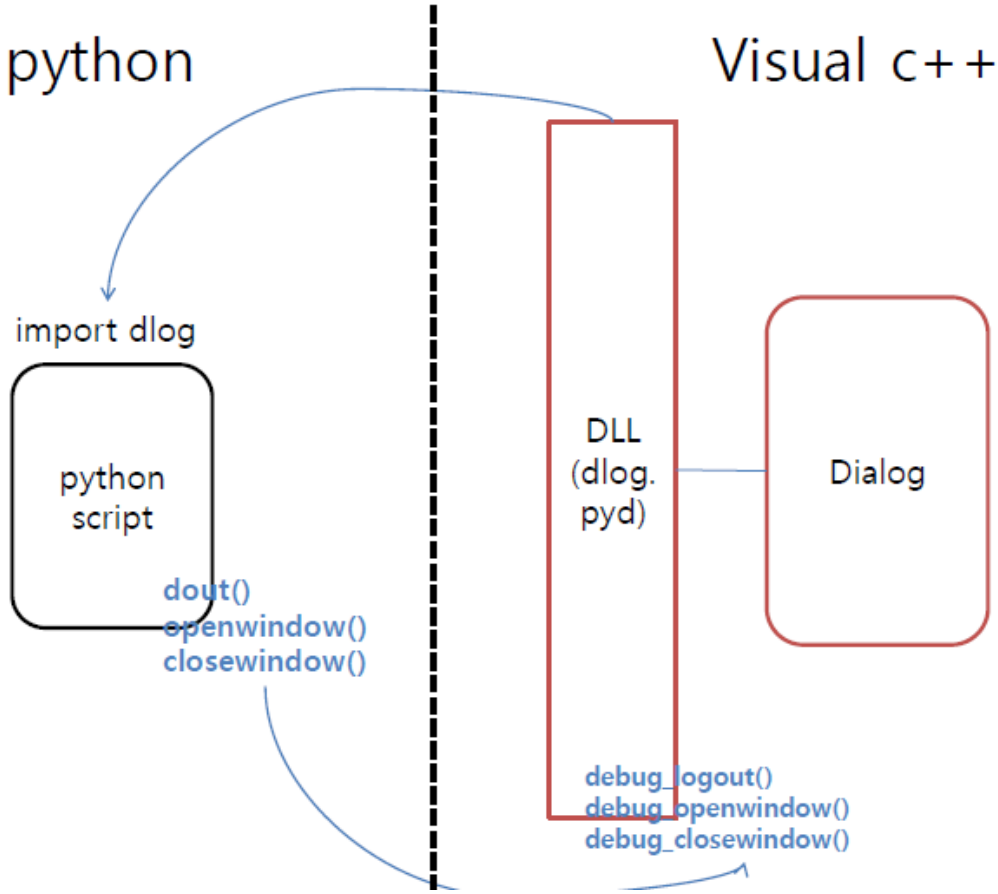
1. 목적

파이썬에서 console로 로그를 출력하다 보면 나중에 캡쳐하기가 불편해진다. 그러한 이유로 윈도우 형태의 GUI가 필요할 때가 있다.



Windows에서 파이썬 프로그램을 개발할 때, 로그 출력을 위한 GUI 부분을 Visual C++로 쉽게 만들어 사용하고자 하는 것이 목적이다. wxPython이나 기타 GUI 라이브러리들이 있으나 자신이 원하는 형태로 만드는 데까지는 학습이 필요하므로 빠르게 GUI를 적용할 경우는 위와 같은 방법도 효율적이다.

