因為我們須動用到當中的 Socket。

(2) Read Only API：

預設是「✓」，但如果我們要透過 API 下達交易指示，這裡必須留空。

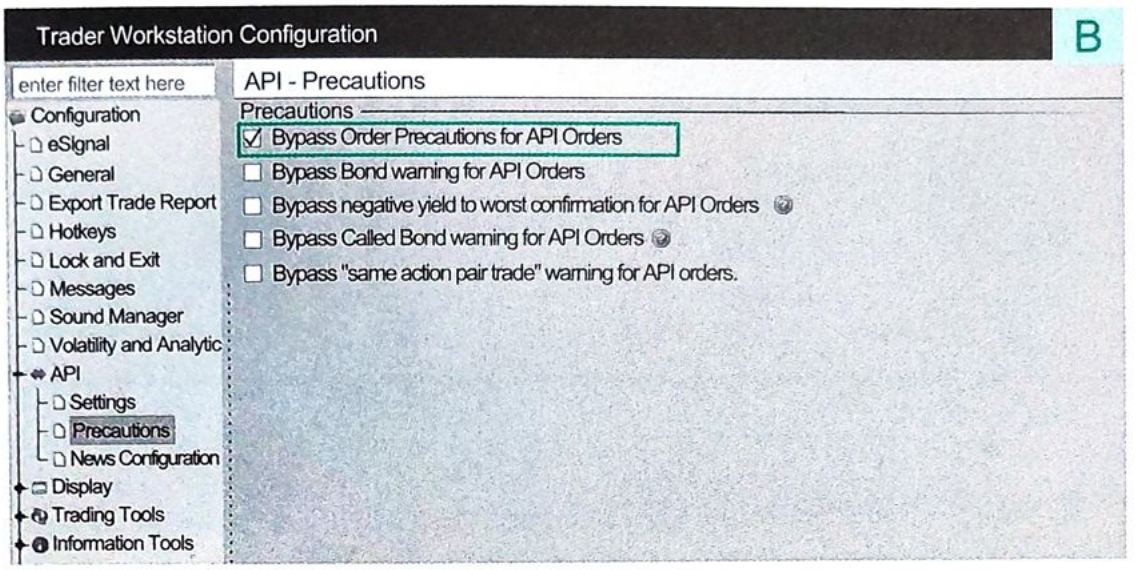
(3) Socket Port：

當中的「7497」是剛才在 MC Broker Profile 內的 Port Number，不同的 IB 戶口都有各自的 Port Number，TWS Paper Trade 戶口通常是「7496」，Live Account 則是「7497」。如果你須用到 IB Gateway 的話，Port Number 則應該是「4000」或「4001」。建議在連接 MC 前，先親自到 Configuration 檢查一下。

(4) Allow Connection From local host only：

在絕大多數情況下，IB 和 MC 都會在同一部電腦上運作，所以這裡必須選「✓」，這代表 TWS 不接受本機以外的連接，只接受本機上述 Port 的連接。

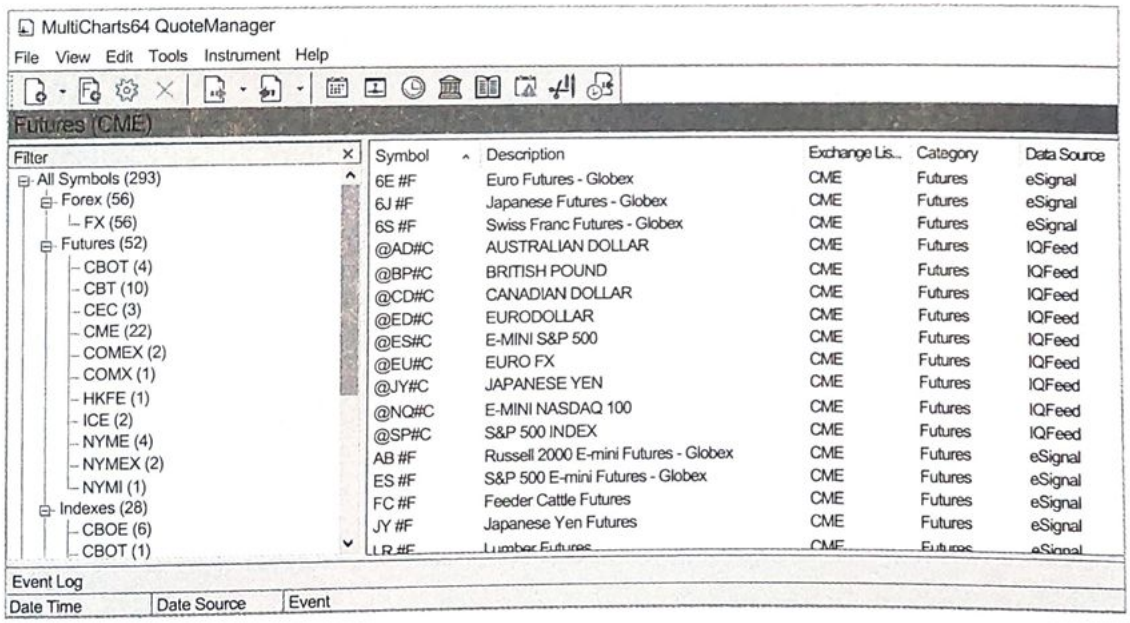
至於 Precautions 的設定。首項「Bypass Order Precaution for API Orders」的預設是空白。但如果你希望透過 API 下達交易指令，這裡就要必須選「✓」，否則每當有新的交易指示透過 API 下達，就會彈出要求確認的 Message Box。這顯然不是程式交易的正常運作方式。


Trader Workstation Configuration
B
enter filter text here
Configuration
eSignal
General
Export Trade Report
Hotkeys
Lock and Exit
Messages
Sound Manager
Volatility and Analytic
API
Settings
Precautions
News Configuration
Display
Trading Tools
Information Tools
API - Precautions
Precautions
Bypass Order Precautions for API Orders
Bypass Bond warning for API Orders
Bypass negative yield to worst confirmation for API Orders
Bypass Called Bond warning for API Orders
Bypass "same action pair trade" warning for API orders.

〈 4.3 〉

數據導入（歷史數據）

運用 MC 進行回測、優化等工作前，必須先有歷史數據。而導入數據的工作可透過 MC 內置的 QuoteManager（下稱 QM）進行。



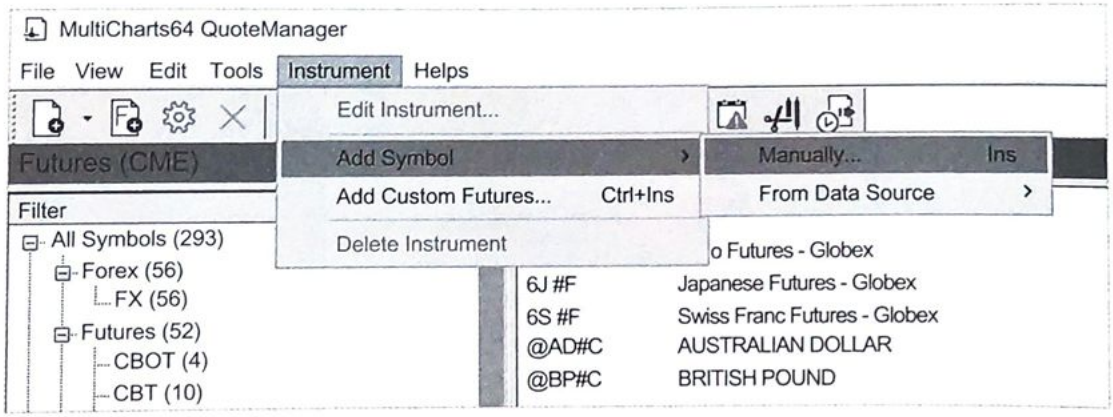
要注意的是，QM 所管理的不只是歷史數據，亦包括即市數據。一般來説，歷史數據的範圍會比較長，而且資料呈靜態，多數用於回測和優化。即市數據的範圍通常會較短，長度只需讓交易策略有足夠數據計算交易訊號便可以。

導入歷史數據

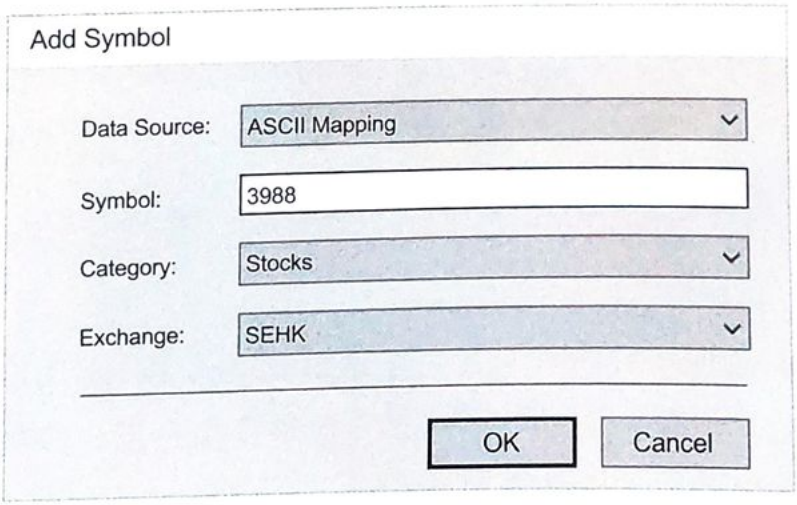
步驟 1：

歷史數據的處理比較簡單，首先要準備以 csv 或 txt 作副檔名的數據。如果從 Yahoo! Finance 下載的免費股價數據，該檔案已經是 csv 格式。舉例說，我們可在該網站下載中國銀行（03988）的每天股價數據（其最高 Resolution 只有每日的 Open、High、Low、Close），而所得的檔案為 3988.HK.csv。

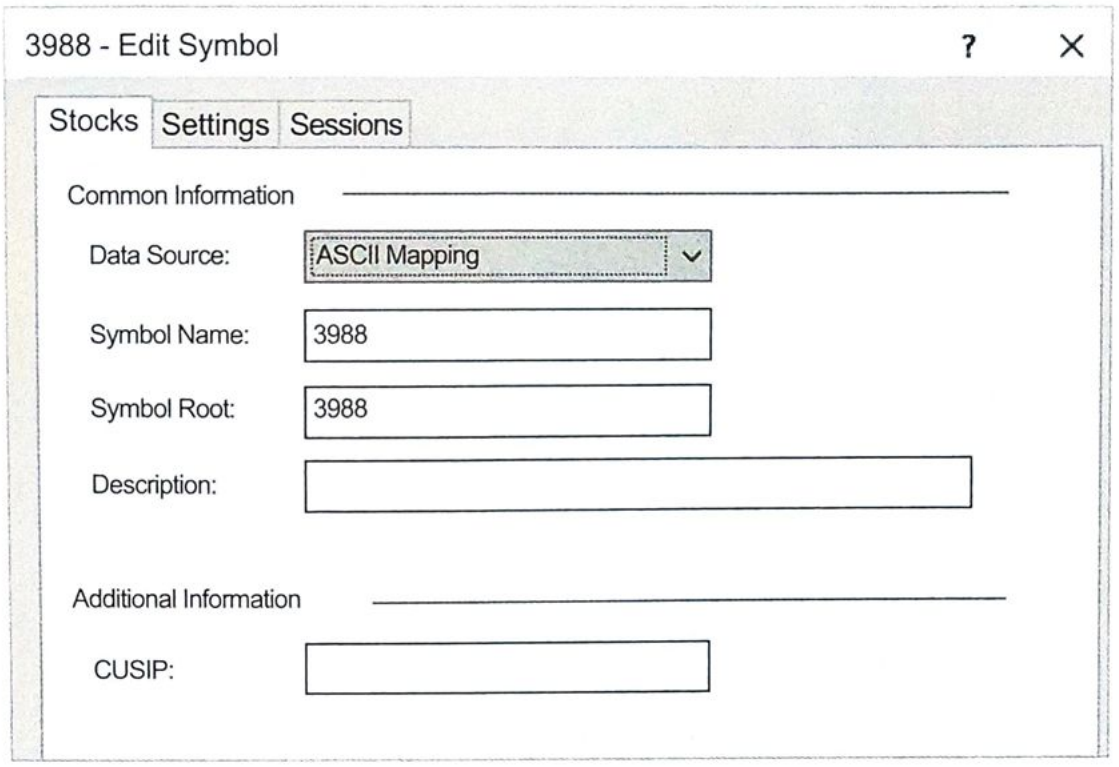
步驟 2：



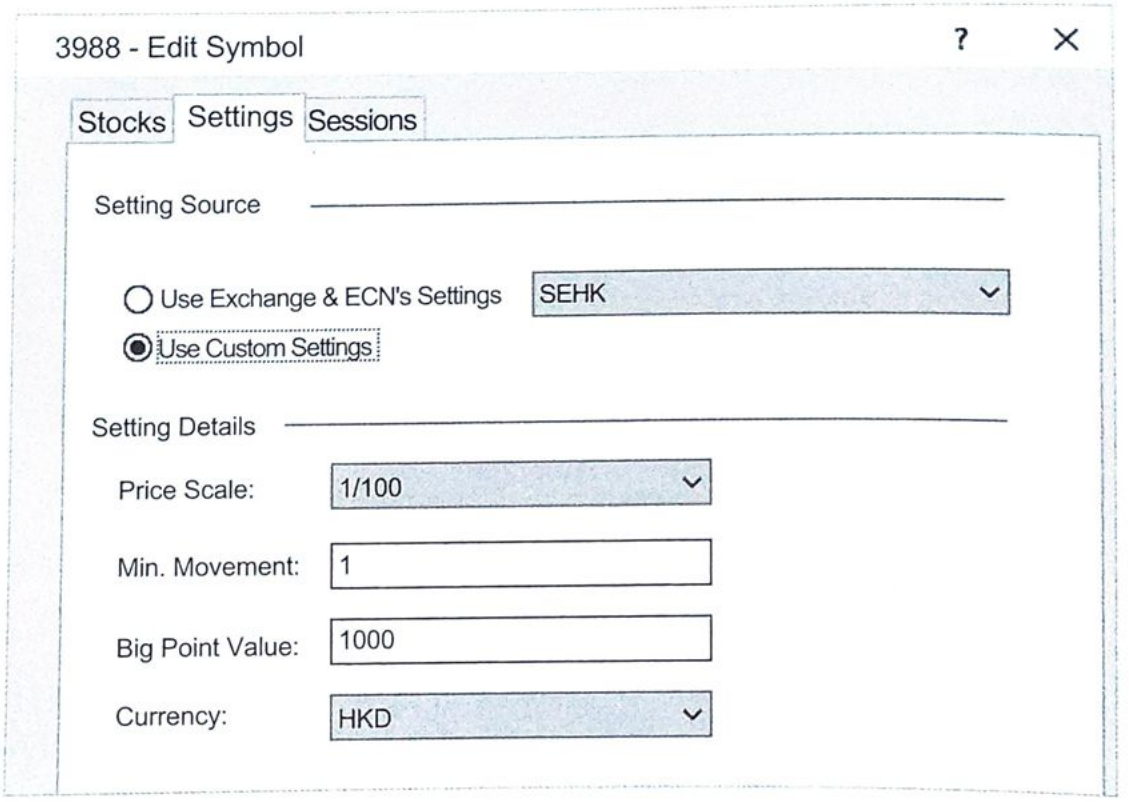
從 QM 內選取 Instrument → Add Symbol → Manually。



凡是 csv 或 txt 的檔案，Data Source 都要選擇 "ASCII Mapping"。Symbol 項填寫資產的名字，可以自行決定。至於 Category 和 Exchange，交易者須選擇項目的種類和交易所，需注意在香港市場，股票是屬於 "SEHK"，而期貨則屬於 "HKFE"。



然後看到 Edit Symbol 一頁，這裡有 3 個 tab，分別是 Stocks、Settings 和 Sessions。基本上，Settings 只是設定名稱，而 Session 則設定該股份的開市和收市時間，因為剛才已選擇交易所為 "SEHK"，所以 Sessions 都不用更改，可使用預設值。而餘下只有 Settings 是需要修改。



Settings 的預設值是使用交易所的設定，但這是不切實際的，因為即使是同一個交易所，旗下的產品亦可能有不同的貨幣、價格等設定，所以我們應為每一項資產選用自己的 Custom Settings。

(1) Price Scale：

這裡的選項比較多，對於股票和期貨，常用的設定為 100、10、as is、1/10、1/100、1/1000 等，其餘的多數用於債券。Price Scale 是指價格的規模，對於某些股票價格較細，最小的 Tick 是小數後兩個位，如 0.005，Price Scale 便是 1/1000。至於一些格價較大的商品，如每個 Tick 是 25，

Price Scale 就會是 10。在本例子中，中國銀行的 Tick 是 0.01，所以 Price Scale 是 1/100。

(2) Min Movement：

意思和 Tick 有點不同。例如，某細價股的 Tick 是 0.005，Price Scale 是 1/1000，Min Movement 就會是 5。因此，Min Movement 就等於要計算 Tick 除以 Price Scale。在中國銀行的例子中，由於 Tick 是 0.01 和 Price Scale 是 1/100，Min Movement 就自然是 1。

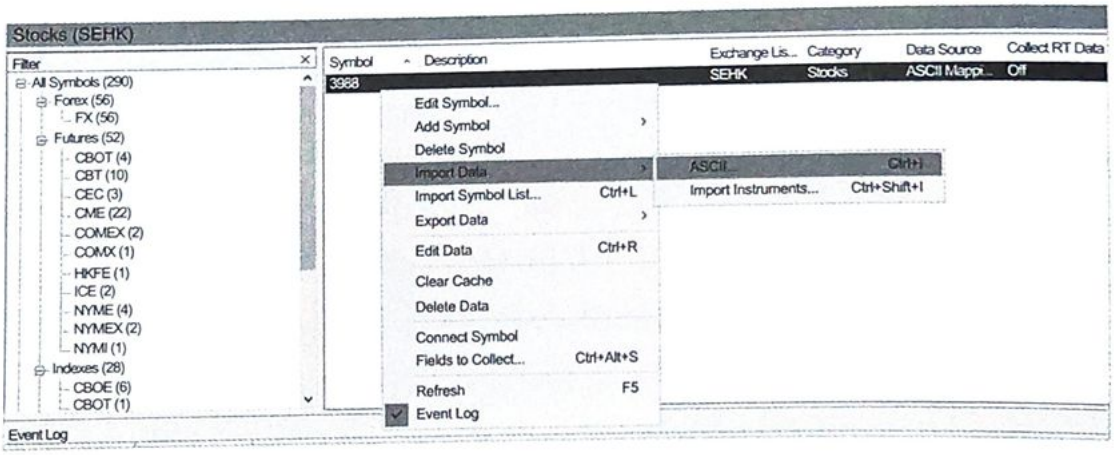
(3) Big Point Value：

不論資產價格的單位如何，Big Point Value 都代表整數 1 的價值。例如，某細價股的股價是 $0.45，每股 2,000 手，Big Point Value 就會是 2,000；又例如，恒指期貨每點 $50，Big Point Value 就會是 $50。由於中國銀行的每手股數是 1,000 股，不論股價是多少，其 Big Point Value 都是 1,000。

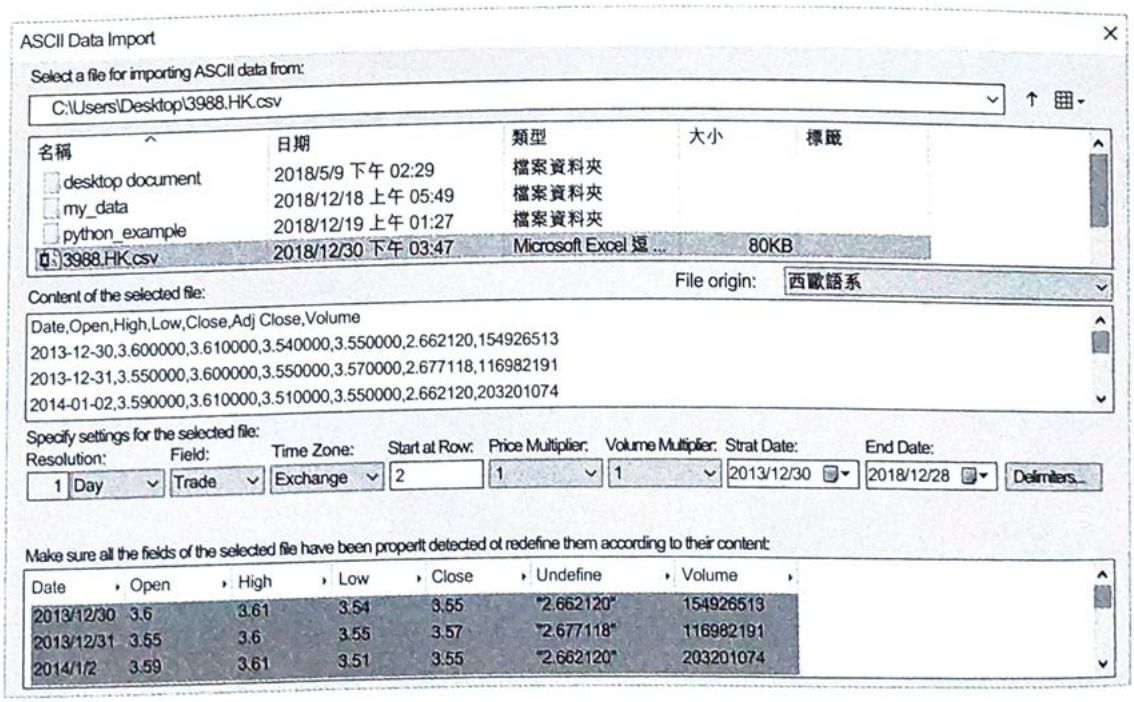
(4) Currency：

不少交易所都有多於一種計價貨幣。以港交所為例，除了以港幣計價之外，亦有以人民幣計價的股票，所以交易者在 Custom Settings 設定貨幣是必要的。

步驟 3 ：



成功新增 Symbol 後，就要導入 csv 檔案內的數據。剛才新增的股票會出現在左右邊相對應的資產和交易所類別內，這時候對住該資產點右鍵→ Import Data → ASCII。

