

○ 「MQL4と外部アプリの連携；(その1)」

amenbo the 3rd

2017. 11. 18

- ・アメンボです、
 実に1年と11ヶ月ぶりの投稿です。
 MQL4への情熱を失った訳ではなく、色々と他に追いかけている
 ことがあり、謀殺されていたためです。

<約2年前の投稿と、方針転換について>

2年前の投稿；

○ 「WebRequest()の使い方(その3)；実際のサイトにログインする」 2015. 12. 26
 <WebRequest()関数を解析していく手順の全体像(予定)>
 ※これから解析・調査していく手順を「現状では」下記のように予定しています。

- ステップ1； 解析環境とツール類の整備・・・(その1) 投稿済み
- ステップ2； WebRequest()関数の解析； 9引数・・・(その2) 投稿済み
- ステップ3； 実際に外部サイトをアクセスする・・・**本稿**
- ステップ4； WebRequest()関数の解析； 7引数・・・「バージョン2」の解析
- ステップ5； ターゲットから得られるタグ・データの中から必要な情報を入手する
- ステップ6； 得られた情報をEAの判断条件に加える方法について考察する

ここで止まってた！

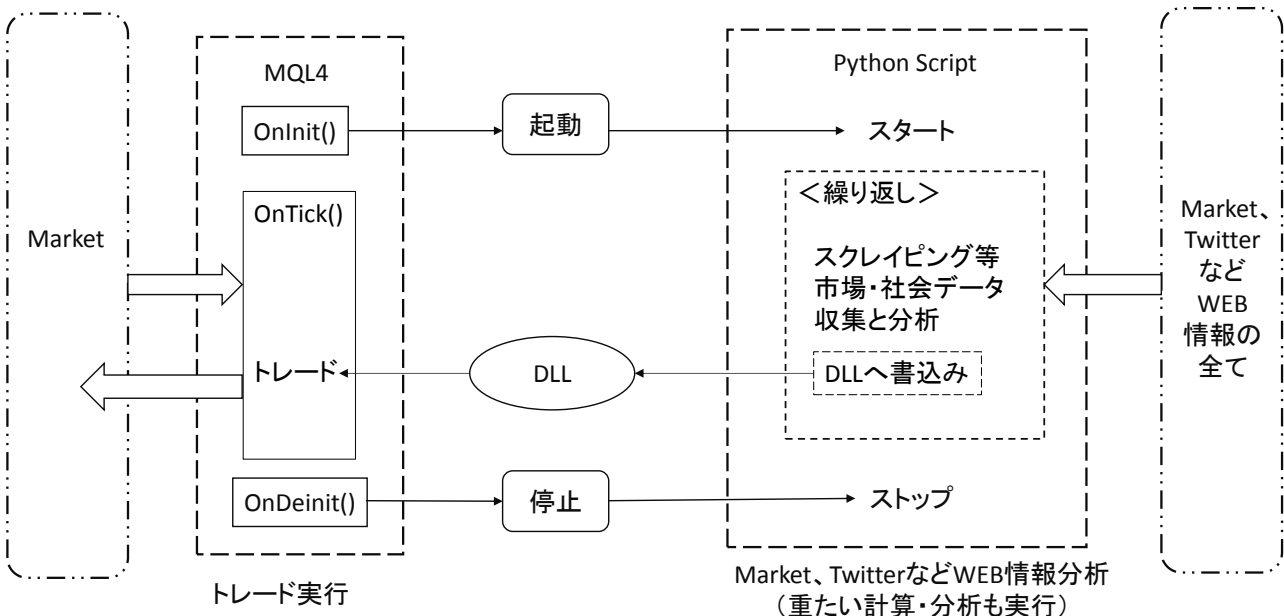
方針転換；

※この2年弱の間にAIが急激に発展し、「金融データ」処理の分野にも影響が及んで
 きました。アメンボのシリーズ再開に際し、上記「ステップ4」以降の処理について、
 最早「MQL言語」のみに頼るべきではないと判断して、大きく方針を変更致しました。

<今後の展開>・・・本シリーズで目指すもの

大方針；他言語(例えばPython)と連携することで、MQL言語の機能を拡張します、
 特に注目のAIを活用する方策を探ることにしました。

[最終目標のイメージ図]



目次 :

- 1. 本稿の概要 ・・・ P 0 2
 - (1) 本稿で実現する「システム図」
 - (2) 本稿の記載範囲
 - (3) 添付プログラムの組合せと基本構成
- 2. 添付プログラムによる動作例 ・・・ P 0 5
 - (1) セット1の使い方
 - (2) セット2の使い方
- 3. 添付プログラムのコード詳細 ・・・ P 1 9
 - (1) セット1
 - (2) セット2
- 4. その他 ・・・ P 2 7
 - (1) 「shared_memory.dll」について

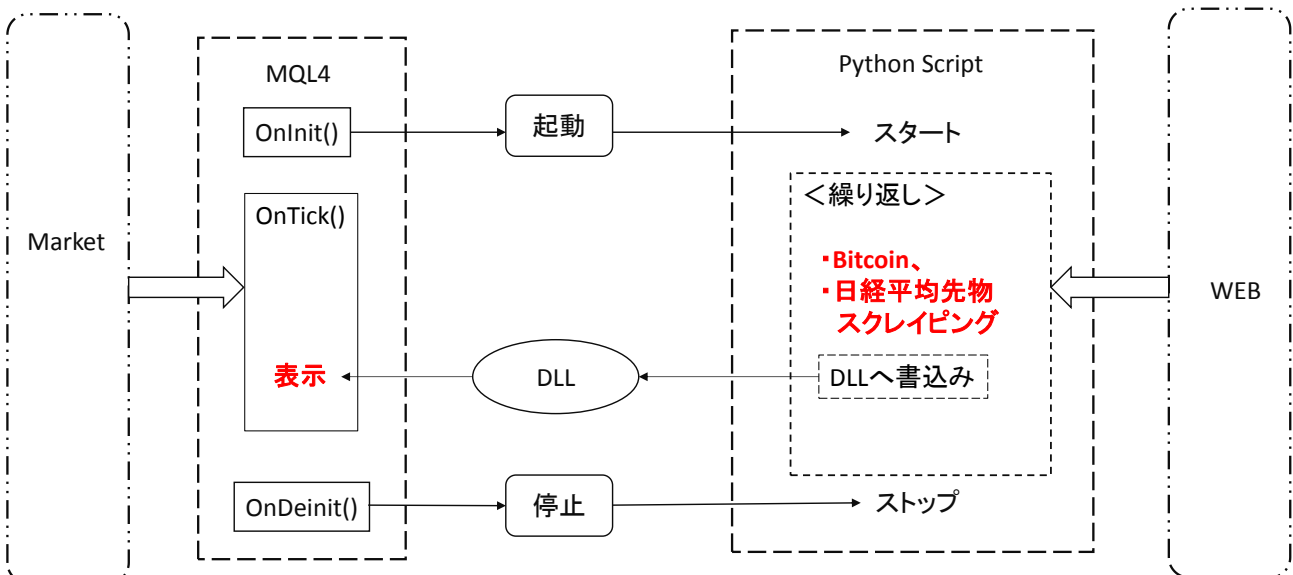
添付コード類と DLL :

「AI&MQL4_01.zip」として添付しています、解凍して使用してください。

1. 本稿の概要

(1) 本稿で実現する「システム図」

再開「第1回目」の投稿は、MQL4 アプリが外部 (Python) アプリから DLL 経由で受け取ったデータを MT4 チャート上に表示する例です。



概要 ; Python でスクレイピングした「ビットコイン」又は「日経平均先物」価格データを、DLL (Dynamic Link Library) を介して MQL4 側に渡します。
(ファイル経由で渡す方法もありますが、本稿では扱いません)
MT4 (MQL4) 側は、受け取ったデータを表示します。(本稿ではココまで、)

(2) 本稿の記載範囲

本稿では、プログラムの詳細な解説は省略して、動作方法と結果のみを解説します。詳細な検討結果やプログラム・コード解説、実施する場合の注意点（制限事項）は、次投稿で行う予定です。

次回以降の投稿内容について；

- 本資料〔最終目標のイメージ図〕を選択した経緯、および本稿で掲載（添付）したコード（MQL4、Python アプリ）の詳細に解説します。
（実は連携する外部アプリとしては Python 以外に R 言語も検討しました）

(3) 添付プログラムの組合せと基本構成

セット	内容	MQL4	Python
1	「ビットコイン」版	Disp_option_1.mq4 (EA)	Scraping_try_1.py
2	「日経平均先物」版	Disp_option_2.mq4 (EA)；起動・終了	Scraping_try_2.py

※セット1；MQL4 アプリには Python アプリの「起動・終了」機能を付けていません

セット2；MQL4 アプリに Python アプリの「起動・終了」機能を付けた例です

役割分担（最終目標）；

- ・MQL4 と Python Script で役割分担する

Python Script = Market、Twitter、および WEB から参照可能な情報すべてを対象として、複雑な計算と分析を実行する。
更に、分析結果の情報を DLL を介して MQL4 側に渡す。

MQL4 = 「テクニカル分析」を行うと共に、DLL のデータを組み合わせた戦略でトレードを実行する。

※本稿では、Python アプリは「スクレイピングとその結果」の DLL への書き込み、MQL4 アプリ（EA）は「DLL から読み込んだデータを表示する」ところまで、です。

<セット1>の特徴

MQL4 アプリ；

※DLL からデータを読み出し、チャート上に表示する。

Python アプリ；

※2つのスレッドから構成されています。（詳細は別途、次の投稿で解説予定）

- ①メイン・スレッド；「スクレイピング・スレッド」の起動と終了を制御
- ②スクレイピング・スレッド；スクレイピングを実行し、結果を DLL へ書き込む

<セット2>

MQL4 アプリ :

※下記の3つの機能構成です

- ①Python プログラムの起動
- ②DLL からデータを読み出し、チャート上に表示する。
- ③Python プログラムの終了

Python アプリ ;

※2つのスレッドから構成されています。(詳細は別途、次の投稿で解説予定)

- ①メイン・スレッド ; 「スクレイピング・スレッド」の起動と終了を制御
- ②スクレイピング・スレッド ; スクレイピングを実行し、結果をDLLへ書き込む

動作確認環境 (重要) ;

※本稿作成に使用したソフト類のバージョンは以下です。(いずれも無料で入手できます)

MT4 ; version 4.00 Build 1090、 MetaEditor ; version 5.00 Build 1601 (19 May 2017)

