DLL

test.cpp

```
////////////////////////////
// test.cpp
//
// A Simple DLL written using C/C++
//
// At the Visual studio command prompt
//
// cl /c test.cpp
//
// link /DLL /out:test.dll /DEF:test.def test.obj
//

#include <stdio.h>

extern "C" __declspec(dllexport) int __stdcall Add(int a , int b ) {

                return a + b;
}
```

Test.def

```
LIBRARY test
EXPORTS
                Add
```

Caller.cpp

```
//////////////////////////////////
// caller.cpp
//
// cl caller.cpp test.lib
//
//
//

#include <stdio.h>

extern "C" int __stdcall Add( int , int );

int main( int argc , char **argv )
{
  printf("The value is %d\n",Add(2,3));

}
```

Visual Studio 2008 Command Prompt

```
c:\PyCon2010\DLL>cl /c test.cpp
Microsoft (R) 32-bit C/C++ Optimizing Compiler Version 15.00.21022.08 for 80x86
Copyright (C) Microsoft Corporation.  All rights reserved.

test.cpp

c:\PyCon2010\DLL>link /DLL /out:test.dll /DEF:test.def test.obj
Microsoft (R) Incremental Linker Version 9.00.21022.08
Copyright (C) Microsoft Corporation.  All rights reserved.

   Creating library test.lib and object test.exp

c:\PyCon2010\DLL>dir test.dll
 Volume in drive C is OS
 Volume Serial Number is 3A5F-6B42

 Directory of c:\PyCon2010\DLL

29-08-2010  10:15            40,960 test.dll
               1 File(s)         40,960 bytes
               0 Dir(s)  27,035,328,512 bytes free

c:\PyCon2010\DLL>_
```

Visual Studio 2008 Command Prompt

```
c:\PyCon2010\DLL>cl caller.cpp test.lib
Microsoft (R) 32-bit C/C++ Optimizing Compiler Version 15.00.21022.08 for 80x86
Copyright (C) Microsoft Corporation.  All rights reserved.

caller.cpp
Microsoft (R) Incremental Linker Version 9.00.21022.08
Copyright (C) Microsoft Corporation.  All rights reserved.

/out:caller.exe
caller.obj
test.lib

c:\PyCon2010\DLL>caller
The value is 5

c:\PyCon2010\DLL>_
```

```
//////////////////////////////
//dyncaller.cpp
//
//
//Call a DLL at the run time
//
//
//
```

```
#include <stdio.h>
#include <windows.h>

//////////////////
//
// typedef for Binary Function ( Add(int , int ) )
//
//

typedef int (__stdcall * BinaryFunc ) ( int , int );


int main( int argc , char **argv )
{

  HMODULE h = LoadLibrary("test.dll");


  if ( h == INVALID_HANDLE_VALUE ) {

            fprintf(stdout,"Failed to load the DLL\n");
            return -1;
  }


  BinaryFunc bfn = (BinaryFunc)GetProcAddress(h,"Add");

  if ( bfn == 0 ) {

     fprintf(stdout,"Failed to retrive func pointer\n");
     return -2;
  }

  printf("The value is %d\n",(*bfn)(2,3));

  FreeLibrary(h);

  return 0;

}
```

```
Visual Studio 2008 Command Prompt

c:\PyCon2010\DLL>cl dyncaller.cpp
Microsoft (R) 32-bit C/C++ Optimizing Compiler Version 15.00.21022.08 for 80x86
Copyright (C) Microsoft Corporation.  All rights reserved.

dyncaller.cpp
Microsoft (R) Incremental Linker Version 9.00.21022.08
Copyright (C) Microsoft Corporation.  All rights reserved.

/out:dyncaller.exe
dyncaller.obj

c:\PyCon2010\DLL>dyncaller
The value is 5

c:\PyCon2010\DLL>
```

HelloWorld Extension (PyExt)

