Power Language 語法構造簡易說明

Power Language 主要是用來定義金融商品的交易條件，並交由電腦自動執行，這種策略執行一定會比人工交易更精準、更效率。

Power Language 的語法構造

一段完整且編譯成功的程式碼可以稱作為腳本（Scripts），Power Language 有 3 種腳本型式：

1. 可產生交易指令的買賣訊號。
2. 可在圖表區中繪出點、線或其他符號，幫助投資人進行視覺分析的技術指標。
3. 函數可獨立運算，且可在其他腳本中被呼叫重複使用。

Power Language 的重要組成元素

1. 關鍵字，也是 Power Language 中內建的函數，透過這些關鍵字可以完成數值運算與羅輯判斷。
2. 忽略字，可讓 Power Language 容易閱讀，但其實沒有作用，程式執行時會跳過這些忽略字。
3. 資料型別，共有數值、字串與布林值（True / False）。
4. 日期與時間，日期 YYYMMDD（YYY 從 1900 起算），時間為 24 小時制。
5. 運算子，分為數學運算子、關係運算子、邏輯運算子與字串運算。
6. 變數，有內部變數與外部變數，用來儲存資料（數值、字串與布林值）。

7. 陣列，含有多個元素的變數，使用時需經宣告。
8. 參數，用來傳遞資料 (數值、字串與布林值)。
9. 標點符號
10. 歷史資料引用，Power Language 提供 N Bars ago ([N]) 的語法，來取得特定期間前的歷史資料。

宣告變數

變數 : 是在程式執行過程中用來存放值的位置，沒有固定的值，可以在程式碼中的任何一個地方存取變數值。

變數宣告及語法

Variables : 變數名稱 (值)

例 : variable : Len (5) / variables : Len (5) / var : Len (5) / vars : Len (5)

變數名稱是變數的名字，在變數宣告時，要同時指定 1 個預設值，這個預設值同時決定了變數的資料型別 (數值、文字串、布林值)，多個變數宣告只需用逗號分隔即可。在 SniperIDE 中 value1 ~ value99 (預設值 : 0) 與 condition1 ~ condition99 (預設值 : False) 為系統內建不需透過變數宣告即可使用。

設定變數值 / 讀取變數值

變數宣告完成時，即可在程式中任一地方指定新的值給變數，但指定值需與變數資料型別一致。而變數最重要的作用是讓我們在程式中可以任意引用，引用方法就是在程式中直接使用變數名稱即可。

例：value1 = (High + Low) / 2，將中價設定為 value1，condition1 = volume > volume[1]，今日成交量是否大於昨日成交量設定為 condition1。

```
Condition1 = High[2] > High[1] And
//昨高小於前高而且
            Low[2] < Low[1] And
            //昨低大於前低而且
            Close > High[2] + 1 Points
            //收盤大於前高加 1 個 Points
If Condition1 Then
//假如 Condition1 為真
    FindPoint1( (Close/High[2]-1)*100, BottomSide, " 母子突破上漲 ")
    //找點並在 K 棒下方顯示 " 母子突破上漲 "
End If
//結束條件判斷式
```

參數

必須經由宣告才能使用，但是在程式執行過程中參數值是不能被修改的，參數是程式時執行預先定義的，我們可以透過變更參數值的方式來調整指標、買賣信號、個股查詢與 Candle pattern。

參數宣告及語法

Parameter : 參數名稱 (值)

例 : Parameter : Len (5) / parameters : Len (5) / Param : Len (5) / Params : Len (5)

參數名稱是參數的名字，在參數宣告時，要同時指定 1 個預設值，這個預設值同時決定了參數的資料型別 (數值、文字串、布林值)，多個參數宣告只需用逗號分隔即可，用參數來代替常數可以增加使用的便利性，參數的另一個重要功能即是參數最佳化。