Python ; Python3.6.1、32ビット版

Python 3.6.1 (v3.6.1:69c0db5, Mar 21 2017, 17:54:52)

[MSC v.1900 32 bit (Intel)]

• Python 実行用のコンソール (2種類で確認) •

- ① IPython コンソール (spyder3) ; IPython 5.3.0
- ② コマンド プロンプト ; Microsoft Windows [Version 6.3.9600]

(IPython コンソール (spyder3) は個人的な好みで選択しました)

※MT4 自体が「32ビット版」なので、DLLを有効にするためPythonも「32ビット版」を選択します。

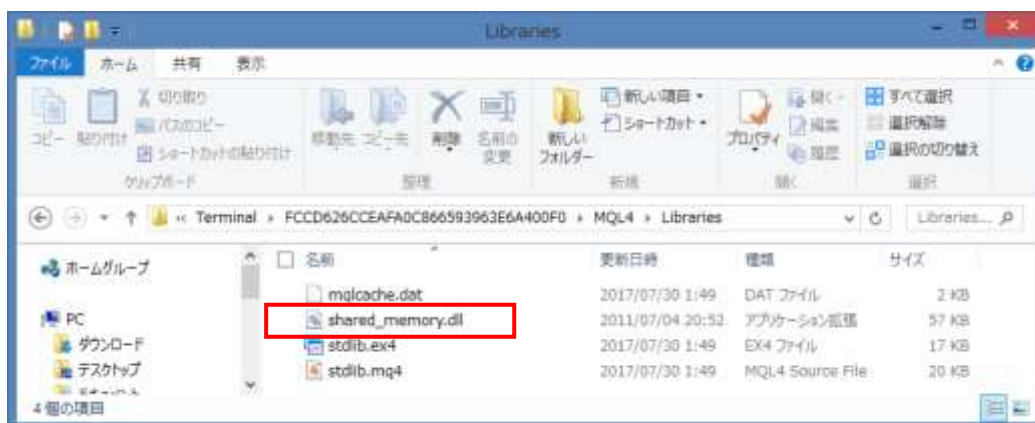
※Pythonの導入は、各自ネット上から行ってください。

(ネットには多くのダウンロードサイトや、参考資料があります)

添付DLL (shared_memory.dll) ;

※このDLLは以前からアメンボが公開している32bit版のシェアード・メモリ(共有メモリ)用DLLです。

MT4で [ファイル] — [データフォルダを開く] — [MQL4] — [Libraries] と開いていき、この中に「shared_memory.dll」を入れます。(コピーを入れればOK)



2. 添付アプリによる動作例

(1) セット1の使い方

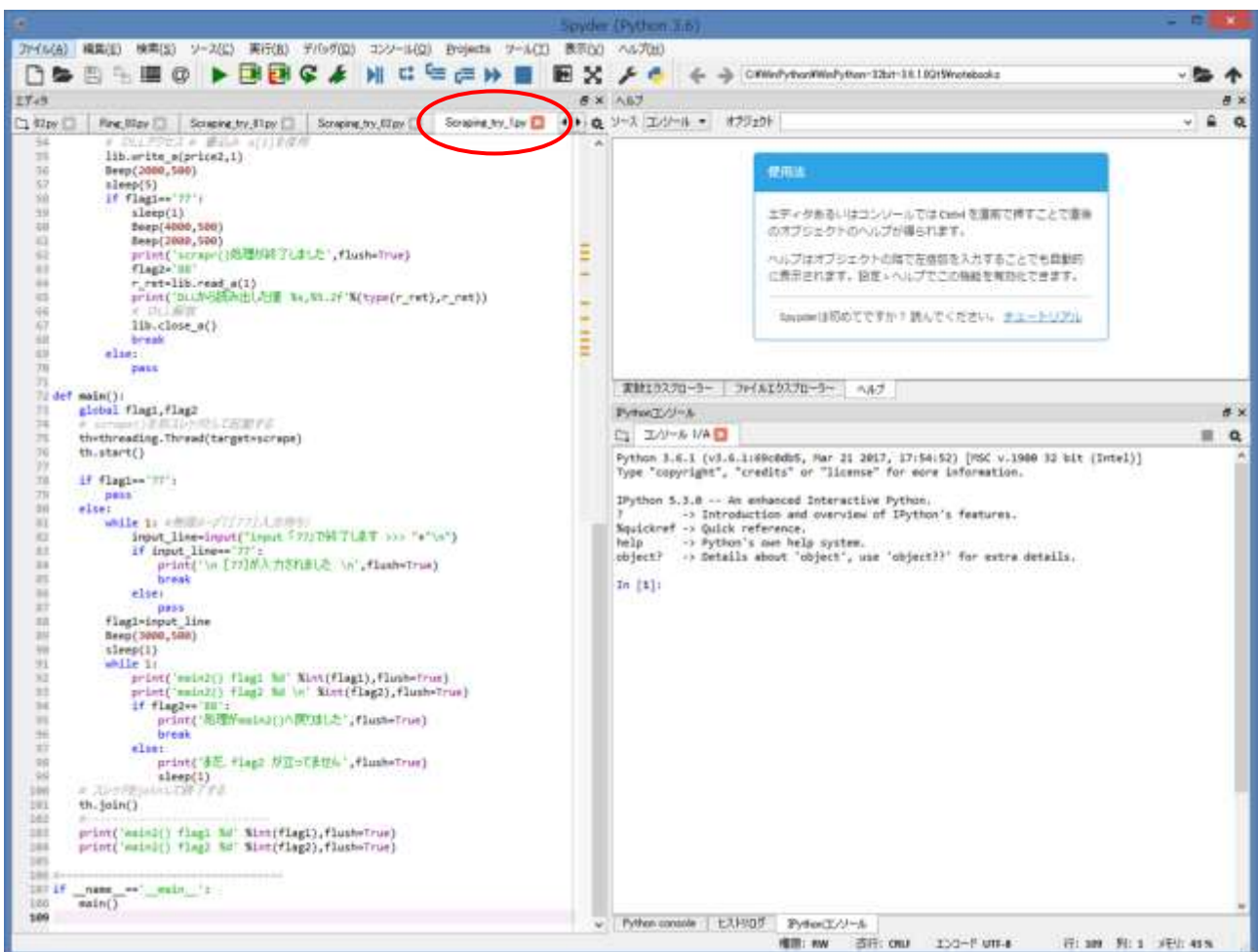
※コンソールとして「IPython コンソール」と「コマンド・プロンプト」を使用する場合で解説します。

(注)「IPython コンソール」、「コマンド・プロンプト」、MT4 (MQL4) の詳細な使い方は省略します。(Pythonの開発環境として本稿では Spyder3 で解説しています、小生の好み)

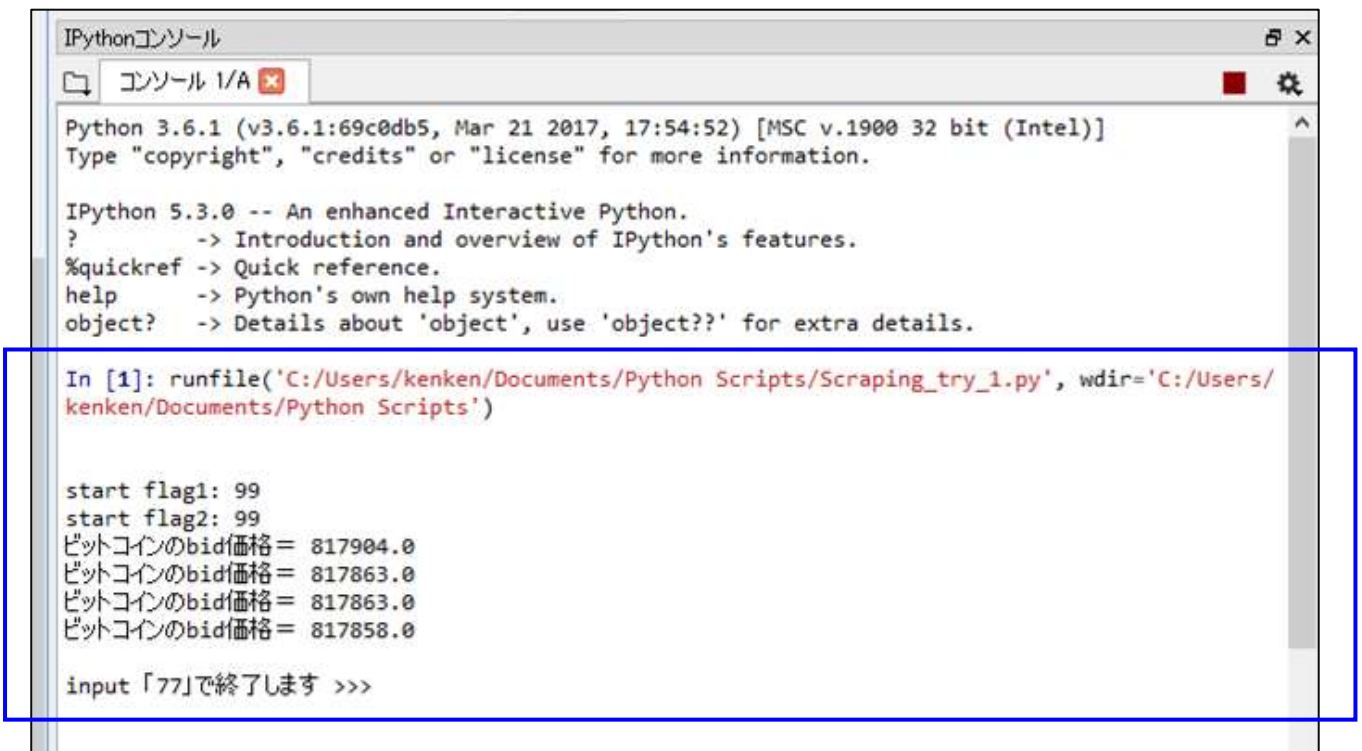
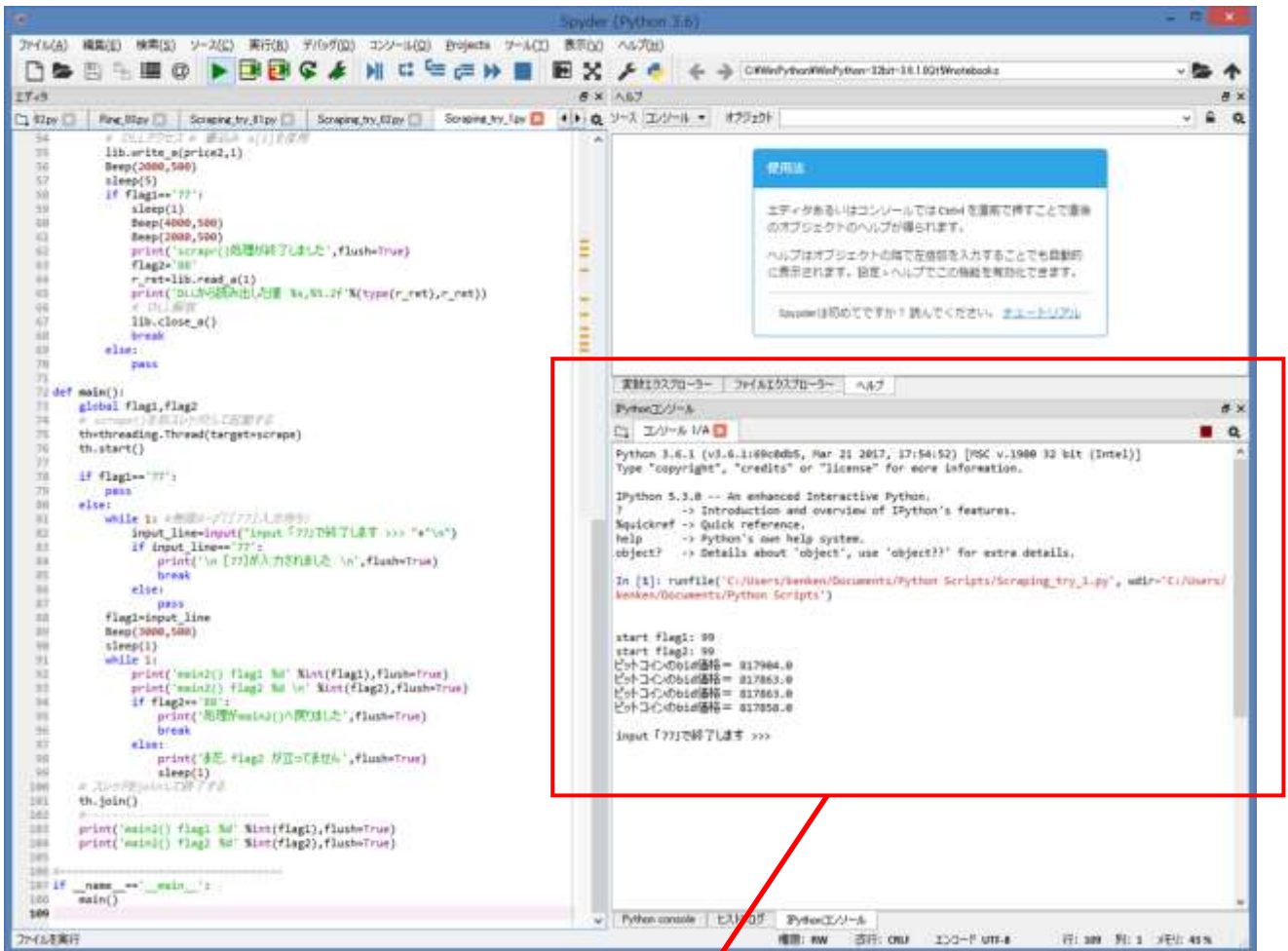
A. 組合せ = [IPython コンソール] + [MT 4]