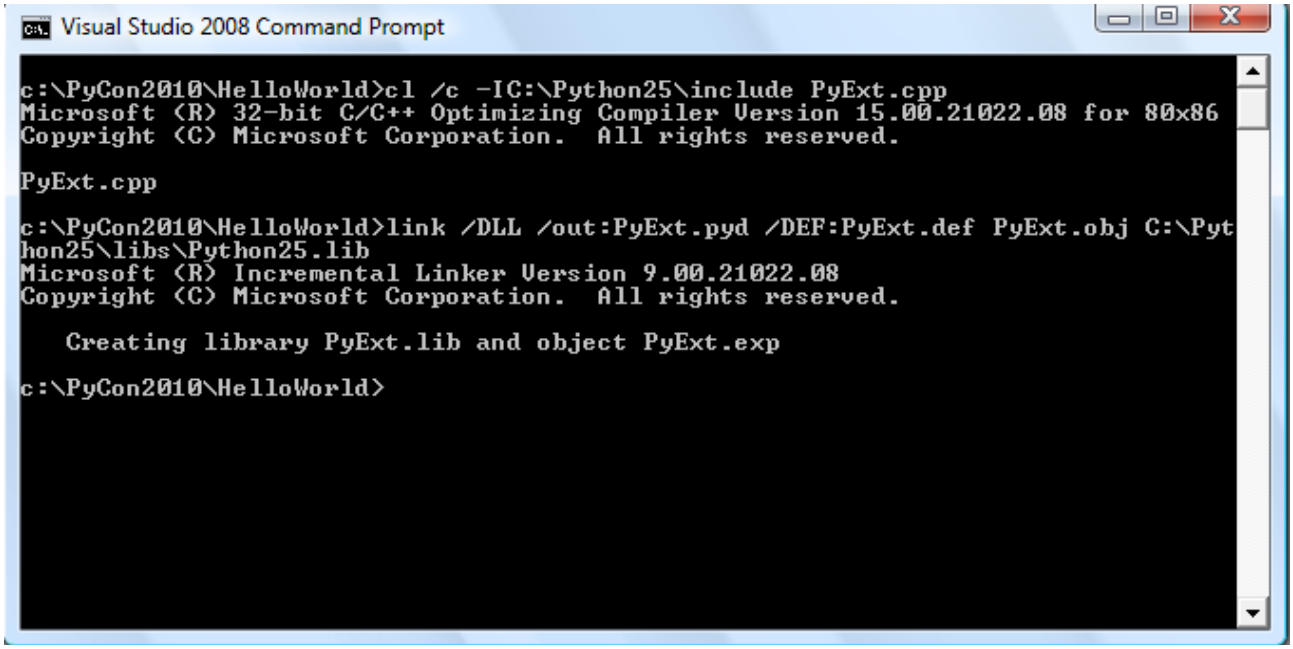
```
//////////////////////////////
//
// PyExt.cpp
//
// A Simple Python Extension under Visual C/C++
//
// Written by Praseed Pai K.T.
//          http://praseedp.blogspot.com
//
//
// At the Visual studio Command prompt
//
// For Python 3.1  => We need to define _PYTHON_3X
// ----------------------------------------------
// The Python path for my machine is C:\Python31
//
// cl /c -IC:\Python31\include /D_PYTHON_3X  PyExt.cpp
//
// link /DLL /out:PyExt.pyd PyExt.obj C:\Python31\libs\Python31.lib
//
// For Python 2.5
// ----------------------------------------------
// The Python path for my machine is C:\Python25
//
// cl /c -IC:\Python25\include  PyExt.cpp
//
```

```
// link /DLL /out:PyExt.pyd PyExt.obj C:\Python25\libs\Python25.lib
//
//

#include <Python.h>
#include <stdlib.h>

//////////////////////////////////
//
// This is the actual routine which returns the string "Hello World.."
//
//
//

static PyObject * SayHello( PyObject *self , PyObject *args )
{
                PyObject *return_value = 0;
                return_value = Py_BuildValue("s","Hello World....");
                return return_value;

}


////////////////////////////////////////
//
// Initialize the PyMethodDef table
//
//
static struct PyMethodDef pyext_methods[] = {
     {"SayHello",SayHello,METH_NOARGS,0},
     { 0 , 0 }

};


#ifdef _PYTHON_3X

///////////////////////////////////
//
// Initialize the table
//
//

static struct PyModuleDef PyExtModule = {
  PyModuleDef_HEAD_INIT,
  "PyExt",   /* name of module */
   0, /* module documentation, may be NULL */
  -1,       /* size of per-interpreter state of the module,
               or -1 if the module keeps state in global variables. */
  pyext_methods
};
```

```
//////////////////////////////////////
//
// Python Interpreter initializes the module by
// calling the routine given below
//
//
PyMODINIT_FUNC  PyInit_PyExt(void) {

 return  PyModule_Create(&PyExtModule);
}


#else

extern "C" __declspec(dllexport) void __stdcall initPyExt() {

   Py_InitModule("PyExt",pyext_methods);
}

#endif
```

