程式交易概論

製作人：黃宥輔

程式交易(Program Trading)是什麼

- 近年來，隨著網路與電腦迅速發展，傳統的金融商品交易也漸漸轉由電腦處理，衍生出程式交易
- 程式交易為利用程式、軟體進行自動化或半自動化的金融商品買賣操作

程式交易特點

優點	缺點
效率高，能在短時間內 比對上百支金融商品資料	電腦設備需要穩定的環境
嚴守紀律，消除因為人性的不安定因素	若邏輯不完善可能會導致預期外的狀況
自動化交易省時省力	以歷史資料回測建構的策略 不一定代表未來的狀況
能在快速變動的市場中搶得先機	某些傳統策略較難轉換為程式邏輯， 如形態學
可以利用歷史資料進行回溯測試(回測)，建構交易策略	學習門檻較高

程式交易主要做法

1. 套裝軟體：Multichart、XQ…
2. 程式語言串接API

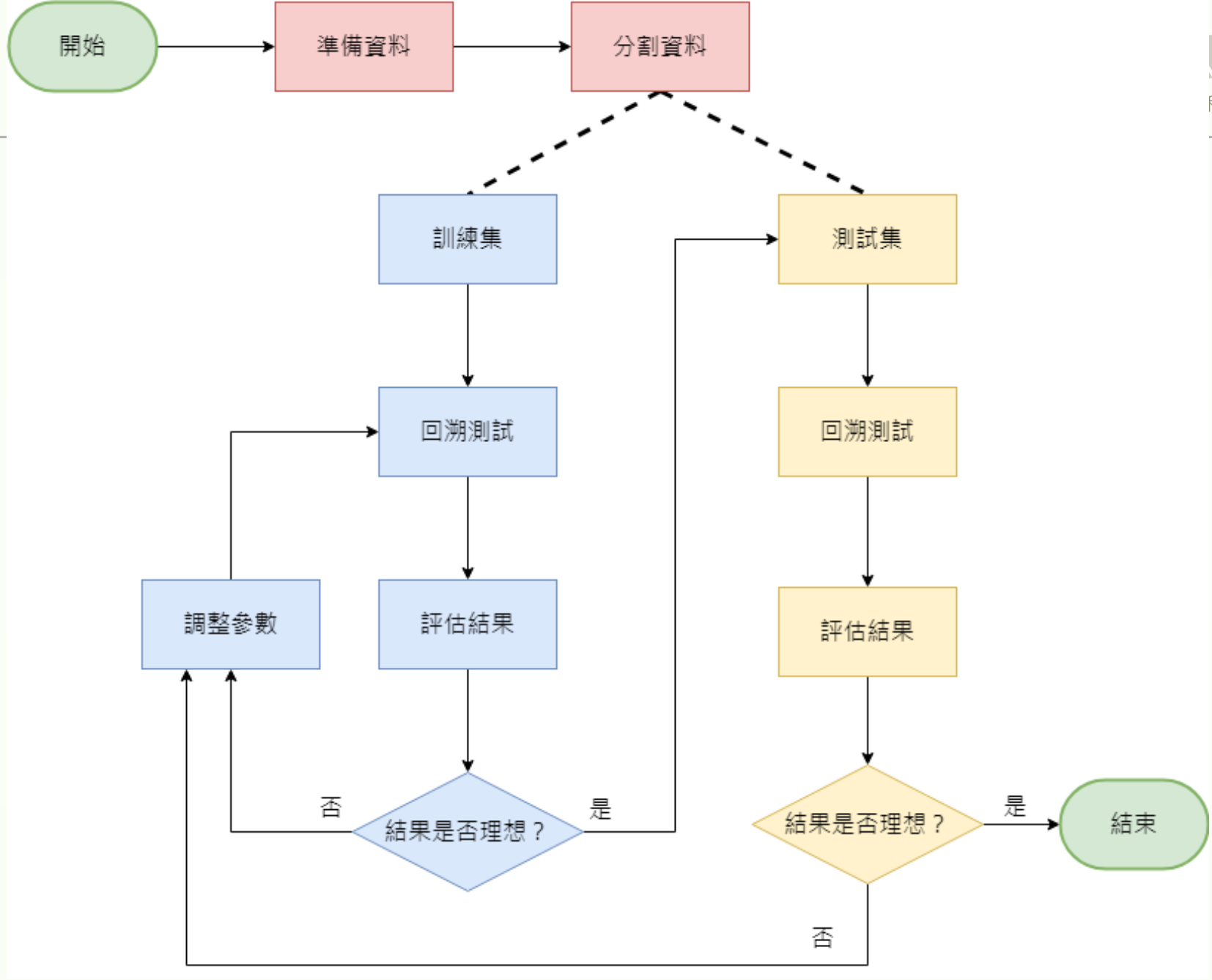
套裝軟體與串接API

比較項目	套裝軟體	Python接API
資料來源	內建	自行抓取
技術指標	豐富	TA-LIB
難易度	較簡易	較困難
靈活度	較低	較高
費用與限制	有	免費

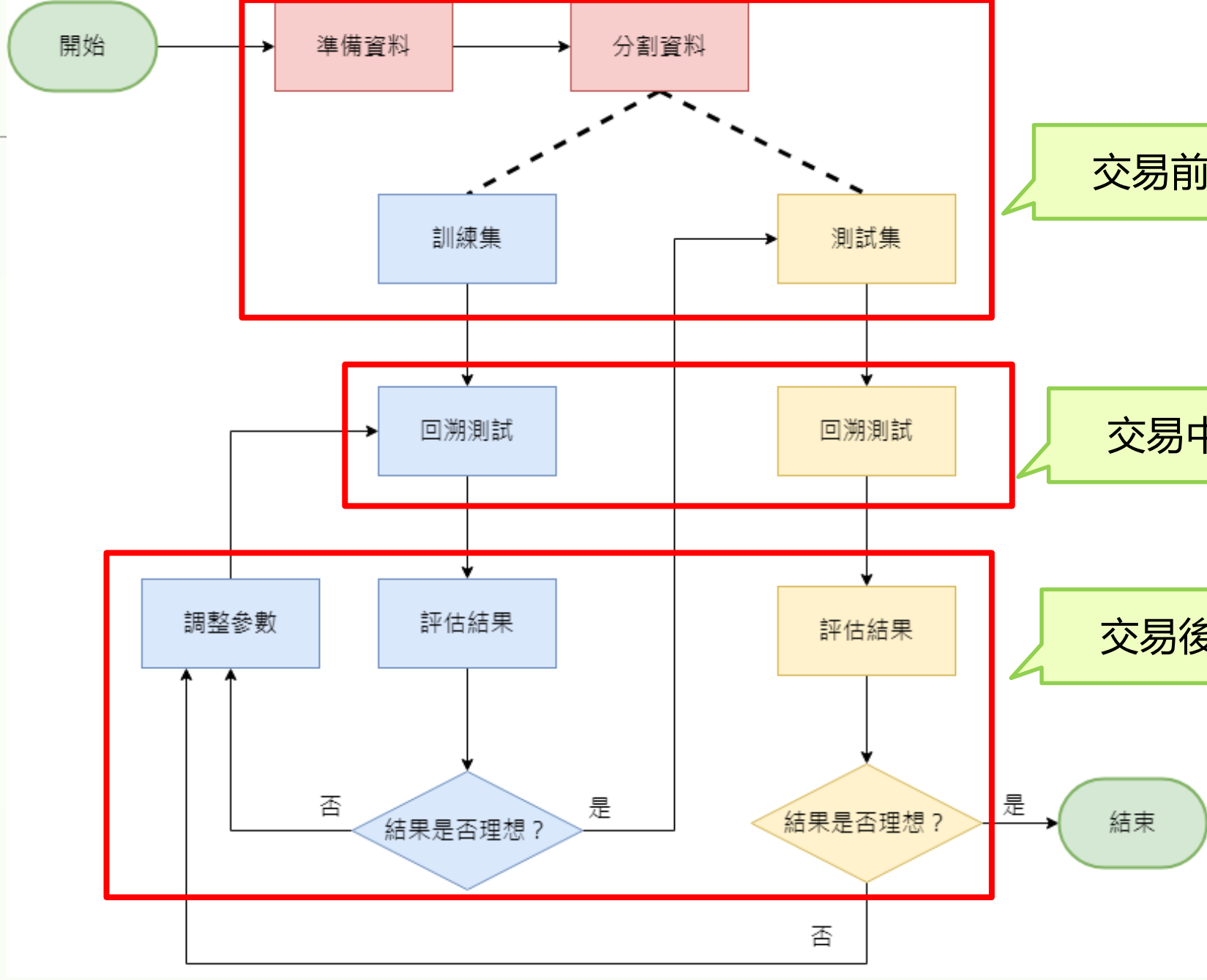
回測要件

時機	基本項目	進階項目
交易前	1. 選擇標的 2. 選擇交易頻率 3. 決定實驗期間	
交易中	1. 進出場條件、趨勢判斷 2. 進出場時機 3. 手續費與交易稅 4. 資金限制	1. 滑價 2. 還原股價 3. 加減碼規則(部位管理)
交易後	1. 產出績效指標	1. 自動化優化參數

程式交易流程



程式交易流程



交易前 - 選擇標的

- 僅挑選金融商品種類，不挑選特定公司
- 挑選特定產業
- 依據條件挑選金融商品，如本益比、營收成長率等
- 挑選基金成分股，如 0050

交易前 - 選擇交易頻率

- 高頻交易(high-frequency trading，HFT)
  - 指在短時間對一個金融商品進行多次的建倉與平倉
  - 通常採日內的資料
  - 需要非常高等級的設備
- 短線價差交易
  - 持有股票賺取價差為目的進行買賣
- 長線投資
  - 投資穩定成長或是穩定發放股息股利

交易前 - 決定實驗期間

- 巴菲特：投資表現最少以3年衡量，5年為佳，並且與大盤比較，若表現不如大盤，那何必白費力氣呢？
- 台灣的大盤可以以大盤指數或 0050 作為比較基準

交易前 – 範例說明

標的：經 NSGA2 篩選過後，相關係數最小的 5 檔 0050 成分股

交易頻率：日

實驗期間：

- 訓練集 – 2015/01/01 到 2016/12/31
- 測試集 - 2017/01/01 到 2017/12/31

交易前 – 準備資料(主程式)

```
28  # 傳入參數，呼叫程式
29  train_data, test_data = get_data(stock_id = 1101,
30            train_start_date = "2015-01-01",
31            train_end_date = "2016-12-31",
32            test_start_date = "2017-01-01",
33            test_end_date = "2018-01-01",
34            )
35
36  # 輸出結果
37  print(train_data)
38  print("---------------")
39  print(test_data)
```

傳入
1. 股票代號
2. 訓練集開始日期
3. 訓練集結束日期
4. 測試集開始日期
5. 測試集結束日期

輸出結果

交易前 – 準備資料(get_data函數)


```python
# -*- coding: utf-8 -*-
import pandas as pd # 資料整理工具

def get_data(**para):
    # 將傳入的參數整理成變數
    stock_id = para["stock_id"] # 股票代號
    train_start_date = para["train_start_date"] # 訓練集開始日期
    train_end_date = para["train_end_date"] # 訓練集結束日期
    test_start_date = para["test_start_date"] # 測試集開始日期
    test_end_date = para["test_end_date"] # 測試集開結束日期

    # 讀取資料
    path = "股價資料_整合/"
    data = pd.read_excel( f"{path}{stock_id}.xlsx" )

    # 帶入日期，篩選訓練集的資料
    train_data = data[ (data["日期"] >= train_start_date) &
                       (data["日期"] <= train_end_date)
                       ]

    # 帶入日期，篩選測試集的資料
    test_data = data[ (data["日期"] >= test_start_date) &
                        (data["日期"] <= test_end_date)
                        ]

    return train_data, test_data
```

交易前 – 準備資料(get_data函數)


```python
# -*- coding: utf-8 -*-
import pandas as pd # 資料整理工具

def get_data(**para):
    # 將傳入的參數整理成變數
    stock_id = para["stock_id"] # 股票代號
    train_start_date = para["train_start_date"] # 訓練集開始日期
    train_end_date = para["train_end_date"] # 訓練集結束日期
    test_start_date = para["test_start_date"] # 測試集開始日期
    test_end_date = para["test_end_date"] # 測試集開結束日期

    # 讀取資料
    path = "股價資料_整合/"
    data = pd.read_excel( f"{path}{stock_id}.xlsx" )

    # 帶入日期，篩選訓練集的資料
    train_data = data[ (data["日期"] >= train_start_date) &
                       (data["日期"] <= train_end_date)
                       ]

    # 帶入日期，篩選測試集的資料
    test_data = data[ (data["日期"] >= test_start_date) &
                      (data["日期"] <= test_end_date)
                      ]

    return train_data, test_data
```