參數值的運用

當參數宣告成功後，可以在程式中直接以參數名稱運用。例 :

```
Parameter: 日期參數( 50 ), 高低比率( 50 )
//參數宣告 : 期間與均線參數 ( 預設值為 50 )，高低比率 ( 預設為 50% )
Value1 = MAFC( Volume, 日期參數)
//50 日均量設定為 Value1
Value2 = Highest( High, 日期參數)
//50 日最高設定為 Value2
Value3 = Lowest( Low, 日期參數)
//50 日最低設定為 Value3


Parameter: 股價( Close ), 連續下跌次數( 2 )
// 參數宣告 : 價格參數 ( 預設值 : 收盤價 )，連續下跌次數 ( 預設值為 2 )
Variables: Count(0)
```

```
// 內部變數宣告 : 計次變數 ( 用來計算下跌次數，預設值為 0)

Count = 0

// 設定計算下跌次數為 0

For Value1 = 0 To 100

//迴圈開始 ( 計數從 0 到 100)

    If 股價[Value1] < 股價[Value1+1] Then

    // 假如價格小於昨日價格

        Count += 1

        // 計算下跌次數 + 1 ( 等同於 Counter = Counter + 1，Counter += 1 是
C++ 寫法，不易解讀 )

    ElseIf 股價[Value1] > 股價[Value1+1] Then

    // 否則假如價格大於於昨日價格

        Exit For

        // 中斷並跳出迴圈

    End if

    //結束條件判斷式

End For

//計數 0 執行結束，計數加 1 後，若未超過 100，則重新進行迴圈，若超過 100 則結
束迴圈
```

在這個例子中，我們宣告了一個名為股價的參數，預設值為 Close，但是我們可以透過參

數值的改變如將 Close 變更為 Low or High or open 或其他運算式，這個時候就不是用收盤價來計算了，這給與程式開發者很大的彈性運用空間。

陣列

陣列是可以儲存很多值的變數，定義好的陣列可以利用索引值（Index）來讀取或變更陣列中的元素值，一般來說，性質類似的值可以使用陣列來存取。

陣列宣告及語法

Array : 陣列名稱 [A1] (預設值)

Arrays : 陣列名稱 [B1] (預設值)

在 Power Language 中，陣列內的元素無限制，但是過大的陣列變數宣告，會佔用過多的記憶體，容易耗盡系統資源，影響程式執行效能。

更多例子 :

Array : Length[7](1),Length1[7](1)

宣告 Length & Length1 為數值陣列，內含 8 (Index : 0 ~ 7) 個元素，每個元素的初始值為 1。

陣列變數的讀取與變更

語法 :

陣列名稱[索引值] = Expression

例 : Array : Volume_High[9](0)

Volume_High[4] = Highest (Volume，10)

將過去 10 根 K Bar 的成交量最高值存入陣列 Volume_High 索引值為 4 (第 5

個元素 ） 中。

陣列變數的讀取

陣列變數宣告成功後，在程式執行過程中可以陣列名稱加上索引值來讀取陣列變數的值

例 ： 讀取陣列 Volume_High 索引值為 3 的元素值，並設定為 Value1。

Array : Volume_High[9](0)

Value1 = Volume_High[3]

使用者函數

“ 使用者函數 ” 的制訂是為了讓特定功能或可重複利用的程式碼片段包裝成使用者函數，讓程式開發者再次執行時只要呼叫函數名稱即可，不需重複進行編碼，可解除開發者的不便及具有易於管理的功能，在程式編碼的過程中，若有可以重複使用的程式片段或執行特定運算功能的程式碼，可以制定 “ 使用者函數 “，制定順序如下 ：

選擇開新檔案並開啟函數 (Function)