PyExt.def

```
LIBRARY PyExt.pyd

EXPORTS
                initPyExt
```

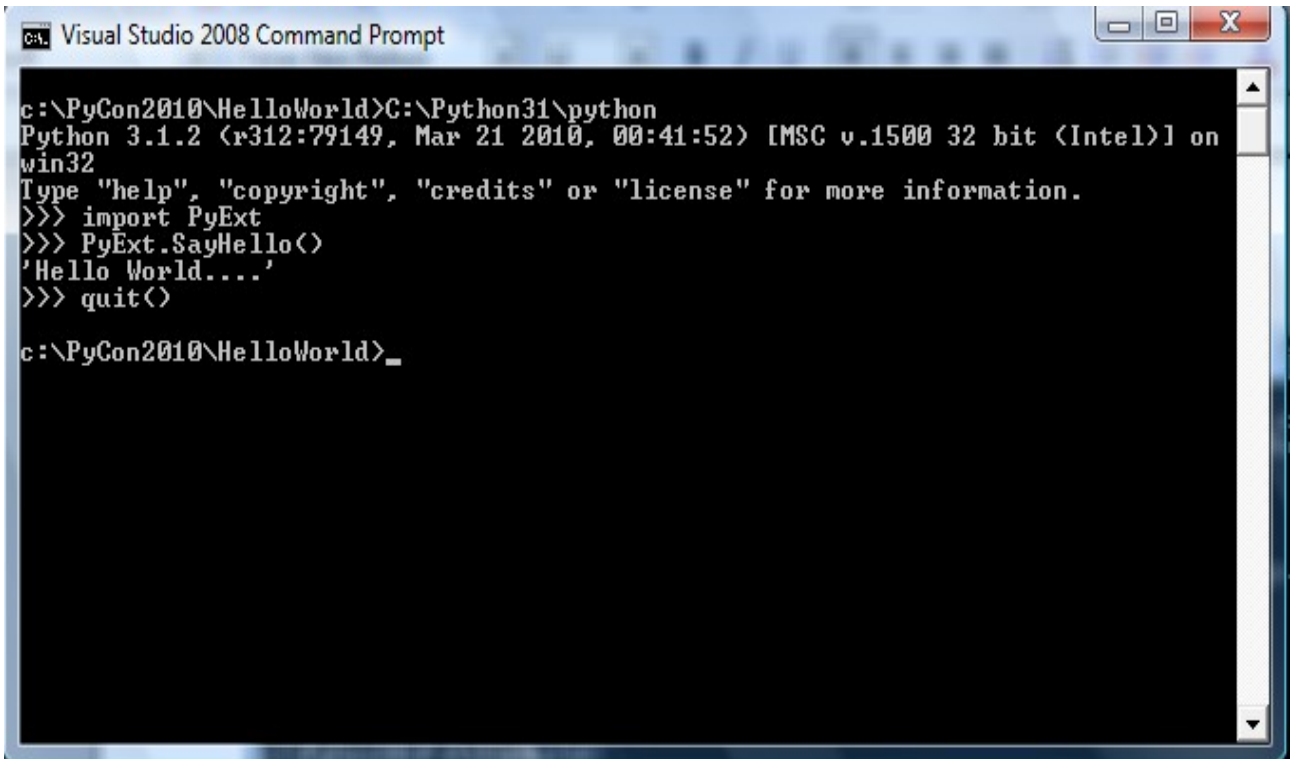


```
Visual Studio 2008 Command Prompt

c:\PyCon2010\HelloWorld>C:\Python25\python
Python 2.5.1 (r251:54863, Apr 18 2007, 08:51:08) [MSC v.1310 32 bit (Intel)] on
win32
Type "help", "copyright", "credits" or "license" for more information.
>>> import PyExt
>>> PyExt.SayHello()
'Hello World....'
>>> quit()

c:\PyCon2010\HelloWorld>
```

