Extending and Embedding Python

Recompiled -
Prof. Naiswita Parmar
Assistant Professor,
CSE, SET,
Navrachana University

Extending and Embedding Python

- Invoking C function from Python is known as Extending.
- Embedding code of Python into C program is known as Embedding.
- Extending is more useful rather than Embedding.
- During extending C function can be invoked as a library from a Python program.
- The only reason to leverage this is due to low execution speed of Python. Python is slow compared to languages such as C, C++ and Java. Hence the compute intensive module can be designed in C or C++ and those function can be invoked from Python to make the task faster.
- The Python API is incorporated in a C source file by including the header "Python.h".
- from ctypes import * # is used to extend functionalities of C to Python.

Extending C to Python Advantages

- Improve the time complexity.
- Parallelize your code. (Multithreading like task)
- Move slow parts of your code to a faster language.

ctypes — A foreign function library for Python

- ctypes is a foreign function library for Python. It provides C compatible data types, and allows calling functions in DLLs or shared libraries.
- It can be used to wrap these libraries in pure Python.

- Link - https://docs.python.org/3/library/ctypes.html

Calling C,C++ program from Python/Extending

STEPS to invoke C program to invoke from Python.

- Create a C program with .c extension
- Create a shared library file using C compiler.
- Follow below command to create shared library with C compiler
- gcc -shared -o libpyfun1.so -fPIC f:\cprg\pyfun1.c
- Here libpyfun1.so is a shared library file created from f:\cprg\pyfun1.c
- From python program file, create a ctypes.CDLL instance from the shared file. Use following command from python
- *from ctypes import **
- *cref = CDLL("libpyfun1.so")*
- **NOTE : PIC stands for Position Independent Code.**
  - If supported for the target machine, emit position-independent code, suitable for dynamic linking and avoiding any limit on the size of the global offset table.

Calling C,C++ program from Python

Pyfun1.c

```
#include<stdio.h>
int add(int x,int y)
{
 int ans=x+y;
 return ans;
}
```

Pyfun1.h

```
int add(int x,int y);
```

----

mypython.py

```
from ctypes import *
cref = CDLL("libpyfun1.so")
cr = cref.add(4,7)
print("Additon is ",cr)
```

Syntax

Calling C++ program from Python

```
//g++ -o libpyfun5.dll -shared -fPIC f:\cprg\pyfun5.cpp //creates a dll file
//g++ -c -fPIC f:\cprg\pyfun5.cpp -o libpyfun5.o
//g++ -shared -Wl,-soname,libpyfun5.so -o libpyfun5.so  libpyfun5.o
#include<iostream>
using namespace std;
class Student
{
        public:
        Student()//Constructor - it is a function which has same name as its classname
        {
             cout<<"Student Constructor"<<endl; //cout is an object //endl is a statement
        }
        void display()// void display(Student *this)
        {
              cout<<"Display function of C++"<<endl;
        }
        void show()
        {
           cout<<"Show function of C++"<<endl;
        }
}; //C++ class ends with semicolon
```

Calling C++ program from Python

```
}; //C++ class ends with semicolon
extern "C"
{
    Student * Student_new() //ConstructorName_new()
         {
           return new Student(); //new allocates memory for Student object
         }
    void Student_display(Student *f)
         {
            f->display();  //-> operator is used to call a member function by using object pointer
         }
    void Student_show(Student *f)
         {
                  f->show();
         }
}
```

Calling C++ program from Python

```
import ctypes # ctypes is a package to invoke C, C++ program from python
lib = ctypes.cdll.LoadLibrary("./libpyfun5.dll") #lib stores the address of the .so or .dll file loaded into RAM
class Student(object):
    def __init__(self):
        #configuring return type of function and arugments if passed
        lib.Student_new.argtypes = [ctypes.c_void_p] #constructor
        lib.Student_new.restype = ctypes.c_void_p  #constructor

        lib.Student_display.argtypes = [ctypes.c_void_p] # display function
        lib.Student_display.restype = ctypes.c_void_p # display function

        lib.Student_show.argtypes = [ctypes.c_void_p]  #show function
        lib.Student_show.restype = ctypes.c_void_p  #show function

        self.obj = lib.Student_new() #invoke C++ constructor from python
    def display(self):
        lib.Student_display(self.obj) #invoking display function of C++

    def show(self):
        lib.Student_show(self.obj)  #invoking show function of C++


st=Student() #invokes python constructor above
st.display() #invokes python display method declared above
st.show()  #invokes python show method declared above
```

Embedding Python in C program

Syntax

```
#include <stdio.h>
#include <Python.h>
//Embedding python program within C
void main()
{
        char filename[] = "F:\\python_oop\\myoop001.py";
        FILE* fp;

        Py_Initialize();//initiate the python environment

        fp = _Py_fopen(filename, "r");
        PyRun_SimpleFile(fp, filename);  //executes python program

        Py_Finalize(); //terminates python environment
        //return 0;
}
```

- Declare a FILE* to store our program file object.
- Now open the Python program file using _Py_fopen(char* program_filename_with_py_extension, char* file_open_mode). This function is similar to the fopen function of standard C/C++.Here we have opened the pyemb7.py in read mode.
- Check the FILE* object returned. If it is NULL, the file cannot be opened, so we cannot proceed further. Report an error and abort.
- Now we have the file opened. We have to execute it using PyRun_SimpleFile(opened_python_program_file_pointer, char* program_filename_which_becomes_argv_0).

References

- Michael H. Goldwasser, David Letscher , “Object-oriented Programming in Python “ , Pearson Prentice Hall, 2008
- Dusty Phillips , “Python 3 Object Oriented Programming” ,PACLT publications.
- Retrieved and referred from docs.python.org, July 2021.
- Retrieved and referred from https://www.pcmag.com/encyclopedia, July 2021.
- Retrieved and referred from https://stackoverflow.com, July 2021.
- Retrieved and referred from https://wikipedia.org, July 2021.
- Retrieved and referred from https://w3schools.com, July 2021.