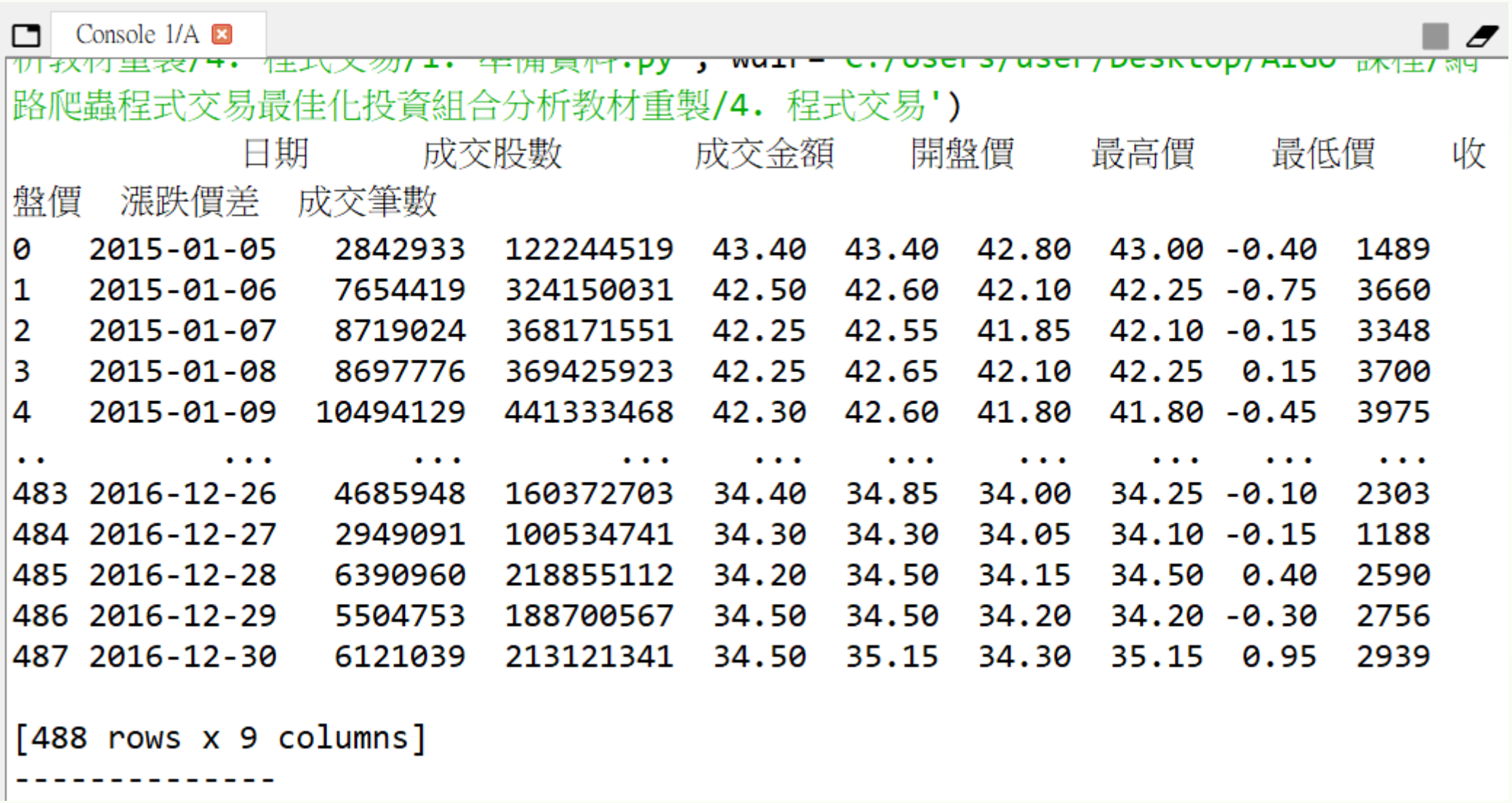
將傳入的參數指派給變數

根據股票代號，讀取資料

根據分割日期，
將資料進行篩選

回傳訓練集、測試集

交易前 – 準備資料

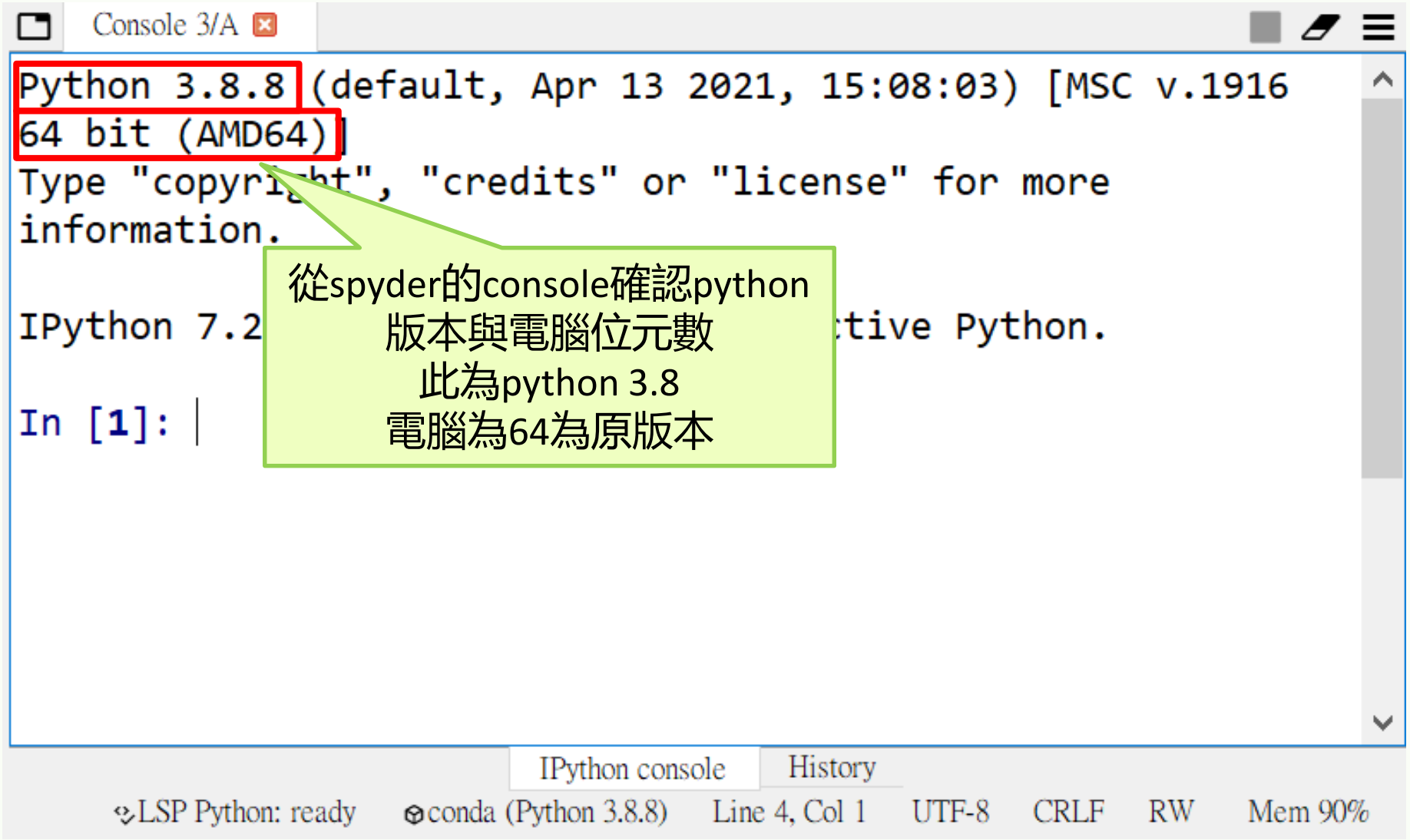


交易前 – 準備資料

AI.FINTECH
AI 金融科技中心

Console 1/A

```
--------------
            日期       成交股數        成交金額     開盤價     最高價     最低價     收
盤價   漲跌價差   成交筆數
488 2017-01-03   2890982  101450502  35.45  35.45  34.80  35.15   0.00  1669
489 2017-01-04   3296820  115895011  35.20  35.30  35.00  35.25   0.10  1737
490 2017-01-05   4098017  143758836  35.00  35.25  34.90  35.25   0.00  1788
491 2017-01-06   3542625  124391293  35.30  35.30  34.90  35.25   0.00  1774
492 2017-01-09   4877900  170970138  35.25  35.35  34.90  34.90  -0.35  1515
..          ...       ...        ...    ...    ...    ...    ...    ...   ...
729 2017-12-25  15153859  543236324  35.30  36.25  35.10  36.15   1.00  5787
730 2017-12-26  18802178  683520276  36.10  36.50  35.95  36.40   0.25  7927
731 2017-12-27   8808450  319904500  36.10  36.45  36.00  36.20  -0.20  3132
732 2017-12-28   7612103  275264758  36.10  36.25  36.00  36.20   0.00  2577
733 2017-12-29  13039478  474385049  36.20  36.80  36.10  36.45   0.25  3330

[246 rows x 9 columns]
```



交易中 – TA-lib 技術指標計算套件安裝

進入此網頁

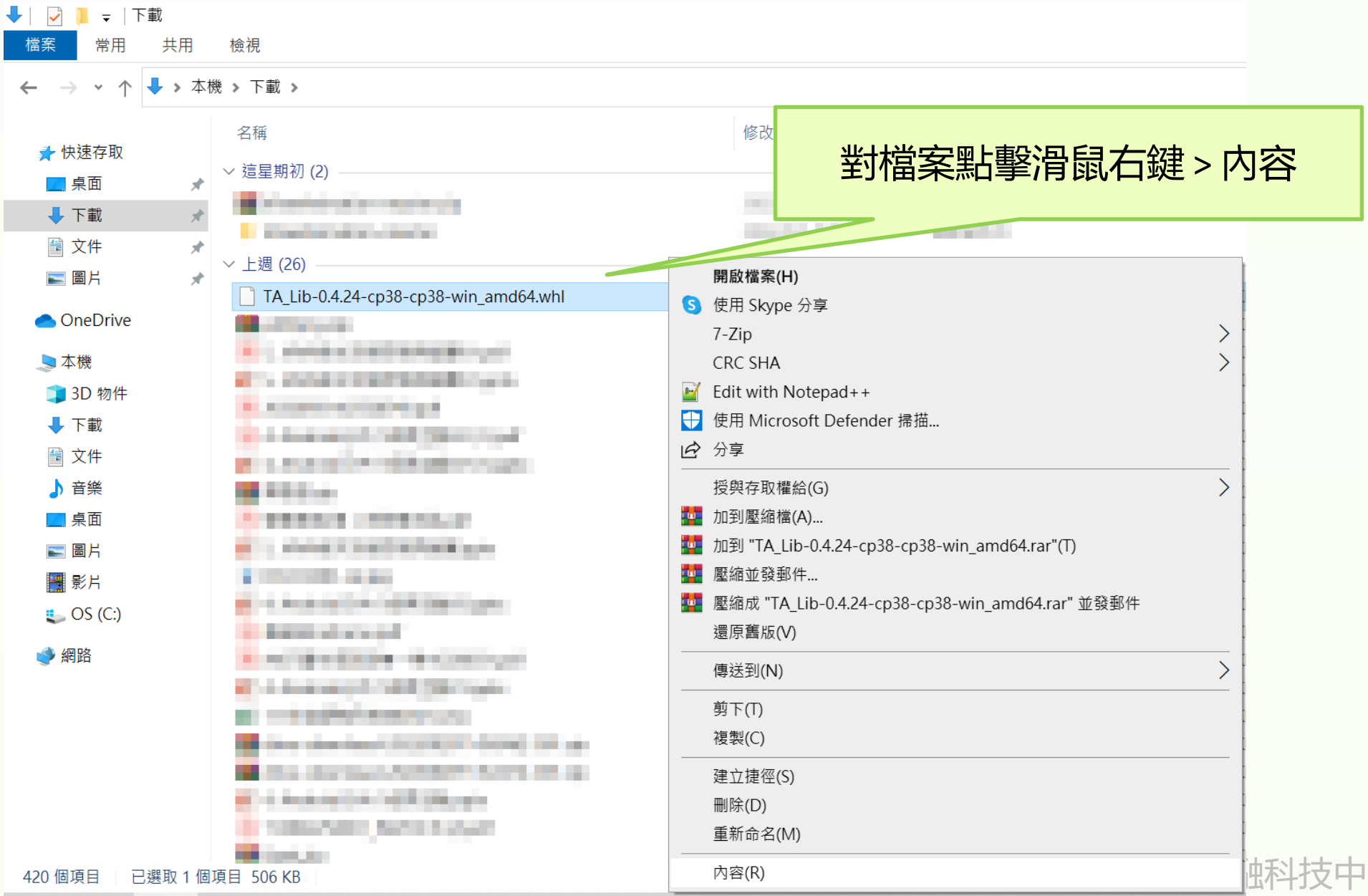
https://www.lfd.uci.edu/~gohlke/pythonlibs/#ta-lib