Now we can compile the stuff for Python 3.x

```
Visual Studio 2008 Command Prompt

c:\PyCon2010\HelloWorld>cl /c -IC:\Python31\include /D_PYTHON_3X PyExt.cpp
Microsoft (R) 32-bit C/C++ Optimizing Compiler Version 15.00.21022.08 for 80x86
Copyright (C) Microsoft Corporation.  All rights reserved.

PyExt.cpp

c:\PyCon2010\HelloWorld>link /DLL /out:PyExt.pyd  PyExt.obj C:\Python31\libs\Pyt
hon31.lib
Microsoft (R) Incremental Linker Version 9.00.21022.08
Copyright (C) Microsoft Corporation.  All rights reserved.

   Creating library PyExt.lib and object PyExt.exp

c:\PyCon2010\HelloWorld>
```

Load it in Python 3.1.2



ArrayExt.pyd

```
//////////////////////////////
//
// Array.cpp
//
// A Simple Python Extension under Visual C/C++
//
// Written by Praseed Pai K.T.
//          http://praseedp.blogspot.com
//
//
// At the Visual studio Command prompt
//
// for Python 3.x
// ----------------
//
// The Python path for my machine is C:\Python31
//
// cl /c -IC:\Python31\include  Array.cpp
//
// link /DLL /out:ArrayExt.pyd Array.obj H:\Python31\libs\Python31.lib
//
// for Python 2.x
// ----------------
//
// The Python path for my machine is C:\Python25
//
// cl /c -IC:\Python25\include  Array.cpp
```

```
//
// link /DLL /out:ArrayExt.pyd Array.obj /DEF:ArrayExt.def
//      C:\Python31\libs\Python31.lib
//

#include <Python.h>
#include <stdlib.h>

//////////////////////////////////
//
// This is the actual routine which returns the string "Hello World.."
//
//
//

static PyObject * SQuare( PyObject *self , PyObject *args )
{

        double cnt=0;
        if ( !PyArg_ParseTuple(args,"d",&cnt) ) {
                        PyErr_SetString(PyExc_RuntimeError,
                    "double argument expected .." );
                return NULL;
        }

                PyObject *return_value = 0;
                return_value = Py_BuildValue("d",cnt*cnt);
                return return_value;

}
//////////////////////////////////////////////
//
//
//
//

static PyObject * IsSolutionForQuad( PyObject *self , PyObject *args )
{

        double a , b , c ;
        a=b=c=0;

        PyArg_ParseTuple(args,"ddd",&a,&b,&c);

        double disc = b*b - 4*a*c;

        if ( disc < 0.0 ) {
                                return Py_BuildValue("i",0);
        }

        double x1 = ( -b + sqrt(disc) ) / (2*a);
        double x2 = ( -b - sqrt(disc) ) / (2*a);
```

```
                PyObject *return_value = 0;
                return_value = Py_BuildValue("dd",x1,x2);
                return return_value;

}

//////////////////////////////////
//
//
//
//
//
static PyObject *SumList( PyObject *self , PyObject *args )
{

  PyObject* myTuple;
  if(!PyArg_ParseTuple(args,"O",&myTuple))
          return NULL;


  if ( !PyList_Check(myTuple) ) {
       PyErr_SetString(PyExc_RuntimeError,
              "List argument expected .." );
       return NULL;

   }

  int length = (int)PyList_Size(myTuple);
  double sum = 0;
  double dbl = 0;
  for(int i=0; i < length; ++i )
  {
     PyObject *next = PyList_GetItem(myTuple,i);

     if ( PyFloat_Check(next) ) {
        dbl = PyFloat_AsDouble(next);
     }
     sum += dbl;
  }

  return Py_BuildValue("d",sum);

}

//////////////////////////////////
//
//
//
//
//
```

```
static PyObject *SumTuple( PyObject *self , PyObject *args )
{

  PyObject* myTuple;
  if(!PyArg_ParseTuple(args,"O",&myTuple))
          return NULL;


  if ( !PyTuple_Check(myTuple) ) {

       PyErr_SetString(PyExc_RuntimeError,
              "Tuple argument expected .." );
       return NULL;

   }

  int length = (int)PyTuple_Size(myTuple);
  double sum = 0;
  double dbl = 0;
  for(int i=0; i < length; ++i )
  {
     PyObject *next = PyTuple_GetItem(myTuple,i);

     if ( PyFloat_Check(next) ) {
        dbl = PyFloat_AsDouble(next);

     }

     sum += dbl;
  }

  return Py_BuildValue("d",sum);

}

//////////////////////////////////////////
//
// Initialize the PyMethodDef table
//
//
static struct PyMethodDef pyext_methods[] = {
     {"SQuare",SQuare,METH_VARARGS,0},
     {"IsSolutionForQuad",IsSolutionForQuad,METH_VARARGS,0},
     {"SumList",SumList,METH_VARARGS,0},
     {"SumTuple",SumTuple,METH_VARARGS,0},
     { 0 , 0 }

};


#ifdef _PYTHON_3X
```

```
//////////////////////////////////
//
// Initialize the table
//
//

static struct PyModuleDef PyExtModule = {
   PyModuleDef_HEAD_INIT,
   "ArrayExt",   /* name of module */
    0, /* module documentation, may be NULL */
   -1,       /* size of per-interpreter state of the module,
                or -1 if the module keeps state in global variables. */
   pyext_methods
};

//////////////////////////////////////
//
// Python Interpreter initializes the module by
// calling the routine given below
//
//
PyMODINIT_FUNC  PyInit_ArrayExt(void) {

 return  PyModule_Create(&PyExtModule);
}


#else
extern "C"
 __declspec(dllexport) void __stdcall initArrayExt()
{

    Py_InitModule("ArrayExt",pyext_methods);

}



#endif
```