ステップ1 ;

まず、Pythonプログラムを立ち上げるため、spyder3 (IPython) を起動します。



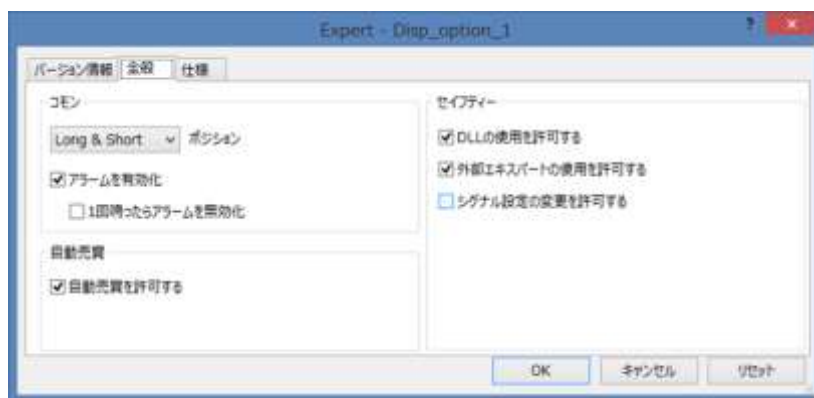
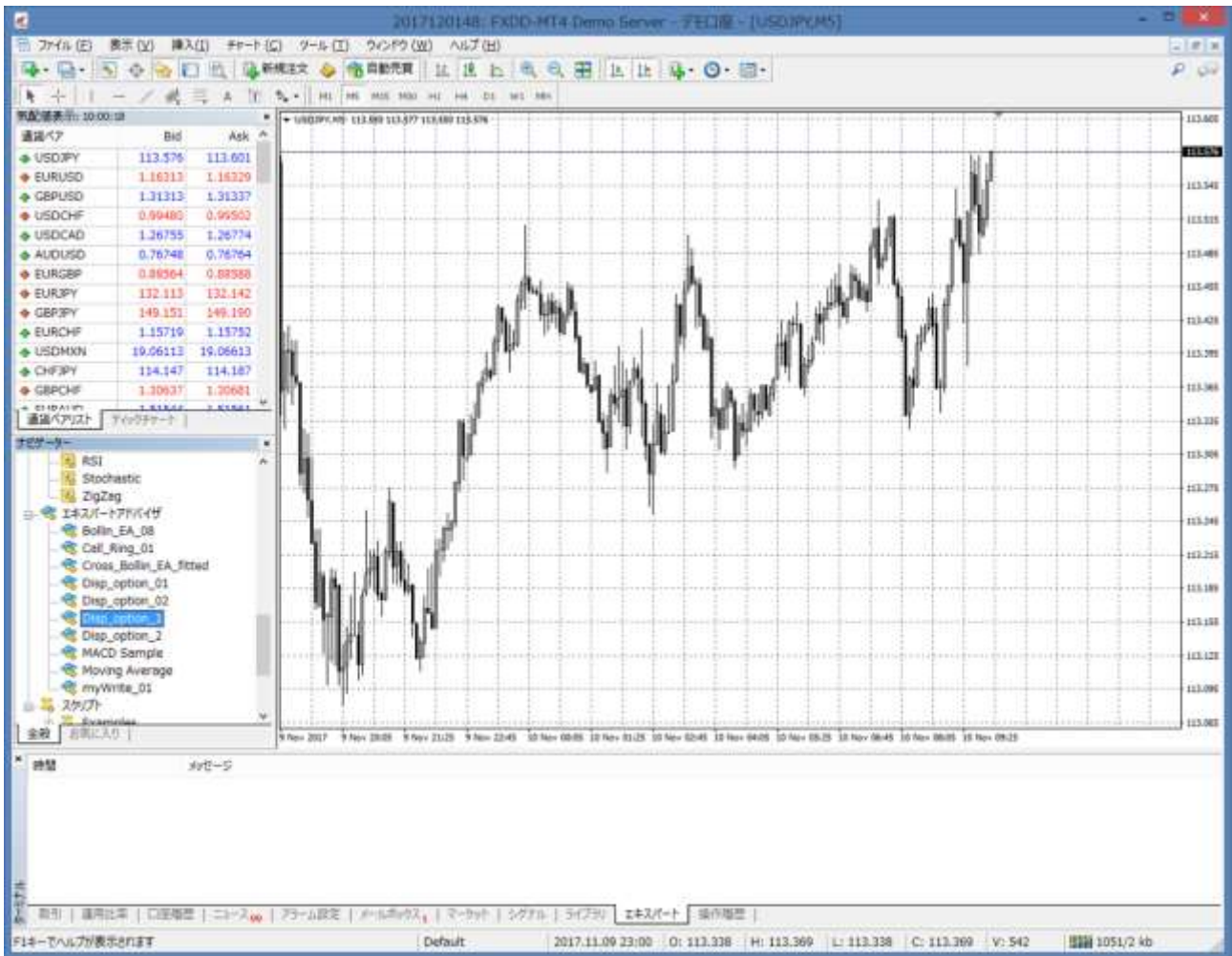
「Scraping_try_1.py」プログラムを選択し、実行します。