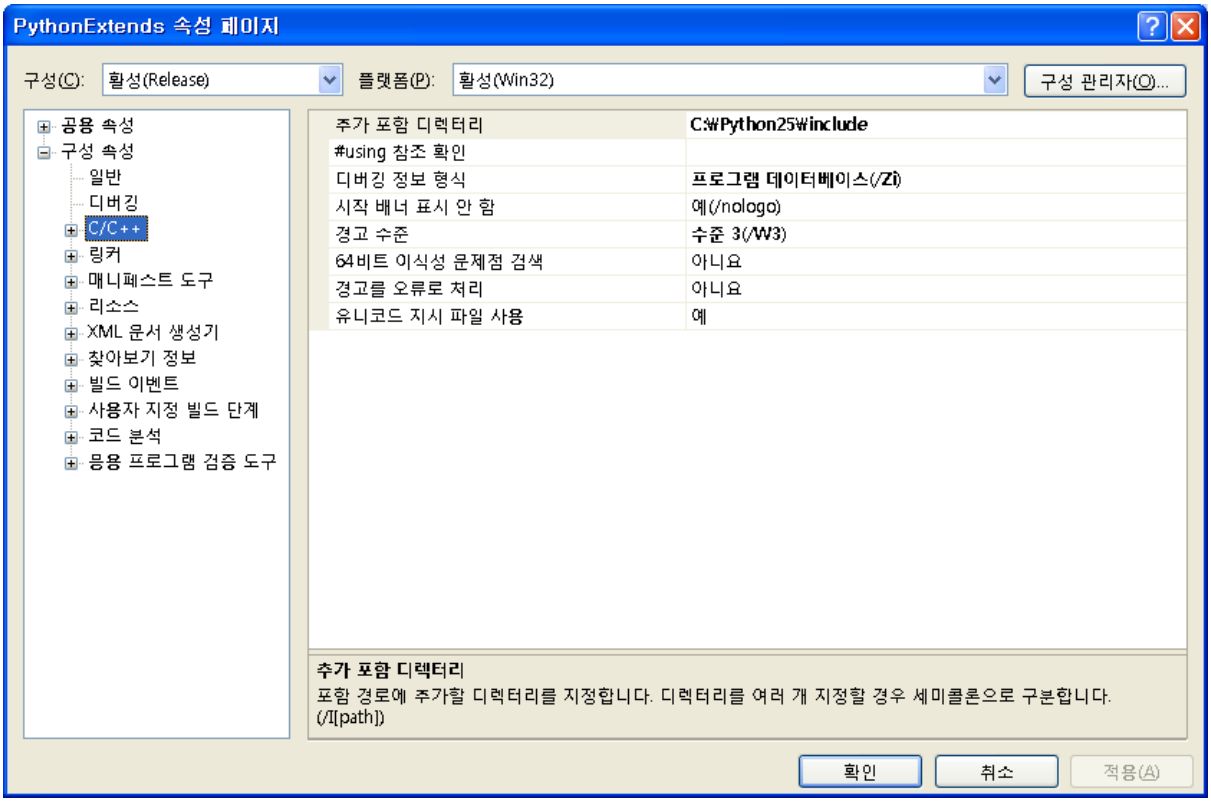
2. 개발환경

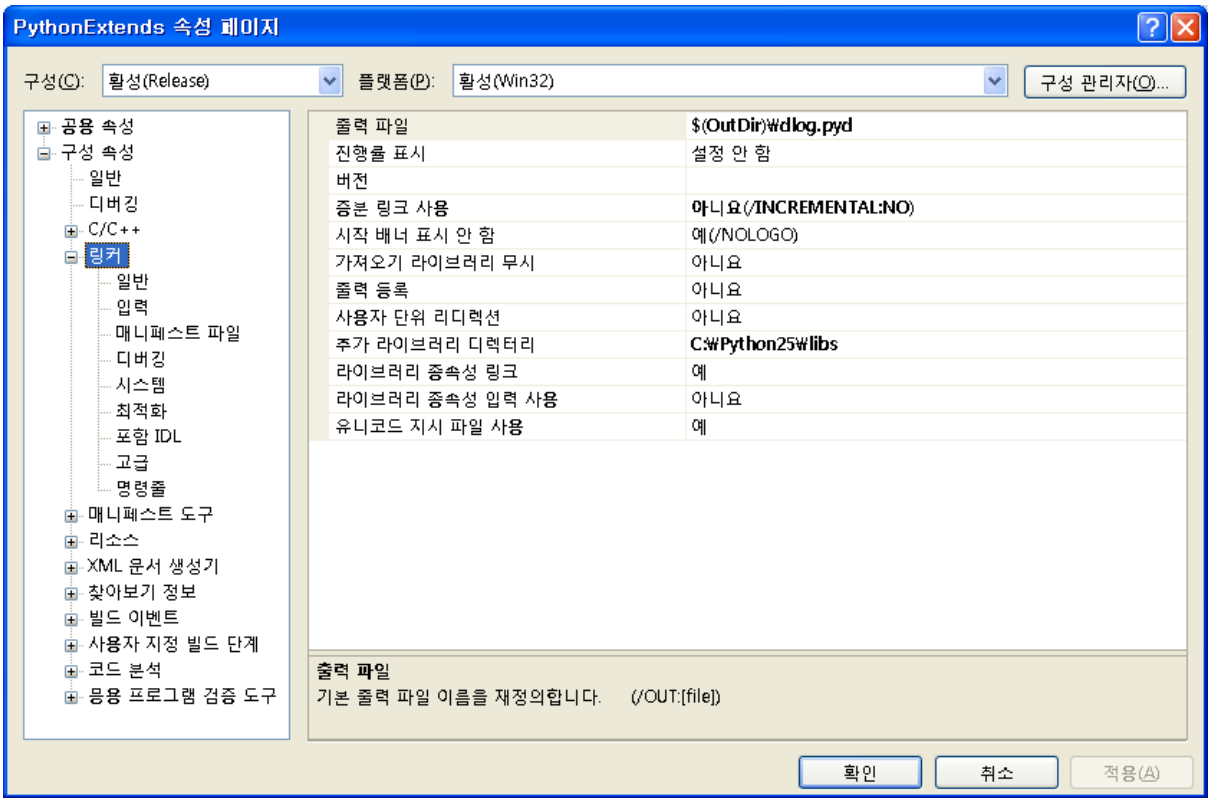
- Python 2.5
- Visual Studio 2008

3. Visual C 2008에서 python 사용하기

Python 2.5를 기준으로 프로젝트를 구성했다.

- MFC DLL 프로젝트를 생성한다.
- include 폴더와 library 폴더를 다음과 같이 추가해주어야 한다.





(*) 개발하는 PC에 설치된 python의 경로를 입력해야 한다.

- #include <python.h>를 소스에 적용한다.
- 프로젝트는 release로 한다(debug시에는 python_d.lib이 필요함)

- 출력파일은 $(OutDir)Wdlog.pyd와 같이 한다(import하는 모듈명과 동일해야 함).

4. Python extend 함수 구현

- 모듈초기화 함수

DLL(.pyd)의 파일명과 동일한 함수가 초기화 함수다. 이곳에서는 python에서 사용할 함수테이블을 등록하는 기능을 수행한다. 네이밍룰은 “init + 모듈명”이다.

```
// Python 함수초기화
PyMODINIT_FUNC initdlog()
{
    Py_InitModule3("dlog", DebugMethods, "Debug log module");
    hWait = CreateEvent(NULL, TRUE, FALSE, NULL);
}
```

- Function 등록

Py_InitModule3() 함수는 첫번째 파라메터는 모듈명이다. 그리고 두번째 파라메터는 사용할 함수들을 정의한 function table이다.