交易中 – TA-lib 技術指標計算套件安裝

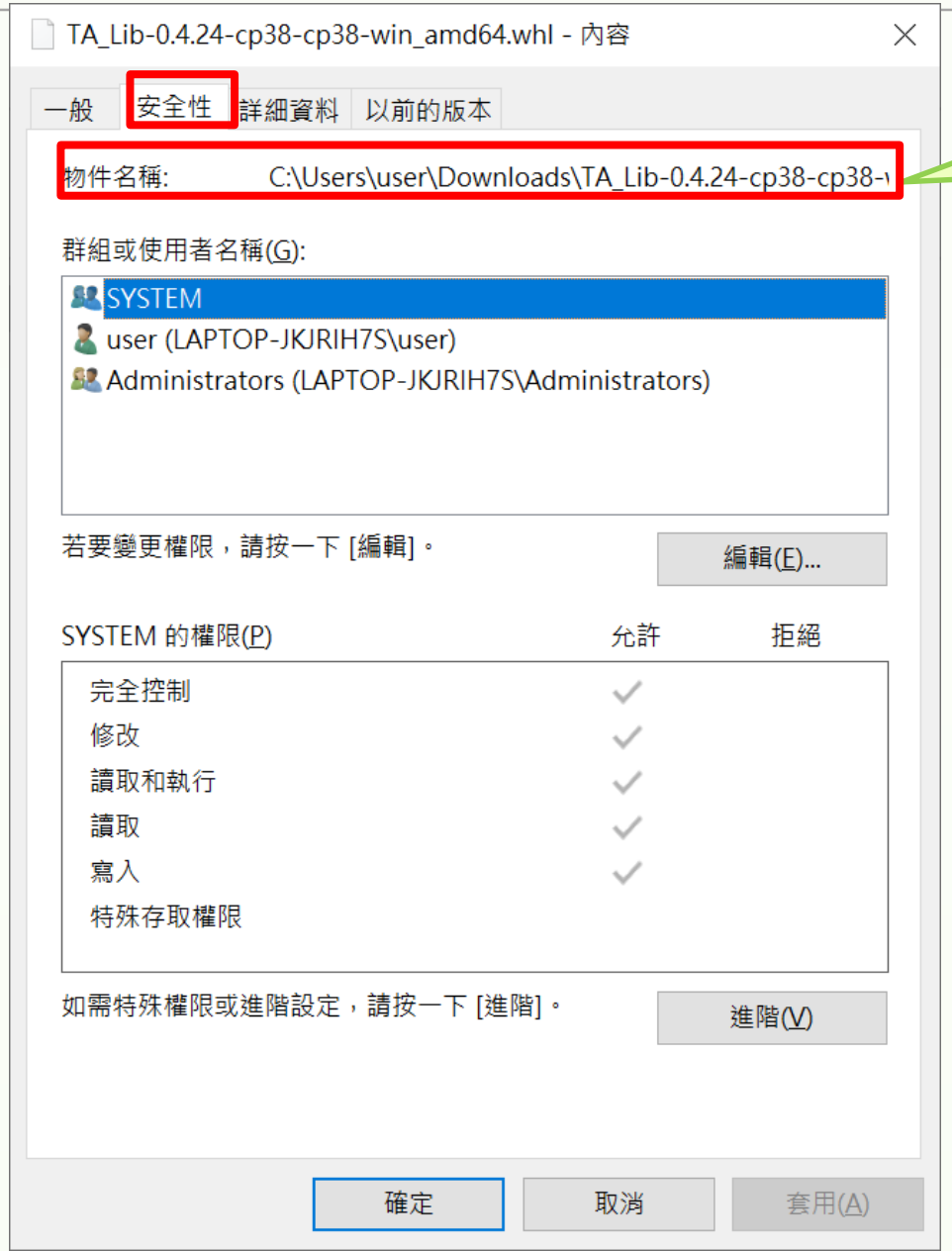
根據安裝的Python版本與電腦位元數，挑選whl檔
範例為Python 3.8版
64位元的電腦
故選擇此檔案

https://www.lfd.uci.edu/~gohlke/pythonlibs/#ta-lib

交易中 – TA-lib 技術指標計算套件安裝



交易中 – TA-lib 技術指標計算套件安裝

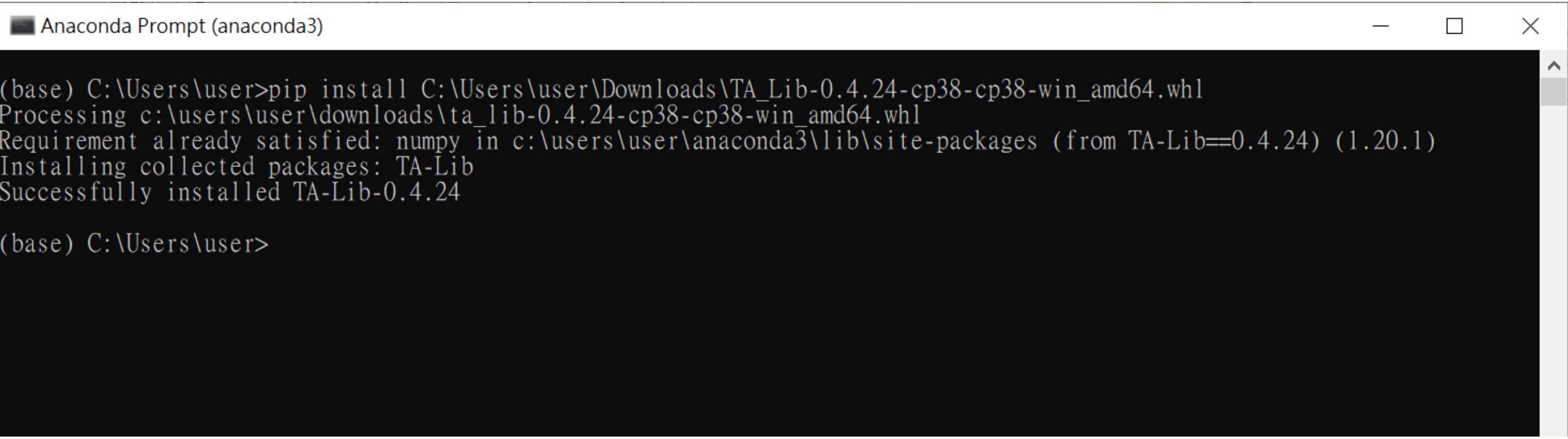


到安全性的頁籤中可以找到完整路徑，將其複製

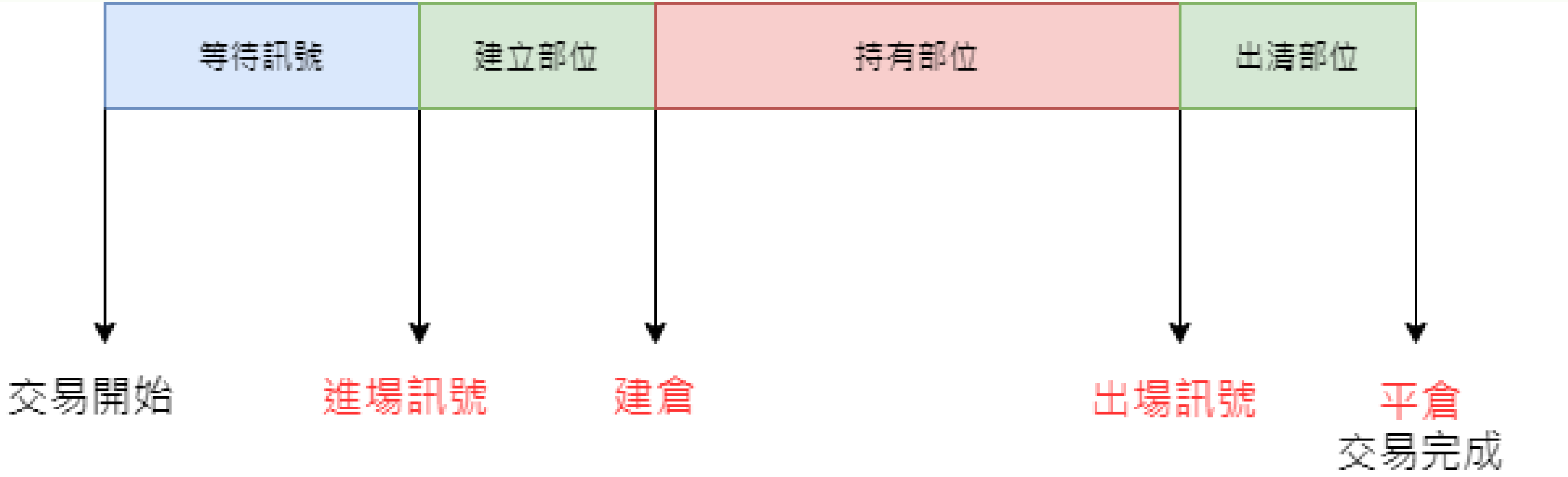
交易中 – TA-lib 技術指標計算套件安裝

在Anaconda Prompt中輸入指令安裝TA-Lib

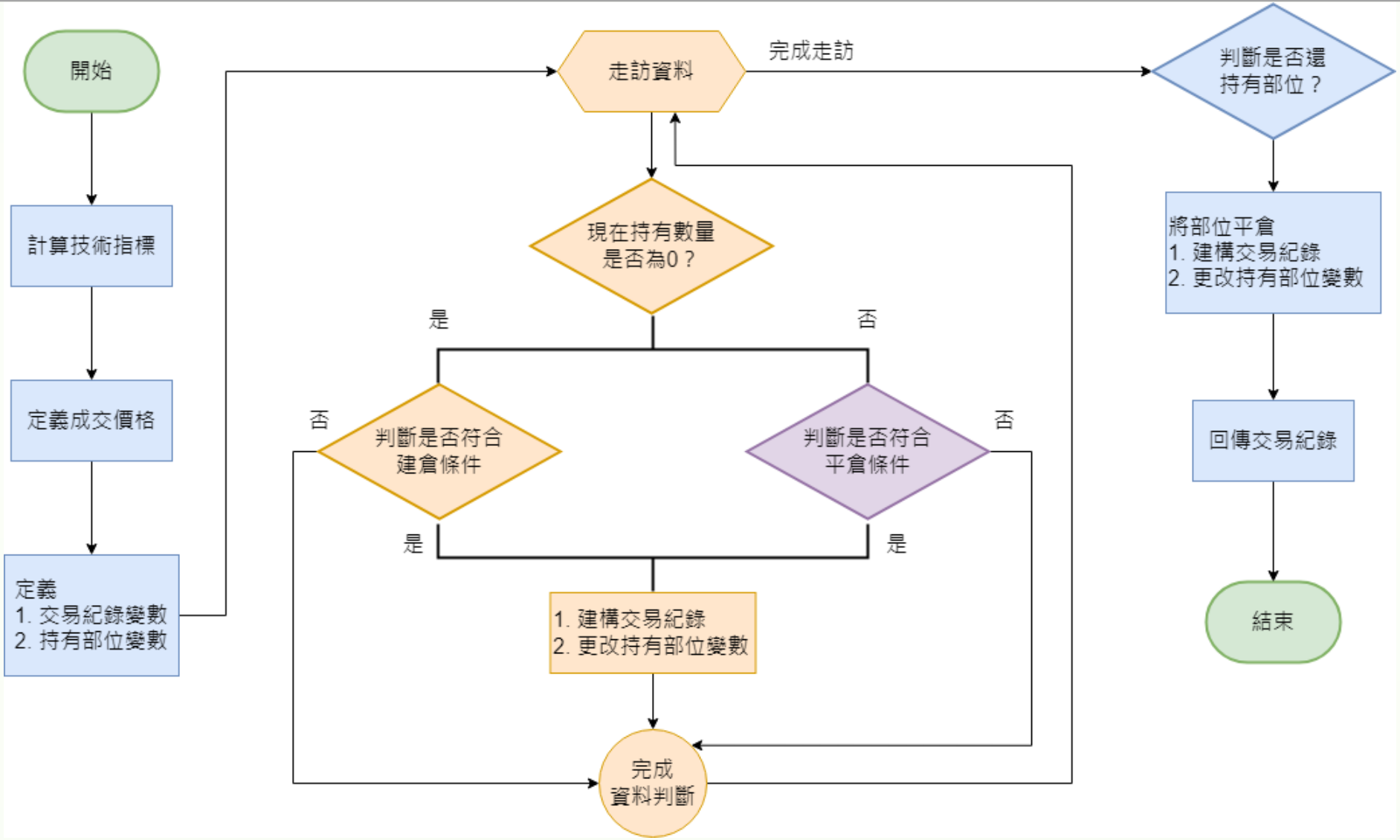
pip install C:\Users\user\Downloads\TA_Lib-0.4.24-cp38-cp38-win_amd64.whl