プログラムが動作し始めます

ステップ2；

次に、MT 4 を立ち上げ、更に「Disp_option_1.mq4 (EA)」を「チャートに表示」します

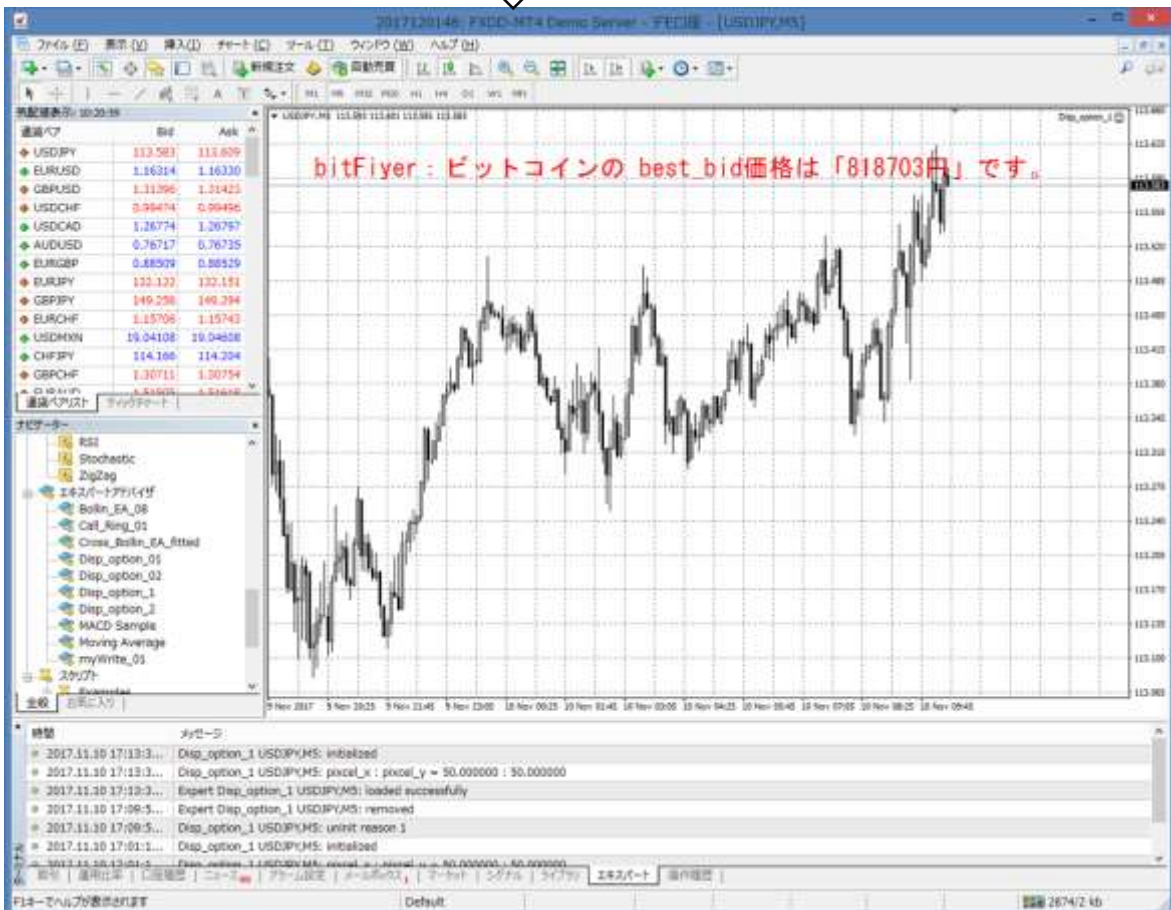


[OK] を選択





「準備中」と「価格」表示を繰り返します



※Python 側の「Scraping_try_1.py」を終了するには；

IPython コンソールで、「77」を入力し、Enter を押します。

(「77」以外の値では終了しません)

```

IPythonコンソール
コンソール 1/A
ビットコインのbid価格 = 819759.0
ビットコインのbid価格 = 818699.0
ビットコインのbid価格 = 818703.0
ビットコインのbid価格 = 819033.0
ビットコインのbid価格 = 819305.0
ビットコインのbid価格 = 819190.0
ビットコインのbid価格 = 819197.0
ビットコインのbid価格 = 819516.0
ビットコインのbid価格 = 820076.0
ビットコインのbid価格 = 819900.0
ビットコインのbid価格 = 819892.0
ビットコインのbid価格 = 819892.0
ビットコインのbid価格 = 819969.0
ビットコインのbid価格 = 820082.0
ビットコインのbid価格 = 820000.0
ビットコインのbid価格 = 819969.0
ビットコインのbid価格 = 819596.0
ビットコインのbid価格 = 819654.0

input 「77」で終了します >>>
77

```

「77」を入力、Return を押す ↓

```

IPythonコンソール
コンソール 1/A
ビットコインのbid価格 = 819227.0
ビットコインのbid価格 = 819392.0
ビットコインのbid価格 = 819857.0
ビットコインのbid価格 = 819857.0

input 「77」で終了します >>>
77

[77]が入力されました

main2() flag1 77
main2() flag2 99

まだ、flag2 が立ってません
main2() flag1 77
main2() flag2 99

まだ、flag2 が立ってません
main2() flag1 77
main2() flag2 99

まだ、flag2 が立ってません
scrapr()処理が終了しました
DLLから読み出した値 <class 'float'>,819857.00
main2() flag1 77
main2() flag2 88

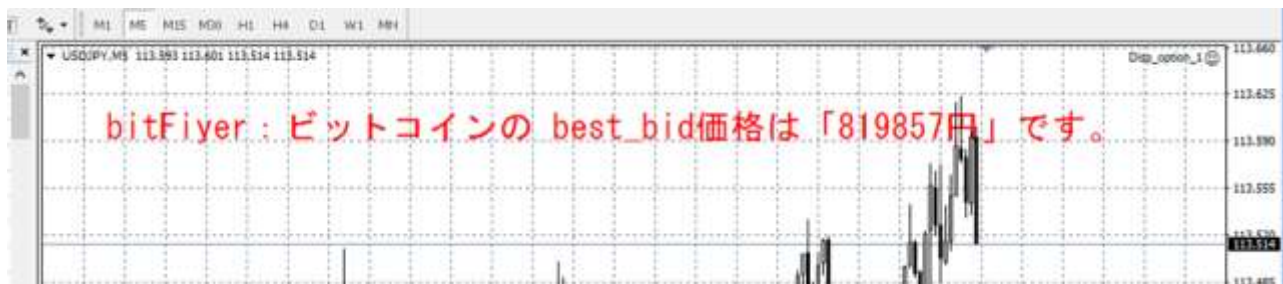
処理がmain2()へ戻りました
main2() flag1 77
main2() flag2 88

In [2]: |

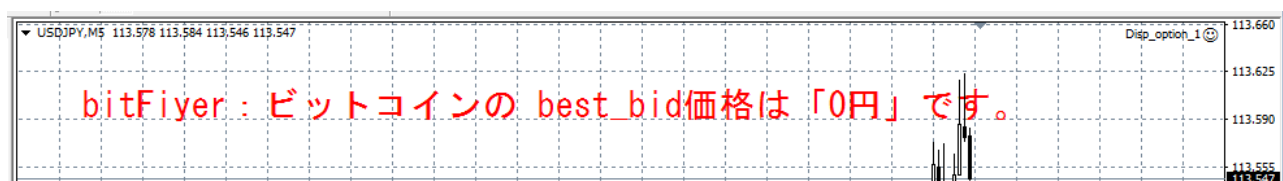
```

終了手順のプログラムが動作し、全ての「スレッド」が終了します。

※Python側の「Scraping_try_1.py」を終了すると、DLL内のデータは更新されないで、MT 4側では常に同じデータを表示します。



※Pythonの起動前にMT 4「Disp_option_1.mq4 (EA)」を立ち上げると、Pythonアプリが動き始めるまでは「ゼロ」表示のままとなります



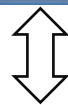
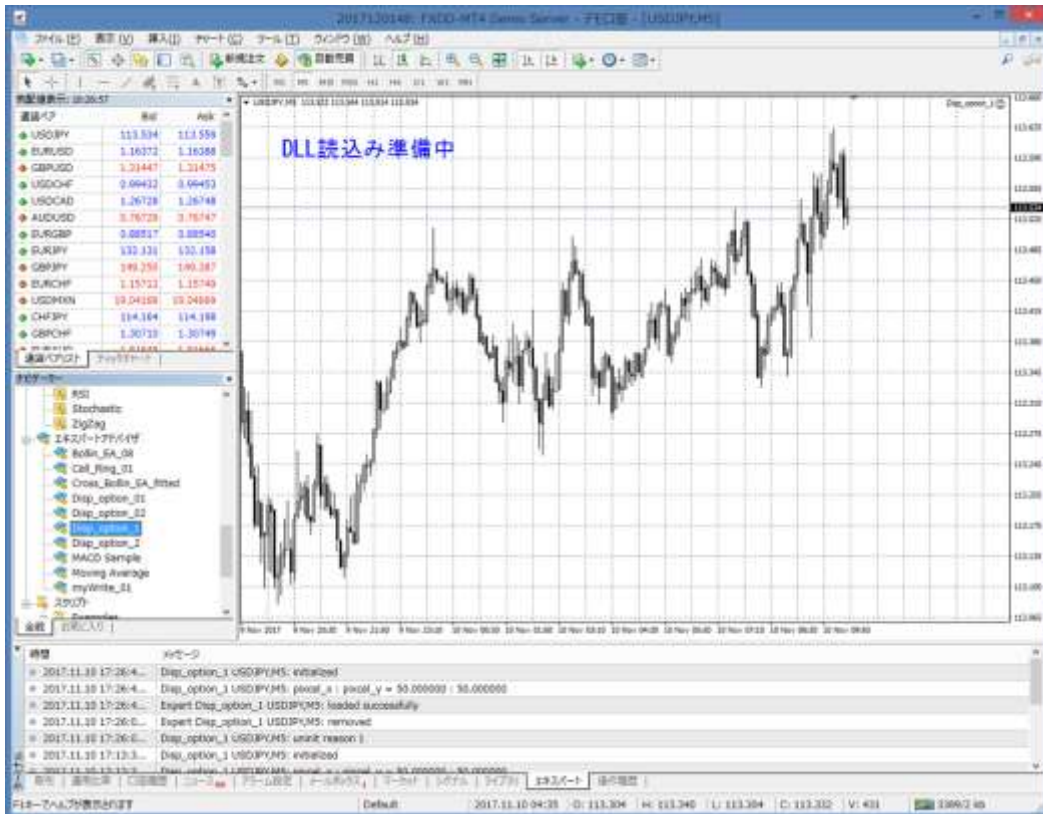
◎特記:

添付アプリでは、「bitFiyer」⇒「bitFlyer」になるように修正済みです！
(やってしまった!!)

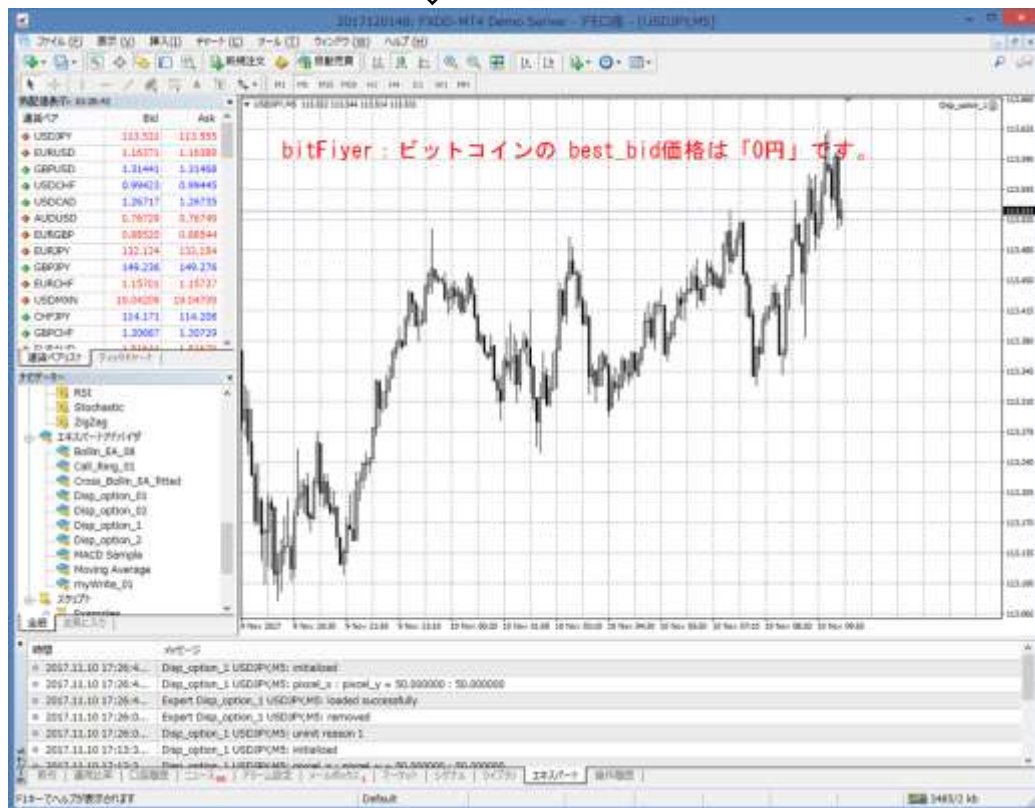
B. 組合せ = [コマンド・プロンプト] + [MT 4]