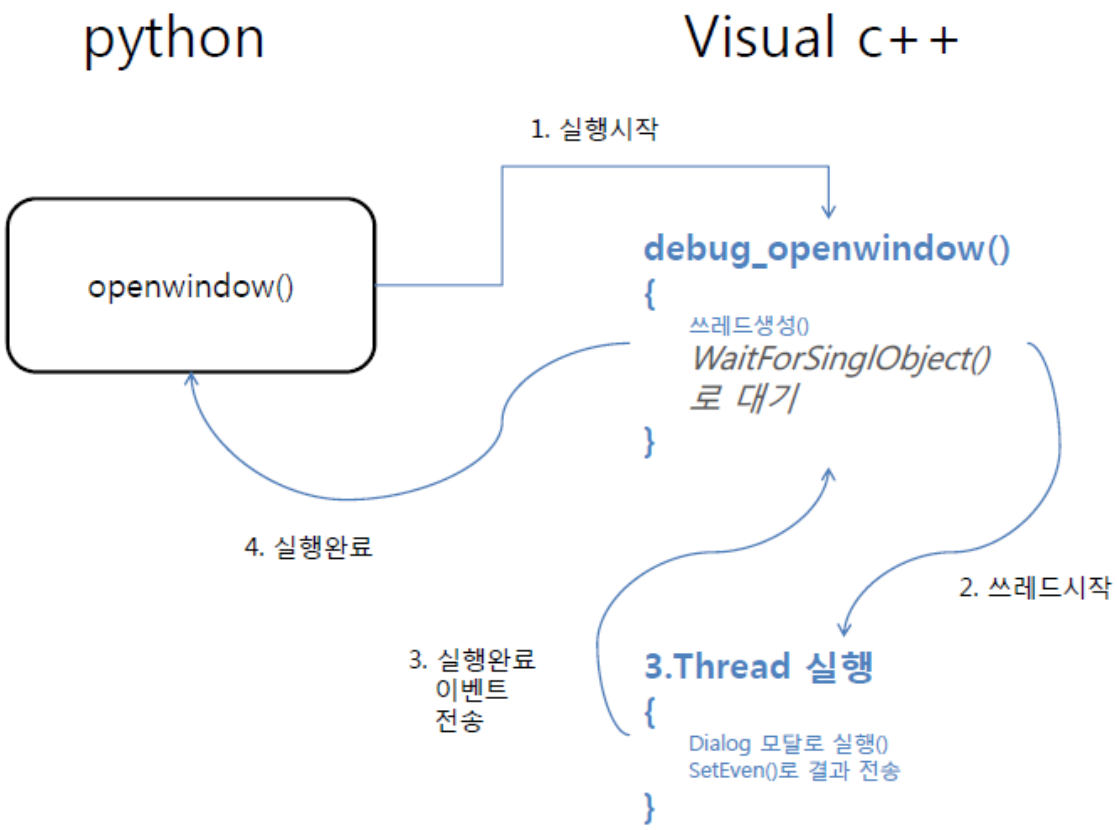
```
static PyMethodDef DebugMethods[] = {
    {"dout",        debug_logout,      METH_VARARGS,"write message"},
    {"openwindow",  debug_openwindow,  METH_VARARGS,"open  window"},
    {"closewindow", debug_closewindow, METH_VARARGS,"close window"},
    {NULL, NULL, 0, NULL}        /* Sentinel */
};
```

- openwindow

Windows 프로그램의 특성상 Form은 DoModal()이 실행되는 순간, 같은 문맥(context)에서는 Form이 종료되지 않는 한, 다음코드들이 진행될 수 없다. 그러므로 “폼을 제어하는 함수”를 다른 문맥에서 처리하기 위해서 다음과 같은 방법을 사용한다.

목적)
openwinow() 함수를 block 방식으로 구현한다. 그러나 python에서는 무한대기하지 않고 Dialog 생성 후에도 Dialog의 메소드를 호출하여 사용할 수 있도록 한다.

- 모듈 초기화 때(initdlog)에 Event를 하나 등록한다.
- openwindow() 함수 호출하면 thread 생성하고 WaitForSingleObject()로 대기한다.
- Thread에서 Dialog에 이벤트 핸들을 넘기고 DoModal()로 화면에 보인다.
- Dialog에서 initDialog 이벤트에서 실행 결과를 SetEvent()로 알려준다.
- openwindow() 함수에서 대기가 풀린다.
- Dialog가 화면에 나타났고 Dialog에 관련 메소드를 사용할 수 있다.



- closewindow

다른 문맥에서 Form을 종료해야 하므로 다음과 같이 종료 메시지를 보낸다.

```
SendMessage(hWnd, WM_CLOSE,0, 0);
```

참고로 종료메시지가 처리되는 순간 Python이 종료되면 윈도우에서 abnormal terminated 에러가 발생하므로 python에서는 closewindow()를 실행 후, 몇 초간 대기해주는 코드를 작성하는 것이 좋다.