交易中 - 進出場時機

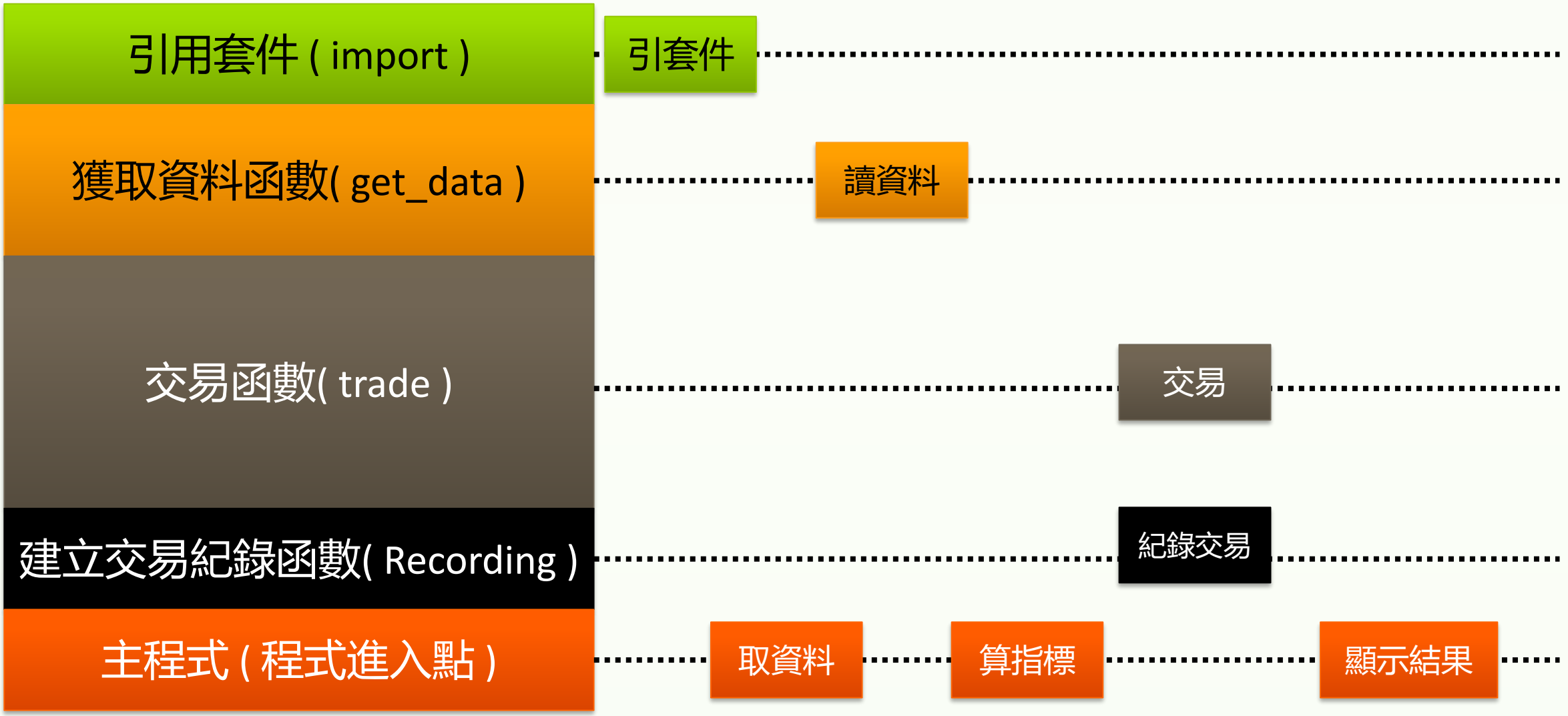


交易中 – 建構回測策略



交易中 – 整體程式擺放順序

交易中 – 整體程式擺放順序



交易中 – RSI 指標

- 相對強弱指標(Relative Strength Index, RSI)，一種用來評估買賣雙方強弱力度的技術指標，公式如下

- $\text{RSI} = \frac{\text{UP}}{\text{UP+DOWN}} \times 100$
- 其中UP為n日內平均上漲值；DOWN為n日內平均下跌值
- 範圍在 0 到 100 之間
- 常用週期為6日、9日、12日等週期，發明人Wilder 偏好用14日

交易中 – 主程式


```
121  # 以隔天的開盤價當作成交價格
122  train_data["成交價格"] = train_data["開盤價"].shift(-1)
123  # 最後一天以收盤價當作交易價格
124  train_data["成交價格"].fillna(method = "ffill", inplace = True)
125  # 計算 RSI5
126  train_data["RSI_5"] = talib.RSI(train_data["收盤價"], 5)
127
128  # 以隔天的開盤價當作成交價格
129  test_data["成交價格"] = test_data["開盤價"].shift(-1)
130  # 最後一天以收盤價當作交易價格
131  test_data["成交價格"].fillna(method = "ffill", inplace = True)
132  # 計算 RSI5
133  test_data["RSI_5"] = talib.RSI(test_data["收盤價"], 5)
134
135  trade(train_data)
136  print(record_df)
137
138  trade(test_data)
139  print(record_df)
```

交易中 – 主程式

```
121  # 以隔天的開盤價當作成交價格
122  train_data["成交價格"] = train_data["開盤價"].shift(-1)
123  # 最後一天以收盤價當作交易價格
124  train_data["成交價格"].fillna(method = "ffill", inplace = True)
125  # 計算 RSI5
126  train_data["RSI_5"] = talib.RSI(train_data["收盤價"], 5)
127
128  # 以隔天的開盤價當作成交價格
129  test_data["成交價格"] = test_data["開盤價"].shift(-1)
130  # 最後一天以收盤價當作交易價格
131  test_data["成交價格"].fillna(method = "ffill", inplace = True)
132  # 計算 RSI5
133  test_data["RSI_5"] = talib.RSI(test_data["收盤價"], 5)
134
135  trade(train_data)
136  print(record_df)
137
138  trade(test_data)
139  print(record_df)
```

接續取得資料
定義成交價格與
計算RSI指標

將資料帶入 trade 函數取得交易紀錄
(下一個部分會開始建構trade函數)

交易中 – 交易函數(trade)

AI.FINTECH
AI 金融科技中心

```
29  def trade(data_df):
30      global record_df, hold_data # 設定全域變數，讓其他函數也可以使用者兩個變數
31
32      # 定義交易紀錄
33      record_df = pd.DataFrame( columns = ["time", "price", "LS", "num"] )
34
35      # 定義持有資訊
36      hold_data = {
37          "price" : 0,
38          "num" : 0 # 做多部位為正數，做空部位為負
39          }
```

交易中 – 交易函數(trade)

將交易紀錄、持有記錄設為全域變數

```
29  def trade(data_df):
30      global record_df, hold_data # 設定全域變數，讓其他函數也可以使用者兩個變數
31
32      # 定義交易紀錄
33      record_df = pd.DataFrame( columns = ["time", "price", "LS", "num"] )
34
35      # 定義持有資訊
36      hold_data = {
37          "price" : 0,
38          "num" : 0 # 做多部位為正數，做空部位為負
39          }
```

交易紀錄，紀錄
1. 交易時間(time)
2. 成交價格(price)
3. 多空方向(LS)
4. 成交數量(num)

持有資訊，紀錄未平倉的
1. 建倉時的成交價格(price)
2. 建倉時的成交數量(num)

其中成交數量做多以正數，做空以負數代表

交易中 – 交易函數(trade)


```
29  def trade(data_df):
        ...
41      for index, row in data_df.iterrows():
42          # 沒有持有部位，建倉判斷
43          if hold_data["num"] == 0:
44              condition1 = row["RSI_5"] < 20 # 若RSI小於20，則做多(Long)
45              condition2 = row["RSI_5"] > 80 # 若RSI大於80，則做空(Short)
46
47              # 符合做多條件
48              if condition1:
49                  trade_price = row["成交價格"] # 取出當下的成交價格
50                  num = 1 # 交易數量
51                  Recording(row["日期"], trade_price, "L", num) # 紀錄做多交易
52
53                  # 更動持有狀況，紀錄成交價格與交易數量
54                  hold_data["price"], hold_data["num"] = trade_price, num
55
56              # 符合做空條件
57              elif condition2:
58                  trade_price = row["成交價格"] # 取出當下的成交價格
59                  num = 1 # 成交數量
60                  Recording(row["日期"], trade_price, "S", num) # 紀錄做空交易
61
62                  # 更動持有狀況，紀錄成交價格與交易數量，做空數量以負數代表
63                  hold_data["price"], hold_data["num"] = trade_price, -num
```

交易中 – 交易函數(trade)

逐列走訪資料

當下持有股票數量為0，則進入建倉判斷

建倉條件分別為
- 做多條件 RSI < 20
- 做空條件 RSI > 80

若符合建倉條件
則利用Recording函數
紀錄交易、
更新持有資料
(hold_data)

做空數量以負數表示

```
29  def trade(data_df):
        ...
41      for index, row in data_df.iterrows():
42          # 沒有持有部位，建倉判斷
43          if hold_data["num"] == 0:
44              condition1 = row["RSI_5"] < 20 # 若RSI小於20，則做多(Long)
45              condition2 = row["RSI_5"] > 80 # 若RSI大於80，則做空(Short)
46
47              # 符合做多條件
48              if condition1:
49                  trade_price = row["成交價格"] # 取出當下的成交價格
50                  num = 1 # 交易數量
51                  Recording(row["日期"], trade_price, "L", num) # 紀錄做多交易
52
53                  # 更動持有狀況，紀錄成交價格與交易數量
54                  hold_data["price"], hold_data["num"] = trade_price, num
55
56              # 符合做空條件
57              elif condition2:
58                  trade_price = row["成交價格"] # 取出當下的成交價格
59                  num = 1 # 成交數量
60                  Recording(row["日期"], trade_price, "S", num) # 紀錄做空交易
61
62                  # 更動持有狀況，紀錄成交價格與交易數量，做空數量以負數代表
63                  hold_data["price"], hold_data["num"] = trade_price, -num
```

交易中 – 紀錄函數(Recording)


```
101  def Recording(time, price, LS, num):
102      if record_df.empty: # 若當下的交易紀錄是空的，則索引值設為 0
103          index_count = 0
104      else: # 若交易紀錄不是空的，則索引值設為最大值 + 1
105          index_count = record_df.index[-1] + 1
106
107      # 將資料寫入交易紀錄
108      record_df.at[index_count, "time"] = time
109      record_df.at[index_count, "price"] = price
110      record_df.at[index_count, "LS"] = LS
111      record_df.at[index_count, "num"] = num
```

交易中 – 紀錄函數(Recording)

傳入
1. 交易時間(time)
2. 成交價格(price)
3. 多空方向(LS)
4. 成交數量(num)

設定索引值

```
101 def Recording(time, price, LS, num):
102     if record_df.empty: # 若當下的交易紀錄是空的，則索引值設為 0
103         index_count = 0
104     else: # 若交易紀錄不是空的，則索引值設為最大值 + 1
105         index_count = record_df.index[-1] + 1
106
107     # 將資料寫入交易紀錄
108     record_df.at[index_count, "time"] = time
109     record_df.at[index_count, "price"] = price
110     record_df.at[index_count, "LS"] = LS
111     record_df.at[index_count, "num"] = num
```