ステップ1；

※今回は、MT 4で「Disp_option_1.mq4 (EA)」を立ち上げ済みの場合で解説します



「準備中」と「0円」表示を繰り返します



DLLにデータが設定されていないので「0円」表示となります

ステップ 2 :

コマンド・プロンプトを開き、目的の Python プログラムのあるディレクトリーまで
 辿ります。(Python のパスが通るディレクトリ)

次に「python Scraping_try_1.py」と打ち込んで(スクレイピング)プログラムを
 起動します。(以下の例はアメンボの場合であることに、ご注意！)

```

  コマンド プロンプト
  2017/10/20 22:14      4,073 Scraping_try_2.py
  2017/08/04 02:54      3,217 scriping_rennshuu.py
  2017/09/17 23:17        783 sh_memory01.py
  2017/06/25 23:01        269 stocks_price.py
  2017/08/02 01:24      1,506 stocks_price_to_mql4.py
  2017/08/05 00:57        644 sukuraping_01.py
  2017/07/27 22:25      1,055 sum01.py
  2017/07/27 22:27        964 sum02.py
  2017/07/27 22:30        877 sum03.py
  2017/09/02 21:08        706 threading_01.py
  2017/09/25 00:52      2,518 thread_try_01.py
  2017/09/27 22:11      2,762 thread_try_02.py
  2017/09/27 22:58      2,078 thread_try_03.py
  2017/06/14 03:06      1,060 timeline.py
  2017/08/11 18:03        828 timeout_input_00.py
  2017/08/11 18:52      1,953 time_out_input_1.py
  2017/07/22 01:07        575 _hatena01.py
  2017/07/22 01:16        621 _hatena02.py
  2017/08/07 19:23      2,276 _mql4_access.py
  2017/07/22 00:37        984 _sum01.py
  2017/07/22 00:55        778 _sum02.py
  2017/07/30 23:17        889 _sum03.py
  35 個のファイル      52,695 バイト
  2 個のディレクトリ  905,325,449,216 バイトの空き領域

  C:\Users\kenken\Documents\Python Scripts>python Scraping_try_1.py
  
```

プログラムが起動します ↓

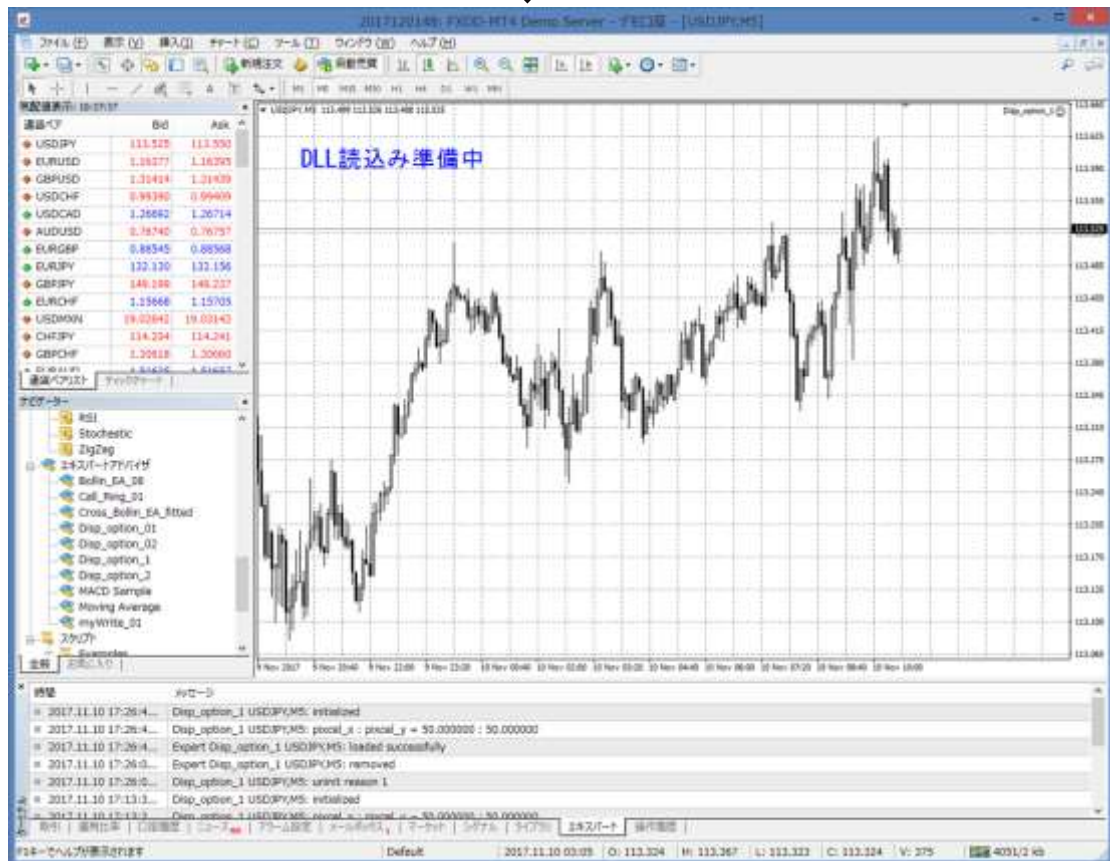
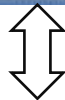
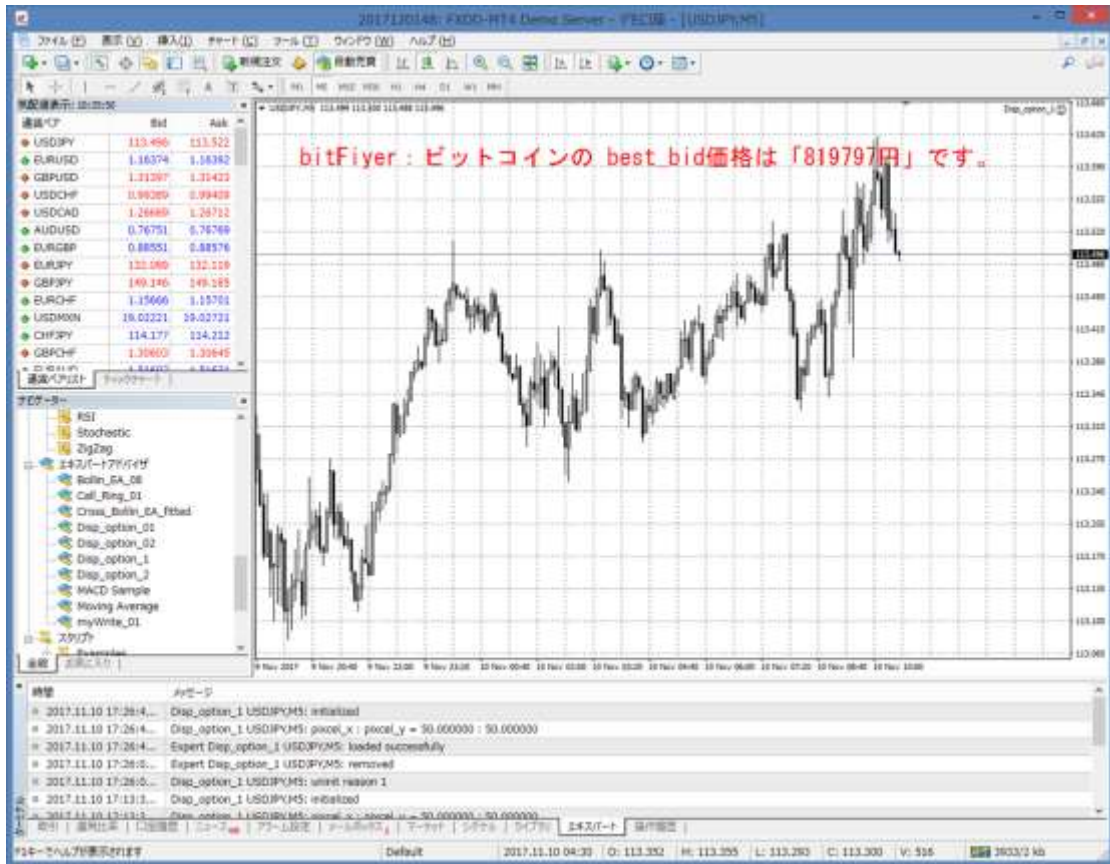
```

  コマンド プロンプト - python Scraping_try_1.py
  2017/08/11 18:52      1,953 time_out_input_1.py
  2017/07/22 01:07        575 _hatena01.py
  2017/07/22 01:16        621 _hatena02.py
  2017/08/07 19:23      2,276 _mql4_access.py
  2017/07/22 00:37        984 _sum01.py
  2017/07/22 00:55        778 _sum02.py
  2017/07/30 23:17        889 _sum03.py
  35 個のファイル      52,695 バイト
  2 個のディレクトリ  905,325,449,216 バイトの空き領域

  C:\Users\kenken\Documents\Python Scripts>python Scraping_try_1.py

  start flag1: 99
  start flag2: 99
  input 「77」 で終了します >>>
  ビットコインのbid価格 = 819971.0
  ビットコインのbid価格 = 819932.0
  ビットコインのbid価格 = 819971.0
  -
  