5. Source

(pyextends.cpp)

```
// PythonExtends.cpp : 해당DLL의초기화루틴을정의합니다.
//

#include "stdafx.h"
#include "PythonExtends.h"

#ifdef _DEBUG
#define new DEBUG_NEW
#endif

#include <Python.h>
```

```
#include "display.h"

// CPythonExtendsApp
BEGIN_MESSAGE_MAP(CPythonExtendsApp, CWinApp)
END_MESSAGE_MAP()

// CPythonExtendsApp 생성
CDisplay m_dlg;
CPythonExtendsApp::CPythonExtendsApp()
{
}


// 유일한CPythonExtendsApp 개체입니다.
CPythonExtendsApp theApp;

// CPythonExtendsApp 초기화
BOOL CPythonExtendsApp::InitInstance()
{
    //TODO: call AfxInitRichEdit2() to initialize richedit2 library.
        AfxInitRichEdit2();

        CWinApp::InitInstance();
    return TRUE;
}

static PyObject * debug_logout(PyObject *self, PyObject *args);
static PyObject * debug_openwindow(PyObject *self, PyObject *args);
static PyObject * debug_closewindow(PyObject *self, PyObject *args);
static PyMethodDef DebugMethods[] = {
    {"dout",        debug_logout,      METH_VARARGS,"write message"},
    {"openwindow",  debug_openwindow,  METH_VARARGS,"open  window"},
        {"closewindow", debug_closewindow, METH_VARARGS,"close window"},

    {NULL, NULL, 0, NULL}        /* Sentinel */
};


// WaitForSingleObject() 대기하기위한핸들
static HANDLE hWait = NULL;

void DebugOut(char* str, int spec)
{
        HWND hWnd = FindWindow(NULL, L"python debug view");

        if( hWnd != NULL)
        {
                const int MAX_SIZE = 1024 * 5;
                TCHAR szSend [MAX_SIZE];
        ZeroMemory(szSend, MAX_SIZE);

                MultiByteToWideChar(CP_ACP, MB_PRECOMPOSED, str, strlen(str), szSend,
```

```
MAX_SIZE);
                m_dlg.WriteMessage(szSend, spec);
        }
}

UINT ThreadFunc(LPVOID lpParam)
{
        m_dlg.SetHandle(hWait);
    m_dlg.DoModal();
        return 0L;
}

static PyObject* debug_openwindow(PyObject *self, PyObject *args)
{
    if (!PyArg_ParseTuple(args, ""))
        return NULL;

        HWND hWnd = FindWindow(NULL, L"python debug view");
        if( hWnd != NULL) return Py_BuildValue("i", 1);

        AfxBeginThread(ThreadFunc, NULL);
        WaitForSingleObject(hWait, INFINITE);

        return Py_BuildValue("i", 0);
}

static PyObject* debug_logout(PyObject *self, PyObject *args)
{
    char *message;
        int  spec;
    if (!PyArg_ParseTuple(args, "si", &message, &spec))
        return NULL;

        DebugOut(message, spec);

        return Py_BuildValue("i", 0);
}

static PyObject * debug_closewindow(PyObject *self, PyObject *args)
{
    if (!PyArg_ParseTuple(args, ""))
        return NULL;

        HWND hWnd = FindWindow(NULL, L"python debug view");
        if( hWnd == NULL) return Py_BuildValue("i", 1);

        SendMessage(hWnd, WM_CLOSE,0, 0);

        return Py_BuildValue("i", 0);

}
```

```
// Python 함수초기화
PyMODINIT_FUNC initdlog()
{
    Py_InitModule3("dlog", DebugMethods, "Debug log module");
        hWait = CreateEvent(NULL, TRUE, FALSE, NULL);
}
```