ArrayExt.def

```
LIBRARY ArrayExt.pyd

EXPORTS
       initArrayExt
```

Visual Studio 2008 Command Prompt - C:\Python25\python

```
c:\PyCon2010\Other>cl /c -IC:\Python25\include Array.cpp
Microsoft (R) 32-bit C/C++ Optimizing Compiler Version 15.00.21022.08 for 80x86
Copyright (C) Microsoft Corporation.  All rights reserved.

Array.cpp

c:\PyCon2010\Other>link /DLL /out:ArrayExt.pyd Array.obj /DEF:ArrayExt.def C:\Py
thon25\libs\Python25.lib
Microsoft (R) Incremental Linker Version 9.00.21022.08
Copyright (C) Microsoft Corporation.  All rights reserved.

   Creating library ArrayExt.lib and object ArrayExt.exp

c:\PyCon2010\Other>C:\Python25\python
Python 2.5.1 (r251:54863, Apr 18 2007, 08:51:08) [MSC v.1310 32 bit (Intel)] on
win32
Type "help", "copyright", "credits" or "license" for more information.
>>> import ArrayExt
>>> ArrayExt.SumList([10.0,20.0,30.0])
60.0
>>> ArrayExt.SumTuple((10.0,20.0,30.0))
60.0
>>> _
```

Visual Studio 2008 Command Prompt

```
c:\PyCon2010\Other>cl /c -IC:\Python31\include /D_PYTHON_3X Array.cpp
Microsoft (R) 32-bit C/C++ Optimizing Compiler Version 15.00.21022.08 for 80x86
Copyright (C) Microsoft Corporation.  All rights reserved.

Array.cpp

c:\PyCon2010\Other>link /DLL /out:ArrayExt.pyd Array.obj  C:\Python31\libs\Pytho
n31.lib
Microsoft (R) Incremental Linker Version 9.00.21022.08
Copyright (C) Microsoft Corporation.  All rights reserved.

   Creating library ArrayExt.lib and object ArrayExt.exp

c:\PyCon2010\Other>C:\Python31\python
Python 3.1.2 (r312:79149, Mar 21 2010, 00:41:52) [MSC v.1500 32 bit (Intel)] on
win32
Type "help", "copyright", "credits" or "license" for more information.
>>> import ArrayExt
>>> ArrayExt.SumList([10.0,20.0])
30.0
>>> ArrayExt.SQuare(20)
400.0
>>> quit()
```