```

MT 4側に戻ると、「0」円以外の価格」と「準備中」の表示を繰り返しています



※Python 側の「Scraping_try_1.py」を終了するには；

- ①まず、コマンド・プロンプト画面で、[Enter] を押します
(「77」以外なら、何でも(「88」「66」でも) OK)
- ② [input 「77」で数量します >>>] が表示されたら、
「77」を入力し、Enter を押します。(「77」以外の値では終了しません)
⇒ スレッド終了プロセスが始まり、やがてプログラムは終了します。
終了すると、「コマンド入力待ち」の状態が表示されます。

```
ビットコインのbid価格= 819299.0
ビットコインのbid価格= 819305.0

input 「77」で終了します >>>
77

[77] が入力されました

main2() flag1 77
main2() flag2 99

まだ、flag2 が立ってません
main2() flag1 77
main2() flag2 99

まだ、flag2 が立ってません
main2() flag1 77
main2() flag2 99

まだ、flag2 が立ってません
scrapr()処理が終了しました
DLLから読み出した値 <class 'float'>,819305.00
main2() flag1 77
main2() flag2 88

処理がmain2()へ戻りました
main2() flag1 77
main2() flag2 88

C:\Users\kenken\Documents\Python Scripts>
```



MT 4側は、最後にDLLに書き込まれた「価格」を表示し続けます。



(2) セット2の使い方

※こちらの場合は、MT 4 (MQL4) の「Disp_option_2.mq4 (EA)」に、Python アプリの「立上げ」と「終了」を行う MQL4 コードを設定しています。

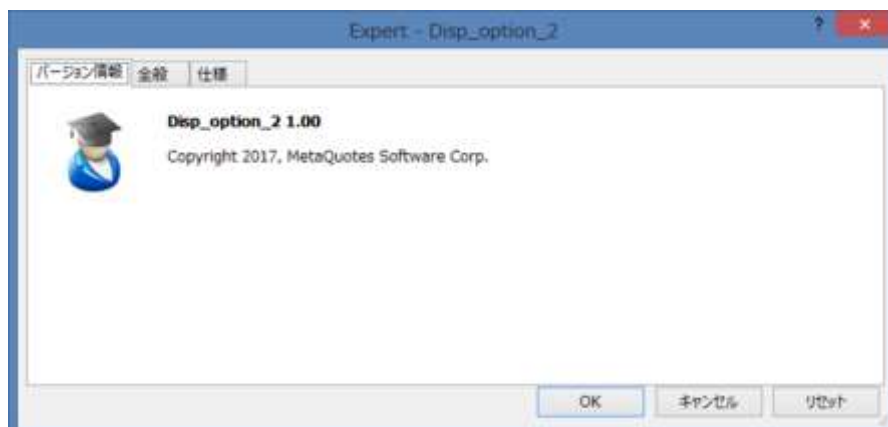
従って、「Disp_option_2.mq4 (EA)」を起動 (チャートに表示) させるだけで、Python アプリによるスクレイピングと、MT 4 (チャート画面) への価格表示を実行させることができます。

- ①Python 側のソフトは自動で立上げる
- ②スクレイピングを実行し、その結果をチャート上に表示します
- ③EA を終了すると、Python ソフトも自動で終了します。

MQL4 アプリを起動する；



[Disp_option_2.mq4] を選択

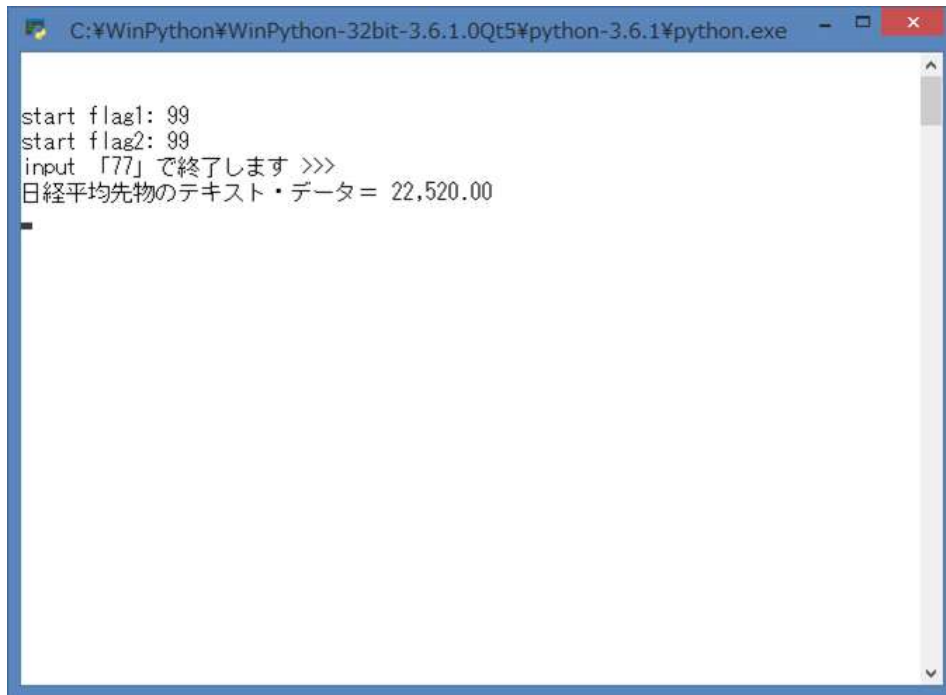


[OK]

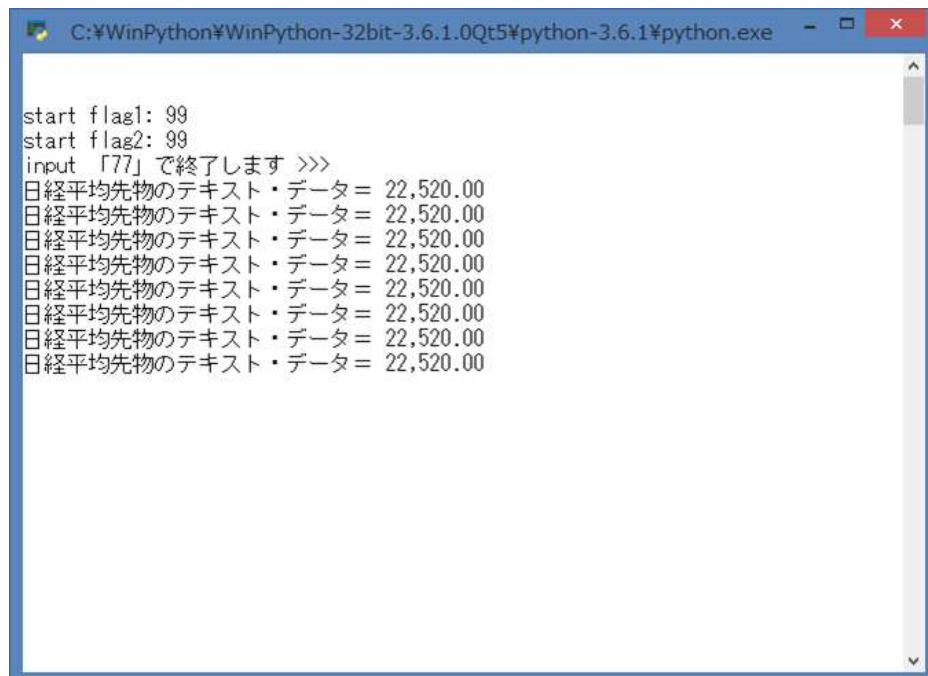


M T 4の横、あるいは重なる様に「コマンド・プロンプト画面」が立上り、同時に Python アプリの「Scraping_try_2.py」が起動します。

コマンド・プロンプト；



```
C:\WinPython\WinPython-32bit-3.6.1.0Qt5\python-3.6.1\python.exe
start flag1: 99
start flag2: 99
input 「77」で終了します >>>
日経平均先物のテキスト・データ = 22,520.00
```



```
C:\WinPython\WinPython-32bit-3.6.1.0Qt5\python-3.6.1\python.exe
start flag1: 99
start flag2: 99
input 「77」で終了します >>>
日経平均先物のテキスト・データ = 22,520.00
日経平均先物のテキスト・データ = 22,520.00
日経平均先物のテキスト・データ = 22,520.00
日経平均先物のテキスト・データ = 22,520.00
日経平均先物のテキスト・データ = 22,520.00
日経平均先物のテキスト・データ = 22,520.00
日経平均先物のテキスト・データ = 22,520.00
日経平均先物のテキスト・データ = 22,520.00
```

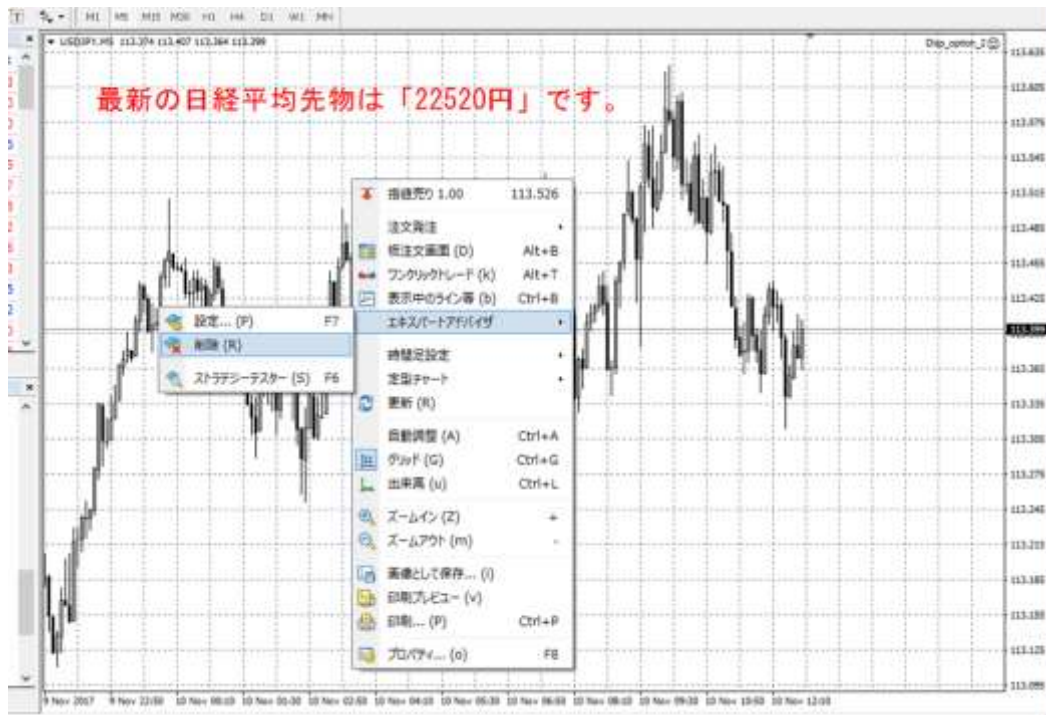


MT 4 ;