(display.cpp)

```
// Display.cpp : 구현파일입니다.
//

#include "stdafx.h"
#include "PythonExtends.h"
#include "Display.h"


// CDisplay 대화상자입니다.

IMPLEMENT_DYNAMIC(CDisplay, CDialog)

CDisplay::CDisplay(CWnd* pParent /*=NULL*/)
        : CDialog(CDisplay::IDD, pParent)
{

}

CDisplay::~CDisplay()
{
}

void CDisplay::DoDataExchange(CDataExchange* pDX)
{
        CDialog::DoDataExchange(pDX);
        DDX_Control(pDX, IDC_RICHEDIT21, m_display);
}


BEGIN_MESSAGE_MAP(CDisplay, CDialog)
END_MESSAGE_MAP()


BOOL CDisplay::OnInitDialog()
{
        CDialog::OnInitDialog();

        // 시스템메뉴에"정보..." 메뉴항목을추가합니다.

        // IDM_ABOUTBOX는시스템명령범위에있어야합니다.
        ASSERT((IDM_ABOUTBOX & 0xFFF0) == IDM_ABOUTBOX);
        ASSERT(IDM_ABOUTBOX < 0xF000);
```

```
	CMenu* pSysMenu = GetSystemMenu(FALSE);
	if (pSysMenu != NULL)
	{

	}

	// 이대화상자의아이콘을설정합니다. 응용프로그램의주창이대화상자가아닐경우에는
	//  프레임워크가이작업을자동으로수행합니다.
	SetIcon(m_hIcon, TRUE);			// 큰아이콘을설정합니다.
	SetIcon(m_hIcon, FALSE);		// 작은아이콘을설정합니다.

	// TODO: 여기에추가초기화작업을추가합니다.

	// 리치에디터배경설정
	m_display.SetBackgroundColor(FALSE, RGB(0xDF, 0xED, 0x7F));

	// 윈도우가실행된것을Python 호출함수에알려준다.
	SetEvent(hWait);

	return TRUE;  // 포커스를컨트롤에설정하지않으면TRUE를반환합니다.
}

void CDisplay::OnSysCommand(UINT nID, LPARAM lParam)
{
	CDialog::OnSysCommand(nID, lParam);
}

// 대화상자에최소화단추를추가할경우아이콘을그리려면
//  아래코드가필요합니다. 문서/뷰모델을사용하는MFC 응용프로그램의경우에는
//  프레임워크에서이작업을자동으로수행합니다.

void CDisplay::OnPaint()
{
	if (IsIconic())
	{
		CPaintDC dc(this); // 그리기를위한디바이스컨텍스트

		SendMessage(WM_ICONERASEBKGND, reinterpret_cast<WPARAM>(dc.GetSafeHdc()),
0);

		// 클라이언트사각형에서아이콘을가운데에맞춥니다.
		int cxIcon = GetSystemMetrics(SM_CXICON);
		int cyIcon = GetSystemMetrics(SM_CYICON);
		CRect rect;
		GetClientRect(&rect);
		int x = (rect.Width() - cxIcon + 1) / 2;
		int y = (rect.Height() - cyIcon + 1) / 2;

		// 아이콘을그립니다.
		dc.DrawIcon(x, y, m_hIcon);
	}
```

```
        else
        {
                CDialog::OnPaint();
        }
}

// 사용자가최소화된창을끄는동안에커서가표시되도록시스템에서
//  이함수를호출합니다.
HCURSOR CDisplay::OnQueryDragIcon()
{
        return static_cast<HCURSOR>(m_hIcon);
}

// 문자열출력
void CDisplay::WriteMessage(TCHAR* msg, int spec)
{

        COLORREF cr[] = {
                RGB(100, 100, 55),
                RGB(255, 55, 11),
                RGB(0, 111, 255),
                RGB(12, 24, 48),
                RGB(48, 24, 112),
                RGB(0, 71, 255),
                RGB(0, 255, 155)
        };

        spec = spec % sizeof(cr);

        CHARFORMAT ufFontNormal;
        memset(&ufFontNormal, 0, sizeof(CHARFORMAT));
        ufFontNormal.crTextColor = RGB(11, 22, 11);
        ufFontNormal.dwMask      = CFM_EFFECTS;
        ufFontNormal.dwEffects   = CFE_BOLD  | CFE_STRIKEOUT;
        ufFontNormal.yHeight     = 180;
        _tcscpy(ufFontNormal.szFaceName, L"굴림체");

        // 자동스크롤을위한편법
        m_display.HideSelection(FALSE,FALSE);

        // 맨마지막으로캐럿을옮기고
        CHARRANGE rng;
        rng.cpMax = m_display.GetTextLength();
        rng.cpMin = rng.cpMax;
        m_display.SetSel( rng );
        m_display.SetSelectionCharFormat(ufFontNormal);

        // 라인넘버치환
        CString str =L"";
        str.Format(L"%04d:", nLineNum);
        m_display.ReplaceSel( str.GetBuffer() );
```

