http://blog.naver.com/PostView.nhn?blogId=stop2y&logNo=221111049566&categoryNo=136&parentCategoryNo=0&viewDate=&currentPage=1&postListTopCurrentPage=1&from=postView

[Python - Win32API] 동적 라이브러리(dll) 호출

[](http://blog.naver.com/stop2y) [댕이댕이](http://blog.naver.com/stop2y) *・*2017. 10. 5. 15:54

[URL 복사](http://blog.naver.com/PostView.nhn?blogId=stop2y&logNo=221111049566&categoryNo=136&parentCategoryNo=0&viewDate=&currentPage=1&postListTopCurrentPage=1&from=postView) [이웃추가](http://blog.naver.com/PostView.nhn?blogId=stop2y&logNo=221111049566&categoryNo=136&parentCategoryNo=0&viewDate=&currentPage=1&postListTopCurrentPage=1&from=postView)

파이썬 동적 라이브러리 호출

파이썬에서 import를 통하여 모듈에 대하여 사용할 수 있다. 하지만, C로 구성된 dll 파일이나 기타 Win32API에 사용되는 kernel32.dll 같은 라이브러리를 사용하고자 할 경우엔 별도의 호출 방식이 필요하다.

실습 내용

비쥬얼 베이직으로 C로 구성된 동적 라이브러리(dll)을 만든 뒤 파이썬에서 ctypes 모듈을 이용하여 dll을 호출하여 dll 내 함수를 사용한다.

비쥬얼 베이직으로 C 언어로 구성된 dll 파일 만들기

#include <stdio.h>

#include <windows.h>

#define EXPORT extern "C" \_\_declspec(dllexport)

EXPORT int plus(int nA, int nB);

EXPORT int minus(int nA, int nB, int nVal);

BOOL APIENTRY DllMain(HANDLE hModule,

DWORD ul\_reason\_for\_call,

LPVOID lpReserved)

{

return TRUE;

}

EXPORT int plus(int a, int b)

{

return a + b;

}

EXPORT int minus(int a, int b, int val)

{

if (a > b)

{

val = a - b;

return TRUE;

}

else

return FALSE;

}

 C언어에서 동적 라이브러리 파일에 대하여 코딩을 하기 위해선 EXPORT extern "C" \_\_declspec(dllexport) 를 입력하여 완성된 dll 에서 응용프로그램(파이썬)으로 함수를 호출을 가능케 하도록 해줍니다.  
 또한, 함수 선언 시 EXPORT를 붙여 외부로 EXPORT되는 함수임을 정의해줍니다. 위 선언된 EXPORT 함수는 plus와 minus 함수로 단순하게 정수형 인자를 받아 더하거나 빼거나 하는 연산을 위한 함수입니다. 이 함수는 후에 파이썬으로 부터 인자를 받아와 연산을 출력시켜주게 됩니다.  
 완성된 코드에 대하여 컴파일을 하면 아래와 같이 dll 파일이 생성됩니다.

http://postfiles4.naver.net/MjAxNzEwMDVfMjE2/MDAxNTA3MTgxNjUzMjg2.WKa3Soy0xPdNsVL3f09LCPZFP6rrTFViS7t9vvnxpS0g.jTRrdCyN4avHdlXk-AqjgjLnEBOlroz3XzxFvfEPi7Eg.PNG.stop2y/1.PNG?type=w966

파일로 첨부해드리겠습니다. 참조하실분은 참조해주시기 바랍니다.

[dll\_test.dll](http://blogattach.naver.net/b82da41700333287ab4c2b1f22c5bfc66038cd9e/20171005_215_blogfile/stop2y_1507181671013_TBG21l_dll/dll_test.dll)

파이썬으로 컴파일된 dll 호출하여 사용하기

우선은 이 과정을 위해 사용되어질 모듈은 ctypes 입니다.  
ctypes는 C언어의 type을 쉽게 다룰 수 있도록 파이썬 2.5 버전 이후부터 표준 라이브러리에 포함된 모듈로, C언어의 Native 코드를 작성할 필요 없이 동적 라이브러리를 바로 불러 입출력 타입만 지정하고 사용할 수 있습니다.  
쉽게 말해서 ctypes 모듈은 C언에서 사용되어지는 것들을 파이썬에서 최대한 활용토록 해주는 모듈입니다.

