Call Python/Numpy Function within Metatrader 4

PyData Osaka #3 2017/01/21

@fx_kirin

Self-Introduction

- Individual professional Forex trader since 2010
- Specialized in automatic algorithm trading
- Python / Ruby / C++ / Hacking
- Data Analytics
- Road bike
- Twitter : @fx_kirin
- Blog : http://fx-kirin.com

WIZARD IN THE MARKET

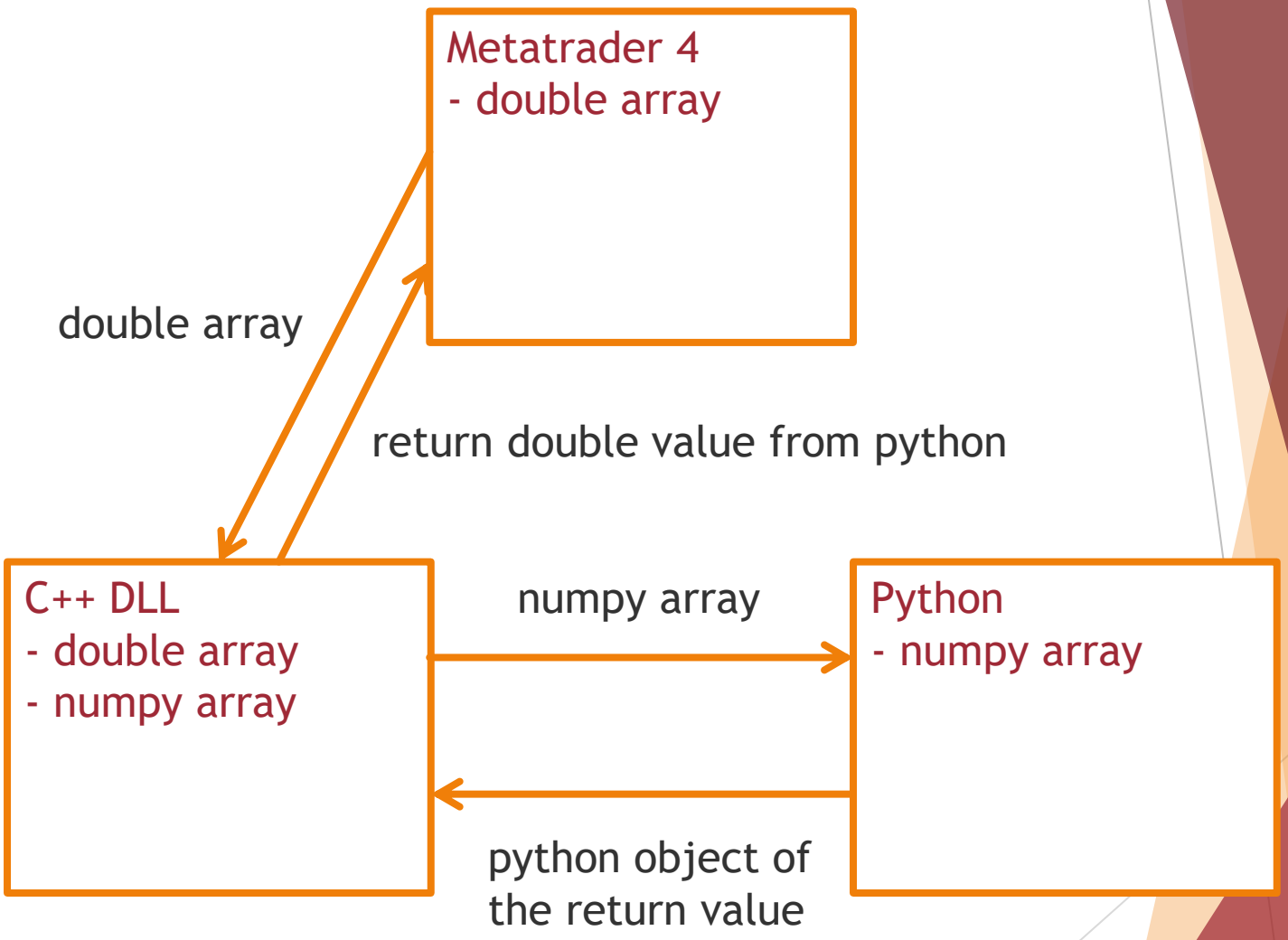
What is Metatrader 4

- Automatic Trading Platform
- It has own programing language called MQL which specifically focuses on trading.
  - MQL is close to C or C++.
- It has comfortable GUI and it's quite fast enough.
- It can call a user built DLL from MQL.
  - I'm going to use this function to accomplish what I want.