MQL4 アプリを終了する；

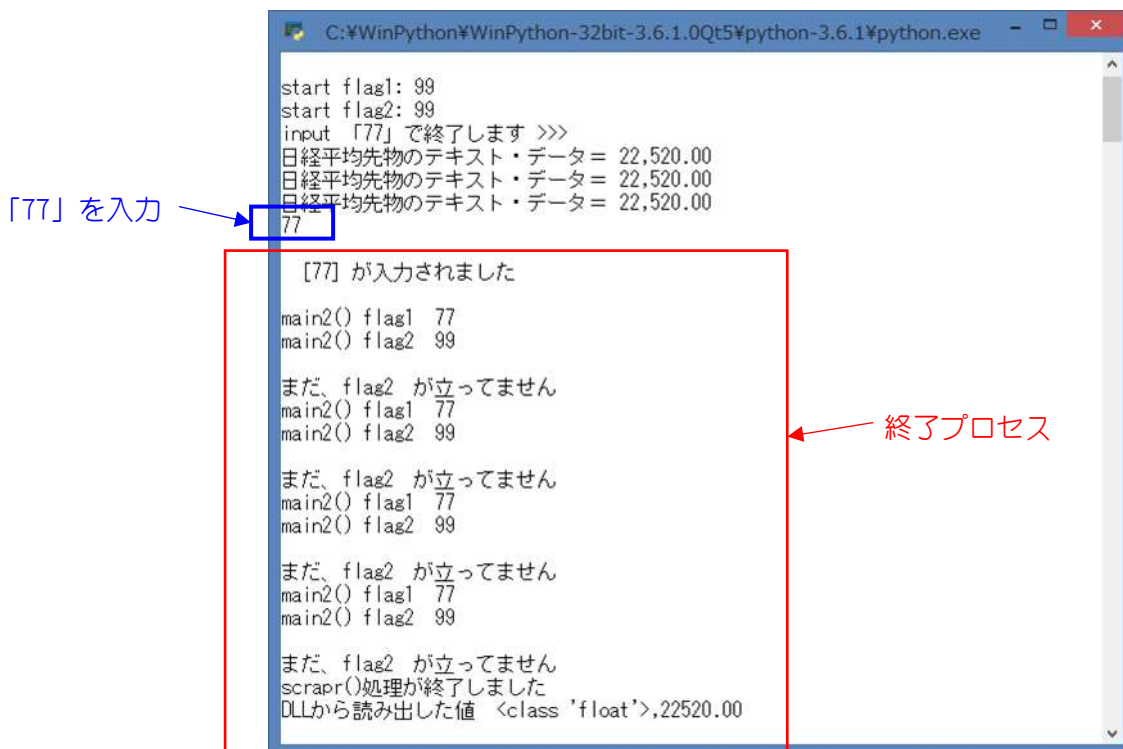
エキスパートアドバイザー「Disp_option_2.mq4」を通常の方法で終了します



すると、コマンド・プロンプトがPython アプリの「Scraping_try_2.py」と共に終了します。

※補足；

コマンド・プロンプト画面で「77」を入力することでも、Python アプリを終了させることもできます。



3. 添付プログラムのコード詳細

セット	内容	MQL4	Python
1	「ビットコイン」版	Disp_option_1.mq4 (EA)	Scraping_try_1.py
2	「日経平均先物」版	Disp_option_2.mq4 (EA) ; 起動・終了	Scraping_try_2.py

(1) セット1

MQL4 コード ; Disp_option_1.mq4 (EA)

```

//+-----+
//|                                     Disp_option_1.mq4 |
//|                                     Copyright 2017, MetaQuotes Software Corp. |
//|                                     https://www.mql5.com |
//+-----+
#property copyright "Copyright 2017, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property strict
//
#import "shared_memory.dll"
    double set_a();
    double write_a(double, int);
    double read_a(int);
    double close_a();
#import
//+-----+
//| Expert initialization function |
//+-----+
int OnInit()
{
//--- //DLL 初期化
    set_a();
    //「ラベル・オブジェクト」設定
    ObjectsDeleteAll();
    int pixel_x=50, pixel_y=50;
    ObjectCreate("price_1", OBJ_LABEL, 0, 0, 0);
    ObjectSet("price_1", OBJPROP_XDISTANCE, pixel_x);
    ObjectSet("price_1", OBJPROP_YDISTANCE, pixel_y);
    //チェック用
    printf("pixel_x : pixel_y = %f : %f", pixel_x, pixel_y);
//---
    return(INIT_SUCCEEDED);
}
//+-----+
//| Expert deinitialization function |
//+-----+
void OnDeinit(const int reason)
{
//--- //DLL 解放
    close_a();
    //オブジェクト削除
    ObjectDelete("price_1");
}

```

```

//+-----+
//| Expert tick function |
//+-----+
void OnTick()
{
//---
// 「準備中」を表示
string st1="DLL 読み込み準備中";
ObjectSetText("price_1",st1,20,"M S ゴシック",Blue);
Sleep(200);
//DLL から読み込んで表示する
double op=read_a(1);
string st2="bitFlyer : ビットコインの best_bid 価格は「'+top+'円」です。";
ObjectSetText("price_1",st2,20,"M S ゴシック",Red);
}
//+-----+

```

Python コード ; Scraping_try_1.py

```

# -*- coding: utf-8 -*-
"""
Created on Sat Oct 14 22:37:39 2017
Scraping_try_1.py
    ・Bit Coinの価格をスクレイピングして、DLL に書き込む（無限ループ）
    ・コンソールから「77」を打ち込むと、無限ループから抜ける
@author: kenken
"""
# ----- 初期設定 -----
# Bit coin スクレイピング
import requests
# ビットコイン用の
url="https://api.bitflyer.jp/v1/ticker?product_code=BTC_JPY"
# DLL アクセス用
import sys
import time
from ctypes import *
# DLL のロード -----
lib=windll.LoadLibrary('C:/Users/kenken/AppData/Roaming/MetaQuotes/Terminal'+
                        '/FCCD626CCEAFA0C866593963E6A400F0/MQL4/Libraries'+
                        '/shared_memory.dll');
#DLL 中の関数呼び出し#戻り値型(restype)、引数型(argtypes)を指定
lib.set_a.restype=c_double
lib.set_a.argtype=None
lib.close_a.restype=c_double
lib.close_a.argtype=None
lib.write_a.restype=c_double
lib.write_a.argtypes=[c_double,c_int]
lib.read_a.restype=c_double
lib.read_a.argtype=c_int
# DLL 関数 ; 初期設定
lib.set_a()
#----- マルチ・スレッド用-----
from winsound import Beep
from time import sleep
import threading
# 「global 変数」の宣言 ; 異なるスレッド間で共有可能
flag1='99'

```

```

flag2='99'
#----- 初期設定終了 -----
print(' ¥n', flush=True)
print(' start flag1: %d' %int(flag1), flush=True)
print(' start flag2: %d' %int(flag2), flush=True)

def scrape():
    global flag1, flag2
    # 無限ループ (ビットコイン価格のスクレイピング)
    while 1:
        # 繰り返し (ループ) #-----
        bit_c=requests.get(url)
        json=bit_c.json()
        price2=float(json["best_bid"])
        print(' ビットコインのbid 価格=', price2)
        # DLLアクセス # 書込み a[1]を使用
        lib.write_a(price2, 1)
        Beep(2000, 500)
        sleep(5)
        if flag1=='77':
            sleep(1)
            Beep(4000, 500)
            Beep(2000, 500)
            print(' scrapr()処理が終了しました', flush=True)
            flag2='88'
            r_ret=lib.read_a(1)
            print(' DLL から読み出した値 %s, %5.2f'%(type(r_ret), r_ret))
            # DLL解放
            lib.close_a()
            break
        else:
            pass

def main():
    global flag1, flag2
    # scrape()を別スレッドとして起動する
    th=threading.Thread(target=scrape)
    th.start()

    if flag1=='77':
        pass
    else:
        while 1: #無限ループ ([77] 入力待ち)
            input_line=input("input [77] で終了します >>> "+"¥n")
            if input_line=='77':
                print(' ¥n [77] が入力されました ¥n', flush=True)
                break
            else:
                pass
            flag1=input_line
            Beep(3000, 500)
            sleep(1)
            while 1:
                print(' main2() flag1 %d' %int(flag1), flush=True)
                print(' main2() flag2 %d ¥n' %int(flag2), flush=True)
                if flag2=='88':
                    print(' 処理が main2()へ戻りました', flush=True)
                    break
                else:
                    print(' まだ、flag2 が立ってません', flush=True)

```

```
        sleep(1)
# スレッドを joinして終了する
th.join()
#-----
print('main2() flag1 %d' %int(flag1), flush=True)
print('main2() flag2 %d' %int(flag2), flush=True)

#=====
if __name__=='__main__':
    main()
```

注記:

```
.....
# DLLのロード -----
lib=windll.LoadLibrary('C:/Users/kenken/AppData/Roaming/MetaQuotes/Terminal'+
                        '/FCCD626CCEAFA0C866593963E6A400F0/MQL4/Libraries'+
                        '/shared_memory.dll');
.....
```