[Loadllib.py](http://blogattach.naver.net/6cf970c0d0818854789ff9cef6106b12b2e612fbc8/20171005_237_blogfile/stop2y_1507181884330_INNt55_py/Loadllib.py)

#!python

# -\*- coding: cp949 -\*-

# my\_debugger.py

from ctypes import \*

#파이썬에서 c언어의 함수를 사용하기 위해 ctypes 모듈 정의

Load\_DLL=WinDLL('dll\_test.dll')

#사용하고자할 모듈인 dll\_test.dll 동적라이브러리 파일을 불러온다. ctypes.WinDLL('dll 파일 이름')

plus\_func=Load\_DLL['plus']

#Load\_DLL에는 dll\_test.dll 라이브러리에 대한 정보가 지녀졌으며 이 라이브러리에 포함된 함수인 plus 함수를 plus\_func

#에 인스턴스로 저장한다.

plus\_func.argtypes=(c\_int,c\_int)

#plus\_func 함수의 인자 형식을 지정한다. plus 함수는 int 형 인자 2개를 받으므로 2개의 인자가 들어간다.

plus\_func.restype=c\_int

#plus\_func 함수의 출력 형식을 지정한다.

print('Arg PLUS')

intA=int(input("arg1 : "))

intB=int(input("arg2 : "))

plus\_res=plus\_func(intA,intB)

#함수를 호출한 뒤 반환 값을 plus\_res에 저장한다.

print("Arg1 + Arg2 :",intA+intB,'\n')

minus\_func=Load\_DLL['minus']

minus\_func.argtypes=(c\_int,c\_int,c\_int)

minus\_func.restype=c\_bool

print('Arg MINUS')

intA=int(input("arg1 : "))

intB=int(input("arg2 : "))

minus\_res=minus\_func(intA,intB,0)

if minus\_res==True:

print("Arg1 > Arg2")

print("Arg1-Arg2 :",intA-intB)

else:

print("Arg1 <= Arg2")

print("Arg1-Arg2 :",intA-intB)

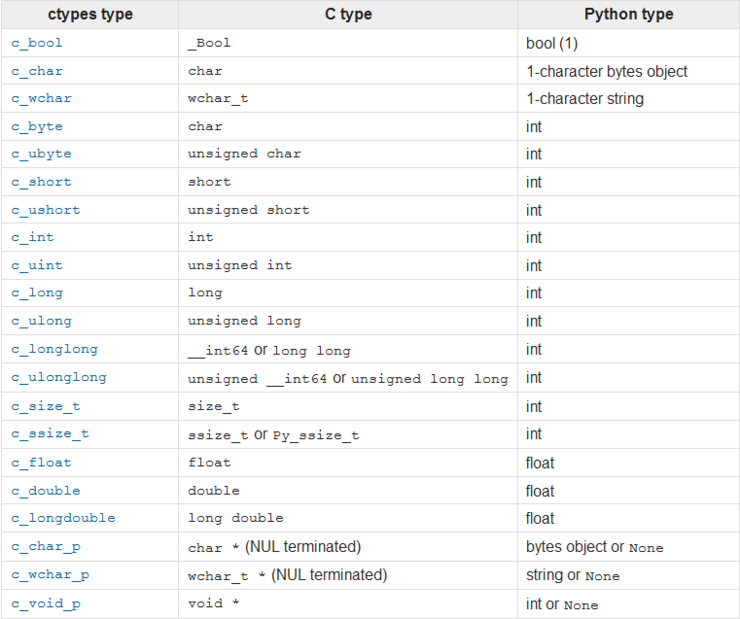
CDLL('dll\_test.dll')

#호출된 dll\_test.dll 연결을 종료한다.

**Load\_DLL=ctypes.WinDLL("라이브러리 파일 이름")**  
라이브러리를 호출하여 호출된 인스턴스를 변수에 저장하여 후에 Dictionary 타입 처럼 함수에 접근 할 수 있습니다.

**plus\_func=Load\_DLL['plus']**  
위에 언급했다시피 라이브러리에 대한 정보를 지닌 인스턴스 Load\_DLL는 Dictionary 형태이며 이 곳에서 라이브러리내 함수를 호출할 수 있습니다. 가령 dll 파일 내 함수인 plus 함수를 호출하기 위해선 위와 같이 ['fuction'] 형태로 접근 한 뒤 정의된 함수를 지정할 수 있습니다.

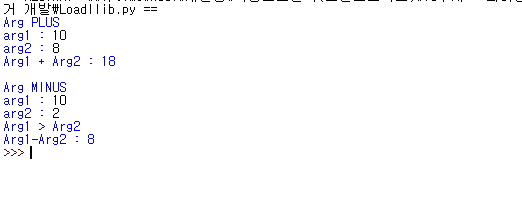
**plus\_func.argtypes=(c\_int,c\_int)**  
위에서 함수에 대하여 '불러오는 과정'을 거쳤다면 함수가 인자를 어떻게 받는지 정의를 해주어야 합니다. fuction.argtypes는 함수의 인자를 설정하는 것으로 인자에 대한 c\_type은 아래를 참조하시면 됩니다.