Embedded Python Application

- Python Interpreter can be called from other languages and execute low level python code in it.
- Numpy has also C-API
  - It can directly access to row binary memory data allocated by numpy.
- It can also be call from DLLs. Let's make it.

https://docs.python.org/2/extending/embedding.html


Metatrader 4
- double array
double array
return double value from python
C++ DLL
- double array
- numpy array
numpy array
Python
- numpy array
python object of
the return value

Howto

- GitHub

https://github.com/fx-kirin/mt4-numpy-example

MQL

```
#import "PythonNumpyMean.dll"
double GetNumpyMean(double &value[], int arraysize, int period, int index);
#import
```

```
//+------------------------------------------------------------------+
//| expert start function
//+------------------------------------------------------------------+
int start()
  {
   double close[];
   int size = ArrayCopySeries(close, MODE_CLOSE);
   double test[];
   if(size <= 0){
      Print("no size.");
      return(0);
   }
   ArrayResize(test, size);
   ArrayCopy(test, close);
  int limit = Bars;
   for(int i=limit-1000; i>=0; i--){
      Buf0[i] = GetNumpyMean(test, size, 120, i);
   }

   return(0);
}
```

```
BOOL APIENTRY DllMain(HANDLE hModule,DWORD ul_reason_for_call,LPVOID lpReserved)
  {
//----
   switch(ul_reason_for_call)
     {
      case DLL_PROCESS_ATTACH:
          Py_Initialize();
          import_array1(-1);
          PyObject *pName;
          pName = PyString_FromString("numpy");
          NumPy = PyImport_Import(pName);
          Py_DECREF(pName);
          if(NumPy != NULL){
             PyNumpyMeanFunc = PyObject_GetAttrString(NumPy, "mean");
             if(PyNumpyMeanFunc == NULL){
                 OutputDebugString("Failed to get mean func.");
             }
          }else{
              OutputDebugString("Failed to load numpy.");
          }
          OutputDebugString("Attached.");
          break;
      case DLL_THREAD_ATTACH:
          break;
      case DLL_THREAD_DETACH:
          break;
      case DLL_PROCESS_DETACH:
          Py_DECREF(NumPy);
          Py_DECREF(PyNumpyMeanFunc);
          Py_Finalize();
          break;
     }
//----
   return(TRUE);
  }
//
```

```
MT4_EXPFUNC double __stdcall GetNumpyMean(double *value, int arraysize, int period, int index)
  {
//----
   if(value==NULL)
     {
      //OutputDebugString("GetArrayItemValue: NULL array¥n");
      return(0.0);
     }
   if(arraysize < index + period)
     {
      //OutputDebugString("GetArrayItemValue: wrong arraysize ¥n");
      return(0.0);
     }

   npy_intp npy_arraysize;

   npy_arraysize = period;
   double* shifted_value = (double*)((int)value + sizeof(double)*index);
   PyObject *np_value = PyArray_SimpleNewFromData(1, &npy_arraysize, NPY_DOUBLE, shifted_value);
   Py_INCREF(np_value);;

   PyObject *pArgs = PyTuple_New(1);
   PyTuple_SetItem(pArgs, 0, np_value);
   PyObject *pResult = PyObject_CallObject(PyNumpyMeanFunc, pArgs);
   Py_DECREF(pArgs);
   Py_DECREF(np_value);

   double mean = PyFloat_AsDouble(pResult);
   Py_DECREF(pResult);
//----
   return(mean);
  }
```

Caution

- Do not call Python Interpreter twice.
  - It’s not supported even if you properly finalize it.
    - It may crashed the process.