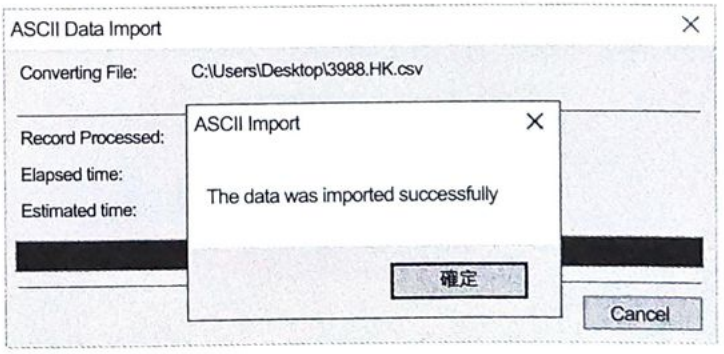


點選之後，大家會見到選擇檔案的頁面，並可以選取剛才從 Yahoo! Finance 下載的 csv 檔案；然後，下方的位置會出現 Date、Open、High、Low、Close、Volume 等數據，這代表 QM 已辨認到檔案內的數據和屬性；但如果出現 undefined 的話，則代表不能辨認。值得留意是 Date 的格式，QM 經常分辨不出月份和日期，所以最好點擊 Date 作進一步確認和選擇，如果成功辨認，你會見到 Start Date 和 End Date 都是正確的開始和結束日期。

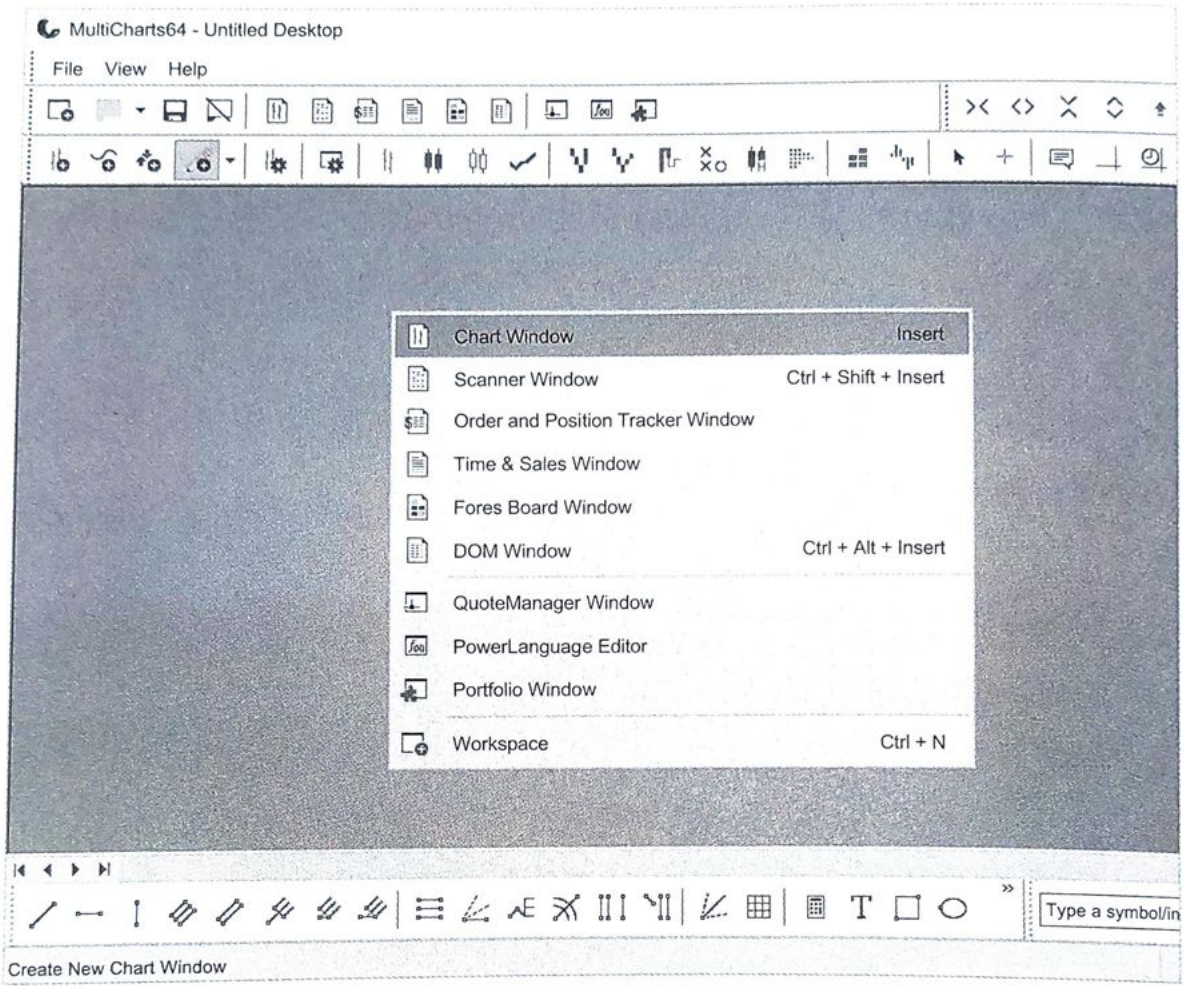
另外，如圖所見，Yahoo! Finance 的數據包含了 Adjusted Close，這項數據是 QM 不能辨認的，但由於它不會影響其他數據，所以毋須理會。

QM 的導入數據最高可以支援 Tick Data，亦支援常見的 1 分鐘、1 日等數據。如果 Resolution 是 Tick Data 或分鐘級，項目會有多一項 Time 的資料，一般的 Excel 格式都可以讓 QM 辨認。不過，分鐘級數據或 Tick Data 一般都要收費的，港股、期指等數據都在港交所網站有售，有興趣的朋友可以到港交所網站查看一下。

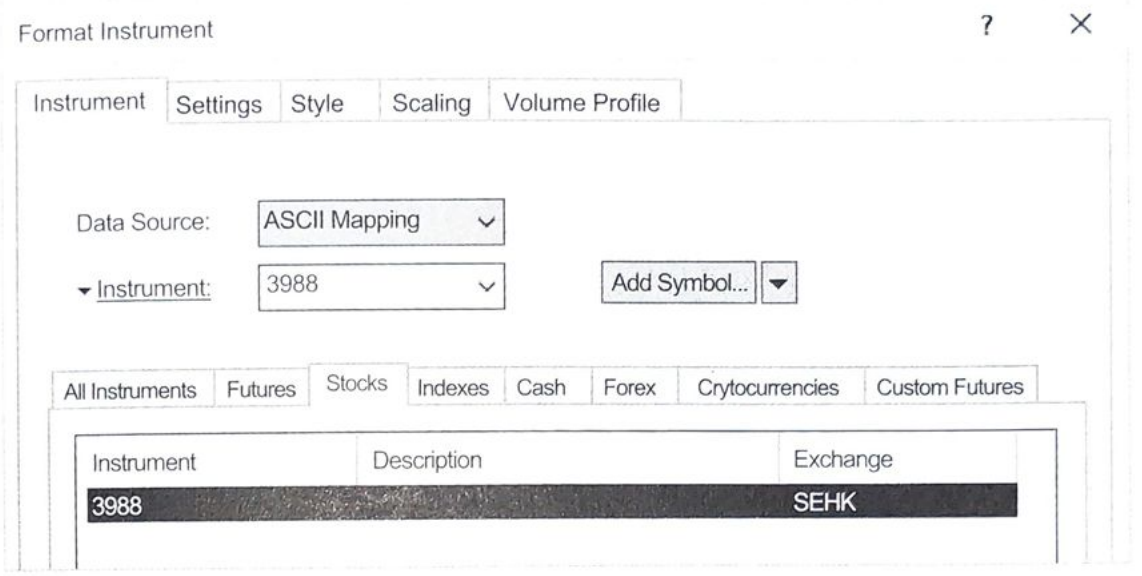
當成功導入數據，就會出現這個確認畫面。



步驟 4 ：



最後一步就是回到主畫面，在空白的地方按右鍵選取 Chart Windows，然後會出現 Format Instrument 視窗。

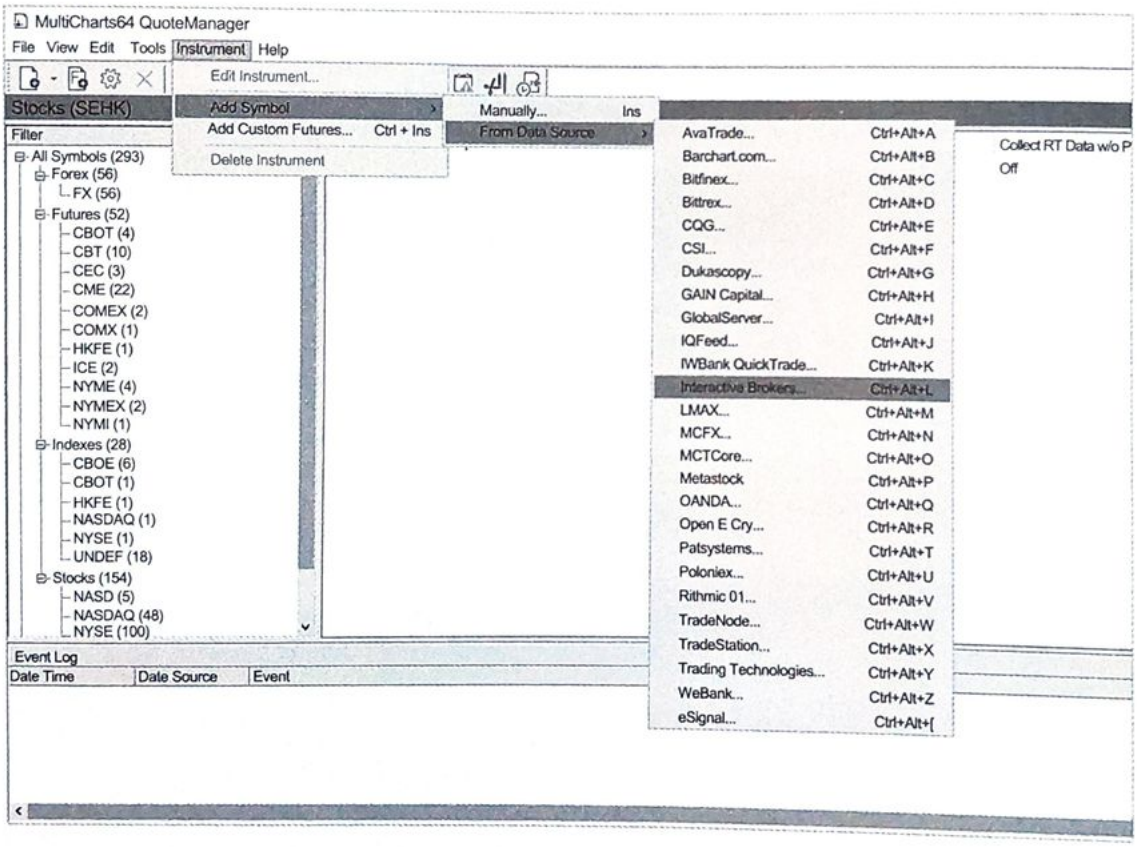


Data Source 選擇 ASCII Mapping，下方的 Tab 選擇之前安排好的分類，就會出現剛才導入的中國銀行數據。在按確認前，大家記得要到 Settings 選擇正確的 Resolution 和 Data Range，然後就可以把圖表放置在主畫面了。

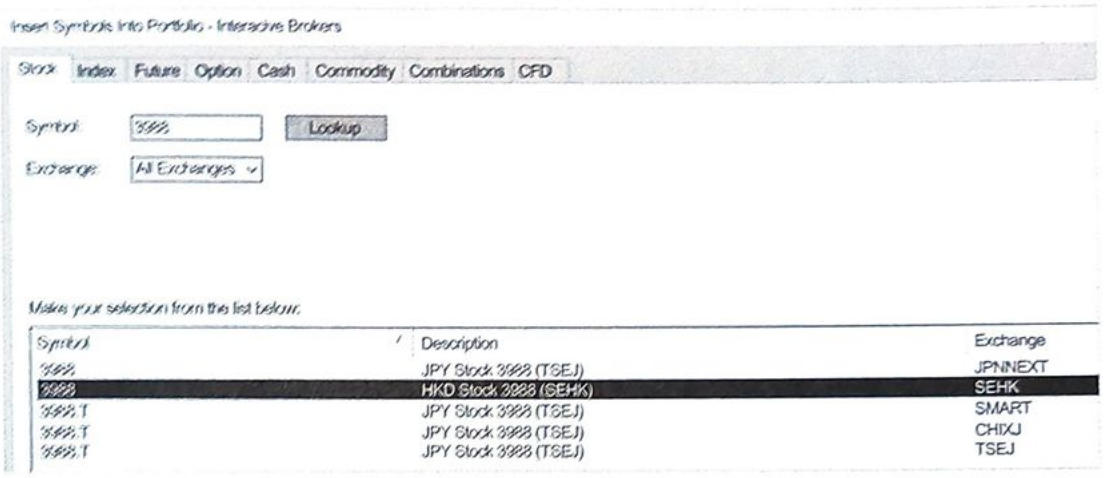
〈 4.4 〉

數據導入（即市數據）

相比起歷史數據，即市數據的導入較為簡單，同樣是運用 QuoteManager（QM），但步驟就會較少，只須確保 MC 和 IB API 已經連接好。



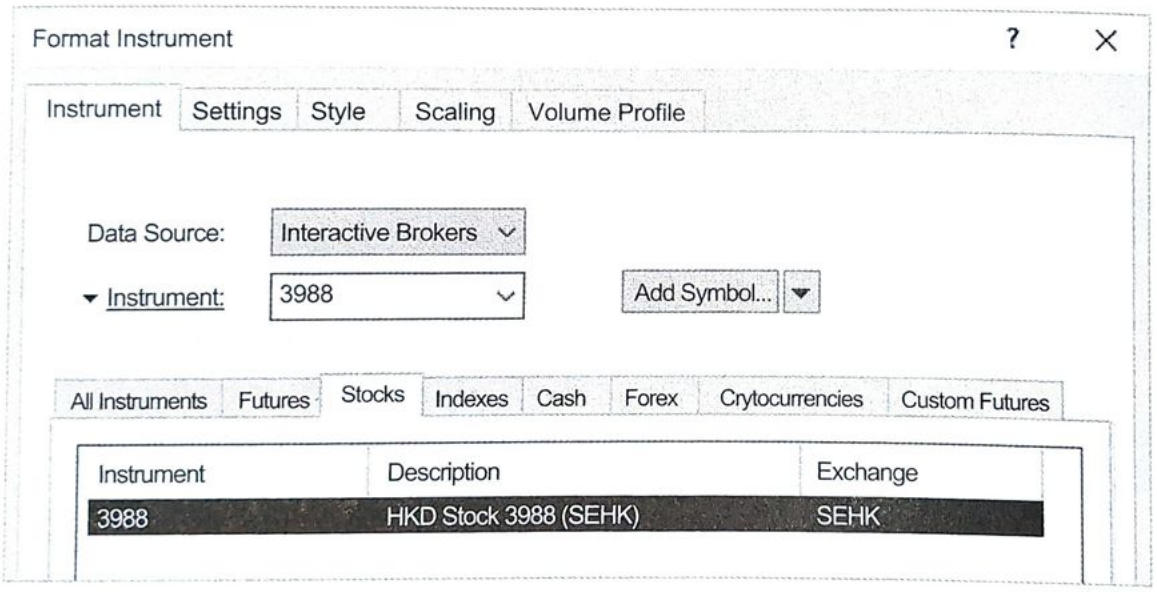
首先，回到 QM 和選擇 Instruments → Add Symbol → From Data Source → Interactive Broker；然後會出現一個搜尋資產的視窗，當中有 Stock、index、Future 等幾個常用的 Tab。我們在 Stock 的搜尋輸入中國銀行的編號 3988，可以看到以下結果。



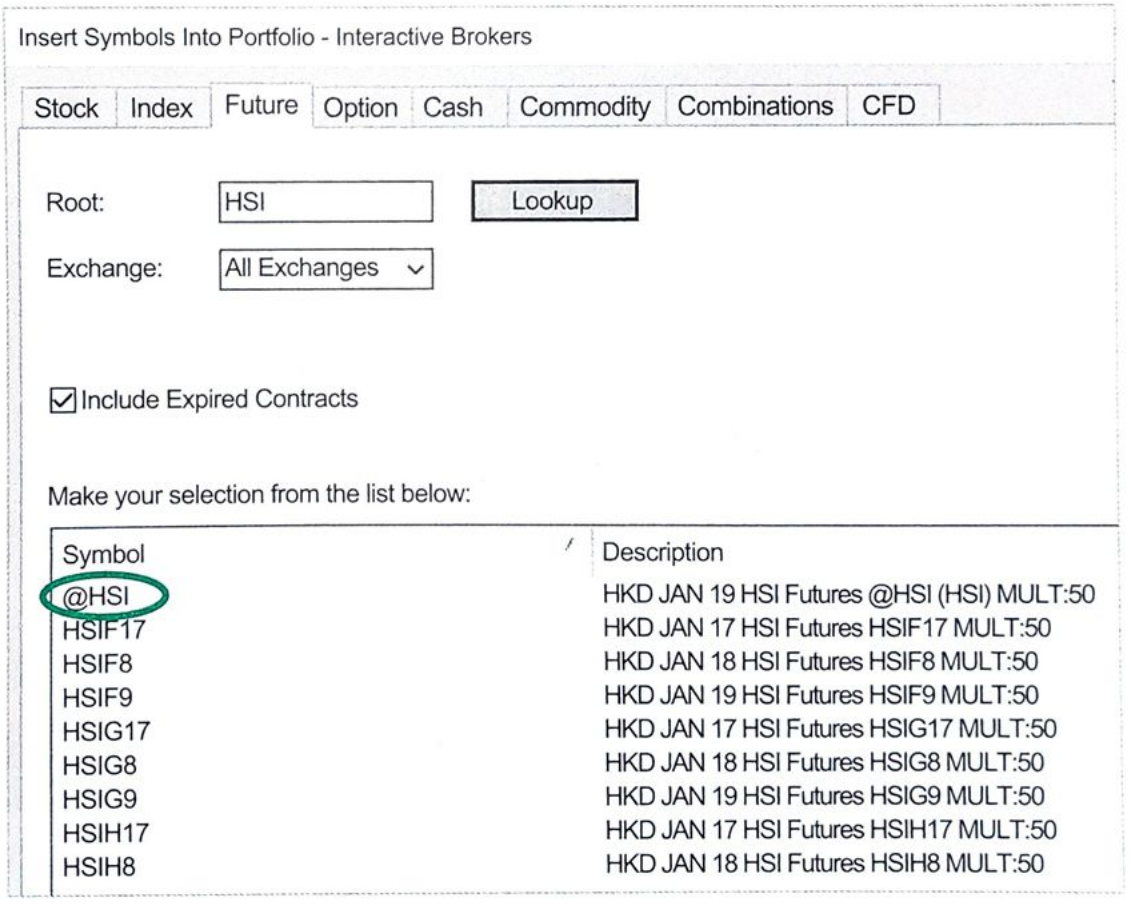
搜尋除了出現中國銀行外，也有其他編號同樣是 3988 的股票，但這些股票都不是中國銀行，而是在日本上市的股票。因此，我們在選擇時，須留意它是 SEHK 的股票，又或是在搜尋前已選擇交易所為 SEHK，這樣就能避免出現港交所以外的股票。完成後，系統會出現確認信息，並且在 SEHK 分類中出現中國銀行的股票。

Symbol	Description	Exchange Lis...	Category	Data Source	Collect RT Data w/o Plotting
3988		SEHK	Stocks	ASCII Mapping	Off
3988	HKD Stock 3988 (SEHK)	SEHK	Stocks	Interactive Brokers	Off

大家可以看到剛加入的3988即市數據，以及之前導入的3988歷史數據，兩者的分別在於Data Source，一個是ASCII Mapping，另一個則是Interactive Broker。有別於之前的歷史數據，我們不須進行Import Data，在主畫面Chart Window設置Format Instrument的時候，選擇Interactive Broker作為Data Source，就已經可以在Stock Tab見到3988，如下圖所示。



如果導入的是期貨數據，在搜尋頁面會見到一個 @HSI 的選項。原因是每張期貨合約都有自己的編號，例如，在 2019 年 1 月，即月的恒指期貨編號為 HSIF9，而 MC 在導入數據產生圖表時，只會導入單一合約，因此當導入「即月」合約的過去數據時，會同時導入該合約在上一個月的歷史價格，然而當時該合約並非即月合約，這樣就會產生數據上的問題，可能令策略採用了錯誤的數據作計算。為此，IB 設計出 @HSI 作為連續合約，在這種連續合約的結構中，過往的數據都會是當時最新的合約。



另外，還要留意是，利用 QM 接駁 IB 導入的數據其實不止即市數據，也包括一定數量的歷史數據，只要在 Format Instrument 設定好數據範圍，就可以運用 IB 提供的歷史數據作回測和優化。而透過接駁 IB 所得到的數據與 ASCII Mapping 數據的最大分別是，前者使用時必須確保 MC 接駁 IB，在交易時段內就會不停導入最新數據，主畫面最右邊的 Bar 的最新價格也會不停更新。在交易時段完結後，所有歷史數據都會保在主畫面內，運用這些數據作回測和優化的步驟，與運用 ASCII Mapping 的步驟相同。