輸入 MA 的名稱並且設定數字型（Numeric），接著即可在編輯視窗中進行編輯。

內部變數 / 外部變數

只有在檔案內部回傳的變數，需要 Variable 變數宣告，可由外部指定的變數，需要 Parameter 變數宣告，使用 Parameter 宣告的變數可在模擬回測中進行參數最佳化以求取各種參數組合的績效表現。

```
Parameter: 上漲顏色(Red), 下跌顏色(Green), 比較棒數(6),權重(1)
//外部變數宣告：上漲顏色（預設值為紅色），下跌顏色（預設值為綠色），比較棒數（預設值為 6），權重（預設值為 1）
Variables: haClose(0), haOpen(0), haHigh(0), haLow(0), color(0)
//內部變數宣告（內部用）: ha 收盤，ha 開盤，ha 最高，ha 最低，顏色
If BarNumber = 1 then
//假如第 1 根 K 棒
    haOpen = open
    //將開盤價設定為 ha 開盤
    haClose = (Open + High + Low + Close) / 4
    //將 4 價平均值設定為 ha 收盤
    haHigh = MaxList( high, haOpen, haClose)
    //將最高、ha 開盤、ha 收盤中最大值設定為 ha 最高
    haLow = MinList( low, haOpen, haClose)
```

```
    //將最低、ha 開盤、ha 收盤中最小值設定為 ha 最低
End If
//結束條件判斷式
if BarNumber > 1 then
//假如 K 棒數大於 1
    haClose = (Open + High + Low + Close * 權重) / (3 + 權重)
    //將收盤加權與開高低取平均值設定為 ha 收盤
    haOpen = (haOpen[1] + haClose[1] * 權重) / (1 + 權重)
    //將昨日 ha 開盤 & 昨日 ha 收盤加權取平均值設定為 ha 開盤
    haHigh = MaxList(High, haOpen, haClose)
    //將最高、ha 開盤、ha 收盤中最大值設定為 ha 最高
    haLow = MinList(Low, haOpen, haClose)
    //將最低、ha 開盤、ha 收盤中最小值設定為 ha 最低
    if haClose > haOpen then
    //假如 ha 收盤大於 ha 開盤
        color = 上漲顏色
        //將紅色設定為 color
    else
    //否則
        color = 下跌顏色
```

```
        //將綠色設定為 color
    End If
    //結束條件判斷式
    for value1 = 1 to 比較棒數
    //迴圈開始 ( 計數由 1 到 6 )
        if haOpen <= MaxList(haOpen[value1], haClose[value1]) and
        //假如 ha 開盤小於等於昨日 ha 開盤 & 昨日 ha 收盤的最大值而且
            haOpen >= MinList(haOpen[value1], haClose[value1]) and
            //ha 開盤大於等於昨日 ha 開盤 & 昨日 ha 收盤的最小值而且
            haClose <= MaxList(haOpen[value1], haClose[value1]) and
            //ha 收盤小於等於昨日 ha 開盤 & 昨日 ha 收盤的最大值而且
            haClose >= MinList(haOpen[value1], haClose[value1]) then
            //ha 收盤大於等於昨日 ha 開盤 & 昨日 ha 收盤的最小值
            color = color[value1]
            //將昨日顏色設定為顏色
        End If
        //結束條件判斷式
    End For
    //計數 1 執行結束，計數加 1 後，若未超過比較棒數 ，則重新進行迴圈，若超過比較棒數則結束迴圈
```

```
        If Color =  上漲顏色  And CurrentContracts <= 0 Then
        //假如顏色是上漲顏色而且目前無未平倉部位或有空頭部位
            Buy ("HAL") next bar at market
            //次一根開盤市價買進 ( 單位數由 UI 設定參數控制 ),並在買進 K 棒下方顯示
" HAL "
            If AlertEnabled Then Alert(" HAL ") End If
        End If
        //結束條件判斷式
        If Color =  下跌顏色  And CurrentContracts >= 0 Then
        //假如顏色是下跌顏色而且目前無未平倉部位或有多頭部位
            Sell ("HAS") next bar at market
            //次一根開盤市價賣出 ( 單位數由 UI 設定參數控制 ),並在賣出 K 棒上方顯示
" HAS "
            If AlertEnabled Then Alert(" HAS ") End If
        End If
        //結束條件判斷式
end If
//結束條件判斷式
```

控制語法

1．IF

IF 函數在您指定的條件結果為 TRUE 時，會傳回一個值，而在結果為 FALSE 時傳回另一個值。例如，如果 A 大於 10，公式 = IF (A > 10 , " Over 10 " , " 10 or less ") 會傳回 " Over 10 "，如果 A 小於或等於 10，則會傳回 " 10 or less "。

格式：

IF (logical_test , [value_if_true] , [value_if_false])

IF 函數語法具有下列引數 (引數 : 將資訊提供給動作、事件、方法、屬性、函數或程序的值。)

logical_test : 必要，評估後可為 TRUE 或 FALSE 的任何值或運算式。例如，Close = 100 是邏輯運算式。如果收盤價等於 100，則運算式會評估為 TRUE。否則，運算式會評估為 FALSE。此引數可以使用任何比較計算運算子。

value_if_true : 選用，這是 logical_test 引數評估為 TRUE 時要傳回的值。例如，如果這個引數的值為文字字串 “ 平盤 “，且 logical_test 引數評估為 TRUE，則 IF 函數會傳回文字 “ 平盤 “。如果 logical_test 評估為 TRUE，且 value_if_true 引數已省略 (也就是說，logical_test 引數後面只有一個逗號)，則 IF 函數會傳回 0 (零)。若要顯示 TRUE 這個字，請在 value_if_true 引數中使用邏輯值 TRUE。

value_if_false : 選用，這是 logical_test 引數評估為 FALSE 時要傳回的值。例如，如果這個引數的值為文字字串 “ 上漲或下跌 “，且 logical_test 引數評估為 FALSE，則 IF 函數會傳回文字 “ 上漲或下跌 “。如果 logical_test 評估為 FALSE，且 value_if_false