A Custom Type in C/C++ for Python

```
//////////////////////////////////////////////
//
// Writing a Custom Type for Consumption from
// Python
//
//
// Written By Praseed Pai K.T.
//          http://praseedp.blogspot.com
//
//
// for Pyton 3.x
// --------------
// cl /c /D_PYTHON_3X -IC:\Python31\include PythonType.cpp
// link /DLL /out:PythonTypeExt.pyd PythonType.obj
//      C:\Python31\libs\Python31.lib
//
// for Pyton 2.5.x
// ----------------
// cl /c -IC:\Python25\include PythonType.cpp
// link /DLL /out:PythonTypeExt.pyd PythonType.obj
//      C:\Python25\libs\Python25.lib
//


#include <Python.h>

//////////////////
//
// for offsetof function
//
//

#include "structmember.h"

#ifdef _PYTHON_3X

#define MEMBER_ATTRIBUTES PyMemberDef
#define PYTHON_STRING_CONVERT(b) PyUnicode_FromString((b))


#else

#define MEMBER_ATTRIBUTES memberlist
#define PYTHON_STRING_CONVERT(b) PyString_FromString((b))

#endif
```

```
//////////////////////
//
// Data structure to store
// an integer and float .
// These will be exposed as
// properties
//

typedef struct {
   PyObject_HEAD
   int vint;
   float vfloat;
}PyConType;

//////////////////////////////
//
//
// Function prototypes
//
//

static PyObject*
PyConTypeNew(PyObject *self, PyObject *args);

static PyObject*
Spit(PyConType *self, PyObject *args);

static void
pycontype_dealloc(PyConType* cons);

/////////////////////////////////
//
//
// Method supported by this object
//


static PyMethodDef pycontype_methods[  ] = {
   {"Spit", (PyCFunction)Spit, METH_VARARGS
   },    {NULL}  /* Sentinel */
};

/////////////////////////////
//
// Attributes inside the type
//
static struct MEMBER_ATTRIBUTES  pycontype_memberlist[] = {
        {"vint",T_INT,offsetof(PyConType,vint)},
        {"vfloat",T_FLOAT,offsetof(PyConType,vfloat)},
        {NULL}
};
```

```
//////////////////////////////////
//
// Method to create a new instance exported
// from the module
//
static PyMethodDef pycon_module_functions[  ] = {
    {"PyConTypeNew",   PyConTypeNew,   METH_VARARGS},
    {0, 0}
};


#ifdef _PYTHON_3X

static PyObject *
pycontype_getvint(PyConType *self, void *closure)
{

    return Py_BuildValue("i", self->vint);
}

static void
pycontype_setvint(PyConType *self, PyObject *value ,void *closure)
{

     int cnt;

     if ( !PyArg_ParseTuple(value,"i",&cnt) ) {
                    PyErr_SetString(PyExc_RuntimeError,
                "integer argument expected .." );
            return ;
        }

     self->vint = cnt;

}

static PyObject *
pycontype_getvfloat(PyConType *self, void *closure)
{

    return  Py_BuildValue("d", self->vfloat);
}

static void
pycontype_setvfloat(PyConType *self,PyObject *value, void *closure)
{

     double cnt;

     if ( !PyArg_ParseTuple(value,"d",&cnt) ) {
                    PyErr_SetString(PyExc_RuntimeError,
                "double argument expected .." );
```

```
            return ;
        }

    self->vfloat = cnt;
}




static PyGetSetDef pycontype_getseters[] = {
    {"vint",
    (getter)pycontype_getvint, (setter)pycontype_setvint,
    0,
    NULL},
    {"vfloat",
    (getter)pycontype_getvfloat, (setter)pycontype_setvfloat,
    0,
    NULL},
    {NULL}  /* Sentinel */
};


static PyTypeObject pycontype_type = {
    PyVarObject_HEAD_INIT(NULL, 0)
    "pycontype",            /* tp_name */
    sizeof(PyConType),      /* tp_basicsize */
    0,                      /* tp_itemsize */
    (destructor)pycontype_dealloc, /* tp_dealloc */
    0,                      /* tp_print */
    0,
    0,
    0,                      /* tp_reserved */
    0,                      /* tp_repr */
    0,                      /* tp_as_number */
    0,                      /* tp_as_sequence */
    0,                      /* tp_as_mapping */
    0,                      /* tp_hash  */
    0,                      /* tp_call */
    0,                      /* tp_str */
    0,                      /* tp_getattro */
    0,                      /* tp_setattro */
    0,                      /* tp_as_buffer */
    Py_TPFLAGS_DEFAULT |
        Py_TPFLAGS_BASETYPE,   /* tp_flags */
    0,          /* tp_doc */
    0,                                  /* tp_traverse */
    0,                                  /* tp_clear */
    0,                                  /* tp_richcompare */
    0,                                  /* tp_weaklistoffset */
    0,                                  /* tp_iter */
    0,                                  /* tp_iternext */
    pycontype_methods,            /* tp_methods */
```

```
   pycontype_memberlist,            /* tp_members */
   pycontype_getseters,                     /* tp_getset */
   0,                       /* tp_base */