〈 4.5 〉
回測設定

MC 的回測設定主要有兩項重要因素，分別是佣金收費和滑價（Slippage）。除非交易策略的交易頻率極低，否則這兩項因素對交易表現會產生極大的影響。它們甚至可視為程式交易中最重要的因素。

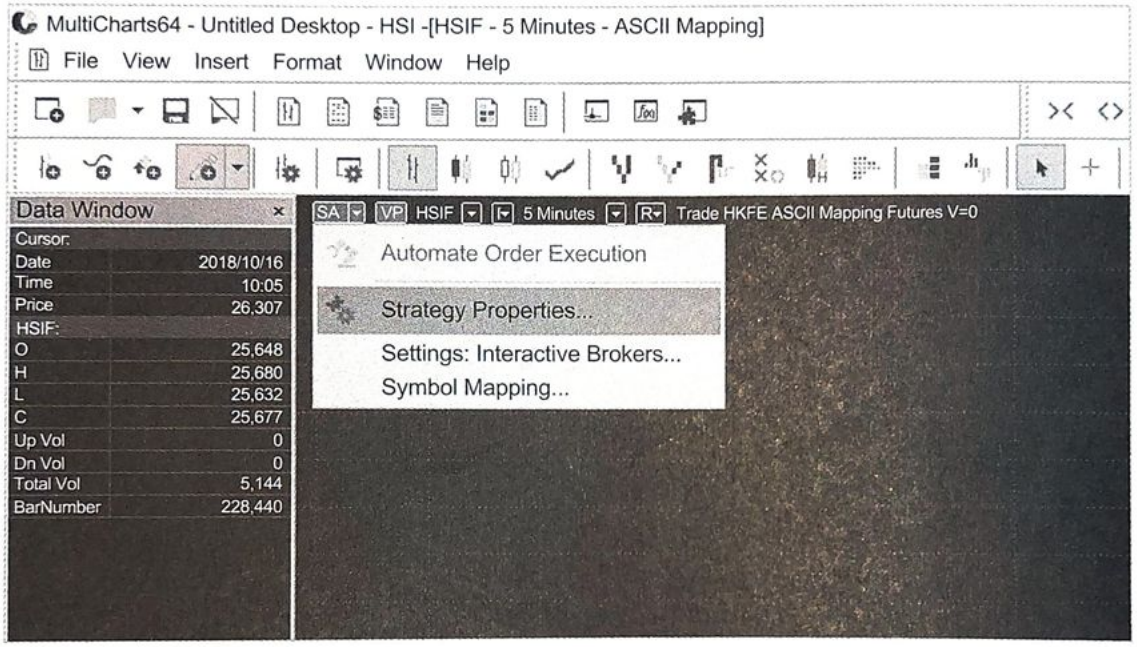
交易的成本：佣金 + 滑價

筆者通常將這兩個因素合稱為交易成本，佣金自然是成本的一部分，而滑價其實也可以稱為成本。這是由於佣金是付給證券行的，而滑價是付給市場的，兩者都是成本，只是付出的對象不同。

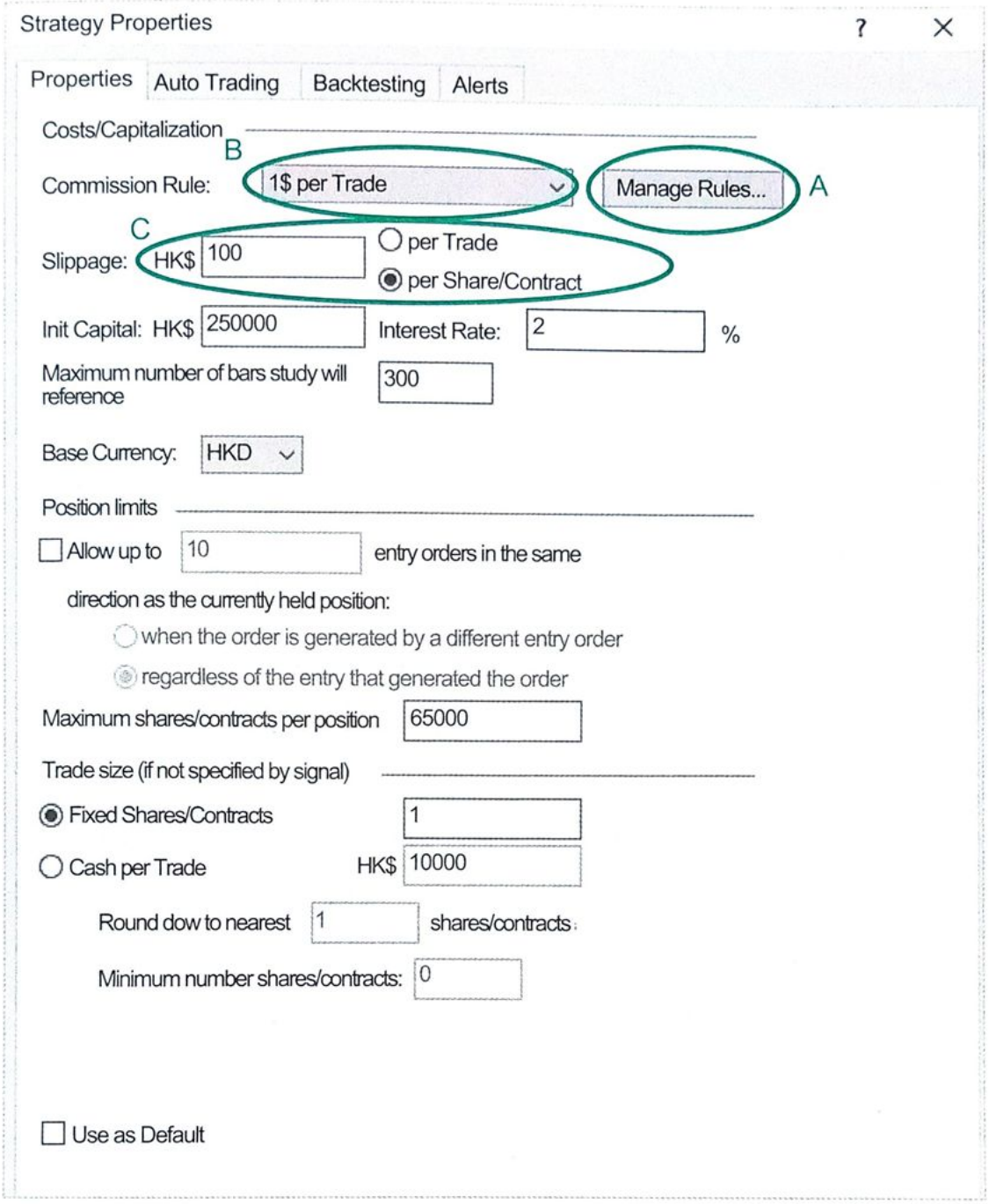
滑價是源於市價盤，或是所有不問價的交易指令，當產生交易條件的價格與實際成交的價格存在差距，就會出現滑價。舉例說，某策略以恒指期貨交易，指定升穿 30,000 點大關便買入；而當價格到達 30,000 點時，期貨的 Bid Ask 報價可能是 30,000 點和 30,002 點，如果用市價盤追入，付出的價格就會比觸發條件的 30,000 點高於 2 點，而這 2 點就是滑價。我們可以理解這 2 點是

我們願意付出市場的額外成本，讓我們的指令可以立即成交，因此筆者會將佣金和滑價統稱為交易成本。

設定交易成本



在主畫面左上角可以找到 Strategy Properties，這裡是控制策略預設設定的地方，包括交易張數、佣金、滑價等數據。我們先了解一下關於交易成本的設定，其餘設定將會稍後再講。



圈 A 是用作管理佣金的設定；完成設定後，交易者可於圈 B 內選擇佣金規則，圖中所顯示的「1$ per trade」就是該規則的名稱。

Tiered Commission Rule

Rule Name: 1$ per Trade

Volume in: Cash

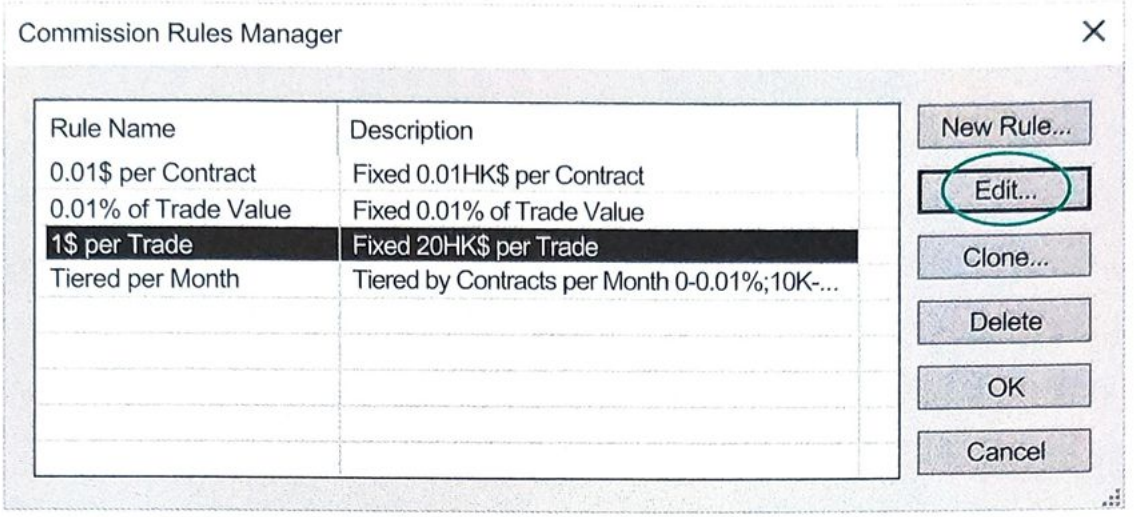
Reset Volume Counter: after every trade

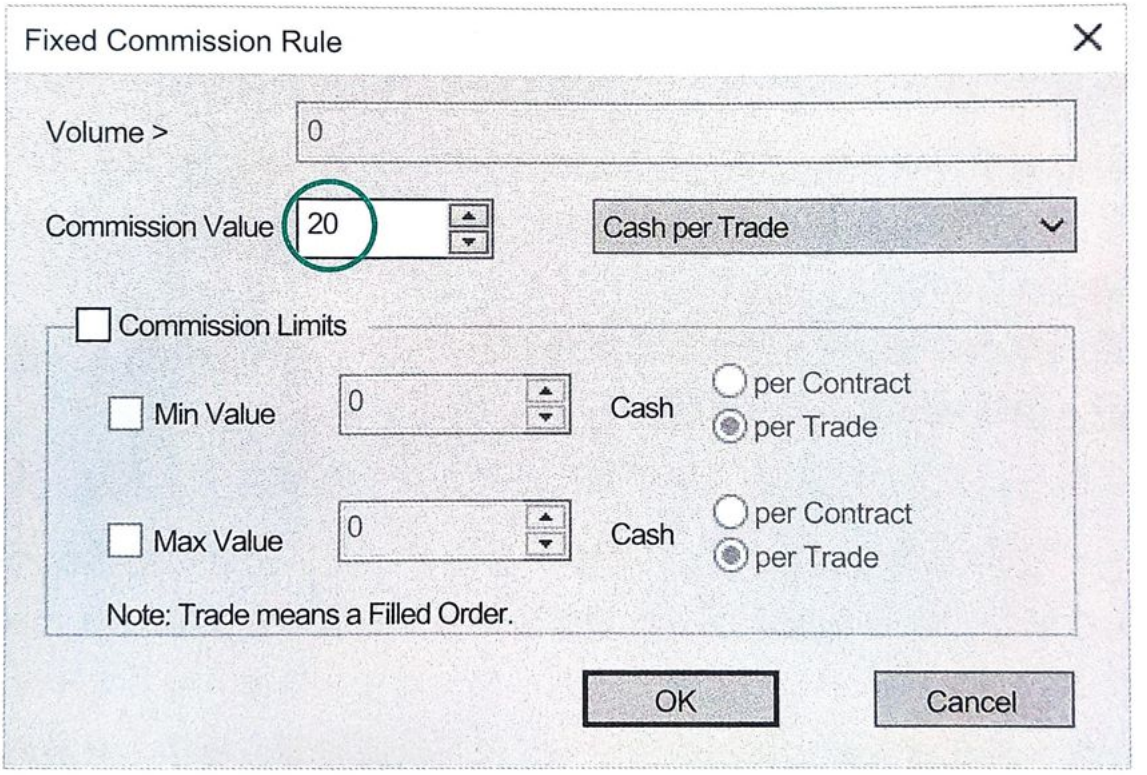
Calculate Commission: in real-time

Volume	Commision	Min Value	Max Value
>0	20 Cash per Tr...	N/A	N/A

Add... | Edit... | Delete

OK | Cancel


Commission Rules Manager
Rule Name
Description
0.01$ per Contract
Fixed 0.01HK$ per Contract
0.01% of Trade Value
Fixed 0.01% of Trade Value
1$ per Trade
Fixed 20HK$ per Trade
Tiered per Month
Tiered by Contracts per Month 0-0.01%;10K-...
New Rule...
Edit...
Clone...
Delete
OK
Cancel


Fixed Commission Rule
Volume >
0
Commission Value
20
Cash per Trade
Commission Limits
Min Value
0
Cash
per Contract
per Trade
Max Value
0
Cash
per Contract
per Trade
Note: Trade means a Filled Order.
OK
Cancel

如圖所示，在按下圈A後一直按Edit，便會進入Fixed Commission Rule，佣金金額的設定就是圓圈的位置，注意這裡不可以選擇貨幣種類，系統會按交易資產的貨幣設定計算佣金。此外，除了付給證券行的佣金外，交易者還要支付證監會徵費、交易所費用、政府印花稅等不同的收費，而由於這些收費通常是固定的，可以連同證券行的佣金一起計算。在這例子中，我們設定恒指期貨的單邊佣金收費為 $20，當中 $10 是港交所費用，另外 $9 是 IB 的佣金，還有極小的尾數是證監會和政府收費。

滑價的設定就在圈 C 內。在 MC 的回測系統中，MC 會以這裡設定的單一滑計作為所有滑價的預設數值，除了限價盤（Limit Order）外，市價盤（Market）和止蝕盤（Stop）都會產生滑價，所有利潤和虧損都會扣除此滑價數值金額，可見滑價只會帶來負面影響。此外，在圈 C 的右邊有兩個選項：「per Trade」和「per Share/Contract」；大家一定要選擇後者，若果選擇前者，即使交易張數高達 10 張，總滑價依然是 $100，平均每張就只得 $10 滑價，很明顯這是不設實際的。

至於滑價金額的設定，又是一個很大學問。在進行回測時，MC 會設定這一個數字為所有滑價的金額，然而在實際交易中，不同的市況會產生不同的滑價，例如，恒指期貨的滑價一般只有 1 至 2 點，但在市況變化激烈的時間，滑價可以高達雙位數字，將平均滑價數值拉高。另外，交易的張數亦會影響滑價，因為香港市場的深度有限，假設單一市價盤的交易張數高達數十張，Best Bid 和

Best Ask 都不足夠應付，可能在下一個和下下個排隊價格才完成交易，導致滑價的金額上升。

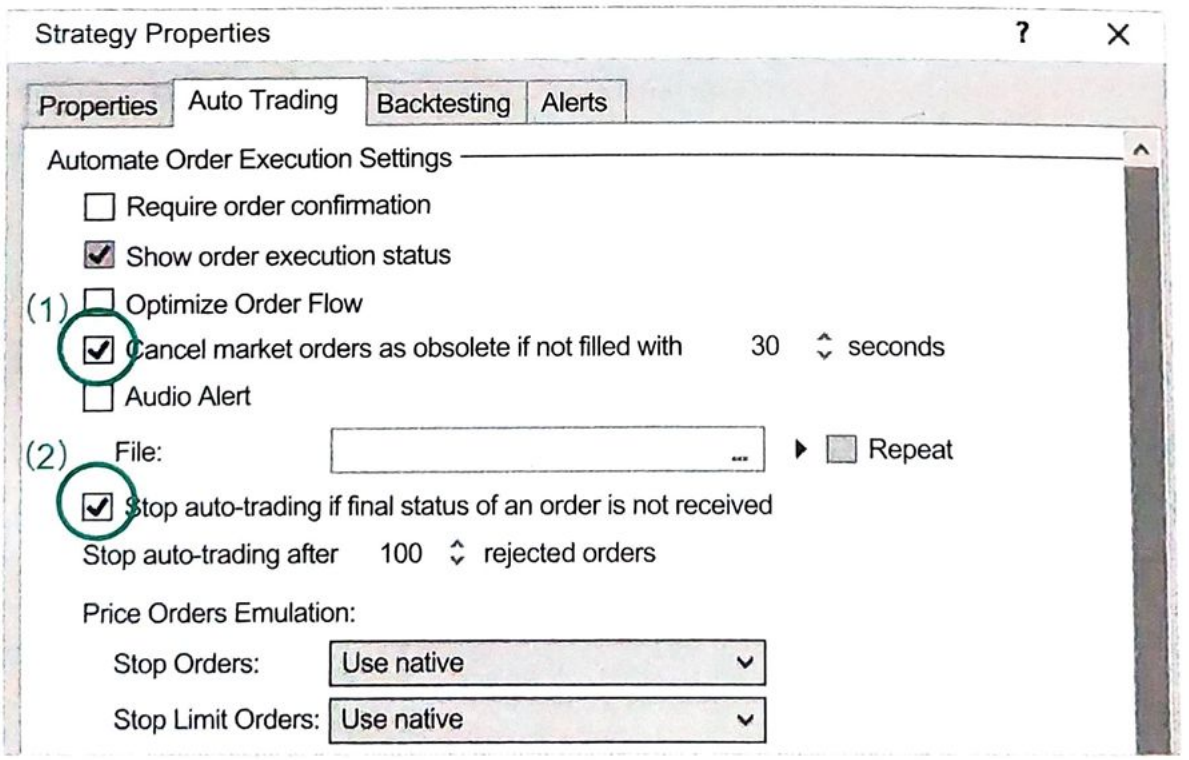
因此，設定太低的滑價數字，交易成本就可能被低估。一般來說，如果交易張數在 10 張以下，滑價設定為 2 點至 3 點都算合理，因為這數字是每張的單邊設定，一個 3 點的設定於一個來回交易（Round Trade）就是 6 點的滑價成本。由此可見，滑價對交易表現的影響是巨大的，一個來回交易可產生 6 點的滑價，10 個交易就會有 60 點滑價，即是在該交易策略，10 個交易必須賺到 60 點才能僅僅回本。因此，若說交易成本是交易中最重要的因素，實在毫不誇張。

既然滑價如此危害表現，減少滑價就顯得非常重要。最簡單的做法就是減少使用市價盤，尤其是開倉的指令，應盡量以限價盤代替，因為即使開倉時的限價盤未能成交，其結果可能是獲利或虧損，未能成交的結果不一定是壞的。不過，平倉時就很難採用限價盤了，因為如果是限價止蝕盤，當交易無法完成，由於虧損會擴大，結果必然是壞的；而如果是限價止賺盤，無法成交亦必然會令獲利減少。因此，平倉指令還是採用市價盤較佳，以免因小失大。

〈 4.6 〉

自動交易設定

最後一個要認識的設定，是關乎自動交易的部分，會直接影響真錢交易時的表現，而交易者在此需要作主觀決定。值得注意是，這裡的設定所帶來的影響並不能在回測中觀察，交易者如果對設定沒有概念，最好先以 Forward Run 的形式試行，觀察一下設定對表現有甚麼影響，因為不同類型的策略可能需要配合不同的設定，沒有一套設定可應用於所有情況。



Strategy Properties

Properties | Auto Trading | Backtesting | Alerts

Stop Limit Orders: Use native

Limit Orders: Use native

(3)

Unfilled Strategy Order Replacement

- [ ] Convert unexpected limit/stop entry orders to market orders after 0 seconds
- [ ] Convert unexpected limit/stop exit orders to market orders after 0 seconds

Partially Filled Orders Replacement

- [x] Convert partially filled limit/stop entry orders to market orders after 30 seconds
- [x] Convert partially filled limit/stop exit orders to market orders after 30 seconds

Recalculate on Broker Events

- [x] Market Position Change
- [x] Order Filled
- [ ] Order Rejected

Get Real-Time Data

- (•) Broker
- ( ) Chart

(4)

Mode Selection

- (•) Entries/Exits are based on the execution confirmation by the broker (Sync)
- ( ) Entries/Exits are independent of the execution status at the broker (Async)

Assign the Initial Market Position at the Broker Settings

- (•) Show the Assign the Initial Market Position at the Broker dialogue
  - [ ] Show always
- ( ) Do not show the Assign the Initial Market Position at the Broker dialogue

(5)

- (•) Assume the initial market position at the broker FLAT
- ( ) Assume the initial market position at the broker the SAME AS on the CHART
- ( ) Use the actual position at the broker

Select Broker Plug-in

Interactive Brokers

- [ ] Use as Default

確定 取消

回到 Strategy Properties，選擇 Auto Trading 後會出現相當多的設定，其中一些非常重要，執行自動交易前應該首先設定好。留意以下解說的設定都必須手動控制，至於沒有在此解說的設定則可沿用預設設定。

(1) Cancel market order as obsolete if not filled in _____ seconds

市價盤有機會成交不到嗎？答案是有可能的，不是每種資產都有足夠的市場深度，甚至連基本的 Bid Ask 都不能維持，所以連 Market Order 也不能成交。不過，對於一般散戶，遇上這情況的機會不大，反而是證券行斷線的機會多一點。證券行伺服器斷線會導致 Market Order 不能成交，保險起見當然是取消交易指令，以避免承受不必要的風險。

(2) Stop auto trading if final status of an order is not received

意思與剛才有點分別，不過目的相同。大多數證券行的 API 設定都會定期回覆 Order Status。如果用戶無法接收到 API 的 Order Status，斷線的機會十分大了，為保險起見，當然最好停止自動交易。