將資料記入 record_df

交易中 – 交易函數(trade)


```
29  def trade(data_df):
        ...
41      for index, row in data_df.iterrows():
            ...
66          else:
67              # 做多平倉條件
68              condition1 = hold_data["num"] > 0 # 持有數量大於0
69              condition2 = row["RSI_5"] > 80 # RSI大於80
70
71              # 做空平倉條件
72              condition3 = hold_data["num"] < 0 # 持有數量小於0
73              condition4 = row["RSI_5"] < 20 # RSI小於20
74                          for index, row in data_df.iterrows():
75              # 做多平倉條件
76              if condition1 and condition2:
77                  num = hold_data["num"] # 當下持有數量
78                  trade_price = row["成交價格"] # 取出當下的成交價格
79                  Recording(row["日期"], trade_price, "S", num) # 紀錄平倉交易
80                  hold_data["price"], hold_data["num"] = "", 0 # 清空持有紀錄
81
82              # 做空平倉條件
83              elif condition3 and condition4:
84                  num = -hold_data["num"] # 當下持有數量，將負數轉正
85                  trade_price = row["成交價格"] # 取出當下的成交價格
86                  Recording(row["日期"], trade_price, "L", num) # 紀錄平倉交易
87                  hold_data["price"], hold_data["num"] = "", 0 # 清空持有紀錄
```

交易中 – 交易函數(trade)

```
29  def trade(data_df):
        ...
41      for index, row in data_df.iterrows():
            ...
66          else:
67              # 做多平倉條件
68              condition1 = hold_data["num"] > 0 # 持有數量大於0
69              condition2 = row["RSI_5"] > 80 # RSI大於80
70
71              # 做空平倉條件
72              condition3 = hold_data["num"] < 0 # 持有數量小於0
73              condition4 = row["RSI_5"] < 20 # RSI小於20
74
75              # 做多平倉條件
76              if condition1 and condition2:
77                  num = hold_data["num"] # 當下持有數量
78                  trade_price = row["成交價格"] # 取出當下的成交價格
79                  Recording(row["日期"], trade_price, "S", num) # 紀錄平倉交易
80                  hold_data["price"], hold_data["num"] = "", 0 # 清空持有紀錄
81
82              # 做空平倉條件
83              elif condition3 and condition4:
84                  num = -hold_data["num"] # 當下持有數量，將負數轉正
85                  trade_price = row["成交價格"] # 取出當下的成交價格
86                  Recording(row["日期"], trade_price, "L", num) # 紀錄平倉交易
87                  hold_data["price"], hold_data["num"] = "", 0 # 清空持有紀錄
```

平倉條件分別為
做多平倉:
1. 持有數量 > 0
2. RSI > 80

做空平倉
1. 持有數量 < 0
2. RSI < 80

若符合條件，紀錄交易、清空持有紀錄

交易中 – 交易函數(trade)


```
29  def trade(data_df):
        ...
89      # 最後一日平倉
90      last_date = data_df.iloc[-1]["日期"] # 最後一日日期
91      last_trade_price = data_df.iloc[-1]["成交價格"] # 最後一日成交價格
92
93
94      if hold_data["num"] > 0: # 若持有數量不為大於0，將做多部位平倉
95          Recording(last_date, last_trade_price, "S", hold_data["num"])
96      elif hold_data["num"] < 0: # 若持有數量不為小於0，將做空部位平倉
97          Recording(last_date, last_trade_price, "L", -hold_data["num"])
98
99      return record_df # 回傳交易紀錄
```

交易中 – 交易函數(trade)

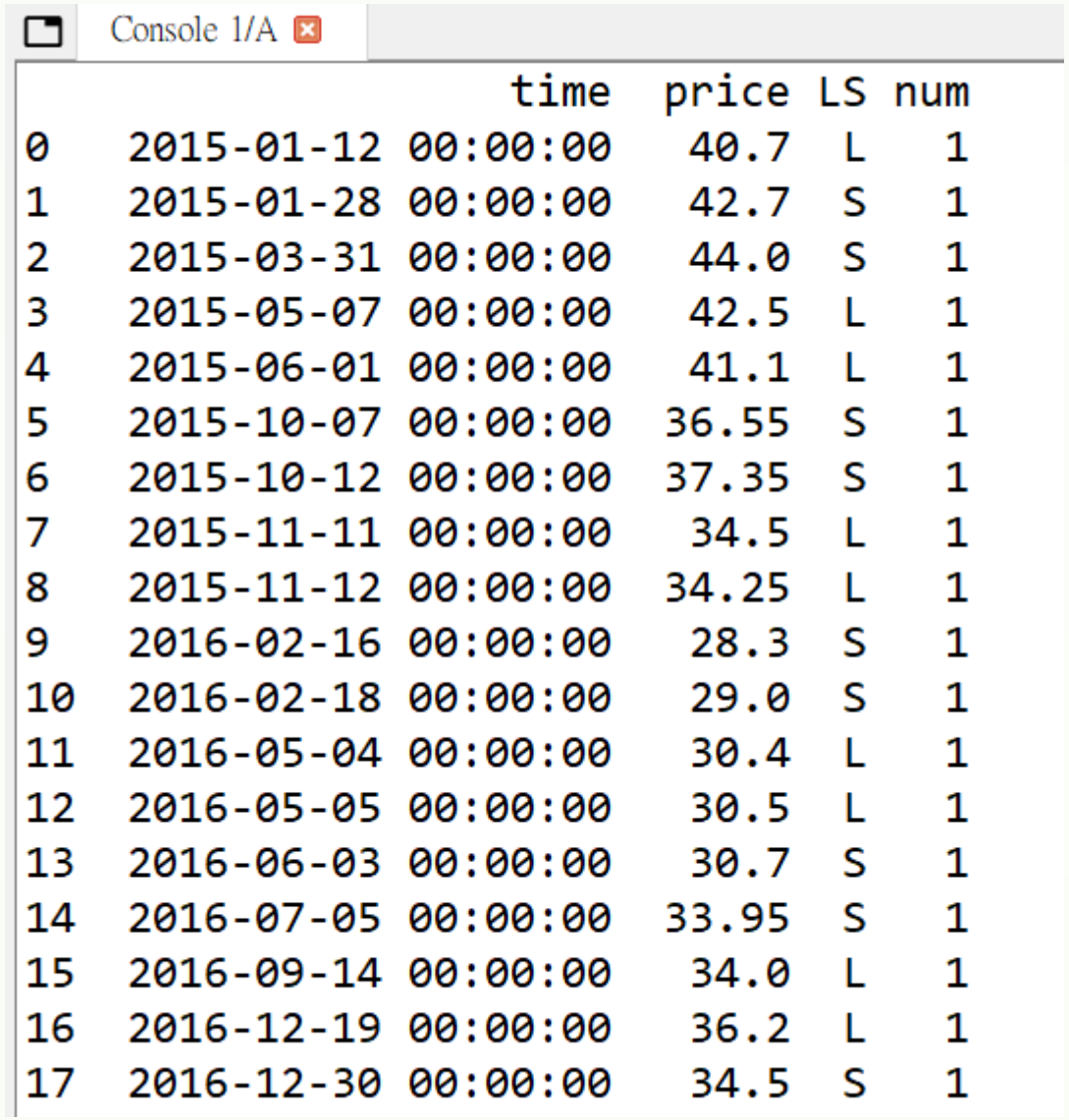

```
29  def trade(data_df):
        ...
89      # 最後一日平倉
90      last_date = data_df.iloc[-1]["日期"] # 最後一日日期
91      last_trade_price = data_df.iloc[-1]["成交價格"] # 最後一日成交價格
92
93
94      if hold_data["num"] > 0: # 若持有數量不為大於0，將做多部位平倉
95          Recording(last_date, last_trade_price, "S", hold_data["num"])
96      elif hold_data["num"] < 0: # 若持有數量不為小於0，將做空部位平倉
97          Recording(last_date, last_trade_price, "L", -hold_data["num"])
98
99      return record_df # 回傳交易紀錄
```

最後一日的日期與成交價格

若持有數量大於0或小於0，則進行平倉

回傳交易紀錄

執行結果



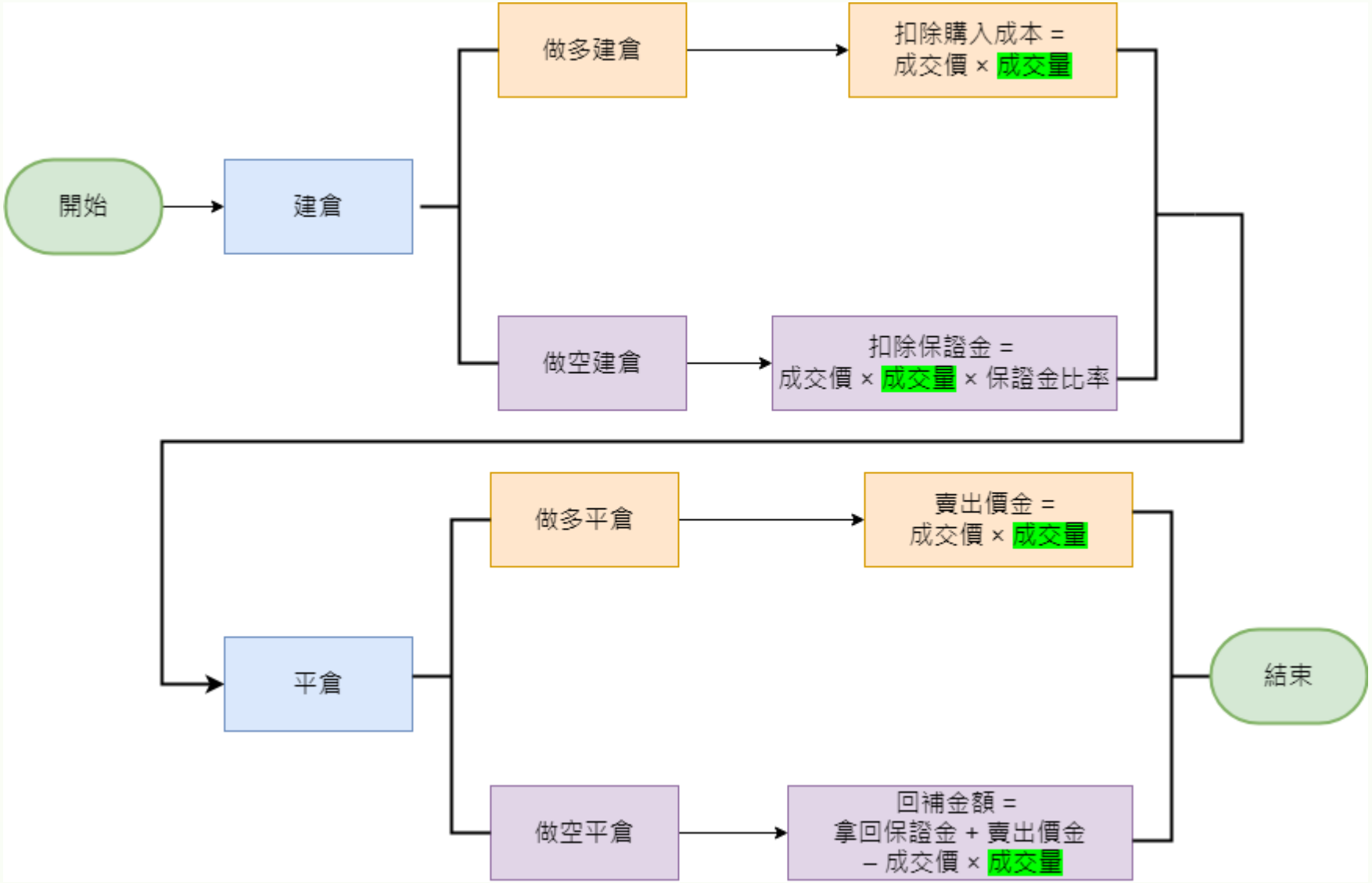
	time	price	LS	num
0	2015-01-12 00:00:00	40.7	L	1
1	2015-01-28 00:00:00	42.7	S	1
2	2015-03-31 00:00:00	44.0	S	1
3	2015-05-07 00:00:00	42.5	L	1
4	2015-06-01 00:00:00	41.1	L	1
5	2015-10-07 00:00:00	36.55	S	1
6	2015-10-12 00:00:00	37.35	S	1
7	2015-11-11 00:00:00	34.5	L	1
8	2015-11-12 00:00:00	34.25	L	1
9	2016-02-16 00:00:00	28.3	S	1
10	2016-02-18 00:00:00	29.0	S	1
11	2016-05-04 00:00:00	30.4	L	1
12	2016-05-05 00:00:00	30.5	L	1
13	2016-06-03 00:00:00	30.7	S	1
14	2016-07-05 00:00:00	33.95	S	1
15	2016-09-14 00:00:00	34.0	L	1
16	2016-12-19 00:00:00	36.2	L	1
17	2016-12-30 00:00:00	34.5	S	1