예를들어 정수형 인자 1개와 문자열 인자 1개를 받는 함수일 시  
fuction.argtypes=(c\_int, c\_char)  
이런 식으로 정의해주시면 됩니다.

**plus\_func.restype=c\_int**  
restype은 함수의 반환 값에 대한 c\_type을 지정합니다. 반환된 값이 정수 형태이면 위와 같이 c\_int 형을 주실 수 있으며 만약 반환 값이 없을 경우엔 c\_void\_p 반환 값이 bool 형태(참,거짓)일 경우엔 c\_bool로 지정하시면 됩니다.  
아래는 그 예 입니다.  
fuction.restype=c\_bool  
  
**intA=int(input("arg1 : "))**  
**intB=int(input("arg2 : "))**  
**plus\_res=plus\_func(intA,intB)**  
plus\_func 함수에 대한 인자 설정과 반환 값 설정이 끝났다면 plus\_func 함수에 인자 값을 집어 넣은 뒤 그 값의 반환 값을 plus\_res로 반환하게 합니다.  
  
**CDLL('dll\_test.dll')**  
메모리에 연결된(로드된) 라이브러리인 dll\_test.dll에 대하여 언로드합니다. 반드시 dll의 사용이 끝났으면 이 함수를 통해서 메모리를 정리해주셔야 합니다.

실행 결과



https://codingcoding.tistory.com/610

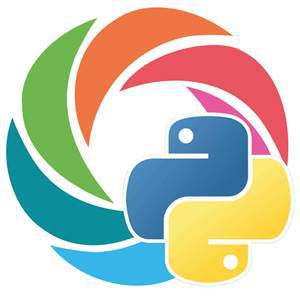
## [**파이썬 CTypes 구조체 자세히 알아보기 (PyThon, 파이선)**](https://codingcoding.tistory.com/610)

* [프로그래밍](https://codingcoding.tistory.com/category/%ED%94%84%EB%A1%9C%EA%B7%B8%EB%9E%98%EB%B0%8D)

* 2018. 2. 23.

**파이썬 CTypes 구조체 자세히 알아보기 (PyThon, 파이선)**

파이썬에서 외부 라이브러리를 쓰기 위해서는 확장 모듈을 만들고 그 속에서 외부 라이브러리를 호출하는 방법이 있습니다. 사실 이보다 훨씬 더 간단한 방법이 있었는데, 바로 파이선에서 제공하는 파이썬 ctypes 모듈을 사용하는 것입니다. ctypes 모듈을 이용하면 C의 데이터 타입이나, DLL 혹은 공유 라이브러리(shared library)의 함수를 직접 사용할 수 있습니다.

[Python 자료형] 확장 모듈과 라이브러리

파이선 ctypes를 사용하는 방법은 매우 간단합니다. 다음 예제를 보겠습니다.

|  |  |  |
| --- | --- | --- |
| 1  2  3  4  5  6 | >>> import ctypes     <- 윈도우에서 테스트 한 결과.  >>> print(ctypes.windll.kernel32)  <WinDLL 'kernel32', handle   ...  at   ...>    >>> print(ctypes.cdll.msvcrt)  <CDLL 'msvcrt', handle  ...  at  ...> | [cs](http://colorscripter.com/info#e) |

파이썬 ctypes를 임포트하는 것으로 모든 준비가 끝납니다.

위 예제에서는 'stdcall' 방식의 함수는 windll로, 'cdecl' 방식의 함수는 cdll을 통해 호출하는 것을 보여줍니다. 이제는 직접 윈도우에 들어 있는 kernel32.dll 안의 함수를 호출해 보겠습니다.

|  |  |  |
| --- | --- | --- |
| 1  2 | >>> print(hex(ctypes.windll.kernel32.GetModuleHandleA(None)))  0x1d000000  [*Colored by Color Scripter*](http://colorscripter.com/info#e) | [cs](http://colorscripter.com/info#e) |

**참조 사항**

stdcall와 cdecl은 함수 호출 규약이며, 함수가 호출될 때 매개변수가 스택에서 어떻게 전달되는지에 따라 stdcall과 cdecl로 나눌 수 있습니다.

stdcall은 불려진 함수가 스택에서 인자를 꺼내는(pop) 것이고, cdecl은 호출한 함수에서 스택의 인자를 꺼냅니다.