(3) Order Replacement

這設定極之重要！還記得先前我們提到滑價的影響嗎？由於滑價對策略表現的影響實在太大，所有交易者都希望盡量減少滑價金額，最簡單的做法就是開倉時用限價盤取代市價盤，不過限價盤始終有可能未法完成交易，所以 MC 系統設

計出 Order Replacement 的設定，讓交易者可以在若干秒數後將未能成交的限價盤 / 止蝕盤變成市價盤。

該設定將未能成交的情況分為 4 種，分別是完全未能成交的「Unfilled Strategy」、部分未能成交的「Partially Filled Strategy」，然後再細分為開倉的「Entry Order」及平倉的「Exit Order」。如果以未能平倉的嚴重性排列，最需要進行 Order Replacement 的必然是 Partially Filled Exit Order，其次是 Unfilled Exit Order，然後才是 Partially Filled Entry Order 及 Unfilled Entry Order。對有使用限價盤或止蝕盤的交易者來說，這個設定是必須調整的。

(4) Mode Selection

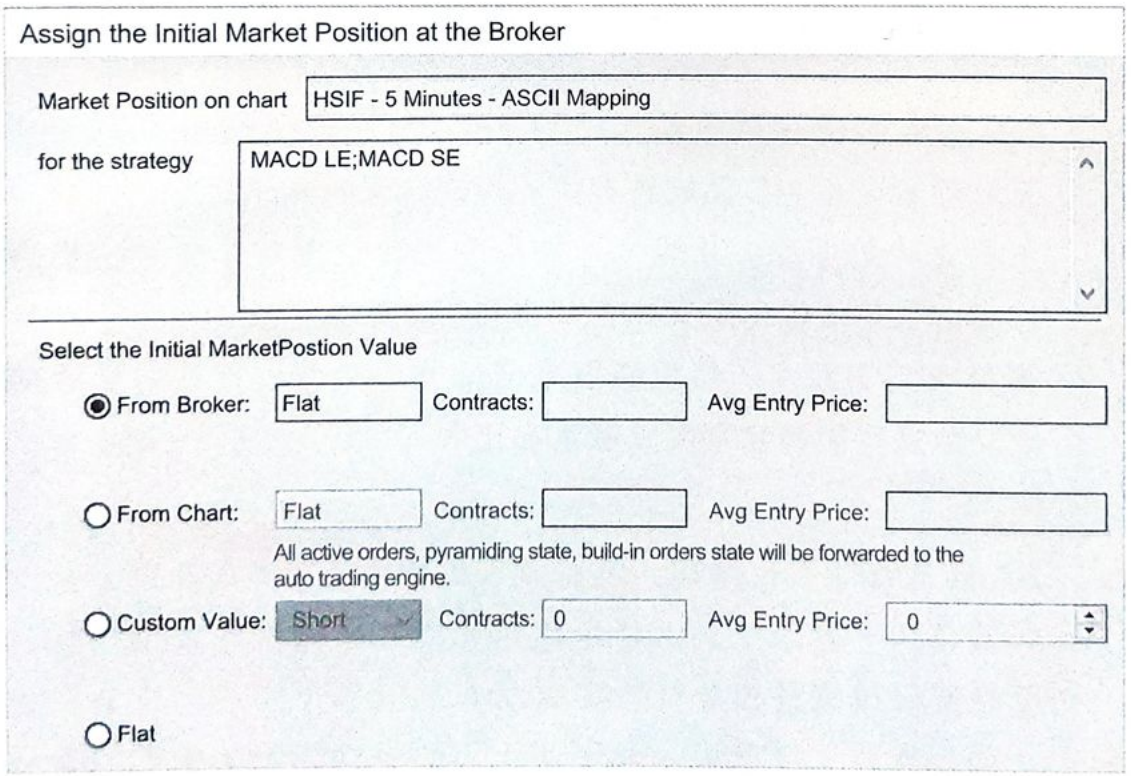
MC 的自動交易可分為 SA（Sync）和 AA（Async）兩種模式，前者是指 MC 系統會與證券行的持倉張數、成交價同步，所有之後的開倉、止賺、止蝕指令都會以實際的交易所數據作準；至於後者，成交數據則完全根據系統預設的設定，並非實際成交的數字，若果採用限價盤或止蝕盤，系統所預設的成交數據可能與實際成交數字有出入。

AA 模式看似不合情理，沒有選擇的理由，但 SA 模式卻有一個致命傷，就是由於 MC 系統對於成交數據是沒有記憶的，當需要關掉電腦和重新開啟 MC 後，SA 模式下的 MC 會失去之前的實際交易數據，產生了不可遇視的問題。但這問題不

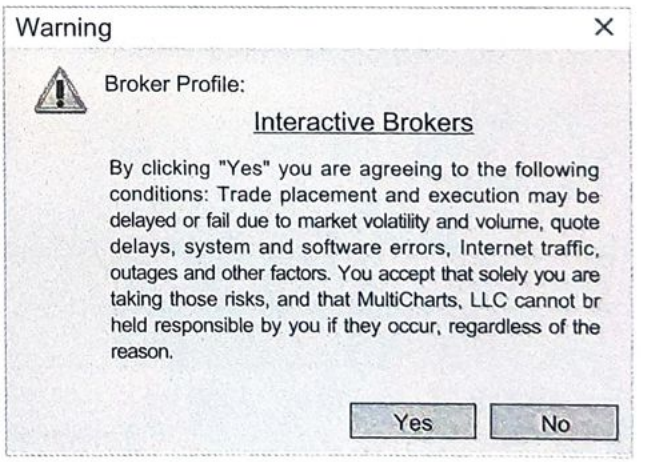
會出現有的 AA 模式，因為 AA 模式完全根據系統預設的設定成交數字，即使重新開啟 MC，系統仍然可以推算出之前成交過的倉位，繼續執行交易策略。

SA 模式與 AA 模式之爭是許多 MC 用戶的長期爭拗點，兩者都有其優缺點。筆者認為，初學者最好先採用 AA 模式，盡量避免無法成交的情況。例如，多用市價盤和減少 Order Replacement 等待轉換秒數，藉以減少預設與實際倉位產生差距的機會。

(5) Initial Market position at Broker Setting

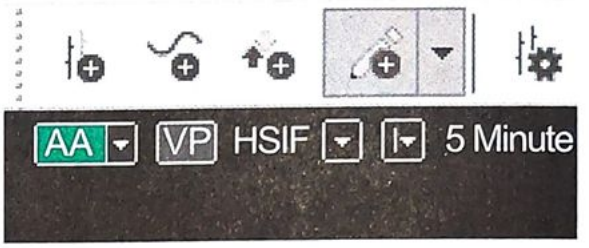


最後是關於初始倉位的設定。這個設定容許交易者在開始交易前對 MC 設定當時的倉位狀態，分別是 From Broker、From Chart、Custom Value 和 Flat。筆者建議開倉前最好保持持倉為零，但在某些情況下，可能證券行內本身已經有持倉，而該持倉與程式交易無關，這樣便需要進行這個設置。留意在 Strategy Properties 內只是設定開啟自動交易前彈出這個視窗，而非在 Strategy Properties 內直接設定 Initial Market Position。

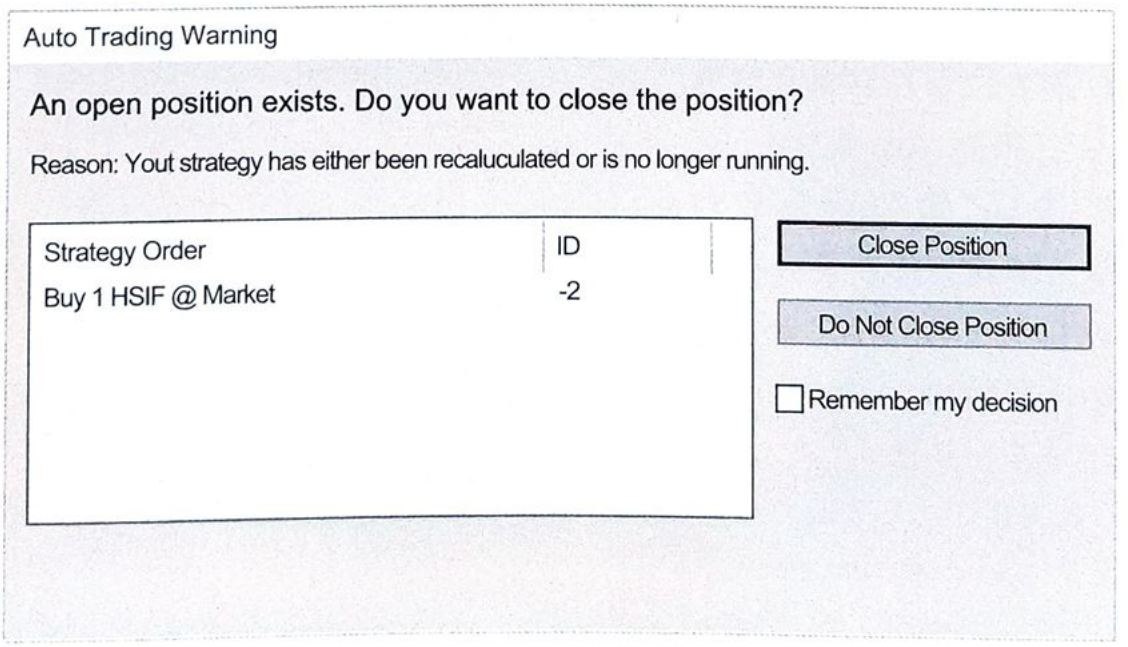


完成一切設定後，就可以啟動自動交易。而啟動前，請先確認已經在同一部電腦上登入了 IB TWS，並且設定 MC 和 IB 使用相同的 Port Number。MC 方面，啟動的方法就是點擊主畫面左上角的 AA 或 SA，這時候 MC 會彈出警告字樣，提醒你即將要進行真錢交易，確認後又會彈出剛才提到的 Initial Market Position 設定視窗，完成設定才會真正開始自動交易。

留意當 MC 進入自動交易模式後，整個畫面的唯一標示就只有主畫面左上角的 AA 或 SA，啟動後會變成綠底白字，除此之外就沒有其他提示了，所以這是一個相當考眼力的提示。



如果想結束自動交易，可以再按一下這個綠底白字的 AA 或 SA。在沒有持倉的情況下，自動交易會直接停止；但若果手上有持倉，或者圖示顯示現時手上應該有持倉（AA 模式），MC 就會彈出一個平倉選項，讓你決定是否將手上倉位平倉。筆者建議不要選擇 remember my decision，因為在 AA 模式下，你的真實持倉與圖表持倉未必相同，如果沒有持倉，MC 幫你「平倉」，這就等於變成開倉。



第 5 章
基本交易程式編寫

- Power Language 基本語法
- 止賺與止蝕
- 常用函數
- 倉位控制
- 小結

〈 5.1 〉

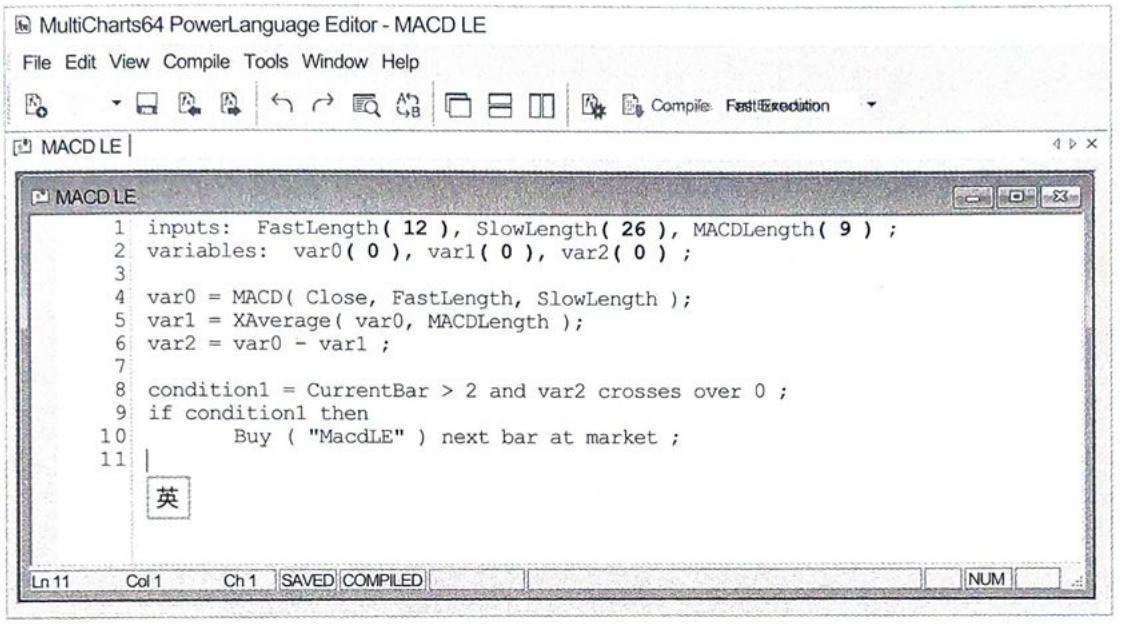
Power Language 基本語法

編寫程式是一門博大精深的學問，而策略編寫則是一個非常深入的題目。本章節的目的是帶領讀者了解編程的基本知識，如果大家希望進一步研究策略編寫，建議閱讀專門講解編程的書籍或修讀相關課程。另外請注意，本書內所有策略僅作示範教學用途，絕不建議讀者直接使用。

本章節會繼續以 MultiCharts（MC）作示範，所採用的程式語言為 Power Language。對於一些基本策略，不同平台的編程邏輯其實都大同小異，相異之處通常只是程式的語法，要去到較高階的交易策略，如波幅率交易、配對交易、期權交易等，大家才需要使用傳統的程式語言。

了解訊號結構

MC 內置了許多策略，全部都放於 Netvigator 的 Signal（訊號）資料夾內，因此在 MC 系統內，策略和訊號的意義是共通的，而雖然這些訊號未有止賺止蝕、倉位控制等元素，但結構上已經算是完整。例如是下圖的MACD LE訊號，大致上可以分為3個區域。



(1) 宣告區：

```
inputs: FastLength( 12 ), SlowLength( 26 ),
MACDLength( 9 ) ;
variables: var0( 0 ), var1( 0 ), var2( 0 ) ;
```

宣告區中的input是定義參數的名稱和數值。在這例子中，MACD的參數名稱是FastLength、SlowLength和MACDLength，而數值則是常見的12-26-9。宣告參數的作用是在計算時不需直接輸入數值，只需輸入參數名稱，電腦就會知道該名稱所代表的數值；更重要的是只有在input宣告過的參數才能在MC進行優化，所以所有需進行優化的參數都要在input中宣告。

至於 variable，宣告的方式與 input 相近，但宣告時的數值通常是 0。這是由於 variable 的角色與 input 不同，variable 是策略計算過程中所使用的媒介。當訊號在實際運行時，它會隨著即市價格變動，而宣告為 0 的作用只是定義該變數的類型。至於變數的類型，除了數值外，常見的還有 boolean（True / False）和文字（String）。

值得留意是，每個宣告完成的句子後都要加上分號（;）。在 Power Language，分號的角色就像在英語中的 full stop 一樣，表示句子的結束。許多新手都經常忘記在句末加上分號，而這亦是部分程式碼不能編譯的常見原因。

（2）計算區：

```
var0 = MACD( Close, FastLength, SlowLength ) ;
var1 = XAverage( var0, MACDLength ) ;
var2 = var0 - var1 ;
```

計算區會應用到在宣告區中的 input 和 variable。Variable 的名稱會放置於等號左邊，而等號右邊則是函數和相關的計算參數。在例子的第一句中， Close 是函數 MACD 的第一個參數，指最新一支 Bar 中的最後成交價，直接等於最新的市價，所以 var0 等於以市價和 MACD 函數計算出來的數值；而第二句的 var1 就是以 var0 和函數 XAverage 計算出來的數值；在最後一句中，var0 與 var1 相減會得出 var2 的數值。

留意是每一支 Bar 都有 Open、High、Low、Close，4 個有關價格的關鍵字。若果關鍵字後面有中括號及數字，當中的數字代表較之前的 Bar 的數值。例如 High[1] 代表之前第 1 支 Bar 的最高價，而 Open[10] 就代表之前第 10 支 Bar 的開市價；若果沒有中括號在後面，其預設意思是最新一支 bar 的數值，如在本例子中的 Close 就等於最新一支 bar 的最後成交價。

另外，在計算區出現的函數，已包含在 Power Language Editor（PLE）右邊的 Function 資料夾內。函數的運用方式非常簡單，只要寫出函數的名稱，再在括號內加入計算所需的參數，若有多於一個參數，便以逗號分開。

（3）執行區：

```
condition1 = CurrentBar > 2 and var2 cross over 0 ;
if condition1 then
Buy ( "MacdLE" ) next bar at market ;
```

這是一個執行區的開倉例子，當中包括開倉的條件 condition1，以及設定 condition1 成立時須執行的動作，即是 if …then statement 句式。首先，condition1 的意思不難理解，當中包括兩項條件，CurrentBar > 2 和 var2 cross over 0，兩者同時成立才算滿足 condition1，並可以執行 then 後的指令，即 Buy ("MacdLE") next bar at market。

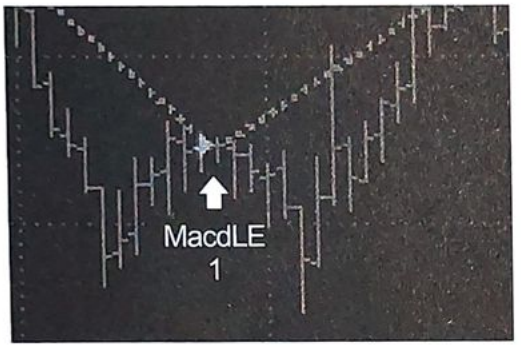
在絕大多數情況下，CurrentBar>2 都是成立的，所以我們可將重點放在 var2 crosses over 0。回想在計算區中，var2 是經過一連串的計算而得來的，而在計算過程中，函數 MACD 包含了最新價格 Close。在交易時段內，Close 的數值會不停變動，導致 var0、var1 和 var2 的數值也在轉變，直至有一刻 var2 從負數變成正數， var2 crosses over 0 的條件便會成立，從而令 condition1 成立。另外，大家要留意 and 的角色是連續兩個條件，凡有 and 出現，所有條件都要符合才能使 condition 成立，而很多時候一個 condition 內會有多於一個條件，所以裡面很可能出現一個甚至多個 and。

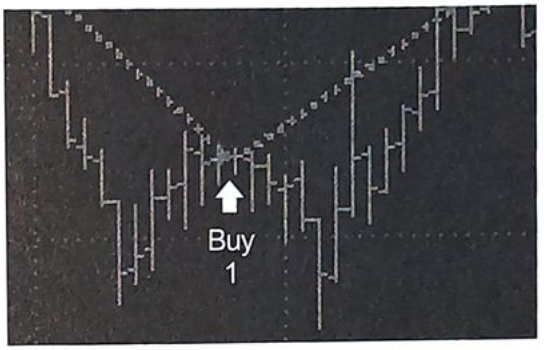
在最後的 if…then statement 中，關鍵是 then 後的動作，如 Buy("MacdLE") next bar at market。Buy 即是建立好倉的意思，除了 Buy 之外，還有 3 個交易動作的關鍵字：

Buy ：建立好倉
Sellshort ：建立淡倉
Sell ：平好倉
BuytoCover ：平淡倉

大家須留意這些關鍵字在意思上的分別。Buy 和 Sellshort 都是用於建倉，而就算原來手上有相反的倉位，例如本來手上有好倉，但只要執行 Sellshort，都會直接反手建淡倉。而 Sell 和 BuytoCover 都是用於平倉，不會建立任何倉位，所以若果手上沒有相對應的倉位，就自然不會有倉可平。

至於之後的（“MacdLE”），所指是主畫面顯示的訊號名稱，就如下圖所示，若果不寫入這名稱，顯示的訊號名稱就會變成「Buy」。





最後是 next bar at market，意思指「在下一支 Bar 剛出現時用市價交易」，所謂的下一支 Bar 剛出現，其實等同現在的 Bar 剛完結，以 resolution 為 1 分鐘的圖表作例，當秒數到達 60 秒時，就是 this bar 剛完結和 next bar 剛開始的時刻，程式會在此時執行 Buy next bar at market 的動作。我們一般都不會用 this bar 進行真錢自動交易，所以在回測時也會設定 next bar 才進行交易，盡量保持回測和真錢交易有一致的效果。

除了按市價盤進行交易的 market 之外，交易指令還有代表限價盤的 limit 和代表止蝕盤的 stop。這兩項的例子如下：

```
Sellshort next bar at 27000 limit;
Sellshort next bar at 27000 stop;
```

第一句的意思是以 27,000 點作為沽出限價盤，如果市價等於或高於 27,000 點，系統就會執行沽出動作；而第二句的意思是一旦市價跌穿 27,000 點，就會執行沽出動作。明顯地，兩者的用途完全不同，前者是寧願等也不願以較低的價格沽出，而後者則要見到跌勢才願意沽出。由於各項指令的意思完全不同，交易者可以按照自己的意願，選擇哪一項指令。例如，若果有非常強的成交意願，就應該採用多點 market 盤；而採用 limit 盤則可節省不少滑價成本。

〈 5.2 〉
止賺與止蝕

上一篇提到，內置的 MACE LE 訊號沒有止賺和止蝕的設定。但其實，即使是一個簡單而完整的交易訊號，最起碼也應該包含止賺和止蝕，因為這樣的交易策略才可以令交易訊號不斷重複，達至重複交易的效果。