   0,                       /* tp_dict */
   0,                       /* tp_descr_get */
   0,                       /* tp_descr_set */
   0,                       /* tp_dictoffset */
   0,     /* tp_init */
   0,                       /* tp_alloc */
   0,               /* tp_new */
};



#else




static PyObject *
pycontype_getattr(PyConType *self , char *attr )
{

   PyObject *res = 0;

   res = Py_FindMethod(pycontype_methods,(PyObject *)self,attr);

   if ( res == 0 ) {
     PyErr_Clear();
     return PyMember_Get((char *)self ,
                    pycontype_memberlist,attr );
   }
   return res;
}

int
pycontype_setattr(PyConType *self , char *attr,PyObject *value )
{

   if ( value == 0 ) {
         return -1;
   }
   return PyMember_Set((char *)self ,
                    pycontype_memberlist,attr,value );
}




static PyTypeObject pycontype_type = {
   PyObject_HEAD_INIT(0)     /* initialize to 0 to ensure Win32 portability  */
```

```
    0,                      /* ob_size */
    "pycontype",                    /* tp_name */
    sizeof(PyConType),          /* tp_basicsize */
    0,                      /* tp_itemsize */
    /* methods */
    (destructor)pycontype_dealloc, /* tp_dealloc */
    0,
    (getattrfunc) pycontype_getattr,
    (setattrfunc) pycontype_setattr
    /* implied by ISO C: all zeros thereafter, i.e., no other method */
};




#endif




static PyObject*
PyConTypeNew(PyObject *self, PyObject *args)
{

    if ( !PyArg_ParseTuple(args,":PyConType"))
                return 0;


    PyConType *cons = PyObject_New(PyConType, &pycontype_type);
    cons->vint = 100;
    cons->vfloat = 3.14159;

    return (PyObject *)cons;
}


static PyObject*
Spit(PyConType *self, PyObject *args)
{

    if ( !PyArg_ParseTuple(args,":PyConType"))
                return 0;

    char buffer[255];
    sprintf(buffer,"%d %f" , self->vint , self->vfloat );
    return PYTHON_STRING_CONVERT(buffer);

}

static void
pycontype_dealloc(PyConType* cons)
{
```

```
    PyObject_Del(cons);
}



#ifdef _PYTHON_3X


static PyModuleDef PyExtModule = {
    PyModuleDef_HEAD_INIT,
    "PythonTypeExt",
    0,
    -1,
    pycon_module_functions,NULL, NULL, NULL, NULL

};

////////////////////////////////////
//
// Python Interpreter initializes the module by
// calling the routine given below
//
//


PyMODINIT_FUNC
PyInit_PythonTypeExt(void)
{
    PyObject* m;

    if (PyType_Ready(&pycontype_type) < 0)
        return NULL;

    m = PyModule_Create(&PyExtModule);
    if (m == NULL)
        return NULL;

    Py_INCREF(&pycontype_type);
    PyModule_AddObject(m, "PythonTypeExt", (PyObject *)&pycontype_type);
    return m;
}


#else


/* module entry-point (module-initialization) function */
extern "C" void __declspec(dllexport) initPythonTypeExt(void)
{
    /* Create the module, with its functions */
    PyObject *m = Py_InitModule("PythonTypeExt", pycon_module_functions);
```

```
    /* Finish initializing the type-objects */
    pycontype_type.ob_type = &PyType_Type;
}

#endif
```

```
Administrator: Visual Studio 2008 Command Prompt

C:\PyCon2010\PythonType>cl -IC:\Python25\include /c PythonType.cpp
Microsoft (R) 32-bit C/C++ Optimizing Compiler Version 15.00.21022.08 for 80x86
Copyright (C) Microsoft Corporation.  All rights reserved.

PythonType.cpp

C:\PyCon2010\PythonType>link /DLL /out:PythonTypeExt.pyd PythonType.obj C:\Pytho
n25\libs\Python25.lib
Microsoft (R) Incremental Linker Version 9.00.21022.08
Copyright (C) Microsoft Corporation.  All rights reserved.

   Creating library PythonTypeExt.lib and object PythonTypeExt.exp

C:\PyCon2010\PythonType>c:\Python31\python
Python 3.1.2 (r312:79149, Mar 21 2010, 00:41:52) [MSC v.1500 32 bit (Intel)] on
win32
Type "help", "copyright", "credits" or "license" for more information.
>>> import PythonTypeExt
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
ImportError: Module use of python25.dll conflicts with this version of Python.
>>> quit()

C:\PyCon2010\PythonType>c:\Python25\python
Python 2.5.1 (r251:54863, Apr 18 2007, 08:51:08) [MSC v.1310 32 bit (Intel)] on
win32
Type "help", "copyright", "credits" or "license" for more information.
>>> import PythonTypeExt
>>> obj = PythonTypeExt.PythonTypeNew()
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
AttributeError: 'module' object has no attribute 'PythonTypeNew'
>>> dir(PythonTypeExt)
['PyConTypeNew', '__doc__', '__file__', '__name__']
>>> obj = PythonTypeExt.PyConTypeNew()
>>> obj.Spit()
'100 3.141590'
>>> obj.vint = 42
>>> obj.Spit()
'42 3.141590'
>>> obj.vfloat = 1001
>>> obj.Spit()
'42 1001.000000'
>>> obj.vfloat
1001.0
>>> obj.vint
42
>>> quit()

C:\PyCon2010\PythonType>
```

