מדריך ל- Wrapper:

* ליצור מחלקת C++:

1. #include <iostream>
2. // A simple class with a constuctor and some methods...
3. class Foo
4. {
5. public:
6. Foo(int);
7. void bar();
8. int foobar(int);
9. private:
10. int val;
11. };
12. Foo::Foo(int n)
13. {
14. val = n;
15. }
16. void Foo::bar()
17. {
18. std::cout << "Value is " << val << std::endl;
19. }
20. int Foo::foobar(int n)
21. {
22. return val + n;
23. }

* ליצור extern ל-C (ctypes לא יודע להשתמש בC++, אז צריך לעשות העברה ל-C, אבל זה נמצא באותו קובץ C++):

extern "C"

{

Foo\* Foo\_new(int n) {return new Foo(n);}

void Foo\_bar(Foo\* foo) {foo->bar();}

int Foo\_foobar(Foo\* foo, int n) {return foo->foobar(n);}

}

* להריץ את השורת קוד הבאה ב-CMD:

g++ **file.cpp** –I <include\_dir> -shared –o **file.so**

דוגמא של Mesh.cpp:

g++ Mesh.cpp C:\Users\User\Downloads\huji-rich-Elad3D\huji-rich-Elad3D\source\misc\int2str.cpp -I C:\Users\User\Downloads\boost\_1\_67\_0\boost\_1\_67\_0 -I C:\Users\User\Downloads\huji-rich-Elad3D\huji-rich-Elad3D\source\misc -shared -o Mesh.so

* ליצור מחלקה זהה ב-Python שמשתמשת בפונקציות של C++:

import ctypes

lib = ctypes.cdll.LoadLibrary('./file.so')

class Foo(object):

def \_\_init\_\_(self, val):

lib.Foo\_new.argtypes = [ctypes.c\_int]

lib.Foo\_new.restype = ctypes.c\_void\_p

lib.Foo\_bar.argtypes = [ctypes.c\_void\_p]

lib.Foo\_bar.restype = ctypes.c\_void\_p

lib.Foo\_foobar.argtypes = [ctypes.c\_void\_p, ctypes.c\_int]

lib.Foo\_foobar.restype = ctypes.c\_int

self.obj = lib.Foo\_new(val)

def bar(self):

lib.Foo\_bar(self.obj)

def foobar(self, val):

return lib.Foo\_foobar(self.obj, val)