MC 有很多內置的止賺止蝕指令，而最常用的有以下 3 個：

setprofittarget
setstoploss
setpercenttailing

以上一篇的 MACD LE 訊號為例，雖然訊號本身沒有止賺與止蝕的設定，但我們可以在程式碼最後加入上述的程式碼，就能令整個策略變得完整。

```
inputs:  FastLength( 12 ), SlowLength( 26 ), MACDLength( 9 ) ;
variables:  var0( 0 ), var1( 0 ), var2( 0 ) ;

var0 = MACD( Close, FastLength, SlowLength ) ;
var1 = XAverage( var0, MACDLength ) ;
var2 = var0 - var1 ;

condition1 = CurrentBar > 2 and var2 cross over 0 ;
if condition1 then
        Buy next bar at market ;

setprofittarget(5000);
setstoploss(2500);
```

最後補充的兩行程式碼的意思是，程式設定以 $5,000 作止賺和 $2,500 作止蝕。從這例子可見，setprofittarget 和 setstoploss 都是預設接受以銀碼作為單位，就如括號內的 5000 和 2000。而若果交易者想採用其他形式作止賺止蝕，就要配合原先參數的格式作變化。舉例說，若果希望以開倉價 n% 作止賺止蝕，可以修改以下程式碼。

```
inputs: FastLength( 12 ), SlowLength( 26 ), MACDLength( 9 ),
SL( 0.01 ), SPT( 0 .01 ) ;
variables:  var0( 0 ), var1( 0 ), var2( 0 ) ,start_price( 0 );

var0 = MACD( Close, FastLength, SlowLength ) ;
var1 = XAverage( var0, MACDLength ) ;
var2 = var0 - var1 ;

condition1 = CurrentBar > 2 and var2 cross over 0 ;
if condition1 then begin
        buy next bar at market ;
        start_price = close;
end;

setprofittarget(start_price*SPT*50);
setstoploss(start_price*SL*50);
```

在宣告區中，我們新增了一個變數 start_price，並在 if…then…begin…end 中，按條件紀錄開倉時的市價。begin…end 的寫法其實只是 if…then…寫法的延伸。在之前提到的 if…then statement 中，系統只會在 Condition 成立時執行一個動作；在加上 begin…end 後，系統便可執行多個動作。

要理解紀錄 close 價格的邏輯其實不難，close 本身會每分每秒隨著市價而改變，但執行 start_price = close 這個動作只會在 condition1 成立的一刻（即交易時刻）才會進行，也就是說紀錄 start_price 與 buy next bar at market 這兩項動作會同步進行，start_price，只會紀錄當時的市價，亦即是開倉價，而在紀錄後就不會改變。

接著下來是解釋 start_price*SPT*50 和 start_price*SL*50。當中，start_price 紀錄的是開倉價，SPT 和 SL 在 input 中宣告為 0.01，即代表 1%，而 50 則是每點期指的價格。假設 start_price 是 30,000 點，在括號內得出的數值就是 $15,000（=30000*0.01*50），代表開倉價 1%（即 300 點期指）的銀碼。

由此可見，setprofittarget 和 setstoploss 原本接受的參數是以銀碼作單位，但透過紀錄開倉價，並在 input 中宣告 SL 和 SPT 兩個以百分比為單位的參數，令 setprofittarget 和 setstoploss 可以開倉價的百分比設定止賺和止蝕。此外，由於 SL 和 SPT 是在 input 內宣告，代表這兩個百分比亦可以在 MC 進行優化。

此外，止賺和止蝕的寫法也不限於 setprofittarget 和 setstoploss，可以完全以 if…then…的寫法代替。

```
inputs:  FastLength( 12 ), SlowLength( 26 ), MACDLength( 9 ),
SL(0.01), SPT(0.01) ;
variables:  var0( 0 ), var1( 0 ), var2( 0 ) ,start_price( 0 );

var0 = MACD( Close, FastLength, SlowLength ) ;
var1 = XAverage( var0, MACDLength ) ;
var2 = var0 - var1 ;

condition1 = CurrentBar > 2 and var2 cross over 0 ;
if condition1 then begin
        buy next bar at market ;
        start_price = close;
end;

if close - start_price > start_price*SPT then
        sell next bar at market;

if start_price – close > start_price*SL then
        sell next bar at market;
```

在最後的兩段程式中，我們可以見到，close-start_price > start_price*SPT 明顯是止賺的條件，而 start_price-close > start_price*SL 則是止蝕的條件，由於 close 代表的是最新的市

價，前者要成立的話，市價必須高於開市價，而後者要成立則必須低於開市價，形成了止賺止蝕的訊號。

setprofittartget、setstoploss 和 setpercentrailing 的問題是，如果一個策略包含多於一個開倉訊號，所有開倉訊號的止賺止蝕都會以同一組數據運作，所以我們才須用 if…then…的方式訂立止賺和止蝕。

至於先前提到的 setpercentrailing，都有同樣的問題，就是同一訊號裡的開倉指示都需要依從一組止賺數值。不過，setpercentrailing 比 setprofittarget 更適用於趨勢策略，因為 setpercentrailing 有追止賺的特性，從以下例子可以了解得更清楚。

```
inputs:  FastLength( 12 ), SlowLength( 26 ), MACDLength( 9 ),
SL( 0.01 ), SPT1( 0.01 ), SPT2( 20 ) ;
variables:  var0( 0 ), var1( 0 ), var2( 0 ) ,start_price( 0 );

var0 = MACD( Close, FastLength, SlowLength ) ;
var1 = XAverage( var0, MACDLength ) ;
var2 = var0 - var1 ;

condition1 = CurrentBar > 2 and var2 cross over 0 ;
if condition1 then begin
        buy next bar at market ;
        start_price = close;
end;

setpercenttrailing(start_price*SPT1*50,SPT2);
setstoploss(start_price*SL*50);
```

這次我們在 input 設定了 SPT1 和 SPT2 兩組止賺參數，SPT1 維持同樣是百分比的數值，即 1%，而 SPT2 則代表 20%。而 setpercentrailing 所表達的是，當賺幅達到 start_price*SPT1*50 的金額後，若果價格回撤達到 20%，就會執行止賺；也就是說，如果價格一直維持向上趨勢，回撤幅度一直不多於 20%，止賺位可以永無止境地推進，直至某個時候回撤幅度達到 20% 為止。由此可見，setpercenttrailing 是一個趨勢策略中非常有用的止賺方法。

止賺止蝕是交易策略中一個頗為深奧的題目，讀者們可以先嘗試本篇例子中的編程方法，對當中的邏輯有充分了解之後，不妨自己設計適用的止賺止蝕方法，最重要的是知道編寫的方法不限於本篇例子中的寫法。

〈 5.3 〉
常用函數

MC 有許多內置的函數，交易者很少需要自行設計函數，光是使用內置函數已經十分夠用。本篇會集中介紹一些常用的函數，建議讀者毋須死記硬背，反而要了解函數的運作原理，因為多數的函數除了本身的用法之外，其實可以和不同的變數或其他函數互相配合，不一定只有預設的用法。

AverageFC

不少技術指標都須計算簡單平均數。例如，保力加通道的中線就是 20 天簡單平均線。雖然 MC 已內置了 Average 函數，但絕大部分的內置指標都會用 AverageFC 去計算平均數，而非 Average。這是由於 AverageFC 中的 FC 意指 Fast Calculation，其編寫方法比 Average 更有效率，所以人們在計算平均數時都會使用 AverageFC。

```
input: len1( 20 );
var: avg( 0 );

avg = AverageFC( Close, len1 );

condition1 = close cross over avg;
if condition1 then
        buy next bar at market;
```

這例子是 AverageFC 的簡單寫法。在例子中，兩個參數分別是 Close 和 len1，Close 通常是指價格，而 len1 則指長度，是可優化的 input。其實，AverageFC 不限於計算價格的平均，任何數值都可以利用 AverageFC 計算出其平均數，我們會在往後的例子再作探討。

另外，大家可能留意到這次 condition1 出現的條件是 cross over，與上一篇 MACD LE 所用到的一樣。其實，與 cross over 相對應的還有 cross under，兩者都是 Power Language 語法中的 Operator，與大家所理解的「>」（大於）和「<」（小於）一樣，都可用於條件設定，分別就是「>」的用法沒有「升穿」的意思。假設這個例子是採用 1 分鐘 resolution 的圖表，cross over 是指「上一分鐘的 close 小於 avg，這一分鐘的 close 大於 avg」，這才是英語中 cross over 的真正意思。由於 MC 系統會根據圖表 resolution 作為其 bar-based 的 refresh rate，若果採用「>」而

非 cross over，MC 會每 1 分鐘檢查一次 close 是否高於 avg，高於的話 condition1 就會成立，然後每 1 分鐘執行一次 buy next bar at market，但這很明顯不是交易者的原意。因此，cross over 和 cross under 這兩個孖寶，以及 AverageFC 是經常同時出現的。

StandardDev

另一個常見的函數應該是 StandardDev 了。這個函數返回的值是標準差，是很多技術指標和統計都需要的數值，最經典的莫過於保力加通道了。雖然筆者不會用保力加通道，但無可否認保力加通道的原理是相當有啟發性的，十分適合用於學習的材料。

```
input: len1( 20 ), n( 2 );
var: avg( 0 ),sd( 0 );

avg = AverageFC( Close, len1 );
sd = StandardDev( close, len1, 1 );

condition1 = close cross under avg - n*( sd );
if condition1 then
        buy next bar at market;
```

在這例子中，我們在 input 加入了 n，又在變數加入 sd。後者是 StandardDev 函數所得的標準差數值。StandardDev 函數有 3 個參數，除了價格 Close 和長度 len1 外，最後的 1 是函數所需的 datatype 參數，1 代表統計標準差的數據是統計學中的 Population，0 則代表採用的數據是 Sample。簡單來說，當中涉及的統計學概念是，Population 會比 Sample 多一個 Degree of Freedom。一般而言，我們會假設所採用的數據是 Sample，所以 StandardDev 的最後一個參數通常設定為 0。

這例子的意思十分簡單，就是如果即價跌穿平均數減兩個標準差的話，就執行買入的動作。留意是 len1 會同時在 avg 和 sd 的計算中出現，若果將 len1 進行優化，優化結果會同時應用於 avg 和 sd 的計算中。而這個設定不是強制的，我們可以在 input 宣告另一個名為 len2 的參數，並將 avg 和 sd 所用的長度數值分開。不過，平均數和標準差在配合使用時，一般會採用相同的長度數值，所以比較常見的做法是兩者同時使用 len1 作為長度的參數。

OpenD、HighD、LowD、CloseD

這 4 個函數是十分有用的價格函數，因為不少 MC 用家都是以分鐘級的 resolution 進行交易，而如果在這 resolution 下想要存取之前日子的 Open High Low Close，就需要以上 4 個函數。在這裡我們以一個簡單的突破寫法作例子。

```
condition1 = close cross over HighD(1);
if condition1 then
        buy next bar at market;
```

這個策略簡單到連 input 和 variable 都不需要。HighD(1) 所代表的是昨天的高位，括號內的 1 使函數返回的是昨天的數值。當然括號內的數值不一定是 1，如改寫成 10［即 HighD(10)］，系統就會返回第 10 天前的高位，同樣道理也適用於其餘 3 個函數。

有些交易者會以昨天的波幅去決定今天是否採用一些趨勢策略，簡單如價格升穿平均線，已經屬於趨勢策略了。因此交易訊號可以配合 HighD(1) 和 LowD(1) 使用，用昨天的高低波幅作為開倉條件之一。

```
input: len1( 20 ), vol( 500 );
var: avg( 0 ),prev_day_vol( 0 );

avg = AverageFC( Close, len1 );
prev_day_vol = HighD( 1 ) - LowD( 1 );

condition1 = prev_day_vol > vol and close cross over avg;
if condition1 then
        buy next bar at market;
```

留意這 4 個函數只需一個參數，初學者有時會不小心與紀錄上 n 支 bar 的 open[n]、high[n]、low[n]、close[n] 等價格程式碼混淆。需知道使用中括號的寫法會受到主畫面設置的 resolution 所影響，例如 high[1] 在 1 分鐘圖表上代表之前 1 分鐘的高位；而在 1 小時圖表上，它就會代表之前 1 小時的高位，返回的數值會隨著 resolution 的分別而不同。但 HighD(1) 所使用的是小括號，反映其性質是一個函數，只要每支 bar 的時間不多於一天，即使圖表採用了不同的 resolution ，返回的都必然是昨天的最高位價格。

順帶一題，除了以上這 4 個之外，另有其他性質類似的函數，分別是：

返回第 n 星期前價格的：

OpenW、HighW、LowW、CloseW

返回第 n 月前價格的：

OpenM、HighM、LowM、CloseM

返回第 n 年前價格的：

OpenY、HighY、LowY、CloseY

除了這些常用函數之外，MC 還內置了許多常見的傳統技術指標函數，如 RSI、MACD 等。由於數量太多，難以在此逐一詳解，但筆者希望透過以下例子，說明函數之間是有可能互相配合的。

```
input: len1( 14 ), len2( 20 );
var: RSI_value( 0 ), RSI_avg( 0 );

RSI_value = RSI( Close, len1 );
RSI_avg = AverageFC( RSI_value,len2 );

condition1 = RSI_value cross over RSI_avg;
if condition1 then
        buy next bar at market;
```

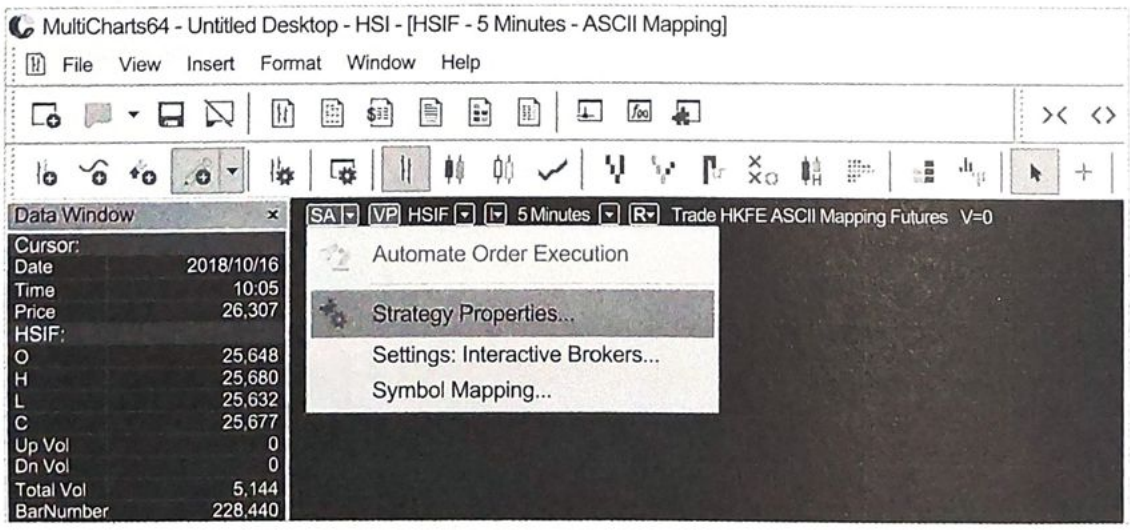
看見了嗎？AverageFC 不只限於計算價格的平均值，也可以計算 RSI 的平均值。這不是傳統技術分析的策略，對於想進行程式交易的新手，可以籍著本例子思考一下，其實數據的處理有很多的可能性，要在市場中成為少數的贏家，思維不能拘泥於傳統的分析方法。

〈 5.4 〉

倉位控制

倉位控制看似不及策略邏輯、止賺止蝕等指令重要，所以許多初學者都很容易忽略倉位控制，但其實，如果欠缺倉位控制，策略會存有嚴重的問題。其中最大問題是回測結果中的風險會被低估，後果是不堪設想的。另外，我們可能想把較多的資金執行較有把握的策略，同時把較少的資金執行較小把握的策略。因此，我們須要把交易張數宣告為可優化的 input，這樣可以提高策略的靈活度和彈性。

交易數量



先前我們在 Strategy Properties 裡面探討過交易成本的設定，現在我們回到 Strategy Properties，但這次要了解的是交易張數的設定。

Strategy Properties

Properties | Auto Trading | Backtesting | Alerts

Costs/Capitalization

Commission Rule: 1$ per Trade | Manage Rules...

Slippage: HK$ 100 ○ per Trade ◉ per Share/Contract

(1)

Init Capital: HK$ 250000 Interest Rate: 2 %

Maximum number of bars study will reference 300

Base Currency: HKD

Position limits (2)

☐ Allow up to 10 entry orders in the same direction as the currently held position:

(3) ○ when the order is generated by a different entry order

(4) ◉ regardless of the entry that generated the order

Maximum shares/contracts per position 65000

Trade size (if not specified bt signal)

◉ Fixed Shares/Contracts (5) 1

○ Cash per Trade HK$ 10000

Round down to neares 1 shares/contracts

Minimum number shares/contracts: 0

☐ Use as Default

與交易張數相關的設定：

（1） Int Capital ：

系統假設的起始資金，作為回測時的資金起步點，請盡量填寫與你計劃中打算使用的資金，讓回測貼近現實；

（2） Position Limit ：

這裡可設定最大張數的數量，只要持倉（好倉或淡倉）達到該數量，即使再有交易指示，都不會繼續開倉；

（3） when the order is generated by a different entry order ：

意思是同一個訊號內，可能有多於一個 if…then…的交易指令，而這些交易指令的方向是一樣的。如果選擇該選項，即使相同的 if…then…指令重複出現都不會產生更多的持倉；

（4） regardless of the entry that generate the order ：

這選項是與上一個選項是相對的，即是說如果同一個 if…then…交易指令重複出現，系統容許增加持倉。最典型的例子是一個普通的升穿平均線買入的指令，若果市價不停在平均線來回穿插，根據這個選項就會不停開倉，直到持倉量達到 position limit 為止。反之，上一個選項則只會開一次倉，就是首次滿足升穿平均線條件的時候；

（5） Trade size (if not specified by signal), fixed shares/contracts ：

在先前的例子中，我們都沒有指定開倉的張數，所以預設的開倉張數會就根據這裡的設定，留意如果在程式碼中指定開倉數量的話，該指定的數量就會取代這個預設設定。

讓我們再以平均線策略作例子，示範一下指定開倉張數的程式寫法。

```
input: len1( 20 );
var: avg( 0 );

avg = AverageFC( Close, len1 );

condition1 = close cross over avg;
condition2 = close cross under avg;
if condition1 then
        buy 2 share next bar at market;

if condition2 then
        sell next bar at market;
```

首先，程式碼內加入了「2 share」，代表開兩張合約，如果交易正股，則代表兩手股票。share 這個關鍵字與 contract 的意思

和用法完全一樣，筆者通常會用 share，因為字數較少。至於負責平倉的 sell 指令，雖然沒有指定張數，但在 Power Language 預設中，如果平倉指令 sell 和 buytocover 沒有指定張數，就代表將所有持倉平倉。

聰明的讀者看到「2 share」這個寫法時，應該聯想到將「2」變成一個 variable。還記得為何我們需要用變數嗎？就是由於該數字在程式運行時不是固定，會隨著不同的原因而變化。那麼，開倉的張數為何有會變化？這是與賺蝕幅度有關，隨著你賺錢愈來愈多，應該不會保持只交易一張期指吧！當賺得足夠多時，當然要增加交易數量，所以開倉張數須設定為變數。

要表達開倉張數隨著利潤增加，須用上 InitialCapital、netprofit 等關鍵字，分別代表起始資金和淨利。此外，我們還需要 intPortion 函數，藉以返回數字的整數部分，捨棄其小數位。

```
input: len1( 20 ),min_buying_power( 150000 );
var: avg( 0 ), n( 0 );

avg = AverageFC( Close, len1 );

if ( InitialCapital + netprofit )/min_buying_power > 1 then
        n = IntPortion(( InitialCapital + netprofit )/min_buying_power )
        else
        n = 1;

condition1 = close cross over avg;
condition2 = close cross under avg;
if condition1 then
        buy n share next bar at market;

if condition2 then
        sell next bar at market;
```

我們在 input 中加入了 min_buying_power，預設數值是 150000，即是 15 萬元。這意思是假設每多賺 15 萬元就增加一張恒指期貨的單位，而當中我們用了 if…then…else…的句式去決定 n 的數值，如果起始資金加淨利除以 15 萬超過某整數，我們就會以該整數作為 n。例如，起始資金為 15 萬、淨利是 20 萬，（15 萬 +20 萬）/15 萬 = 2.33，整數部分就是 2，交易的張數就會是 2。

而如果淨利為負值，就會執行 else 下一行的 n = 1，所以即使一起步時微蝕，都會維持最少 1 張的交易。

這個隨利潤增加的交易張數設定是必需的，如果某策略一直賺錢，卻一直只交易一張期貨合約，回測報告所顯示的捱價百分比就有可能被嚴重低估。試想像，當資金達 200 萬元時， 10 萬元的捱價幅度只佔 5%，但由於交易張數只得一張，若果該捱價金額出現於只有 30 萬資金，捱價幅度就會高達 33%。

除此之外，我們亦可以因應不同的條件，在同一個訊號內調整不同的張數。

```
input: len1(14);
Var: RSI_value(0),start_price1(0),start_price2(0),MP(0);

MP = marketposition*currentcontracts;

RSI_value = RSI(Close,len1);

condition1 = RSI_value cross under 30;
condition2 = RSI_value cross under 20;

if condition1 and MP = 0 then begin
        buy ("R1") 1 share next bar at market;
        start_price1 = close;
        end;

if condition2 and MP = 1 then begin
        buy ("R2") 2 share next bar at market;
        start_price2 = close;
        end;

if close - start_price1 > 100 or start_price1 - close > 100 then
        sell from entry ("R1") next bar at market;

if close - start_price2 > 200 or start_price2 - close > 200 then
        sell from entry ("R2") next bar at market;
```