訓練集交易結果

	time	price	LS	num
0	2017-01-12 00:00:00	36.3	S	1
1	2017-12-29 00:00:00	36.2	L	1

測試集交易結果

交易中 – 加入資金管控



交易中 – 加入資金管控

- 原始資金 1,000,000
- 做多建倉，成交價 65 元，成交量 = 1,000,000 // 65 = 15,384 股
- 現金 = 1,000,000 - 65 × 15,384 = 40

- 做多平倉，成交價 70 元，成交量(持有量) 15,384 股
- 現金 = 40 + 70 × 15,384 = 1,076,920

交易中 – 加入資金管控

- 現金 1,076,920
- 做空建倉，成交價 75 元，成交量 = 1,076,920 // 75 = 14,358 股
- 保證金 = 75 × 14,358 × 100% = 1,076,850
- 現金 = 1,076,920 - 保證金 = 70

- 做空平倉，成交價 72 元，成交量(持有量) 14,358 股
- 現金 = 70 + 保證金 + (75 – 72) × 14,358 = 1,119,994

交易中 – 主程式(加入資金管控)


```
141  # 以隔天的開盤價當作成交價格
142  train_data["成交價格"] = train_data["開盤價"].shift(-1)
143  # 最後一天以收盤價當作交易價格
144  train_data["成交價格"].fillna(method = "ffill", inplace = True)
145  # 計算 RSI5
146  train_data["RSI_5"] = talib.RSI(train_data["收盤價"], 5)
147
148  # 以隔天的開盤價當作成交價格
149  test_data["成交價格"] = test_data["開盤價"].shift(-1)
150  # 最後一天以收盤價當作交易價格
151  test_data["成交價格"].fillna(method = "ffill", inplace = True)
152  # 計算 RSI5
153  test_data["RSI_5"] = talib.RSI(test_data["收盤價"], 5)
154
155  # 定義初始資金
156  cash = 1000000
157
158  trade(train_data, cash)
159  print(record_df)
160
161  trade(test_data, cash)
162  print(record_df)
```

交易中 – 主程式(加入資金管控)


```
141  # 以隔天的開盤價當作成交價格
142  train_data["成交價格"] = train_data["開盤價"].shift(-1)
143  # 最後一天以收盤價當作交易價格
144  train_data["成交價格"].fillna(method = "ffill", inplace = True)
145  # 計算 RSI5
146  train_data["RSI_5"] = talib.RSI(train_data["收盤價"], 5)
147
148  # 以隔天的開盤價當作成交價格
149  test_data["成交價格"] = test_data["開盤價"].shift(-1)
150  # 最後一天以收盤價當作交易價格
151  test_data["成交價格"].fillna(method = "ffill", inplace = True)
152  # 計算 RSI5
153  test_data["RSI_5"] = talib.RSI(test_data["收盤價"], 5)
154
155  # 定義初始資金
156  cash = 1000000
157
158  trade(train_data, cash)
159  print(record_df)
160
161  trade(test_data, cash)
162  print(record_df)
```

持有資金

帶入資金到交易函數中

交易中 –交易函數(加入資金管控)


```
29  def trade(data_df, cash):
30      global record_df, hold_data # 設定全域變數，讓其他函數也可以使用者兩個變數
31
32      # 定義交易紀錄
33      record_df = pd.DataFrame( columns = ["time", "price", "LS", "num"] )
34
35      # 定義持有資訊
36      hold_data = {
37          "price" : 0,
38          "num" : 0 # 做多部位為正數，做空部位為負
39          }
```

交易中 – 交易函數(加入資金管控)

傳入原始金額

```
29  def trade(data_df, cash):
30      global record_df, hold_data # 設定全域變數，讓其他函數也可以使用者兩個變數
31
32      # 定義交易紀錄
33      record_df = pd.DataFrame( columns = ["time", "price", "LS", "num"] )
34
35      # 定義持有資訊
36      hold_data = {
37          "price" : 0,
38          "num" : 0 # 做多部位為正數，做空部位為負
39          }
```

交易中 – 交易函數(加入資金管控)


```
29  def trade(data_df):
        ...
41      for index, row in data_df.iterrows():
42          # 沒有持有部位，建倉判斷
43          if hold_data["num"] == 0:
44              condition1 = row["RSI_5"] < 20 # 若RSI小於20，則做多(Long)
45              condition2 = row["RSI_5"] > 80 # 若RSI大於80，則做空(Short)
46
47              # 若RSI小於20，則做多(Long)
48              if condition1:
49                  trade_price = row["成交價格"] # 取出當下的成交價格
50                  num = cash // trade_price # 整數除法，根據現有資金決定成交股數
51                  Recording(row["日期"], trade_price, "L", num) # 紀錄做多交易
52
53                  # 更動持有狀況，紀錄成交價格與交易數量
54                  hold_data["price"], hold_data["num"] = trade_price, num
55
56                  # 資金更動，扣除買股票的資金
57                  cash-= trade_price * num
58
59              # 若RSI大於80，則做空(Short)
60              elif condition2:
61                  trade_price = row["成交價格"] # 取出當下的成交價格
62                  num = cash // trade_price # 整數除法，根據現有資金決定成交股數
63                  Recording(row["日期"], trade_price, "S", num) # 紀錄做空交易
64
65                  # 更動持有狀況，紀錄成交價格與交易數量，做空數量以負數代表
66                  hold_data["price"], hold_data["num"] = trade_price, -num
67
68                  # 資金更動，假設保證金率為1，且沒有融券手續費、利息
69                  cash-= trade_price * num
```

根據持有資金決定成交量

做多買入時扣除資金

根據持有資金決定成交量

做空賣出時扣除保證金，假設保證金率為100%，沒有融券手續費與利息

交易中 – 交易函數(加入資金管控)

```
29  def trade(data_df):
    ...
41      for index, row in data_df.iterrows():
    ...
71          # 持有部位，平倉判斷
72          else:
73              # 做多平倉條件
74              condition1 = hold_data["num"] > 0 # 持有數量大於0
75              condition2 = row["RSI_5"] > 80 # RSI大於80
76
77              # 做空平倉條件
78              condition3 = hold_data["num"] < 0 # 持有數量小於0
79              condition4 = row["RSI_5"] < 20 # RSI小於20
80
81              # 做多平倉條件
82              if condition1 and condition2:
83                  num = hold_data["num"] # 當下持有數量
84                  trade_price = row["成交價格"] # 取出當下的成交價格
85                  Recording(row["日期"], trade_price, "S", num) # 紀錄平倉交易
86
87                  cash+= trade_price * num # 平倉取得金額
88
89                  hold_data["price"], hold_data["num"] = "", 0 # 清空持有紀錄
90
91              # 做空平倉條件
92              elif condition3 and condition4:
93                  num = abs( hold_data["num"] )# 當下持有數量，將負數轉正
94                  hold_price = hold_data["price"] # 建倉成交價
95                  trade_price = row["成交價格"] # 取出當下的成交價格
96                  Recording(row["日期"], trade_price, "L", num) # 紀錄平倉交易
97
98                  # 拿回保證金、賣出價金，減除回補成本
99                  cash+= hold_price * num + ( hold_price - trade_price ) * num
100
101                 hold_data["price"], hold_data["num"] = "", 0 # 清空持有紀錄
```

交易中 – 交易函數(加入資金管控)

```
29  def trade(data_df):
    ...
41      for index, row in data_df.iterrows():
    ...
71          # 持有部位，平倉判斷
72          else:
73              # 做多平倉條件
74              condition1 = hold_data["num"] > 0 # 持有數量大於0
75              condition2 = row["RSI_5"] > 80 # RSI大於80
76
77              # 做空平倉條件
78              condition3 = hold_data["num"] < 0 # 持有數量小於0
79              condition4 = row["RSI_5"] < 20 # RSI小於20
80
81              # 做多平倉條件
82              if condition1 and condition2:
83                  num = hold_data["num"] # 當下持有數量
84                  trade_price = row["成交價格"] # 取出當下的成交價格
85                  Recording(row["日期"], trade_price, "S", num) # 紀錄平倉交易
86
87                  cash+= trade_price * num # 平倉取得金額
88
89                  hold_data["price"], hold_data["num"] = "", 0 # 清空持有紀錄
90
91              # 做空平倉條件
92              elif condition3 and condition4:
93                  num = abs( hold_data["num"] )# 當下持有數量，將負數轉正
94                  hold_price = hold_data["price"] # 建倉成交價
95                  trade_price = row["成交價格"] # 取出當下的成交價格
96                  Recording(row["日期"], trade_price, "L", num) # 紀錄平倉交易
97
98                  # 拿回保證金、賣出價金，減除回補成本
99                  cash+= hold_price * num + ( hold_price - trade_price ) * num
100
101                 hold_data["price"], hold_data["num"] = "", 0 # 清空持有紀錄
```

做多平倉時回補資金

做空平倉時，拿回保證金、
當初賣出時的價金，
並扣除回補時的價金

交易中 – 交易函數(加入資金管控)


```
29  def trade(data_df):
        ...
103     # 最後一日平倉
104     last_date = data_df.iloc[-1]["日期"] # 最後一日日期
105     last_trade_price = data_df.iloc[-1]["成交價格"] # 最後一日成交價格
106
107
108     if hold_data["num"] > 0: # 若持有數量不為大於0，將做多部位平倉
109         Recording(last_date, last_trade_price, "S", hold_data["num"])
110         cash+= last_trade_price * hold_data["num"] # 平倉取得金額
111     elif hold_data["num"] < 0: # 若持有數量不為小於0，將做空部位平倉
112         num = abs( hold_data["num"] ) # 建倉成交量
113         hold_price = hold_data["price"] # 建倉成交價
114         Recording(last_date, last_trade_price, "L", num)
115         # 拿回保證金、賣出價金，減除回補成本
116         cash+= hold_price * num + ( hold_price - last_trade_price ) * num
117     print(cash)
118     return record_df # 回傳交易紀錄
```

交易中 – 交易函數(加入資金管控)

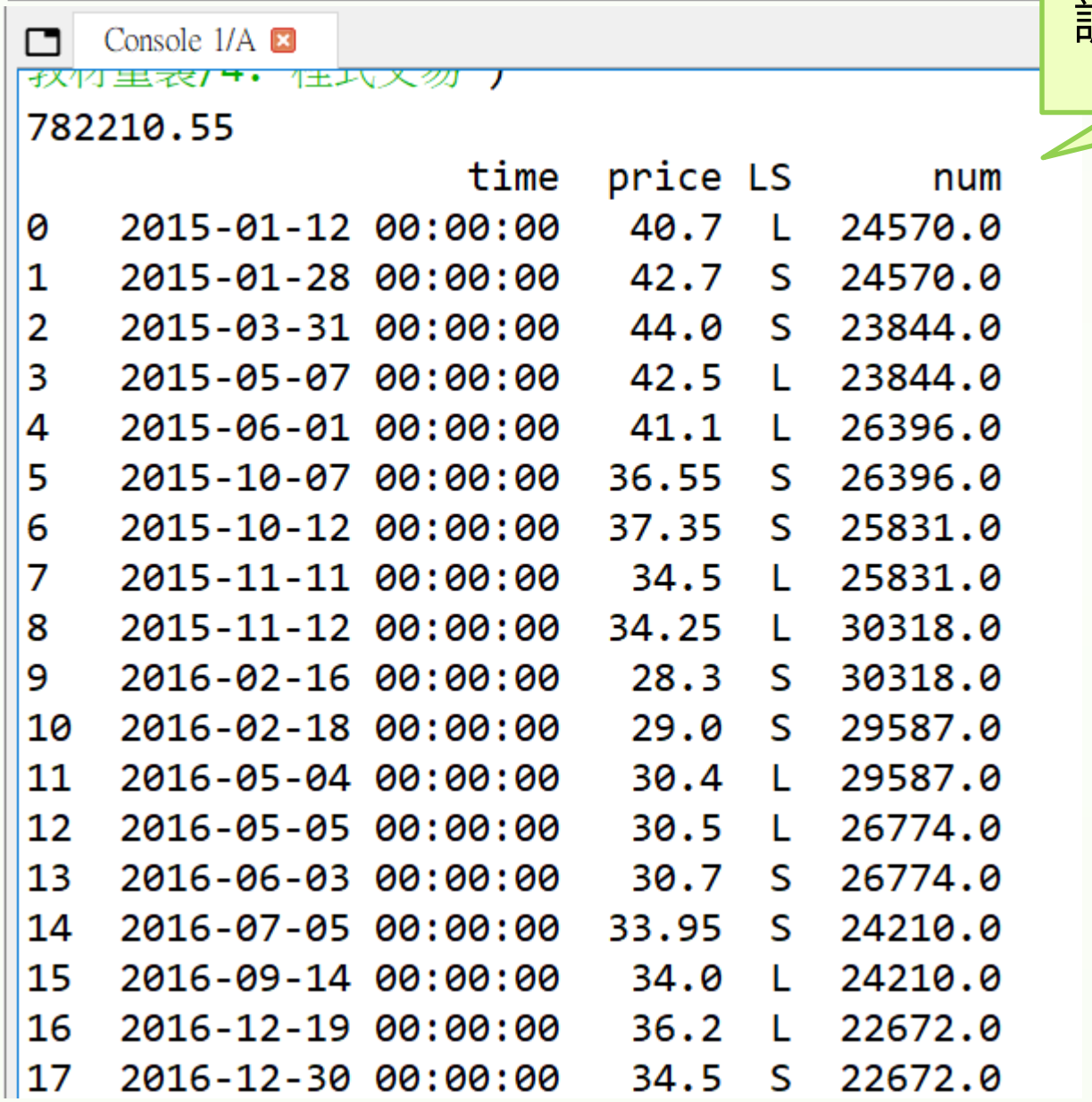
```
29  def trade(data_df):
        ...

103     # 最後一日平倉
104     last_date = data_df.iloc[-1]["日期"] # 最後一日日期
105     last_trade_price = data_df.iloc[-1]["成交價格"] # 最後一日成交價格
106
107
108     if hold_data["num"] > 0: # 若持有數量不為大於0，將做多部位平倉
109         Recording(last_date, last_trade_price, "S", hold_data["num"])
110         cash+= last_trade_price * hold_data["num"] # 平倉取得金額
111     elif hold_data["num"] < 0: # 若持有數量不為小於0，將做空部位平倉
112         num = abs( hold_data["num"] ) # 建倉成交量
113         hold_price = hold_data["price"] # 建倉成交價
114         Recording(last_date, last_trade_price, "L", num)
115         # 拿回保證金、賣出價金，減除回補成本
116         cash+= hold_price * num + ( hold_price - last_trade_price ) * num
117     print(cash)
118     return record_df # 回傳交易紀錄
```