上記の「DLL」配置は、アメンボの設定の場合ですので、各位のMT4設定状況によって変える必要があります。

(2) セット2

ML4 コード : Disp_option_2.mq4 (EA)

```
//+-----+
//|                                     Disp_option_2.mq4 |
//|                                     Copyright 2017, MetaQuotes Software Corp. |
//|                                     https://www.mql5.com |
//+-----+
#property copyright "Copyright 2017, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property strict
//
#define WM_CLOSE 0x0010
#define WM_SYSCOMMAND 0x0112
#define SC_CLOSE 0xF060
//=====Win32API=====
// Win32API 使用宣言
#import "shell32.dll"
    int ShellExecuteW(int handle, int ipVerb, string lpFile, string lpParameters,
string lpDirectory, int nCmdShow);
#import "user32.dll"
    int SendMessageW(int hWnd, int Msg, int wParam, int lParam);
    int PostMessageW(int hWnd, int Msg, int wParam, int lParam);
    int FindWindowW(string lpClassName, string lpWindowName);
    int DestroyWindow(int hWnd);
// 共有メモリ使用宣言
#import "shared_memory.dll"
    double set_a();
    double write_a(double, int);
    double read_a(int);
    double close_a();
#import
//+-----+
//| Expert initialization function |
//+-----+
int OnInit()
{
//---
    //Python アプリを起動する
    // 起動 OK 品 open が暗黙のうちに「open」⇒「数値」に変換される
    int ret=ShellExecuteW(NULL, "open", "C:¥¥WinPython¥¥WinPython-32bit-
3.6.1.0Qt5¥¥python-
3.6.1¥¥python.exe", "Scraping_try_2.py", "C:¥¥Users¥¥kenken¥¥Documents¥¥Python
Scripts", 5);
    //
    PlaySound("alert.wav");
    //共有メモリのセットと、オブジェクトの初期化
    set_a();
    ObjectsDeleteAll();
    int pixel_x=50, pixel_y=50;
    ObjectCreate("price_1", OBJ_LABEL, 0, 0, 0);
    ObjectSet("price_1", OBJPROP_XDISTANCE, pixel_x);
    ObjectSet("price_1", OBJPROP_YDISTANCE, pixel_y);
    //チェック用
    printf("pixel_x : pixel_y = %f : %f", pixel_x, pixel_y);
//---
}
```

```

    return(INIT_SUCCEEDED);
}
//+-----+
//| Expert deinitialization function |
//+-----+
void OnDeinit(const int reason)
{
//---
    //Python アプリの終了
    //OnInit()で起動した「ウインドウの名称」は ⇒ "C:¥¥WinPython¥¥WinPython-
32bit-3.6.1.0Qt5¥¥python-3.6.1¥¥python.exe"
    int PP=PostMessageW(FindWindowW(NULL,"C:¥¥WinPython¥¥WinPython-32bit-
3.6.1.0Qt5¥¥python-3.6.1¥¥python.exe"), WM_SYSCOMMAND, SC_CLOSE, 0);
    PlaySound("alert2.wav");
    //DLL 解放
    close_a();
    ObjectDelete("price_1");
}
//+-----+
//| Expert tick function |
//+-----+
void OnTick()
{
//---
    //「準備中」を表示
    string st1="DLL 読み込み準備中";//O K品
    ObjectSetText("price_1",st1,20,"MS ゴシック",Blue);
    Sleep(200);
    //DLL から読み込んで表示する
    double op=read_a(1);
    //string st2="bitFiyer : ビットコインの best_bid 価格は「top+」円です。";//O K品
    string st2="最新の日経平均先物は「top+」円です。";
    ObjectSetText("price_1",st2,20,"MS ゴシック",Red);
}
//+-----+

```

注記:

```

.....
    // 起動 O K品 open が暗黙のうちに「open」⇒「数値」に変換される
    int ret=ShellExecuteW(NULL,"open","C:¥¥WinPython¥¥WinPython-32bit-
3.6.1.0Qt5¥¥python-
3.6.1¥¥python.exe","Scraping_try_2.py","C:¥¥Users¥¥kenken¥¥Documents¥¥Python
Scripts",5);
.....
    //OnInit()で起動した「ウインドウの名称」は ⇒ "C:¥¥WinPython¥¥WinPython-
32bit-3.6.1.0Qt5¥¥python-3.6.1¥¥python.exe"
    int PP=PostMessageW(FindWindowW(NULL,"C:¥¥WinPython¥¥WinPython-32bit-
3.6.1.0Qt5¥¥python-3.6.1¥¥python.exe"), WM_SYSCOMMAND, SC_CLOSE, 0);
.....

```

上記の設定は、アメンボの「Python 本体とアプリ」配置の場合ですので、各位の「Python とそのアプリ」設定状況によって変える必要があります。

Python コード ; Scraping_try_2.py

```

# -*- coding: utf-8 -*-
"""
Created on Sat Oct 14 00:24:28 2017
    Scraping_try_2.py
    ・日経平均先物の価格をスクレイピングして、DLL に書き込む（無限ループ）
    ・コンソールから「77」を打ち込むと、無限ループから抜ける
@author: kenken
"""

# ----- 初期設定 -----
# スクレイピング用 -----
import urllib.request
from bs4 import BeautifulSoup
# 日経平均先物
url="https://stocks.finance.yahoo.co.jp/stocks/detail/?code=5040469.0"
# DLL アクセス用
import sys
import time
from ctypes import *
# DLL のロード -----
lib=windll.LoadLibrary('C:/Users/kenken/AppData/Roaming/MetaQuotes/Terminal'+
                        '/FCCD626CCEAF40C866593963E6A400F0/MQL4/Libraries'+
                        '/shared_memory.dll');
# DLL 中の関数呼び出し#戻り値型(restype)、引数型(argtypes)を指定
lib.set_a.restype=c_double
lib.set_a.argtype=None
lib.close_a.restype=c_double
lib.close_a.argtype=None
lib.write_a.restype=c_double
lib.write_a.argtypes=[c_double, c_int]
lib.read_a.restype=c_double
lib.read_a.argtype=c_int
# DLL 関数 ; 初期設定
lib.set_a()
#----- マルチ・スレッド用-----
from winsound import Beep
from time import sleep
import threading
# 「global 変数」の宣言 ; 異なるスレッド間で共有可能
flag1='99'
flag2='99'
#----- 初期設定終了 -----
print('\n', flush=True)
print(' start flag1: %d' %int(flag1), flush=True)
print(' start flag2: %d' %int(flag2), flush=True)

def scrape():
    global flag1, flag2
    # 無限ループ（日経平均先物のスクレイピング）
    while 1:
        # 繰り返し（ループ） #-----
        res=urllib.request.urlopen(url)
        soup=BeautifulSoup(res, 'html.parser')
        stoksPrice=soup.select('.stoksPrice')
        print("日経平均先物のテキスト・データ=", stoksPrice[1].text)
        #-----読み取りデータを数値に変換する-----
        # 例 ; [19.985]の「,」が邪魔
        price=stoksPrice[1].text

```

```

price1=price.replace(",","")
price2=float(price1)
# DLLアクセス # 書込み a[1]を使用
lib.write_a(price2,1)
Beep(2000,500)
sleep(5)
if flag1=='77':
    sleep(1)
    Beep(4000,500)
    Beep(2000,500)
    print('scraper()処理が終了しました',flush=True)
    flag2='88'
    r_ret=lib.read_a(1)
    print('DLLから読み出した値 %s,%5.2f'%(type(r_ret),r_ret))
    # DLL解放
    lib.close_a()
    break
else:
    pass

def main():
    global flag1,flag2
    th=threading.Thread(target=scrape)
    th.start()

    if flag1=='77':
        pass
    else:
        while 1:
            input_line=input("input 「77」で終了します >>> "+'\n')
            if input_line=='77':
                print('\n 「77」が入力されました \n',flush=True)
                break
            else:
                pass
            flag1=input_line
            Beep(3000,500)
            sleep(1)
            while 1:
                print('main2() flag1 %d' %int(flag1),flush=True)
                print('main2() flag2 %d \n' %int(flag2),flush=True)
                if flag2=='88':
                    print('処理がmain2()へ戻りました',flush=True)
                    break
                else:
                    print('まだ、flag2 が立ってません',flush=True)
                    sleep(1)

    th.join()
    #-----
    print('main2() flag1 %d' %int(flag1),flush=True)
    print('main2() flag2 %d' %int(flag2),flush=True)

#=====
if __name__=='__main__':
    main()

```

注記:

```
.....  
# DLL のロード -----  
lib=windll.LoadLibrary('C:/Users/kenken/AppData/Roaming/MetaQuotes/Terminal' +  
                        '/FCCD626CCEAFA0C866593963E6A400F0/MQL4/Libraries' +  
                        '/shared_memory.dll');  
.....
```

上記の「DLL」配置は、アメンボの設定の場合ですので、各位のMT4設定状況によって変える必要があります。

4. その他

(1) 「shared_memory.dll」について

※従来から、アメンボが公開している32ビット版の「シェアード・メモリ（共有メモリ）」用のDLLです。

過去に詳細内容を投稿済みですが、念のために、ポイントを記載しておきます。

機能概要:

- 共有メモリ上で、1個のレジスタ (r) と、3個の配列 (a, b, c) をMT4のチャートに設定された「全てのEA、インディケータ」から、共有することが出来ます。ただし、排他処理を入れていませんので取り扱いには注意が必要です。
- ベースの作成技術は同じですが、使用する場面の違いを意識できるように、レジスタと配列ではメモリ容量に差をつけました。
- 共有メモリ（レジスタ、配列）を通して、EAやインディケータはデータのやり取りを行うことが出来ます。

DLLの置場所:

- 「Libraries」フォルダ内に置きます。

DLLの呼出し方:

- 使う機能のみを「#import」で宣言すればOKです、例えば「配列a」のみを使うのであれば、下記のコードを先頭かヘッダファイルに書いておけば充分です。

```
#import "shared_memory.dll"  
double set_a();  
double write_a(double, int);  
double read_a(int);  
double close_a();  
#import
```

関数機能一覧；

	関 数	機 能
レジスタ r	set_r()	レジスタ r 領域を設定します 使用する場合に宣言を実施
	write_r(データ, 要素 No) ・データ; double 型 ・要素 No; int 型	要素 No にデータ (数値) を書込む 要素 No は「0~199」可能 (レジスタ数は 200 個)
	read_r(要素 No) ・要素 No; int 型	要素 No 中のデータを読み出す
	close_r()	レジスタ r を閉じる
配列 a	set_a()	配列 a 領域を設定します
	write_a(データ, 要素 No) ・データ; double 型 ・要素 No; int 型	要素 No にデータ (数値) を書込む 要素 No は「0~1999」可能 (要素数は 2000 個)
	read_a(要素 No) ・要素 No; int 型	要素 No 中のデータを読み出す
	close_a()	配列 a を閉じる
配列 b	set_b()	配列 b 領域を設定します
	write_b(データ, 要素 No) ・データ; double 型 ・要素 No; int 型	要素 No にデータ (数値) を書込む 要素 No は「0~1999」可能 (要素数は 2000 個)
	read_b(要素 No) ・要素 No; int 型	要素 No 中のデータを読み出す
	close_b()	配列 b を閉じる
配列 c	set_c()	配列 c 領域を設定します
	write_c(データ, 要素 No) ・データ; double 型 ・要素 No; int 型	要素 No にデータ (数値) を書込む 要素 No は「0~3999」可能 (要素数は 4000 個)
	read_c(要素 No) ・要素 No; int 型	要素 No 中のデータを読み出す
	close_c()	配列 c を閉じる

以 上