這個策略是應用了傳統的 RSI，跌穿 30 博反彈，跌穿 20 再博反彈，而止賺止蝕方法則與上一章的例子一樣，都是以 if…then…begin…end…配合 start_price 紀錄開倉時價格，再用 if…then…方法去平倉。這裡的平倉條件用了「or」，意思是只要止賺或止蝕的其中一項條件成立，系統就會執行平倉。與上一章例子不同，根據 condition1 的開倉指令會以「R1」作為名稱，而根據 condition2 的開倉指令則會以「R2」作為名稱，這做法是為了在平倉時可以名稱區分指令。最後，在平倉時，我們只需寫上「from entry」，並加上指令的名稱，就可以在指定的倉位平倉，而不影響其他方向一樣的倉位。

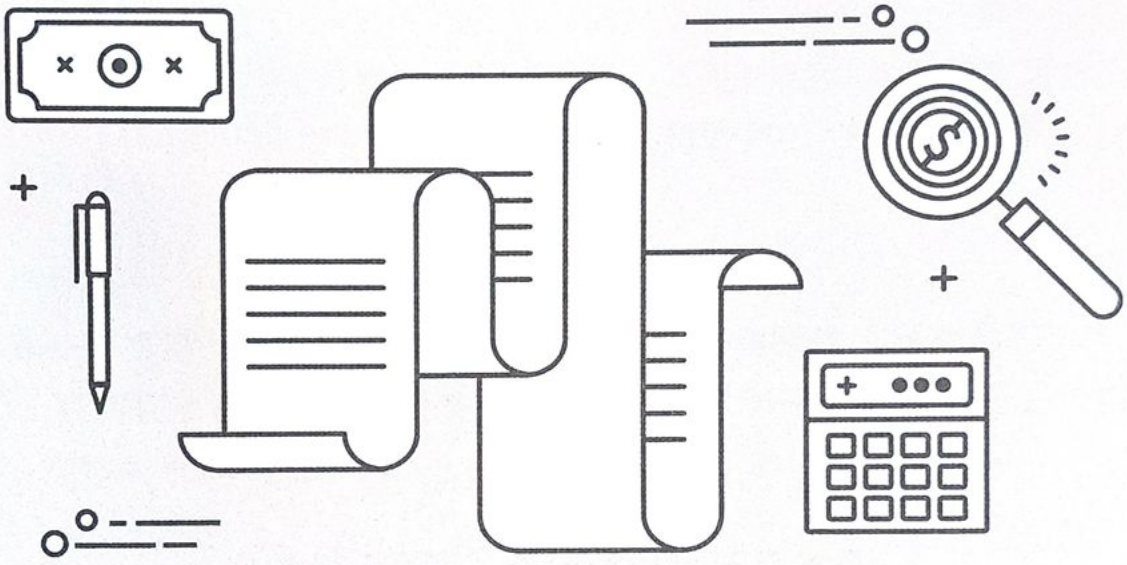
另外留意這裡我們用了 marketposition 和 currentcontracts 兩個關鍵字，前者只會返回 1、0、-1，分別代表好倉、沒有倉和淡倉，而後者則無論是好倉或淡倉，都只會返回倉位的正數數量，而不分方向。因此，兩者相乘就可以得出具備正負數的持倉數字。例如，持有 10 個淡倉，兩者相乘就會得出 -10，這就是我們定義的 MP 變數。而在之後運作中，我們可以加入 MP 作為條件的一部分，例如，MP = 0，這樣就可以控制持倉數作為開倉條件。

〈小結〉

正如先前所說，程式編寫本身是一個博大精深的學問，並不可能在一本書之內完全覆蓋，本篇旨在讓大家了解最基本的語法和必須懂得的關鍵字，只要學懂本篇的內容，之後再深入研究就會容易得多。

對於沒有編程經驗的讀者，筆者的忠告是編程是一種需要練習才能獲得的技能，就如所有語言一樣，都要經過模仿和練習的階段，才能流暢地表達心中所想。此外，由於 Power Language 在多年來已經累積了不少用戶，大家在編程時遇到問題，最快捷的解決方法就是善用 Google，直接將自己的問題或關鍵字在 Google 搜尋，久而久之就會慢慢上手。記住，學習編程是沒有捷徑的，加油！

第 6 章
策略表現判斷



- 風險與修復時間
- 風險與回報
- 穩定性
- 統一應用與放大性
- 小結

〈 6.1 〉

風險與修復時間

"Risk comes from not knowing what you're doing."

- 巴菲特

正當大家在摩拳擦掌，想開始著手建構交易程式。但之前，我們先一起來想像一下，心目中最理想的交易程式究竟有甚麼特質。

程式交易者心目中的「聖盃」特質

高利潤	低風險	高穩定性	高放大性

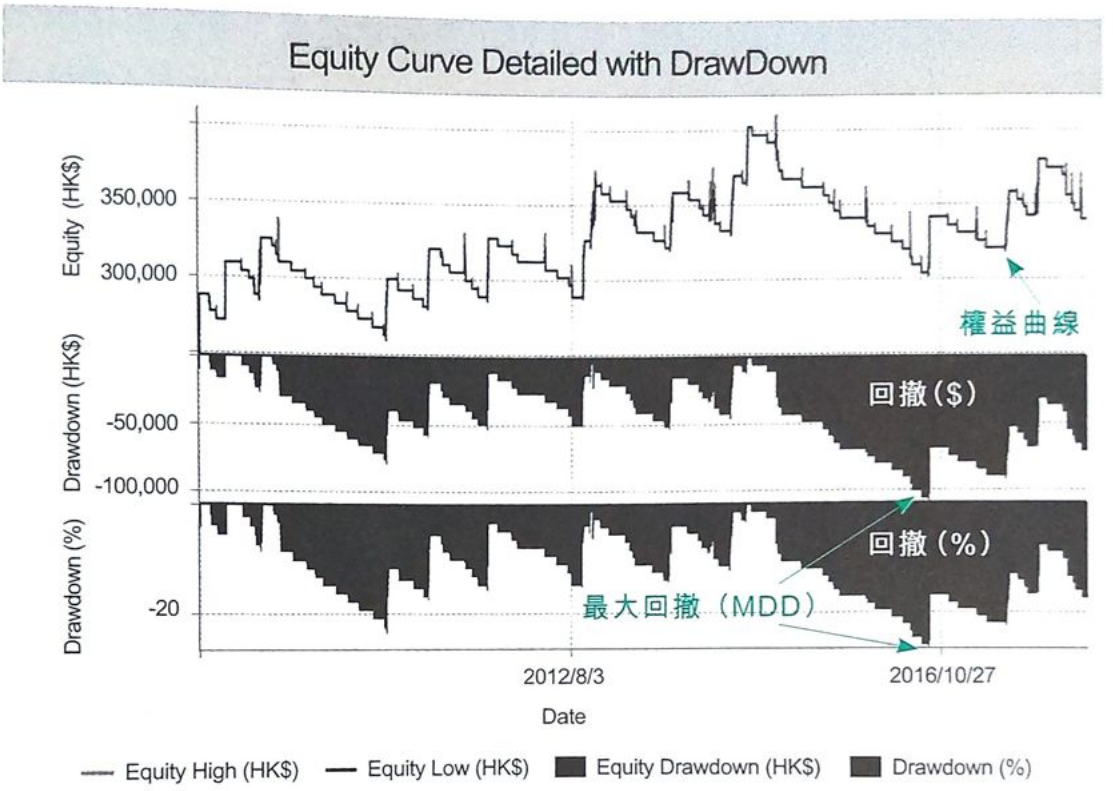
「聖盃」是程式交易界的術語，因為在早期的發展史中，不少人視程式交易為通往必勝之路的唯一方法。諷刺的是，儘管「聖盃」這個名詞充滿著神聖的意味，詞義在今時日已成為「空中樓閣」的代名詞，意思是只能在想像中出現，但在現實中卻是永遠遍尋不獲的聖物。

以日常的語言來說，程式交易者所追求的程式不單要能賺錢，又要有低風險和穩定回報，而且獲利表現不會隨著資金增加而減少（高放大性）。在這 4 個條件中，利潤是所有人都懂的，穩定性亦容易了解，高放大性的特質就對大戶較重要，而當散戶的資金量達到一定規模後，亦會開始追求高放大性；然而，風險卻是 4 項條件之中最難捉摸的，因為其定義和量度方法已有很多種。其實，風險管理本身就是一個可獨立成科的學問，本港的幾間大學都有開辦風險管理科學的本科和碩士課程，可見風險管理絕非三言兩語可以概括。

在本章裡，我們會先探討風險和回報的話題，包括量度的準則、跨策略的比較和取捨，然後在 4.2、4.3 和 4.4 會再探討穩定性和放大性的問題。

MDD——不幸的最大值

還記得嗎？在「3.2 回溯測試」中，我們提到權益曲線（Equity Curve）和回撤（Drawdown）的概念，其中一個我們還未深入探討過的名詞為最大回撤（Maximum Drawdown，簡稱 MDD）。



回撤的定義是「到達該時間點時，曾經見過權益曲線的最高位，與現時權益曲線水平的差距」，而在整個回測的時間範圍內，最大的回撤幅度便是 MDD 了。我們在使用 MDD 時，通常會以百分比作為單位，以銀碼作單位則比較少。

使用 MDD 作為量度風險指標的意義是頗為特別的，亦可反映程式交易員制定策略時的保守取向。須知道，以客觀條件作交易的策略，無論它是否有真實運行，回測結果都是存在的。

舉一個簡單例子，若有一個策略純粹以 1 分鐘恒指期貨圖表作訊號，其客觀條件是每當恒指期貨出現 1 分鐘內升幅多於 30 點就

做好倉，這是典型的順勢策略（Trend Following），當市場在該時段內有明顯趨勢，見升就買的簡單策略當然是賺錢的；然而市場不是永遠只會有 Trend，過了一段時間後，趨勢不再明顯，恒指期貨變成區間行走的形態，1 分鐘內升幅多於 30 點就買入的策略便會輸錢。這種「有勢→冇勢→有勢→冇勢」的周期循環在過去和未來一直不斷重複，只是周期的長度不確定而已。

即使這個策略沒有用真錢運行，該順勢策略的權益曲線都會存在，因為市場還是會不斷從「有勢→冇勢」的周期中不停改變，若果交易者的運氣非常差，於「有勢」市況的最後期才啟動上述的順勢策略，這樣他便要承受整個「冇勢」時段內的所有損失。**而 MDD 所代表的正是在過往歷史中這個最差的啟動時機所面對的最大捱價百分比，也就是在該段回測時段內持倉者所面對的最大風險。**

理論上，任何一個策略都有這樣的捱價風險，所以 MDD 是程式交易者最值得參考的風險數據，因為它可以讓交易者知道該策略在過往歷史的最大捱價幅度，以作為可能面對的最大風險值的參考水平。其實細心一想，其邏輯與一般買股票前會預計捱價幅度一樣，只是買入股票後坐艇要捱的是股價，而啟動交易策略後要捱的卻是價格特性。

另一個操盤心理的重點，關乎剛開展策略的「終極止蝕位」問題。理論上任何回測數據所反映的都是過往的表現，而金融產品的價格特性是建基於該產品的基本面，若果該基本面有重大變化，又或者市場環境的逐漸改變累積至某個程度，過往價格特性的參

考意義就會減少。例如，在開放滬港通前後的港交所（388.HK）價格特性已經有明顯差異，影響統計歷史數據的基本前設「資料同質性」。在最極端的例子下，某策略的回測表現優異，但後來，該金融產品的價格特性完全改變，而該改變在初期不為人知，只能在後期才確認，如果交易者不幸地在改變的初期就啟動該策略，就算先前的回測表現多麼優異，該策略都只會帶來虧損。此外，交易者在不知道價格特性已經完全改變下啟用該策略，也不可能永無止境地承受著虧損，必須有一個「終極止蝕位」用作宣告投降。而這個「終極止蝕位」的設定，就是以 MDD 作為一個參考。

終極止蝕位未必剛好等於 MDD，通常交易者會設定一個緩衝值，包容實際倉位回撤幅度稍為大於回測 MDD 的情況。例如，回測顯示某策略的 MDD 為 15%，設定緩衝值為 5%，終極止蝕位便是實際回撤達到 20% 的水平。設定緩衝值的邏輯是，即使未來出現回撤大於過往 MDD 的情況，也不一定代表該策略已經失效；若果太早放棄策略，可能會錯過之後權益曲線回升的利潤，所以 MDD 加緩衝值所代表的是「包容回撤打破過往紀錄，但策略仍未失效的可能性」，當中假設自己有可能比以往更不幸。

然而，所有招式都有罩門，即使事先設定緩衝值，真正不幸的人無論如何都會遇上不幸，例如，當策略邏輯仍未失效，但實際回撤剛好大於 MDD 加緩衝值，而交易者作出終極止蝕後，權益曲線才築底反彈……這種最不幸的情況有可能出現的，所以無論有否設定緩衝值，終極止蝕的最大重點是避開真正失效的策略，而代價是這種不幸的低位止蝕情況。大家不難發現，當中的問題其實與一般

買股票時設定的止蝕一樣，同樣有機會出現止蝕後股價／權益曲線回升。由此可見，止蝕的邏輯放諸四海皆一樣，都是為同一個目標服務，但會受同樣問題限制。

因此，緩衝值不是必然的，甚至終極止蝕位也不一定要低於 MDD。例如，某策略回測的 MDD 是 15%，終極止蝕位卻只是 10% 的實際回撤，雖然錯誤止蝕的機會會增加，但若果止蝕正確，虧損會比 MDD 少 5%，更不用提節省了的緩衝值。

由此可見，MDD 的實際效用除了可以反映策略的穩定性，更重要的是作為捱價幅度的參考。反之，傳統的風險指標（如標準差）就只可反映策略的穩定性，不能作為捱價幅度的參考指標，更不用說用以設定終極止蝕位。

〈 6.2 〉

風險和回報

"The biggest risk is not taking any risk... In a world that changing really quickly, the only strategy that is guaranteed to fail is not taking risks."

- Mack Zuckerberg（Facebook 創辦人）

風險和回報是交易策略的兩項最重要特質，剛才我們探討了用作評估風險的指標 MDD，接下來會介紹風險和回報的關係，當中包括兩個回報風險比例的指標 ROM 和 Profit Factor。

毫無疑問，所有交易者都希望淨利潤愈高愈好，可是世界上沒有免費的午餐，高回報所伴隨的必然是更高的風險。這個道理人人都懂，但問題在於，當你把策略回報調整得愈來愈進取，歷史回報愈來愈高時，風險增加的幅度卻往往更大，由於這兩項因素增加時不合比例，我們需要量度回報風險指標作參考。

ROM（Return on MDD）

$$ROM = \frac{Net\ Profit}{MDD}$$

Net Profit 是策略在回測時段內能夠賺取的淨利潤。對於具備槓桿的期貨合約，它和 MDD 的比例尤其重要，因為當某策略的虧損擴大至倉位連按金都支付不起時，該策略便不能繼續運行。我們不可能為了丁點兒的利潤，承受期貨合約隨時被斬倉的風險，為了補償某百分比的 MDD，我們希望該策略可以帶來更大百分比的淨利潤，這就是 ROM 的基本邏輯。

由此可見，ROM 是一個愈大愈好的指標，即是策略的 MDD 愈小愈好，同時淨利潤愈多愈好。實際上，當進行策略制定的時候，你會可能遇上幾個能獲利的策略，各自有著不同的 ROM。

個案：

	策略 A	策略 B
年回報	30%	10%
MDD	20%	5%
ROM	1.5	2

單看這個例子，策略 B 比策略 A 優勢，因為策略 B 的 ROM 較高。雖然策略 B 的回報（10%）低於策略 A（30%），但只要策略 B 把槓桿放大至 4 倍，其 MDD 會由 5% 變成 20%，而年回報就會由 10% 變成 40%，高於策略 A 的年回報。

淨利潤和 MDD 的變化是同步的，交易者可以透過增減起始資金或持倉數量，輕易地控制回報率和 MDD，卻不能改變 ROM。舉例說，某策略的起始資金是 $200,000，年回報和 MDD 金額分別是 $60,000 和 $40,000，所以兩者的百分比就分別是 30% 和 20%，而 ROM 則是 1.5。若果把起始資金翻倍至 $400,000，而持倉數量保持不變，它們年回報和 MDD 金額也會不變，但它們的百分比則會降至 15% 和 10%，而 ROM 依然是 1.5。

由此可見，ROM 不反映資金運用的效率，因為即使 ROM 很高，都有可能是因為 MDD 太低所致。以典型的套戥交易策略為例，MDD 可以低至數個百分點，甚至低於 1%；假設年回報為 3%，MDD 為 1%，ROM 就會高達 3。但是，該策略的年回報只有 3%，甚至比美國 10 年債債息更低，是槓桿不足的表現，除非有隱憂（如潛在黑天鵝事件導致突發性的超大風險），否則交易者都應該增加該套戥交易的槓桿以增加回報。

不過，上述都只是美好和簡化的例子，實際上是未必容易更改起始資金和持倉數量。我們可以在剛才的例子中加入起始資金和按金的限制，大家就會明白問題出自哪裡。

個案：

	策略 A	策略 B
起始資金	$1,000,000	$1,000,000
按金	$400,000	$400,000
按金佔起始資金	40%	40%
年回報	30%	10%
MDD	20%	5%
ROM	1.5	2

同樣是策略 A 和策略 B，運用同樣的起始資金和持倉數據。執筆時恒指期貨按金大約是每張 $100,000，所以 4 張的按金約是 $400,000，而按金佔起始資金的比例是 40%。若以剛才的邏輯，以增減起始資金或持倉數量來調整年回報或 MDD，都會帶來其他問題。

例如，若果把策略 B 的槓桿放大至 4 倍，令持倉數量由 4 張期指增加至 16 張期指，這顯然是行不通的，因為按金要求會高達 $1,600,000（=$400,000×4），大幅高於起始資金。最佳做法是把策略 B 的槓桿「僅僅」放大至兩倍，按金只需要 $800,000（=$400,000×2），MDD 亦只從 5% 升至 10%，即是 $100,000（=$1,000,000×10%），而且仍有剩下 $100,000 作額外的緩衝；放大槓桿後，策略 B 的年回報會提升至 $20%，雖然還是低於策略 A，但卻有較高的 ROM，風險較容易能承受，亦更值得承受。

由此可見，我們未必每次都能透過控制槓桿來達至心目中的 MDD 和回報率。我們當然希望能將策略 B 的 MDD 調整至策略 A 的水平，以同樣的風險享受更大的回報，但現實是策略 B 的按金金額限制其槓桿只能放大至兩倍，導致回報依然低於策略 A。這解釋了為何有些交易者會混合策略使用，只要策略 A 的 MDD 在接受範圍之內，即使 ROM 較低，都會同時運行策略 A 和策略 B，以追求較大的回報。

〈 6.3 〉
穩定性

若說風險和回報是交易策略最重要的兩個特質，緊接其後的就必然是穩定性了。原因顯然易見，回報穩定的交易策略可以讓交易者放心，並對未來的資金狀況有更好的預期。

穩定性對大戶尤其重要，因為大戶的資金通常都是行內所謂的 OPM（Others People Money），就像基金一樣，其客戶可以隨時提出贖回，即使基金條款一般設有贖回期，以保障基金的流動性，但大量的同時贖回並非不可能的，若果整體回報不夠穩定，低潮時的贖回輕則可影響回報，重則會帶來致命的傷害。至於散戶，穩定的策略可以讓他們隨時從交易戶口中提金錢，以應付隨時出現支出需要，所以這一點也是相當重要的。

在金融世界裡，量度穩定性的常見方法是計算回報的標準差（Standard Deviation）。假設某策略的月回報 5%，每個月都不多不少，剛剛好賺 5%（要是有這個策略，它肯定是聖盃級數），由於月回報固定不變，標準差自然是 0。這是最高穩定性的極端例子。

不過，現實始終沒有這麼完美的聖盃，標準差為 0 的交易策略幾乎是不存在的，只有銀行存款、美國國債等超穩定資產才有接近 0 的標準差，但享受穩定性的代價是非常低的回報率。一般來說，即使是穩定性最高的套戥交易策略，其回報都有一定的波動，因此穩定性對策略的影響是存在的。

穩定性的重要性除了在於讓交易者隨時可以提錢消費外，最大的重點就是穩定性和風險之間的密切關係。

穩定性和風險是兩個不同概念，但實際上，兩者是難兄難弟，有共同進退的關係（共同進退但不一定成正比）。

要解釋這個概念，我們需要明白以下幾個要點。

（1） 風險來自策略的 downside

對交易者來說，風險就是虧損的風險，如果某策略像銀行存款一樣永不虧損，風險是不存在的。換句話說，哪怕虧損的金額有多輕微、出現虧損的機會有多低，只要有 downside 的存在，就必定有風險。此外，只有虧損機率為 0 的情況下，才是風險和穩定性完全無關的唯一特殊例子；

（2） 只要回報率不確定，就必然存在回報不穩定性

例如，某策略虧損的機會是零，但月回報率介乎 0.15% 至 0.25% 之間，交易者不能確定下個月有多少利潤，而回報率

亦會存在標準差。現實中，這情況出現於銀行同業拆息市場中，不確定的回報帶來不穩定性，即使明知必賺，也不能預先確認未來的收益；

(3) 風險的不穩定來自回報的不穩定

一直以來，我們所指的穩定性是針對回報。其實，如果回報不夠穩定，這會令風險的評估變得困難，例如，風險指標 MDD 的可信度都會降低。原因是回報的不穩定性愈高，出現尾端事件（Tail Event，意指極端事件）的機會亦會增加，導致未來虧損幅度的分佈會更散，所以風險更難以估計。