[Python 자료형] 확장 모듈과 라이브러리

파이썬 ctypes는 외부 함수 호출 말고도 C의 데이터 타입을 쓸 수가 있습니다. 아래는 기본적인 ctypes 함수가 지원하는 타입니다.

|  |  |  |
| --- | --- | --- |
| **ctypes 타입** | **C 타입** | **Python 타입** |
| c\_short | Short | int |
| c\_ushort | unsigned short | int |
| c\_int | Int | int |
| c\_uint | unsigned int | int |
| c\_long | Long | int |
| c\_ulong | usigned long | int |
| c\_longlong | \_\_int64 혹은 long long | int |
| c\_ulonglong | unsigned \_\_int64 혹은   unsigned long long | int |
| c\_float | Float | float |
| c\_double | Double | float |
| c\_longdouble | long double | float |
| c\_char\_p | char \* (NULL terminated) | unicode 혹은 None |
| c\_wchar\_p | wchar\_t \* (NULL terminated) | unicode 혹은 None |

이를 이용해 간단한 예제를 만들겠습니다.

|  |  |  |
| --- | --- | --- |
| 1  2  3  4  5 | >>> from ctypes import \*  >>> i = c\_int(10)    >>> print("ctypes : ", i ,", value : ", i.value )  ctypes : c\_long(10), value : 10 | [cs](http://colorscripter.com/info#e) |

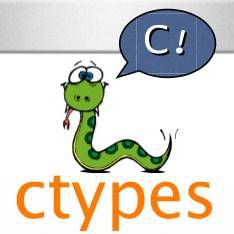
또한 다음 예제와 같이 파이썬 ctypes는 포인터도 지원합니다.

|  |  |  |
| --- | --- | --- |
| 1  2  3  4 | >>> k = c\_int(10)  >>> pk = pointer(k)     <- c\_int(10)의 포인터 값을 가져옵니다.  >>> print("position :", pointer(k), ", contents :", pk.contents)  position : <\_\_main\_\_.LP\_c\_long object at 0x018931C0> , contents : c\_\_long(10) | [cs](http://colorscripter.com/info#e) |

게다가 구조체와 유니온(union)은 다음과 같이 표현할 수 있습니다.

|  |  |  |
| --- | --- | --- |
| 1  2  3  4  5  6  7  8  9  10 | >>> import ctypes  >>> class POINT(ctypes.Structure):      \_fields\_ = [("X", ctypes.c\_int), ("y", ctypes.c\_int)]  >>> point = POINT(10, 20)  >>> print(point.x, point.y)  10 20    >>> point = POINT(y=5)  >>> print(point.x, point.y)  0 5  [*Colored by Color Scripter*](http://colorscripter.com/info#e) | [cs](http://colorscripter.com/info#e) |

구조체와 유니온은 파이선에서 class로 표현되기 때문에 일단 class 객체를 생성하고, \_fields\_ 멤버 변수를 이용해 각 멤버 변수를 설정해주는 것입니다.

[Python 자료형] 확장 모듈과 라이브러리

다음 예제에서는 실제로 파이썬 ctypes을 어떻게 사용하는지 보겠습니다. 우선 윈도우에는 사용자에게 메시지 박스를 보여주는 MessageBoxA()라는 함수가 정의돼 있습니다. winuser.h 헤더파일에 정의돼 있는 MessageBoxA는 다음과 같습니다.

|  |  |  |
| --- | --- | --- |
| 1  2  3  4  5  6 | WINUSERAPI int WINAPI  MessageBoxA(      HWND hWnd,      LPCSTR lpText,      LPCSTR lpCaption,      UINT uType); | [cs](http://colorscripter.com/info#e) |

ctypes 모듈을 이용해 이를 래핑(wrapping)해 보겠습니다.

|  |  |  |
| --- | --- | --- |
| 1  2  3  4  5 | >>> from ctypes import c\_int, WINFUNCTYPE, windll  >>> from ctypes.wintypes import HWND, LPCSTR, UINT  >>> prototype = WINFUNCTYPE(c\_int, HWND, LPCSTR, LPCSTR, UINT)  >>> paramflags = (1, "hwnd", 0), (1, "text", "Hi"), (1, "caption", None), (1, "flags", 0)  >>> MessageBox = prototype(("MessageBoxA", windll.user32), paramflags) | [cs](http://colorscripter.com/info#e) |

이제 Python에서 MessageBox는 윈도우의 MessageBoxA() 함수를 가리킵니다. MessageBox를 아래와 같이 호출해 보겠습니다.

|  |  |  |
| --- | --- | --- |
| 1  2 | >>> MessageBox(text="Please, programming with python")  >>> MessageBox(text="Do you know about python?", flags=3) | [cs](http://colorscripter.com/info#e) |

그리하면 윈도우의 메시지 창이 나타나는 것을 볼 수 있습니다.