引數已省略 (也就是說，value_if_true 引數後面沒有逗號)，則 IF 函數會傳回邏輯值 FALSE。如果 logical_test 評估為 FALSE，且 value_if_false 引數的值已省略 (也就是說，在 IF 函數中，value_if_true 引數後面沒有逗號)，則 IF 函數會傳回值 0 (零)。

巢狀流程控制

格式 :

```
IF ( logical-expression-1 ) THEN
    statements-1
ELSE IF ( logical-expression-2 ) THEN
    statements-2
ELSE IF ( logical-expression-3 ) THEN
    statement-3
ELSE IF ( ..... ) THEN
    ..........
ELSE
    statements-ELSE
END IF
```

若 logical-expression-1 為 TRUE，statements-1 會被執行，若 logical-expression-1 為 FALSE，則接下來會評估 logical-expression-2，其結果若為真則執行 statements-2 依此類推。一般來說假設 logical-expression-n 條件判斷為真，則 statements-n 會被

執行，然後結束條件判斷式，否則程式會繼續進行下一個判斷條件評估，假如所有的條件判斷皆評估為 FALSE，則直接執行 statements-ELSE。

2．四種不同類型的 Do 迴圈

Do 迴圈的類型	說明	範例
Do While ... Loop	Do While ... Loop 會驗算條件，如果條件為 True，就接著驗算條件之後的陳述式。驗算結束後，此迴圈會繼續驗算條件，而如果條件為 True，又會再次驗算陳述式。這個程序會繼續重複進行，直到條件為 False 時。	Do While condition statements Loop
Do Until ... Loop	Do Until ... Loop 與 Do While ... Loop 類似，不同之處在於 Do Until Loop 會一直進行陳述式的驗算，直到條件驗算結果為 True，而不是當條件為 True 時進行驗算。	Do Until condition statements Loop
Do ...Loop While	不論什麼情況之下，Do ... Loop While 只會驗算一次陳述式。 然後驗算條件，如果條件為 True，則再驗算一次陳述式。這個程序會繼續重複進行，直到條件為 False 時。	Do statements Loop While condition
Do ...Loop Until	與 Do ... Loop While 類似，不同之處在於 Do ... Loop Until 在驗算出條件結果為 True 前，都會一直驗算陳述式。	Do statements Loop Until condition

Do 迴圈支援 Exit Do 陳述式立即跳出迴圈。Exit Do 陳述式與 For ... Next 迴圈中的 Exit For 類似。

範例：

```
Do While sum < 100
   sum = sum + 10
Loop
```

Do While 會評估變數 sum，查看它是否小於 100，如果是，就執行下一行程式碼；如果不是，就移到 Loop 之後的下一行程式碼。Loop 關鍵字會命令程式碼回到 Do While 行，然後評估 sum 的新值。

```
Do Until sum >= 100
   sum = sum + 10
Loop
```

這段程式碼與 Do ... While 陳述式的程式碼類似，但是這次程式碼是評估 sum 變數，查看它是否等於或大於 100。

3．For ..... End For 迴圈

在變數中設定初始值與終止值及增加一定的增加值（Step 以下若省略的話以基本增價值+1）比最終值小或相同為止就是執行 Statement – 1、Statement – 2，遇到 End For 的話就會結束 For 迴圈，For 迴圈支援 Exit For 陳述式立即跳出迴圈，Exit For 陳述式與 Do ... Loop 迴圈中的 Exit Do 類似。

格示：

```
For 變數 = 開始 To 結束 [Step 增減值]
    Statement - 1
  [Exit For]
    Statement - 2
End For
```

巢狀式 For 迴圈

```
For I=1 To 10
    For J=1 To 10
        For K=1 to 10
            Statement
        End For
    End For
End For
```