最後一日持有部位處理

做多平倉時回補資金

做空平倉時，拿回保證金、
當初賣出時的價金，
並扣除回補時的價金

執行結果



訓練集交易後剩餘資金與交易結果

```
1002754.7999999998
                    time price LS      num
0  2017-01-12 00:00:00  36.3  S  27548.0
1  2017-12-29 00:00:00  36.2  L  27548.0
```

測試集交易後剩餘資金與交易結果

交易中 – 交易函數(加入交易成本)

- 手續費率 = 1.425 ‰ = 0.001425 (依據券商不同而有差異)
- 買賣都需要繳交手續費，不足 20 元，以 20 元計價

- 證交稅率 = 3 ‰ = 0.003
- 僅賣出時需要繳納

交易中 – 加入交易成本

- 原始資金 1,000,000
- 做多建倉，成交價 65 元，

  成交量 = 1,000,000 // [65 × (1 + 手續費率)]= 15,362 股
- 手續費 = 65 × 15,362 × 手續費率 = 1,423
- 現金 = 1,000,000 – 65 × 15,362 – 1,423 = 47

- 做多平倉，成交價 70 元，成交量(持有量) 15,384 股
- 手續費 = 70 × 15,362 × 手續費率 = 1,532
- 證交稅 = 70 × 15,362 × 證交稅率 = 3,226
- 現金 = 47 + 70 × 15,362 – 1,532 – 3,226 = 1,070,652

交易中 – 加入交易成本

- 現金 1,070,652
- 做空建倉，成交價 75 元，

  成交量 = 1,070,652 // [75 × (1 + 手續費率 + 證交稅率)] = 14,212 股
- 保證金 = 75 × 14,212 × 100% = 1,065,900
- 手續費 = 75 × 14,212 × 手續費率 = 1,519
- 證交稅 = 75 × 14,212 × 證交稅率 = 3,198
- 現金 = 1,070,652 – 1,065,900 – 1,519 – 3,198 = 35

- 做空平倉，成交價 72 元，成交量(持有量) 14,212 股
- 現金 = 35 + 保證金 + (75 – 72) × 14,358 = 1,109,009

交易中 – 主程式(加入交易成本)


```
177  # 以隔天的開盤價當作成交價格
178  train_data["成交價格"] = train_data["開盤價"].shift(-1)
179  # 最後一天以收盤價當作交易價格
180  train_data["成交價格"].fillna(method = "ffill", inplace = True)
181  # 計算 RSI5
182  train_data["RSI_5"] = talib.RSI(train_data["收盤價"], 5)
183
184  # 以隔天的開盤價當作成交價格
185  test_data["成交價格"] = test_data["開盤價"].shift(-1)
186  # 最後一天以收盤價當作交易價格
187  test_data["成交價格"].fillna(method = "ffill", inplace = True)
188  # 計算 RSI5
189  test_data["RSI_5"] = talib.RSI(test_data["收盤價"], 5)
190
191  # 定義初始資金
192  cash = 1000000
193
194  fee_rate = 0.001425 # 手續費率，千分之1.425
195  min_fee = 20 # 不足20元以20元計價
196  tax_rate = 0.003 # 證交稅
197
198  trade(train_data, cash)
199  print(record_df)
200
201  trade(test_data, cash)
202  print(record_df)
```

交易中 – 主程式(加入交易成本)

```
177  # 以隔天的開盤價當作成交價格
178  train_data["成交價格"] = train_data["開盤價"].shift(-1)
179  # 最後一天以收盤價當作交易價格
180  train_data["成交價格"].fillna(method = "ffill", inplace = True)
181  # 計算 RSI5
182  train_data["RSI_5"] = talib.RSI(train_data["收盤價"], 5)
183
184  # 以隔天的開盤價當作成交價格
185  test_data["成交價格"] = test_data["開盤價"].shift(-1)
186  # 最後一天以收盤價當作交易價格
187  test_data["成交價格"].fillna(method = "ffill", inplace = True)
188  # 計算 RSI5
189  test_data["RSI_5"] = talib.RSI(test_data["收盤價"], 5)
190
191  # 定義初始資金
192  cash = 1000000
193
194  fee_rate = 0.001425 # 手續費率，千分之1.425
195  min_fee = 20 # 不足20元以20元計價
196  tax_rate = 0.003 # 證交稅
197
198  trade(train_data, cash)
199  print(record_df)
200
201  trade(test_data, cash)
202  print(record_df)
```

設定證手續費率、最低手續費、證交稅率

交易中 – 交易函數(加入交易成本)


```
29  def trade(data_df):
        ...
41      for index, row in data_df.iterrows():
42          # 沒有持有部位，建倉判斷
43          if hold_data["num"] == 0:
44              condition1 = row["RSI_5"] < 20 # 若RSI小於20，則做多(Long)
45              condition2 = row["RSI_5"] > 80 # 若RSI大於80，則做空(Short)
46
47              # 若RSI小於20，則做多(Long)
48              if condition1:
49                  trade_price = row["成交價格"] # 取出當下的成交價格
50                  num = cash // (trade_price * ( 1 + fee_rate ) ) # 整數除法，根據現有資金決定成交股數
51                  Recording(row["日期"], trade_price, "L", num) # 紀錄做多交易
52
53                  # 更動持有狀況，紀錄成交價格與交易數量
54                  hold_data["price"], hold_data["num"] = trade_price, num
55
56                  # 交易手續費
57                  fee = fee_cal( trade_price, num )
58
59                  # 資金更動，扣除買股票的資金
60                  cash-= trade_price * num + fee
```

交易中 – 交易函數(加入交易成本)

```
29  def trade(data_df):
        ...
41      for index, row in data_df.iterrows():
42          # 沒有持有部位，建倉判斷
43          if hold_data["num"] == 0:
44              condition1 = row["RSI_5"] < 20 # 若RSI小於20，則做多(Long)
45              condition2 = row["RSI_5"] > 80 # 若RSI大於80，則做空(Short)
46
47              # 若RSI小於20，則做多(Long)
48              if condition1:
49                  trade_price = row["成交價格"] # 取出當下的成交價格
50                  num = cash // (trade_price * ( 1 + fee_rate ) ) # 整數除法，根據現有資金決定成交股數
51                  Recording(row["日期"], trade_price, "L", num) # 紀錄做多交易
52
53                  # 更動持有狀況，紀錄成交價格與交易數量
54                  hold_data["price"], hold_data["num"] = trade_price, num
55
56                  # 交易手續費
57                  fee = fee_cal( trade_price, num )
58
59                  # 資金更動，扣除買股票的資金
60                  cash-= trade_price * num + fee
```

做多計算成交量時，
將手續費視為成交價的增加

交易手續費計算函數
(下一頁詳細說明)

做多買入時扣除資金與手續費

交易中 – 交易函數(手續費計算)

```
146  # 計算手續費
147  def fee_cal(trade_price, num):
148      # 交易手續費
149      fee = trade_price * num * fee_rate
150
151      # 手續費不足20元則以20元計價
152      if fee < min_fee:
153          fee = min_fee
154
155      return fee # 回傳手續費
```

傳入成交價、成交量
計算手續費，並判別若受續費
小於min_fee，即 20 元，
則以20元計算

交易中 – 交易函數(手續費計算)

```
29  def trade(data_df):
        ...
41      for index, row in data_df.iterrows():
42          # 沒有持有部位，建倉判斷
43          if hold_data["num"] == 0:
                ...
47              # 若RSI小於20，則做多(Long)
48              if condition1:
                    ...
62              # 若RSI大於80，則做空(Short)
63              elif condition2:
64                  trade_price = row["成交價格"] # 取出當下的成交價格
65                  num = cash // (trade_price * ( 1 + fee_rate + tax_rate) )  # 整數除法，根據現有資金決定成交股數
66                  Recording(row["日期"], trade_price, "S", num) # 紀錄做空交易
67
68                  # 更動持有狀況，紀錄成交價格與交易數量，做空數量以負數代表
69                  hold_data["price"], hold_data["num"] = trade_price, -num
70
71                  # 交易手續費
72                  fee = fee_cal( trade_price, num )
73
74                  # 證交稅
75                  tax =  trade_price * num * tax_rate
76
77                  # 資金更動，假設保證金率為1，且沒有融券手續費、利息
78                  cash-= trade_price * num + fee + tax
```

交易中 – 交易函數(手續費計算)

```
29  def trade(data_df):
        ...
41      for index, row in data_df.iterrows():
42          # 沒有持有部位，建倉判斷
43          if hold_data["num"] == 0:
                ...
47              # 若RSI小於20，則做多(Long)
48              if condition1:
                    ...
62              # 若RSI大於80，則做空(Short)
63              elif condition2:
64                  trade_price = row["成交價格"] # 取出當下的成交價格
65                  num = cash // (trade_price * ( 1 + fee_rate + tax_rate) ) # 整數除法，根據現有資金決定成交股數
66                  Recording(row["日期"], trade_price, "S", num) # 紀錄做空交易
67
68                  # 更動持有狀況，紀錄成交價格與交易數量，做空數量以負數代表
69                  hold_data["price"], hold_data["num"] = trade_price, -num
70
71                  # 交易手續費
72                  fee = fee_cal( trade_price, num )
73
74                  # 證交稅
75                  tax =  trade_price * num * tax_rate
76
77                  # 資金更動，假設保證金率為1，且沒有融券手續費、利息
78                  cash-= trade_price * num + fee + tax
```

做空計算成交量時，
將手續費與證交稅視為成交價的增加

計算手續費、證交稅

做空賣出時扣除資金、手續費與證交稅

交易中 – 交易函數(手續費計算)


```
29  def trade(data_df):
        ...
41      for index, row in data_df.iterrows():
42          # 沒有持有部位，建倉判斷
43          if hold_data["num"] == 0:
                ...
80          # 持有部位，平倉判斷
81          else:
                    ...
90              # 做多平倉條件
91              if condition1 and condition2:
92                  num = hold_data["num"] # 當下持有數量
93                  trade_price = row["成交價格"] # 取出當下的成交價格
94                  Recording(row["日期"], trade_price, "S", num) # 紀錄平倉交易
95
96                  # 交易手續費
97                  fee = fee_cal( trade_price, num )
98
99                  # 證交稅
100                 tax = trade_price * num * tax_rate
101
102                 cash+= trade_price * num - fee - tax # 平倉取得金額
103
104                 hold_data["price"], hold_data["num"] = "", 0 # 清空持有紀錄
```