Administrator: Visual Studio 2008 Command Prompt

```
C:\PyCon2010\PythonType>cl -IC:\Python31\include /D_PYTHON_3X /c PythonType.cpp
Microsoft (R) 32-bit C/C++ Optimizing Compiler Version 15.00.21022.08 for 80x86
Copyright (C) Microsoft Corporation.  All rights reserved.

PythonType.cpp

C:\PyCon2010\PythonType>link /DLL /out:PythonTypeExt.pyd PythonType.obj C:\Pytho
n31\libs\Python31.lib
Microsoft (R) Incremental Linker Version 9.00.21022.08
Copyright (C) Microsoft Corporation.  All rights reserved.

   Creating library PythonTypeExt.lib and object PythonTypeExt.exp

C:\PyCon2010\PythonType>c:\Python31\python
Python 3.1.2 (r312:79149, Mar 21 2010, 00:41:52) [MSC v.1500 32 bit (Intel)] on
win32
Type "help", "copyright", "credits" or "license" for more information.
>>> import PythonTypeExt
>>> dir(PythonTypeExt)
['PyConTypeNew', 'PythonTypeExt', '__doc__', '__file__', '__name__', '__package_
_']
>>> obj = PythonTypeExt.PyConTypeNew()
>>> obj.value = 10
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
AttributeError: 'pycontype' object has no attribute 'value'
>>> obj.vint = 10
>>> obj.Spit()
'10 3.141590'
>>> obj.vfloat = 10001
>>> obj.Spit()
'10 10001.000000'
>>> quit()

C:\PyCon2010\PythonType>
```