範例：

```
for value1 = 1 to 6
//迴圈開始 ( 計數由 1 到 6 )
    if haOpen <= MaxList(haOpen[value1], haClose[value1]) and
    //假如 ha 開盤小於等於昨日 ha 開盤 & 昨日 ha 收盤的最大值而且
        haOpen >= MinList(haOpen[value1], haClose[value1]) and
```

```
            //ha 開盤大於等於昨日 ha 開盤 & 昨日 ha 收盤的最小值而且
            haClose <= MaxList(haOpen[value1], haClose[value1]) and
            //ha 收盤小於等於昨日 ha 開盤 & 昨日 ha 收盤的最大值而且
            haClose >= MinList(haOpen[value1], haClose[value1]) then
            //ha 收盤大於等於昨日 ha 開盤 & 昨日 ha 收盤的最小值
            color = color[value1]
            //將昨日顏色設定為顏色
        End If
        //結束條件判斷式
    End For
    //計數 1 執行結束，計數加 1 後，若未超過比較棒數 ，則重新進行迴圈，若超過比較棒數則結束迴圈（If 條件判斷式之後的比較判斷會因為 For 迴圈的重複執行而與昨日、前 2 日、 ..... 、前 6 日等進行比較。
```

演算關係參考

1．算術運算子

記號	使用法
+(加法)	Close + Low
-(減法)	Close – Low
*(乘法)	Close * Low

/(除法)　Close / Low

^(指數)　Close ^ 2

2．關係運算子

記號		使用法
<	(小於)	Close < Low(1)
<=	(小於或等於)	Close <= Low(1)
<>	(不等於)	Close <> 100
=	(等於)	Close = 100
>	(大於)	Close > Close(1)
>=	(大於或等於)	Close >= High(1)

3．邏輯運算子

在邏輯值中判斷真或假的結果值，隨著邏輯演算者所使用的演算結果中設定的條件，若滿足的話變成 TRUE (真) 的值，若設定的條件不能滿足的話是變成 FALSE (假) 的值。

Or : 2 個之中只有一個是真的話就變換為真 (TRUE)。

And : 2 個都是真的話變換為真 (TRUE)。

範例：

```
If Condition2 And CurrentContracts >= 0 Then
//假如 Condition2 為真 & 目前無未平倉合約或有多頭部位
    Sell (" Reverse to Down ") next bar at market
```

```
        //次 1 根 K 棒開盤市價賣出 & 並在 K 棒上方顯示 " Reverse to Down "
    End If
    //結束條件判斷式

If Condition1 Or Condition2 Or Condition3 Or Condition4 Or Condition5 Or
Condition6 Then
 //若符合上述 6 種狀況其中之 1
    FindPoint1( (Close-Low) / Low * 100, BottomSide, " 上升轉換 ")
    //找點，並在 K 棒下方顯示 " 上升轉換 "
End if
//結束條件判斷式
```

4．運算優先順序

制定策略時各種類的運算子會混合使用，這時以一般性的方式進行由左側至右側演算，並遵循運算子的運算優先順序，另外為了明確的意義傳達和順序設定可利用括弧“ ( ) ”，以優先處理括弧內的運算式。