交易中 – 交易函數(手續費計算)

```
29  def trade(data_df):
        ...
41      for index, row in data_df.iterrows():
42          # 沒有持有部位，建倉判斷
43          if hold_data["num"] == 0:
                ...
80          # 持有部位，平倉判斷
81          else:
                    ...
90              # 做多平倉條件
91              if condition1 and condition2:
92                  num = hold_data["num"] # 當下持有數量
93                  trade_price = row["成交價格"] # 取出當下的成交價格
94                  Recording(row["日期"], trade_price, "S", num) # 紀錄平倉交易
95
96                  # 交易手續費
97                  fee = fee_cal( trade_price, num )
98
99                  # 證交稅
100                 tax = trade_price * num * tax_rate
101
102                 cash+= trade_price * num - fee - tax # 平倉取得金額
103
104                 hold_data["price"], hold_data["num"] = "", 0 # 清空持有紀錄
```

做多平倉時，計算手續費、證交稅

做多平倉時回補資金並扣除手續費、證交稅

交易中 – 交易函數(手續費計算)


```
29  def trade(data_df):
        ...
41      for index, row in data_df.iterrows():
42          # 沒有持有部位，建倉判斷
43          if hold_data["num"] == 0:
                ...
80          # 持有部位，平倉判斷
81          else:
                    ...
90              # 做多平倉條件
91              if condition1 and condition2:
                        ...
106             # 做空平倉條件
107             elif condition3 and condition4:
108                 num = -hold_data["num"] # 當下持有數量，將負數轉正
109                 hold_price = hold_data["price"] # 建倉成交價
110                 trade_price = row["成交價格"] # 取出當下的成交價格
111                 Recording(row["日期"], trade_price, "L", num) # 紀錄平倉交易
112
113                 # 交易手續費
114                 fee = fee_cal( trade_price, num )
115
116                 # 拿回保證金，賣出價金，減除回補成本
117                 cash+= hold_price * num + (hold_price - trade_price) * num - fee
118
119                 hold_data["price"], hold_data["num"] = "", 0 # 清空持有紀錄
```

交易中 – 交易函數(手續費計算)

```
29  def trade(data_df):
        ...
41      for index, row in data_df.iterrows():
42          # 沒有持有部位，建倉判斷
43          if hold_data["num"] == 0:
                ...
80          # 持有部位，平倉判斷
81          else:
                    ...
90              # 做多平倉條件
91              if condition1 and condition2:
                        ...
106             # 做空平倉條件
107             elif condition3 and condition4:
108                 num = -hold_data["num"] # 當下持有數量，將負數轉正
109                 hold_price = hold_data["price"] # 建倉成交價
110                 trade_price = row["成交價格"] # 取出當下的成交價格
111                 Recording(row["日期"], trade_price, "L", num) # 紀錄平倉交易
112
113                 # 交易手續費
114                 fee = fee_cal( trade_price, num )
115
116                 # 拿回保證金，賣出價金，減除回補成本
117                 cash+= hold_price * num + (hold_price - trade_price) * num - fee
118
119                 hold_data["price"], hold_data["num"] = "", 0 # 清空持有紀錄
```

做空平倉時，計算手續費

做空平倉時，
拿回保證金並結算損益

交易中 – 交易函數(手續費計算)


```
29  def trade(data_df):
        ...
```

```
121     # 最後一日平倉
122     last_date = data_df.iloc[-1]["日期"] # 最後一日日期
123     last_trade_price = data_df.iloc[-1]["成交價格"] # 最後一日成交價格
124
125     if hold_data["num"] > 0: # 若持有數量不為大於0，將做多部位平倉
126         num = hold_data["num"]
127         Recording(last_date, last_trade_price, "S", hold_data["num"])
128         # 交易手續費
129         fee = fee_cal( last_trade_price, num )
130         # 證交稅
131         tax = last_trade_price * num * tax_rate
132
133         cash+= last_trade_price * hold_data["num"] - fee - tax # 平倉取得金額
134
135     elif hold_data["num"] < 0: # 若持有數量不為小於0，將做空部位平倉
136         num = -hold_data["num"]
137         hold_price = hold_data["price"] # 建倉成交價
138         Recording(last_date, last_trade_price, "L", num)
139         # 交易手續費
140         fee = fee_cal( last_trade_price, num )
141         # 拿回保證金、賣出價金，減除回補成本
142         cash+= hold_price * num + ( hold_price - last_trade_price ) * num - fee
143     print(cash)
144     return record_df # 回傳交易紀錄
```

交易中 – 交易函數(手續費計算)

```
29  def trade(data_df):
        ...
121     # 最後一日平倉
122     last_date = data_df.iloc[-1]["日期"] # 最後一日日期
123     last_trade_price = data_df.iloc[-1]["成交價格"] # 最後一日成交價格
124
125     if hold_data["num"] > 0: # 若持有數量不為大於0，將做多部位平倉
126         num = hold_data["num"]
127         Recording(last_date, last_trade_price, "S", hold_data["num"])
128         # 交易手續費
129         fee = fee_cal( last_trade_price, num )
130         # 證交稅
131         tax = last_trade_price * num * tax_rate
132
133         cash+= last_trade_price * hold_data["num"] - fee - tax # 平倉取得金額
134
135     elif hold_data["num"] < 0: # 若持有數量不為小於0，將做空部位平倉
136         num = -hold_data["num"]
137         hold_price = hold_data["price"] # 建倉成交價
138         Recording(last_date, last_trade_price, "L", num)
139         # 交易手續費
140         fee = fee_cal( last_trade_price, num )
141         # 拿回保證金、賣出價金，減除回補成本
142         cash+= hold_price * num + ( hold_price - last_trade_price ) * num - fee
143     print(cash)
144     return record_df # 回傳交易紀錄
```

最後一日持倉處理

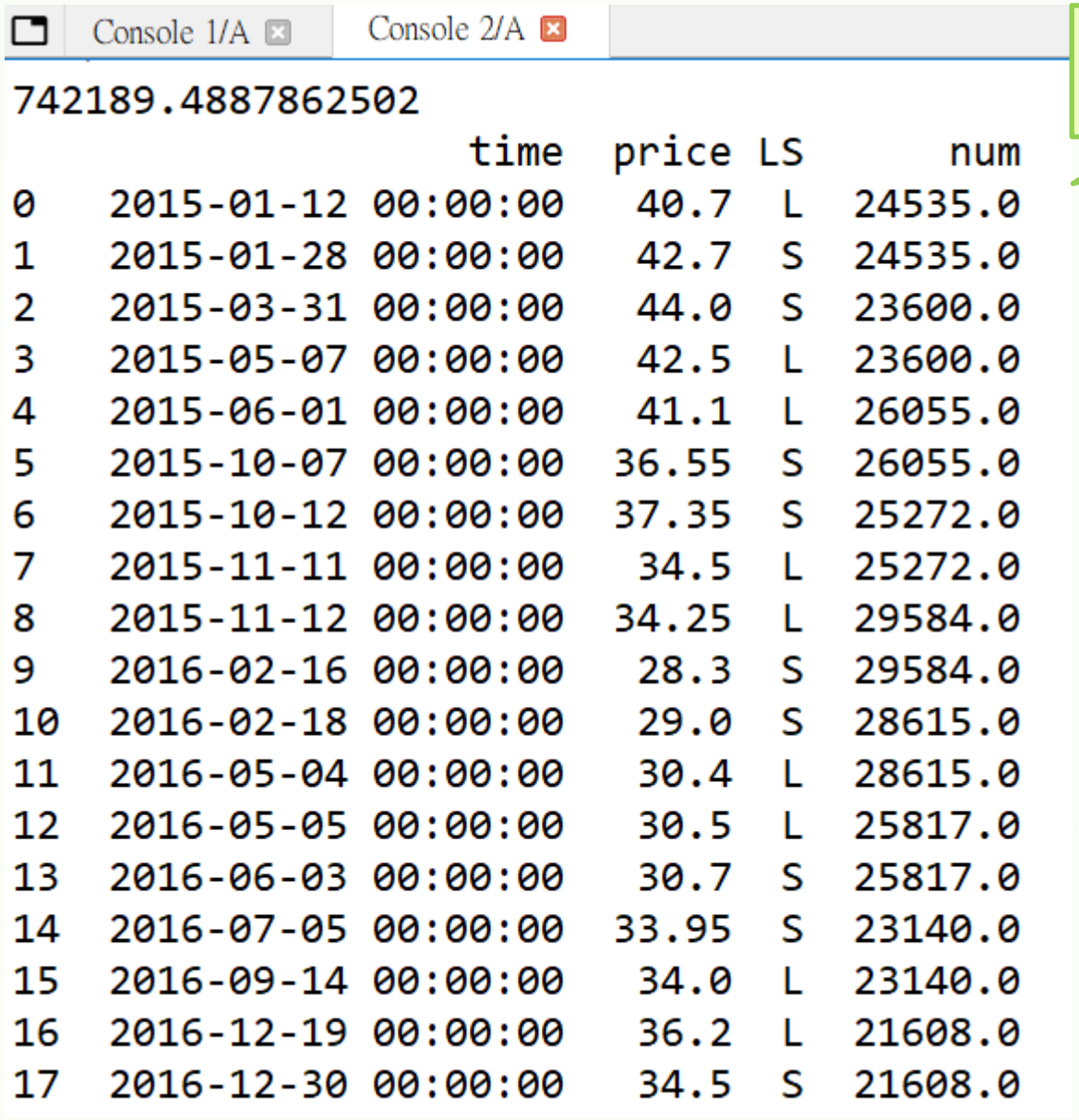
做多平倉時，計算手續費、證交稅

做多平倉時回補資金並扣除手續費、證交稅

做空平倉時，計算手續費

做空平倉時，拿回保證金並結算損益

執行結果



訓練集交易後剩餘資金與交易結果

```
996922.4599749999
                    time price LS      num
0  2017-01-12 00:00:00  36.3  S  27426.0
1  2017-12-29 00:00:00  36.2  L  27426.0
```

測試集交易後剩餘資金與交易結果

執行結果比較

AI.FINTECH
AI 金融科技中心

無交易成本

Console 1/A

782210.55

	time	price	LS	num
0	2015-01-12 00:00:00	40.7	L	24570.0
1	2015-01-28 00:00:00	42.7	S	24570.0
2	2015-03-31 00:00:00	44.0	S	23844.0
3	2015-05-07 00:00:00	42.5	L	23844.0
4	2015-06-01 00:00:00	41.1	L	26396.0
5	2015-10-07 00:00:00	36.55	S	26396.0
6	2015-10-12 00:00:00	37.35	S	25831.0
7	2015-11-11 00:00:00	34.5	L	25831.0
8	2015-11-12 00:00:00	34.25	L	30318.0
9	2016-02-16 00:00:00	28.3	S	30318.0
10	2016-02-18 00:00:00	29.0	S	29587.0
11	2016-05-04 00:00:00	30.4	L	29587.0
12	2016-05-05 00:00:00	30.5	L	26774.0
13	2016-06-03 00:00:00	30.7	S	26774.0
14	2016-07-05 00:00:00	33.95	S	24210.0
15	2016-09-14 00:00:00	34.0	L	24210.0
16	2016-12-19 00:00:00	36.2	L	22672.0
17	2016-12-30 00:00:00	34.5	S	22672.0

有交易成本

Console 1/A Console 2/A

742189.4887862502

	time	price	LS	num
0	2015-01-12 00:00:00	40.7	L	24535.0
1	2015-01-28 00:00:00	42.7	S	24535.0
2	2015-03-31 00:00:00	44.0	S	23600.0
3	2015-05-07 00:00:00	42.5	L	23600.0
4	2015-06-01 00:00:00	41.1	L	26055.0
5	2015-10-07 00:00:00	36.55	S	26055.0
6	2015-10-12 00:00:00	37.35	S	25272.0
7	2015-11-11 00:00:00	34.5	L	25272.0
8	2015-11-12 00:00:00	34.25	L	29584.0
9	2016-02-16 00:00:00	28.3	S	29584.0
10	2016-02-18 00:00:00	29.0	S	28615.0
11	2016-05-04 00:00:00	30.4	L	28615.0
12	2016-05-05 00:00:00	30.5	L	25817.0
13	2016-06-03 00:00:00	30.7	S	25817.0
14	2016-07-05 00:00:00	33.95	S	23140.0
15	2016-09-14 00:00:00	34.0	L	23140.0
16	2016-12-19 00:00:00	36.2	L	21608.0
17	2016-12-30 00:00:00	34.5	S	21608.0

章節到此結束，有任何問題歡迎提出來討論！