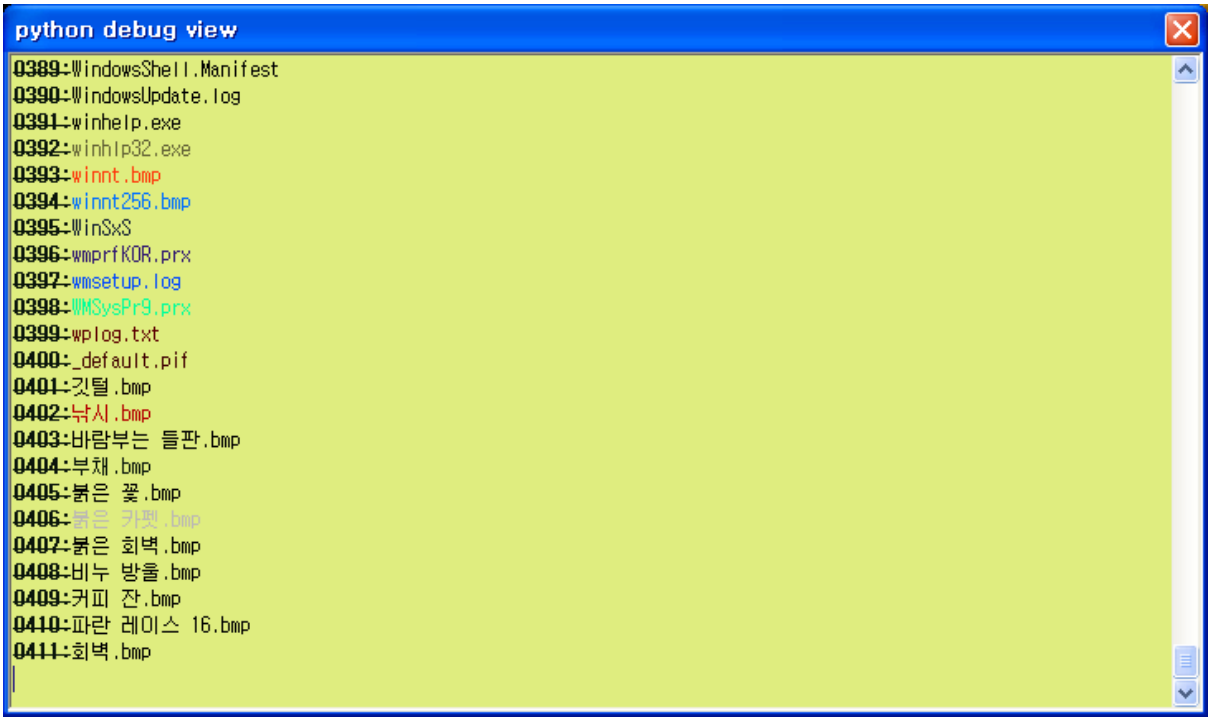
```
        // 자동스크롤을위한편법
        m_display.HideSelection(FALSE,FALSE);

        // 맨마지막으로캐럿을옮기고
        ufFontNormal.crTextColor = cr[spec];
        ufFontNormal.dwEffects   = 0;
        rng.cpMax = m_display.GetTextLength();
        rng.cpMin = rng.cpMax;
        m_display.SetSel( rng );
        m_display.SetSelectionCharFormat(ufFontNormal);

        m_display.ReplaceSel( msg );
        m_display.ReplaceSel( L"WrWn" );
        nLineNum++;

}
```

6. python에서 테스트

test.py를 다음과 같이 작성한다.

```
# -*- coding: cp949 -*-
import dlog
import os

dlog.openwindow()
dlog.openwindow()

findlist = os.listdir("c:WWwindowsWW")
i = 0
for f in findlist:
    dlog.dout(f, i)
    i = i + 1

raw_input("아무키나 누르세요 - 윈도우 종료")
dlog.closewindow()
raw_input("아무키나 누르세요 - 윈도우 시작")

dlog.openwindow()
dlog.openwindow()

findlist = os.listdir("c:WWwindowsWW")
i = 0
for f in findlist:
    dlog.dout(f, i)
    i = i + 1
```

dlog.pyd를 같은 폴더에 넣는다.
탐색기나 console 창에서 test.py를 실행시킨다.


python debug view
0389:WindowsShell.Manifest
0390:WindowsUpdate.log
0391:winhelp.exe
0392:winhlp32.exe
0393:winnt.bmp
0394:winnt256.bmp
0395:WinSxS
0396:wmprfKOR.prx
0397:wmsetup.log
0398:WMSysPr9.prx
0399:wplog.txt
0400:_default.pif
0401:깃털.bmp
0402:낚시.bmp
0403:바람부는 들판.bmp
0404:부채.bmp
0405:붉은 꽃.bmp
0406:붉은 카펫.bmp
0407:붉은 회벽.bmp
0408:비누 방울.bmp
0409:커피 잔.bmp
0410:파란 레이스 16.bmp
0411:회벽.bmp