運算子優先順序

1．求指數的次方值（ ^ ）

2．負值運算子（ - ）

3．乘法（ * ）和除法（ / ）

4．加法（＋）和減法（－）

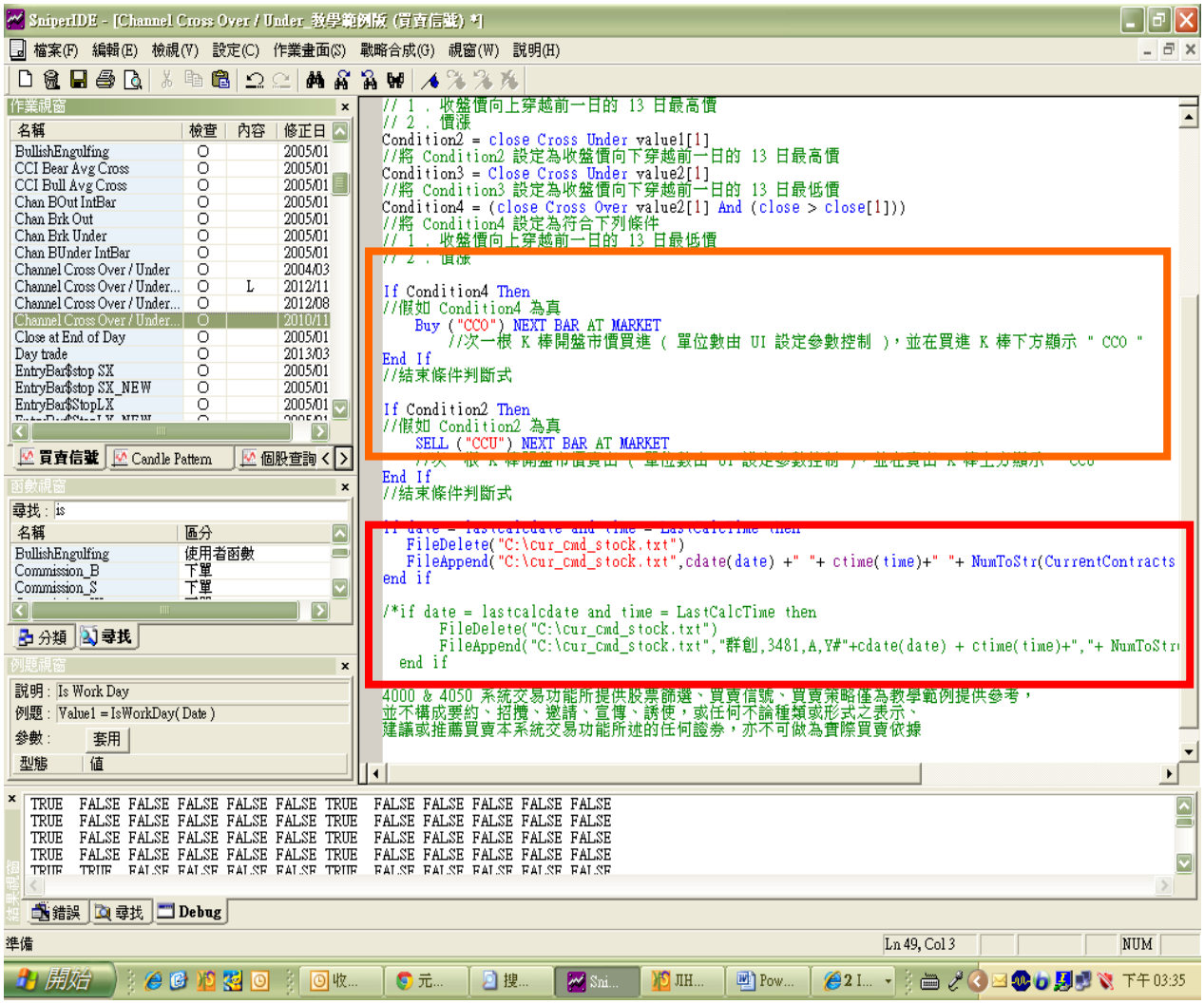
5．邏輯運算子（AND , OR）

[參考] 利用括弧（ ）可以調整優先順序。

注釋文

使用者制定函數時，在 program 的長度很長或很複雜的情形下，program 的制定中需要記錄有關聯的注釋文以利解讀，注釋文所表示的情形有兩個,表示單行的情形下用 “ // “，在文章開頭時若使用 “ /* “，則 /* 之後皆視為註釋文。

範例：



歷史值的引用 - [N] 的用法

函數或變數值可以引用歷史值，只要在其後加上 [N] 即可。

範例：

在 Price[X]中, [X]是一種標記，參照 X 根 K Bar 前的值。

```
For i = 1 To 長期均線
//迴圈開始 ( 計數由 1 到 30 )
    If Low[i] < MovingAverage[i] Then
    //判斷前幾根 K 棒 (i) 最低是否小於前幾根 K 棒 (i) 的 30 日移動平均線
        UpperTerm = i - 1
        //若最低小於長期均線則將最低在均線上方的棒數設定為最低在長期均線上方期間
        Exit For
        //因最低小於長期均線，迴圈終止
    End If
    //結束條件判斷式
End For
//計數 1 執行結束，計數加 1 後，若未超過 30，則重新進行迴圈，若超過 30 則結束迴圈
```

畫圖函數 ー https://fsmarket.fbs.com.tw/hts/MainPage_sts_QnA000C10.pdf

SniperIDE 使用說明 ー https://fsmarket.fbs.com.tw/hts/MainPage_sts_QnA000C4.pdf