就以上 3 點，特點是第 3 點，我們需要運用蒙地卡羅方法，來解釋風險和不確定性之間的關係。

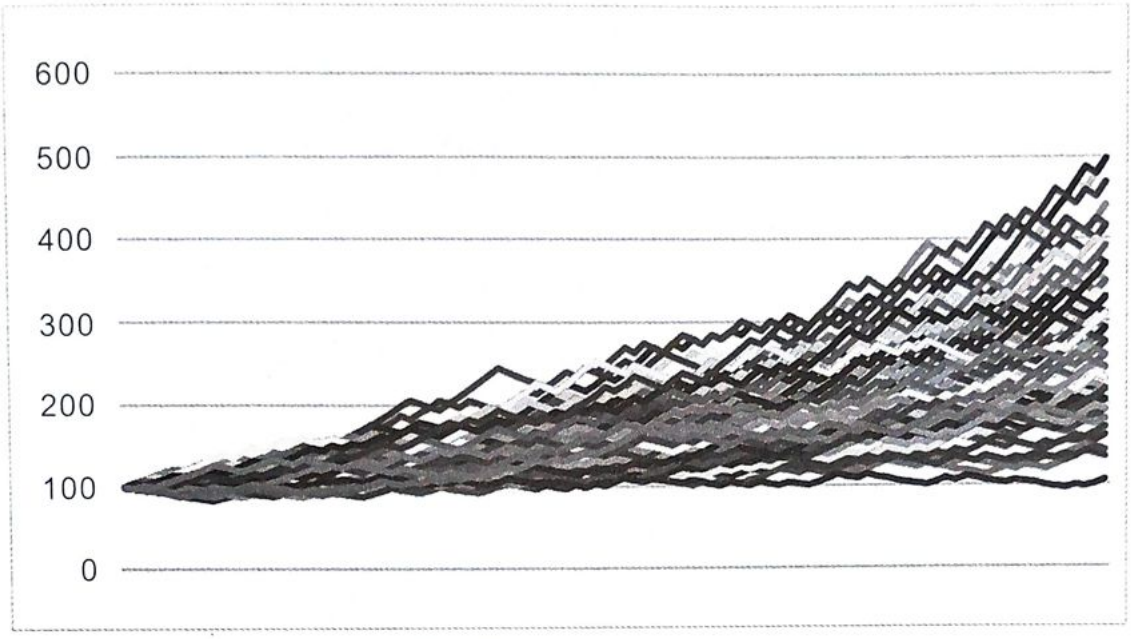
蒙地卡羅方法（Monte Carlo Method）

蒙地卡羅方法是指透過大量重複隨機數所產生的結果，以找出目標數據的理論值，只要重複次數愈多，得出數據就會愈接近理論值。這個方法來源於二次大戰時的曼克頓計劃，當時有一班參與製作原子彈的科學家，他們利用非常有限的電腦運算能力，以蒙地卡羅方法計算核子分裂的連繫反應，後來這方法被大量運用於金融工程學、量子生物學、流體力學、人工智能等牽涉大量隨機過程的範疇。

請不要急著指出真實的資產價格變化並非隨機，這一點筆者怎會不知道。要運用隨機數的原因是，現實的歷史價格數據早已固定，任何策略只得一條權益曲線，亦只得一個 MDD 和回報標準差，不足夠推算 MDD 和標準差之間的關係。

要能夠推算兩者之間的關係，我們必須有大量的 MDD 和標準差數據，而且這些數據必須在其他額外條件完全相同的情況下採樣的。而隨機數的角色就是要在固定的策略命中率、單次止賺幅度和單次止蝕幅度的設定下，產生出不同的「獲利→虧損」路徑。原理就像硬幣擲公字一樣，假設只有 20 次歷史數據，當中出現最多連續 4 隻公，但當你重複 n 次擲 20 次時，可能會出現連續 5 隻公、7 隻公、甚至 10 隻公都不足為奇。因此，若果只以歷史數據中的 4 隻公作為 MDD，恐怕多多錢都不夠輸。

由於我們的目標是尋找 MDD 和回報標準差的關係，策略命中率和單次賺蝕幅度都必須是固定，唯一變化的是資產價格路徑。透過隨機產生路徑，固定的策略特性可以套用於完全不同的價格路徑上，以產生不同的 MDD 和回報標準差。路徑數量愈多，收集得來的大量 MDD 和回報標準差數據就可以推算兩者之間的關係。



▲ 典型的蒙地卡羅結果，此圖與例子無關

現在讓我們設定一個交易策略的蒙地卡羅測試，具體的設定如下：

(1) 單次命中率：50%（固定）
(2) 單次獲利幅度：8%、16%、26%
(3) 單次虧損幅度：比獲利幅度少 6%
(4) 周期：120 次
(5) 重複次數：100 次

單次命中率固定為 50%，分 3 組不同的賺蝕幅度進行，分別為「賺 8% + 蝕 2%」、「賺 16% + 蝕 10%」、「賺 26% + 蝕 20%」，由於每組的獲利幅度都比虧損幅度大 6%，策略的期望值理應為正數。每組各自重複進行 100 次模擬，每次模擬都有 120

個周期的交易。周期為時間單位，可以是 120 個月（即總共 10 年），亦可以是 120 小時，即是大約 20 個港股交易日。其實，周期是設定是沒有規定的，在這例子中我們假設周期為月。

經過 100 次模擬後，得出的結果如下。

Accuracy	50%
Upside (%) 8	Downside (%) 2
Avg monthly return	4%
Avg annual return	51%
Avg of monthly return SD	0.03%
Avg of all MDD	12%
Max MDD	23%
MDD SD	3%
Sharpe Ratio	128

Accuracy	50%
Upside (%) 16	Downside (%) 10
Avg monthly return	3%
Avg annual return	35%
Avg of monthly return SD	0.05%
Avg of all MDD	58%
Max MDD	89%
MDD SD	12%
Sharpe Ratio	53

Accuracy	50%
Upside (%) 26	Downside (%) 20
Avg monthly return	0%
Avg annual return	5%
Avg of monthly return SD	0.15%
Avg of all MDD	90%
Max MDD	100%
MDD SD	8%
Sharpe Ratio	1

名詞說明

(1) Avg monthly return：每個路徑得出的幾何平均月回報（Geometric Average Return）的平均數，等於所有路徑的月回報除以重複次數（100 次）

(2) Avg of monthly return SD：每個路徑得出的幾何平均月回報的標準差

(3) Avg of all MDD：每個路徑的最大回撤的平均數，等於所有路徑的 MDD 除以重複次數

(4) Max MDD：所有路徑的最大回撤的最大值

(5) MDD SD：所有路徑的最大回撤的標準差

(6) Sharpe Ratio：相等於 Avg monthly return 除以 Avg of monthly return SD

以上資料中，Avg of all MDD 和 Avg of monthly return SD 是我們的目標數據，前者代表 MDD，後者則代表回報標準差。不難看出，MDD 本身都有標準差，這是由於不同價格路徑會影響連續出現虧損的機率和幅度，所以我們有需要以蒙地卡羅方法找出 MDD 的平均值。反之，月回報的標準差本身不會因為不同的路徑有太大變化。

從以上資料可見，MDD 和回報標準差的確是共同進退的，雖然難以確定是否屬正比關係，但起碼方向是一致的。

此外，以上資料亦有一個驚人發現，即使賺蝕幅度的差距固定在 6%，隨著賺蝕幅度的提升，不單會令 MDD 和回報標準差大幅上升，也會使月回報和夏普比率（Sharpe Ratio）下降，後者的跌幅更加明顯。比較簡單的解釋是，當資金出現 n% 虧損，要令資金返回家鄉的獲利幅度是 m%，則 m 必然大於 n，而隨著 n% 的增加，m% - n% 的差距亦會擴大，造成愈來愈難返家鄉的效果。因此，即使賺蝕的差距是固定的，單次虧損幅度的增加會對回報造成更大的打擊，導致月回報減少，而夏普比率的跌幅更大。

因此，我們可以斷言，穩定性雖然在表面上與回報和風險沒有直接關係，但實際上在交易策略內會對兩者構成微妙的影響。交易者在能力範圍內應該追求較低的單次虧損幅度，同時盡量拉闊賺蝕差距，並且提高交易頻率，藉以提高回報的穩定性和降低 MDD，代價是手續費和滑價都會與交易頻率成正比上升。由此可見，策略制定是一項平衡各方利益的工作，頻率的高低就是穩定性和交易費用的取捨，在策略制定時必須同時兼顧這兩點。

另外，從蒙地卡羅方法的結果可見，回報標準差的上升除了會蠶食利潤和使 MDD 上升外，亦會令 MDD 本身的標準差上升，即是 MDD 的穩定性會下降。在實際交易和回測時，價格只會有一個路徑，所以可以理解 MDD 標準差為回測 MDD 的可信度。須知道，對於一個不穩定的策略，其 MDD 或會很低，，但這可能只是巧合產生的結果。

因此，若果回測結果出現較高的回報標準差和較低的 MDD，大家應該有所警惕，對 MDD 的可信度打個折扣。就算真的要採用該策略，也應該預備較大的緩衝值，使終極止蝕位擺得較遠。而若果 MDD 本身已經很大，沒有空間容納緩衝值，更應該選擇放棄或再改良這個策略。

〈 6.4 〉

統一應用與放大性

在先前的篇章中，我們提到了 4 個聖盃策略的特質，分別是高回報、低風險、高穩定性和高放大性，當中的回報、風險、穩定性都有各自的量度方法，例如，以 MDD 量度風險，以回報標準差量度穩定性，以 ROM 量度風險與回報比例等，唯獨是放大性沒有一個獨特的指標，這是因為策略的適用資產數量已可直接反映資產的放大性。

甚麼是策略的適用資產數量呢？舉例說，如果交易的資產是恒指期貨，理論上該策略除了適用於恒指期貨外，也同時適用於其他與恒指期貨相關度非常高的資產，例如是國指期貨、盈富基金等。此外，不少恒指成份股與恒指期貨的相關度亦相當高，特別是重磅股，如騰訊、內銀股、港交所等。若果某策略適用於恒指期貨，運用於這些相關資產的表現也應該不會太差。

為甚麼策略需要具備放大性？原因不外乎兩個：

（1）單一資產的價格深度有限

大家都希望交易策略可以一直賺錢，而隨著資金增長，交易的注碼會愈來愈大，資產的價格深度逐漸不能承受，導致滑價大幅增加。如果將現有策略分散於其他相關度高的資產身上，便可以達到 scale up 的效果；

（2）降低組合風險

兩項資產可能有著不低的相關度，然而兩者走勢始終會有些微分別，如恒指和內銀股，價格的變化雖然相關，但總有點分別，這特點有助降低組合風險。

舉例說，建設銀行是恒指的重磅成份股，走勢與恒指相關度很高，但其短線價格走勢的高低點總會與恒指高低點有先後之分，即是所謂的 Time Lag。假設我們把投入於恒指期貨的 100% 資金分成兩份，並同時運用相同的策略於恒指期貨及建行上。如果該策略是旨在捕捉低位搏反彈，在不同的 Time Lag 下，可能會導致以下幾個效果。

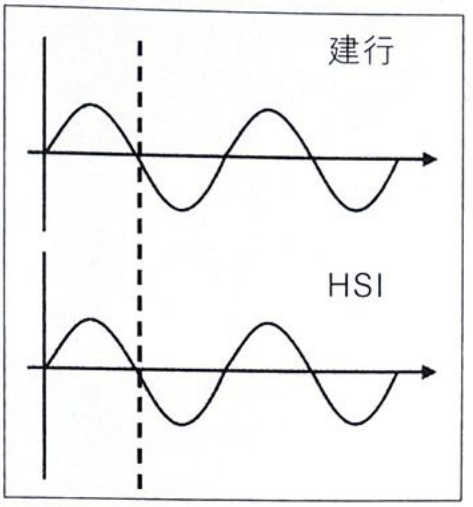


圖 1

x 軸：時間
y 軸：股價 / 指數
建行與恒指完全同步

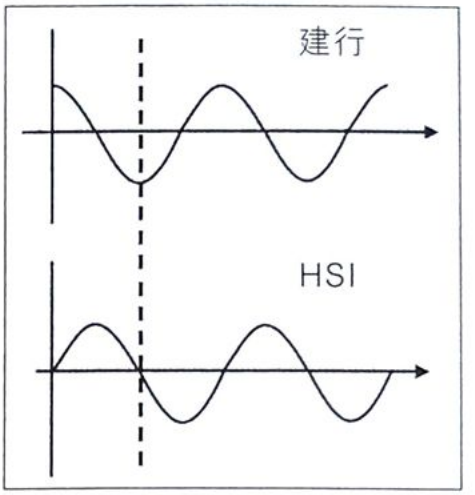


圖 2

x 軸：時間
y 軸：股價 / 指數
建行比恒指更早見低位

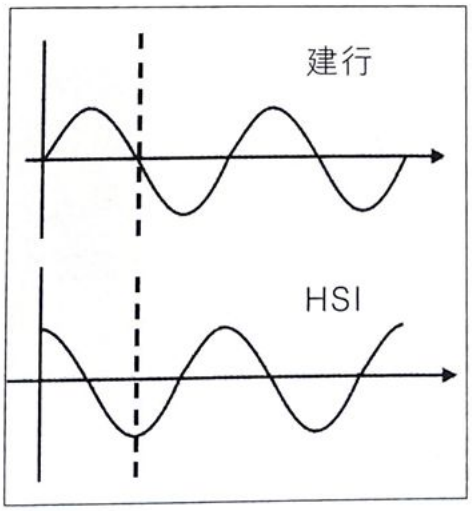


圖 3

x 軸：時間
y 軸：股價 / 指數
恒指比建行更早見低位

上圖虛線代表著搏反彈倉位的開倉時間。圖 1 代表建行與恒指完全同步；圖 2 代表建行領先，恒指滯後，所以在訊號出現時，建行已經見底，而恒指則仍未見底；圖 3 是圖 2 的相反情況，恒指比建行領先，所以恒指比建行更早見底。

在圖 1，建行和恒指的升跌幅度和時間點是完全同步的。由於搏反彈訊號在建行和恒指皆過早出現，兩者都需要捱價，而且捱價的幅度和時點亦完全同步。也就是說，如果該捱價幅度是策略的 MDD，無論將資金 100% 放置於建行或恒指，抑或是平均放置於建行和恒指，整個組合的 MDD 都不會有改變。

假設整體組合捱價幅度為 n%，在圖 1 的完全同步情況下，建行和恒指的捱價幅度都會是 n%，當兩者各佔總資金 50%，建行和恒指造成的整體捱價幅度都是 n%。

至於圖 2，建行比恒指更早見低位，該策略用於建行是不須要捱價的，但用於恒指則要捱價，而捱價幅度為 n%。套用剛才的設定，建行和恒指各佔總資金 50%，恒指為組合造成捱價幅度為 n/2%，而建行為組合造成的捱價幅度為 0%，所以整體組合的捱價幅度是 n/2%。

圖 3 與圖 2 的情況相似，只是換成建行比恒指滯後，整體組合捱價幅度同樣是 n/2%。

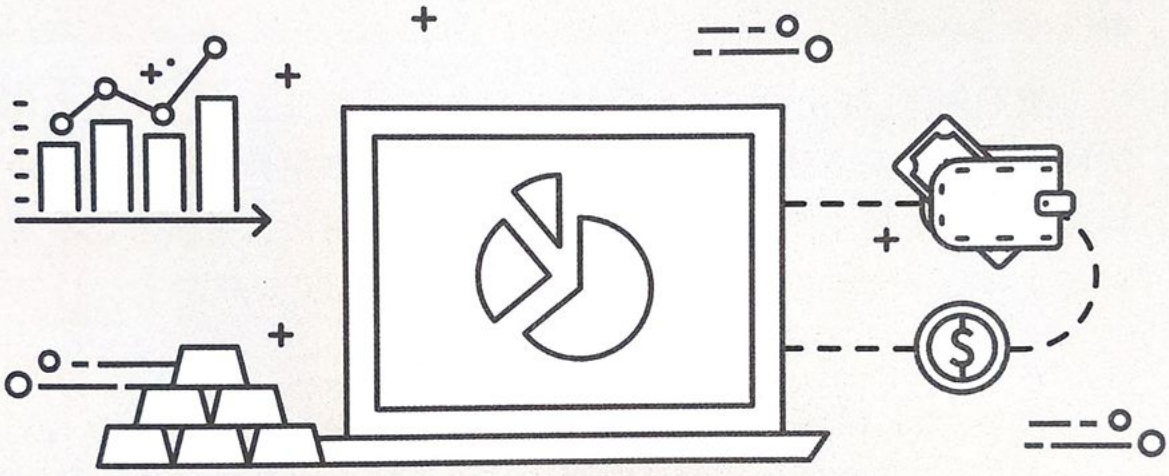
由此可見，將交易策略分散於不同資產身上，當兩個資產完全同步時，對 MDD 不會有任何影響，頂多是維持原狀。而如果兩個資產的短線走勢稍為有點不同步，便能夠造成減輕 MDD 的效果，可見這是一個有利無害的方法。若要達致這效果，關鍵是兩項資產都必須有高度的相關性，令交易策略可同樣在兩者身上長線獲利，而不同步之處僅限於在短線把捱價點分散，達到減輕組合 MDD 的效果。

〈小結〉

這一章的內容比較深入，尤其是風險的部份，很多問題都需要親自回測過才會明白，因此初學者不必一開始便追求完美，畢竟聖盃策略的境界是很難做到的。反而，大家可以在嘗試研究出一大堆可能有用的策略後，逐一比較它們各自的優劣，從而感受程式交易者在選擇策略的過程中的取捨。

要記得的是，世界上沒有東西是完美的，交易程式也是，而不同人會有不同的風險取向，膽子大的人可能面對 30% 的 MDD 都面不改容，悲觀的人面對 10% 的 MDD 已經叫苦連天，因此選擇策略的關鍵其實不是策略本身，而是對自己的了解，只有充分了解自己的人才可以選擇適合自己的策略。由此可見，程式交易的本質還是離不開交易，而人的因素依然佔一席位，心理質素、膽色、耐性都是決勝負的因素之一，只不過這些因素的表現形式與手動交易不同而已。

第 7 章
交易的數學



- 為何獲利因子不是好指標？
- 回報標準差和 MDD 的衝突
- 極端命中率與風險管理
- 小結

〈 7.1 〉

為何獲利因子不是好指標？

獲利因子（Profit Factor）是一個常見於第三方程式交易軟件的回報風險比例指標，因此筆者不得不花一段篇幅介紹，但個人覺得獲利因子使用上問題多多，甚至連是否真正屬於回報風險比例指標都成疑問，大家在了解甚麼是獲利因子後可以思考一下。

獲利因子是總獲利和總虧損相除的結果，而淨利潤（Net Profit）則是總獲利和總虧損相減的結果，兩者雖然同樣是總獲利和總虧損的產物，但所代表的意義卻不相同。例如，在兩個擁有相同淨利潤的策略，其獲利因子則未必相同。此外，交易頻率較高的策略通常會有較低的獲利因子，但由於交易頻率牽涉到策略的穩定性和可信度，獲利因子不能單獨運用，須配合更多的因素作參考，所以其應用性不如 ROM。

	策略 C	策略 D
起始資金	$1,000,000	$1,000,000
年回報（%）	30%	30%
年回報（$）	$300,000	$300,000
獲利因子	2	3

策略 C 和策略 D 的淨利潤都是 $300,000，而起始資金和回報率都相同，但策略 C 的獲利因子就低於策略 D。驟眼看，策略 C 的回報風險比例似乎較低，但若果我們深入探討其中的數字，就會發現情況未必一樣。

	策略 C	策略 D
合約數量	4	4
交易次數	100	20
獲利因子	2	3
獲利次數	50	15
每次每合約獲利點數	60	150
總利潤	$600,000	$450,000
虧損次數	50	5
每次每合約虧損點數	30	150
總虧損	$300,000	$150,000
年回報 ($)	$300,000	$300,000

來計一計數吧，以 4 張恒指期貨合約為例，策略 C 過去一年交易了 100 次，勝負次數各佔 50 次，獲利時能賺 60 點，虧損時會蝕 30 點，所以總利潤為 $600,000（= 50 次 x 4 張 x 60 點 X 50 元），而總虧損則是 $300,000（= 50 次 x 4 張 x 30 點 x 50 元），兩者相減的淨利潤為 $300,000。

至於策略 D，過去一年交易了 20 次，勝負次數分別是 15 次和 5 次，獲利時能賺 150 點，虧損時會蝕 150 點，總利潤為 $450,000（=15次 ×4張 ×150 點 ×50 元），而總虧損為 $150,000（= 5次 ×4張 ×150點 ×50 元），兩者相減的淨利潤為 $300,000，與策略 C 相同。

策略 C 的勝率類似擲銀，雖然命中率不高，但命中時的利潤會高於失敗時的虧損；至於策略 D，雖然每次輸贏的金額相近，但憑藉較高的命中率，都能賺取長線利潤。明顯地，這兩種策略風格迥異，背後代表著不同的交易邏輯。

那麼，究竟哪種策略會較優勝呢？答案是不知道。原因是在挑選策略時，不同的交易者會有不同的著眼點，例如，大戶或會傾向選擇同時運用十幾個類似策略 D 的策略，雖然策略 D 交易次數較少，穩定性和可信度都較低，但這能夠透過分散策略的方式去克服。此外，由於大戶的資金量太大，市場深度不足以消化，容易出現滑價（Slippage）問題，策略 C 的單次獲利和虧損點數較小，所以滑價的影響較大，這亦是大戶一般難以克服的。

假設滑價上升 5 點，令策略 C 和策略 D 的利潤表現下降。

	策略 C	策略 D
合約數量	4	4
交易次數	100	20
獲利因子	2	3
獲利次數	50	15
每次每合約獲利點數	**60 → 55**	**150 → 145**
總利潤	$550,000	$435,000
虧損次數	50	5
每次每合約虧損點數	**30 → 35**	**150 → 155**
總虧損	$350,000	$155,000
年回報（$）	**$200,000**	**$280,000**

由於滑價會同時令利潤減少和虧損增加，當中對於點數較小的利潤和虧損都有較大影響。在策略 D，單次利潤和虧損都比較大，滑價所佔比例較小，加上其交易頻率低得多，所以受滑價影響的程度比較小，例如，策略 D 的年回報受滑價影響較小，只微跌了 $20,000，但策略 C 的年回報卻足足減少了 $100,000，可見策略 C 是屬於 Slippage-sensitive 的策略。

滑價只是其中一個考慮的因素，除此之外，謹慎的程式交易者會追求統計學上的可信性，認為交易次數愈多，代表該策略由巧合所致的機會愈低，可信度就愈高。追求可信度和追求減少滑價剛好相反，前者希望交易次數愈多愈好，後者則希望交易次數愈少愈好。可見選擇交易策略的時候不同交易者會有不同的著眼點，適合別人的不一定適合你。

由此可見，需要用作輔助獲利因子的額外數據太多，大大降低獲利因子作為指標的實用性，這可解釋為何許多程式交易者都不太看重獲利因子。

筆者甚至質疑獲利因子是否屬於回報風險比例指標。這是由於獲利因子的公式是以總虧損計算，但總虧損欠缺風險定義中的「不確定性」特質，亦不像 ROM 般以最大回撤（MDD）作為評估風險的最保守估計。因此筆者認為獲利因子的實用性不及 ROM，大家以後在第三方程式交易軟件看見獲利因子時，可不用放太多的注意力在該項數據之上。

〈 7.2 〉

回報標準差和 MDD 的衝突

在先前的篇章中我們探討過，穩定性與風險是一對關係密切的好兄弟，雖然它們非完全相等的概念，但會共同進退。在蒙地卡羅方法的測試中，我們知道回報標準差愈低的策略會傾向有較低的 MDD，多數的回測都會呈現這個結果，這不是必然的，因為 MDD 本身也有標準差，所以我們可能會遇到回報標準差大，但 MDD 小的情況，例如是以下兩個策略。

◀ 策略 E

Strategy Performance Summary	
	All Trades
Net Profit	HK$324,920.00
Max Strategy Drawdown (%)	(17.19%)
Monthlt Return StdDev	HK$24,776.09

◀ 策略 F

Strategy Performance Summary	
	All Trades
Net Profit	HK$370,330.00
Max Strategy Drawdown (%)	(12.17%)
Monthlt Return StdDev	HK$28,676.01

由於策略回測只會運用歷史數據，只得一條路徑，不會像蒙地卡羅方法一樣會產生多個 MDD。因此，歷史數據的路徑絕對有可能剛好是 MDD 比較小的路徑，出現「高回報標準差 + 低 MDD」的回測結果是正常的。

問題是我們很難得知 MDD 的標準差。在先前蒙地卡羅方法的例子中，我們假設了賺蝕幅度是固定的，但在真正的程式交易策略中，許多止賺和止蝕方法都不是固定的，就如簡單的推止賺方法，止賺位會一直跟隨價格走，所以獲利幅度要視乎資產價格的連續性，與市場心理特性有莫大關係，難以用隨機數作出模擬。因此，在選擇策略時，我們要以其他因素幫助判斷。

說回上圖的例子，策略 F 的回報標準差較高，反映 MDD 標準差亦會較高，這是因為我們從蒙地卡羅方法知道回報標準差愈高會導致 MDD 標準差愈高，也就是說策略 F 的 MDD 較不穩定，可信度較低；相反策略 E 雖然 MDD 較大，但回報標準差較小，可推斷策略 E 的 MDD 的標準差亦同樣較小，所以 MDD 較為穩定，可信度亦較高。

當然，MDD 的穩定性是一項相對概念，而非絕對概念，就是說策略 E 的 MDD 比策略 F 的 MDD 穩定，但我們不知道有多穩定，說不定兩者的 MDD 標準差都是非常大。由此可見，MDD 的穩定性會令問題變得更為複雜。

由於缺乏量化 MDD 穩定性的手段，制定策略時唯有盡量選擇回報標準差較小的策略，如在上圖的策略 E 和策略 F 中，筆者便會選擇策略 E。比較策略 F，雖然策略 E 的回報較小及 MDD 較大，但其回報也算相當不錯，而且 MDD 亦是處於可接受範圍內；更重要是預計策略 E 的 MDD 的穩定性較高，可以推斷未來在運作時出現 MDD 破紀錄的機會亦較低。由此可見，如果回測結果的回報和 MDD 都符合預期的範圍，而且差距不太遠，我們就可以回報標準差較低者為首選。

從這個例子可見，程式交易往往會牽涉到取捨的問題，我們很難寫出聖盃策略，而比較可能的是寫出一堆表面上各有優缺點的策略，但由於資金有限，總要作出取捨，而選擇時永遠要將風險因素放在首位，即使一些策略的回報看似很好，若果風險太大，則只能以小量資金運作，不能作為主力。由此可見，風險是取捨過程中的最重要因素。

〈 7.3 〉

極端命中率與風險管理

這個話題其實是延續 6.3「穩定性」中的蒙地卡羅方法。在 6.3，我們嘗試透用蒙地卡羅方法展示出回報標準差與 MDD 及 MDD 標準差的關係。但在例子中，我們只更改了獲利和虧損幅度，但沒有改變命中率。因此，我們要探討一下，當命中率不再是 50%，而是極端大或極端小時，對策略風險會有甚麼影響。

首先，我們要明白，命中率的高低與策略是否值博沒有太大關係。正如在 6.3 中的例子，雖然命中率只有一半，但單次回報比虧損大，該策略在多次重複後就必然有正數回報，這就是數學上所說的期望值（Expected Value）。同樣道理，低命中率策略可能有超高回報，而低回報策略亦可能有很高的命中率，這些策略都可以有正數期望值，理論上都是值得使用的。

然而，比較少人提及的是極端命中率所帶來的問題，其風險會比較難管理。

先不要直接考慮蒙地卡羅方法，大家可用常理分析，一個命中率達 9 成的策略，都通常是以大博小。假設單次回報為 2%，而單

次虧損為 9%。在 10 個回合中，理應有 9 次獲利，總獲利為 18%（= 9×2%），而餘下的 1 次虧損則帶來 9% 的總虧損，所以 10 個回合後的期望值應該是 9%（= 18% - 9%）。

然而，10 個回合虧損 1 次只是理論推算，我們永遠不知道實際的虧損事件會如何分佈。舉例說，某人可能運氣欠佳，遇上如下表的盈虧分佈，首 10 個回合已經遇上 2 次虧損，這會令人懷疑該策略真實的虧損機率不是 10%，而是 20%，於是停止了交易；但原來只要堅持多額外 10 個回合，便會發現第 11 至第 20 回合全部獲利，總數 20 回合裡有 18 次獲利和 2 次虧損，與期望的結果一致。

第 1 回合	獲利	第 11 回合	獲利
第 2 回合	獲利	第 12 回合	獲利
第 3 回合	獲利	第 13 回合	獲利
第 4 回合	獲利	第 14 回合	獲利
第 5 回合	**虧損**	第 15 回合	獲利
第 6 回合	獲利	第 16 回合	獲利
第 7 回合	獲利	第 17 回合	獲利
第 8 回合	獲利	第 18 回合	獲利
第 9 回合	獲利	第 19 回合	獲利
第 10 回合	**虧損**	第 20 回合	獲利

這帶出了實際操盤時的問題，當一個本應高命中率的策略在頭幾次已出現虧損，我們應否繼續執行下去？如果不停止，我們又怎知道這幾次虧損是因為運氣問題，還是價格基本面已經改變？

除了信心問題外，另一個問題是虧損事件的分佈難以捉摸，導致準備按金和緩衝金的預算困難，以剛才 9 成命中率的策略為例，除了首 10 個回合可能出現 2 次虧損，首 20 回合出現 3 次虧損，而且分佈非常緊密也是意料中的事，但這會對按金水平造成巨大壓力，並且令策略產生巨大的回撤。

第 1 回合	獲利	**第 11 回合**	**虧損**
第 2 回合	獲利	第 12 回合	獲利
第 3 回合	獲利	第 13 回合	獲利
第 4 回合	獲利	第 14 回合	獲利
第 5 回合	獲利	第 15 回合	獲利
第 6 回合	獲利	第 16 回合	獲利
第 7 回合	獲利	第 17 回合	獲利
第 8 回合	獲利	第 18 回合	獲利
第 9 回合	**虧損**	第 19 回合	獲利
第 10 回合	**虧損**	第 20 回合	獲利

千萬別以為出現這種情況的機率很小就可以不理，雖然風險往往是小機率事件，但一出事就足以致命。這種情況常見於期權

short 倉策略，或者大止蝕小止賺的期貨策略；更嚴重的是，資產價格的高波幅往往會展現出自相關性（Autocorrelation），例如，當連續虧損事件出現，萬一按金準備不足，交易者只能黯然出局，即使往後的回合如何賺錢，都是無仇報了。

低命中率但是單次回報超高的策略都有類似的問題。當獲利事件出現得非常少之後，信心問題就會比較明顯，因為不停的小幅度虧損累積起來亦會對資金造成壓力。由於我們不知道會否等到天荒地老才出現獲利事件，這種策略能運用的注碼不能太多。以一個命中率只有 1 成（即是虧損機率為 90%）的策略為例，就算命中回報大得足夠令期望值出現正數，連續 10 個回合虧損的機率達到 90%^10 = 35%，連續 15 回合虧損的機率也有 90%^15 = 20%，這都不是非常低的機率，意味著運氣稍為差一點便會遇上，你怎能肯定自己的運氣不差呢？

反過來說，命中率約 50% 的策略就沒有太大的事件分佈問題。雖然說買大小 10 次，連續輸超過 5 次的機會還是相當不小，約有 3% 的機率。但是，在典型的正期望值和命中率為 50% 的策略中，單次回報都必然高於單次虧損，所以其捱價壓力一般不會太大。這一點可以從蒙地卡羅方法的結果表現出來。

策略 I

Accuracy	20%
Upside (%) 8.5	Downside (%) 1.0
Avg monthly return	0.9%
Avg annual return	12%
Avg of monthly return SD	0.32%
Avg of all MDD	16.1%
Max MDD	39.8%
MDD SD	5.8%
Sharpe Ratio	2.94
% of loser	3%
min equity	68
max equity	1282
median equity	258
sd equity	148

策略 II

Accuracy	50%
Upside (%) 2.8	Downside (%) 1.0
Avg monthly return	1.1%
Avg annual return	13%
Avg of monthly return SD	0.01%
Avg of all MDD	6.3%
Max MDD	19.8%
MDD SD	2.1%
Sharpe Ratio	92.29
% of loser	0%
min equity	163
max equity	468
median equity	287
sd equity	66

策略 III

Accuracy	90%
Upside (%) 2.0	Downside (%) 9.0
Avg monthly return	1.0%
Avg annual return	12%
Avg of monthly return SD	0.44%
Avg of all MDD	23.4%
Max MDD	58.2%
MDD SD	7.5%
Sharpe Ratio	2.20
% of loser	2%
min equity	62
max equity	682
median equity	274
sd equity	114

以上 3 項策略的期望值都同樣是 0.9［註］，但策略 I 和策略 III 的平均 MDD、最大 MDD 和 MDD 標準差都明顯較大，風險比

起策略 II 高得多。然而，三者的月回報、年回報都非常接近，可見策略 II 的回報風險比例是比較出眾。

註：3 項策略的期望值

策略 I： 20% x 8.5 + 80% x (-1) = 0.9

策略 II： 50% x 2.8 + 50% x (-1) = 0.9

策略 III： 90% x 2 + 10% x (-9) = 0.9

更令人憂慮的是 % of Loser。在策略 I 和策略 III，竟然有小部分的測試結果竟然會是輸錢的，須知道這個 120 個回合的測是試足 100 次。雖然回合次數絕對不算少，但可反映即使是正期望值的策略，依然有機會輸錢。事實上，由於策略 I 和策略 III 的 Equity SD 較大，即使經過 120 回合，利潤仍然是較難預測的。

由此可見，即使是期望值相同，極端命中率策略必然有較大的風險，而最可取的策略是命中率只有一半一半，但其單次回報須高於單次虧損。這結論帶出了一個道理，程式交易策略的命中率不一定是愈高愈好。

補充多一點，筆者並非認為極端命中率的策略不可取。舉例說，如果你找到一個策略命中率達 9 成，但單次獲利和單次虧損幅度相差無幾的策略，難道你不會用嗎（當然這策略是否 too good to be true 又會是另一問題）？反過來說，我們應該稍為調高極端命中率策略的要求，藉以彌補當中較大的風險。此外，在資金控制

上，我們對這種策略應該比較保守，雖然未必會以它們作為交易的主力，但加入整體策略池裡面，亦可達到策略分散的效果。因此，極端命中率策略也有其存在意義。

〈小結〉

進入程式交易的世界就等同半隻腳進入了統計學的世界，除了編程知識之外，初學者最難掌握的就是統計學上的概念，很容易會被簡單回測結果騙到，以為自己設計了一個很好的策略，但到真錢交易時卻發現效果遠不如預期。這種問題主要是因為對各種風險因素的誤解，不小心墜進了數字的圈套。

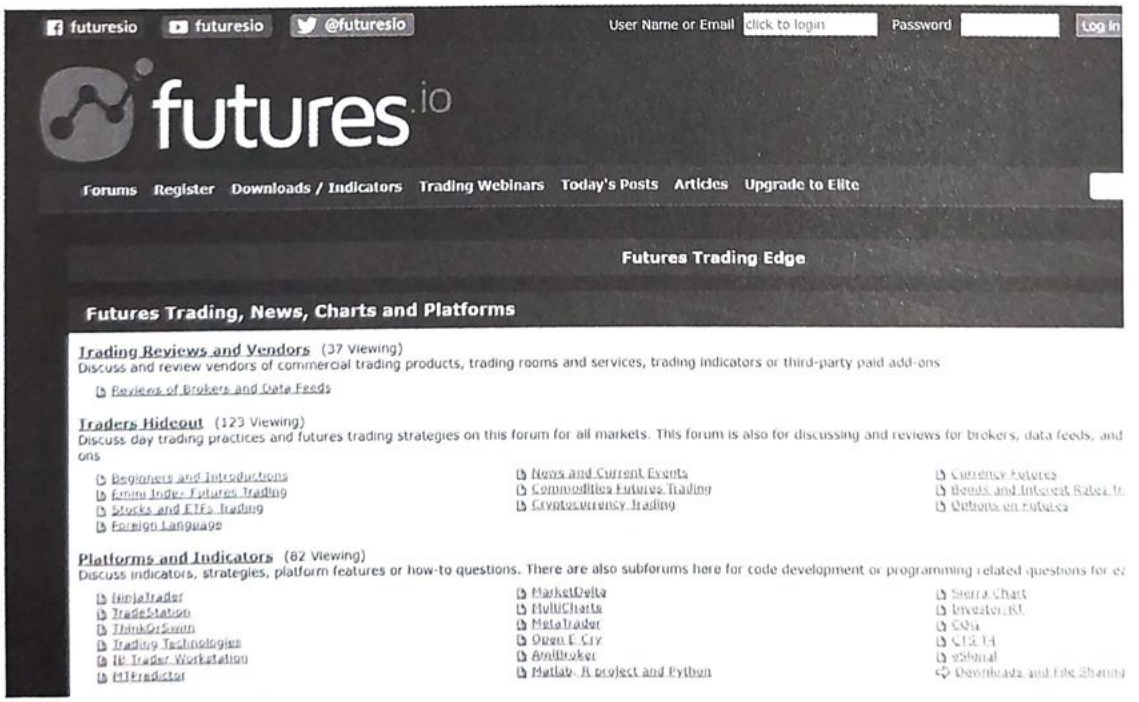
交易者必須牢記的是，在選擇和制定策略時，首要元素必然是風險，其餘所有元素都是後話。此外，我們必須了解回測的限制，回測的價格已經是歷史，雖然對未來有啟示作用，但相同價格軌跡未必會一模一樣地出現，甚至可能出現先前完全未見過的模式，典型例子是 2015 年的大時代，短短幾個月內出現急升急跌的模式是以前從未見過的，若果對這種從未見過的模式缺乏準備，吃虧的必然是自己。因此，我們須為未知的情況做足準備，才算有萬全的風險管理。

〈附錄〉

網上教學資源

網上的資源非常之多，要一一瀏覽幾乎不可能，因為時間成本也是交易者的成本之一。要節省時間成本就要選擇最有用的資源探索，在此向大家介紹幾個筆者認為最有用的網站，足夠大家深入研究。

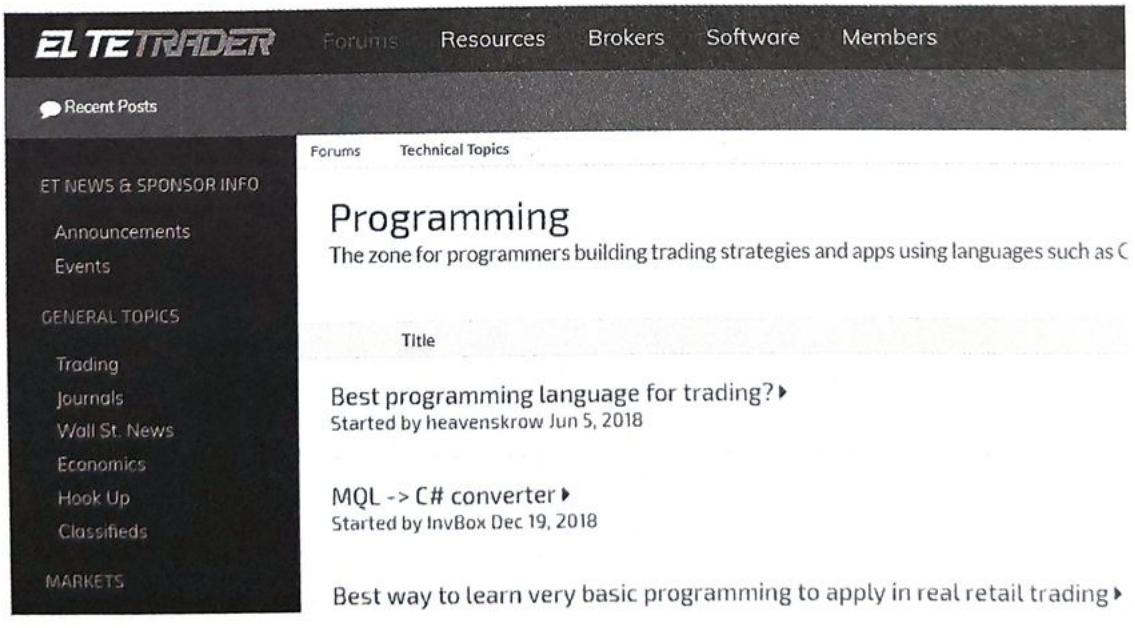
（1）futures.io



futures.io 可說是歷史最悠久的期貨交易討論區，當中一大部分是關於平台、程式語言、策略的討論。這個討論區有來自全世界的熱心網友，他們毫不吝嗇地分享知識和經驗，筆者在學習程式交易時也經常在此取經。由於這個討論區的地位實在崇高，不少第三方程式交易軟件都會派出官方代表在此解答網友的難題，間中更有不少神級交易員在此帶出非常深入的討論。

對於初學者，最有用的莫過於某些網友的程式碼分享。程式語言本身也是一種語言，所以學程式語言的過程就像學外語一樣，需要多閱讀其他人的編程方法，了解寫法背後的意義，透過不斷模仿和練習才會成功。futures.io 設有很多這樣的主題分享，新手老手同樣得益不淺。

（2）Elite Trader



同樣是一個交易員的討論區，Elite Trader 不限於討論程式交易的話題，你甚至可以討論想得到的交易品種。但根據筆者多年的瀏覽經驗，Elite Trader 的 Programming 討論一點都不會輸蝕 futures.io，甚至可以找到更多技術含量極高的話題，是一個高手必到的技術討論區。

(3) HKPTRC

若果嫌外國討論區太多英文，可以換個本地口味的。香港程式交易研究中心（Hong Kong Program Trading Research Center，HKPTRC）是筆者創立的一個實用資訊網站，有鑑於香港的程式交易風氣實在太弱，HKPTRC 的網站提供許多免費的教

學文章和短片，內容包括交易策略、實戰經驗、程式分享，希望可讓更多有興趣的人一起進步，以及能夠凝聚一班香港的程式交易者，讓程式交易不再是小眾的玩意。

程式交易的
理論與實踐

作者：	蔡嘉民（Calvin）
執行編輯：	呂泰康
美術設計：	周凱玲
出版及發行：	經濟日報出版社
	香港北角渣華道321號
	柯達大廈二期6樓
電話：	（852）2880 2444
傳真：	（852）2516 9989
網址：	www.etpress.com.hk
電郵：	etpress@hket.com
出版日期：	2019年7月（初版）
承印：	美雅印刷製本有限公司
定價：	港幣$180（台幣$810）
ISBN：	978-988-8501-52-6

散戶操盤勝率偏低，
經常高買低賣，
原因並非散戶沒優勢，
而是不了解大戶操盤方法。

要知道，投資炒賣絕非賭博，
程式交易運用大量歷史數據進行測試，
各種策略賺與蝕皆一目了然，
從此不須再為交易感到茫然。

此書帶你從機構投資者的角度，
深入淺出地認識程式交易、量化分析，
充分運用科技找出最高值博率的策略，
助你於市場中與大戶一決高下！

深望此書令大眾有系統化地認識程式交易，有助大家活學活用，不單帶來穩健的財富，還能於各種市況下保持身心康泰。

黃實誠

史丹福大學統計學博士、香港大學統計及精算學系名譽教授、
香港中文大學統計學系兼任教授

程式語言不是外星語言，程式交易亦非想像中的困難，大家所欠缺的，往往是一條清晰的學習階梯，以及有經驗人士的指引……

歐陽一心

香港程式交易研究中心董事、《期權投機客》系列作者

ISBN：978-988-8501-52-6
9 789888 501526 >

et press 經濟日報出版社

定價：HK$180（NT $810）
陳列類別：股票投資