

# Интеграция приложений Python и C/C++

Nizhny Novgorod / 20.05.2021

**Luxoft** | training  
A DXC Technology Company

# INTRODUCTION

**Andrei Sorokin**

---

and.e.s@yandex.ru

**Consulting & developing**



# Интеграция приложений Python и C/C++

## План семинара

- Введение
- `ctypes`
- `python extensions`

# Интеграция приложений Python и C/C++

## Введение

# Интеграция приложений Python и C/C++

## ctypes – python интерфейс к библиотекам C/C++

- Состав, документация, исходники (<https://docs.python.org/3/library/ctypes.html>)
- Как создать и подключить библиотеку
- Требования к библиотеке c/c++
- Резервирование и освобождение ресурсов
- Работа с простыми типами
- Строки и указатели
- Массивы и коллекции
- Структуры и классы

## ctypes – python интерфейс к библиотекам C/C++

### Создаём библиотеку.

```
extern "C" {  
using namespace std;  
  
void swap(int* a, int* b){ int c = *a; *a=*b; *b = c;}  
// другие C-методы  
// методы оболочки для c++  
}
```

```
g++ -O0 -g3 -Wall -c -fmessage-length=0 -o "src\\lib_02.o" "..\\src\\lib_02.cpp"  
g++ -shared -o lib_02.dll "src\\lib_02.o"
```

Необходимо протестировать библиотеку средствами c/c++

## ctypes – python интерфейс к библиотекам C/C++

### how to look Exports?

`dumpbin /exports path_to_dll`

`readelf -s path_to_so`

```
.....
1  0 00001404 _ZN5lib0210address_clC1EPcS1_
2  1 00001404 _ZN5lib0210address_clC2EPcS1_
3  2 0000142E _ZN5lib026get_dtB5cxx11ENS_2mmE
4  3 0000172F _ZN5lib027swap_ppEPiS0_
.....
64 3F 00001818 get_dt2
65 40 000018C2 get_int
66 41 00001928 sum_of_int
67 42 00001764 swap
.....
```

# ctypes – python интерфейс к библиотекам C/C++

- libffi (<https://github.com/libffi/libffi>)
- ctypes (<https://github.com/python/cpython/tree/main/Lib/ctypes>)
- \_ctypes([https://github.com/python/cpython/tree/main/Modules/\\_ctypes](https://github.com/python/cpython/tree/main/Modules/_ctypes))



**ctypes – python интерфейс к библиотекам C/C++.**  
**Base types**

<b>ctypes type</b>	<b>C type</b>	<b>Python type</b>
c_bool	_Bool	bool
c_char	char	1-character bytes object
c_wchar	wchar_t	1-character string
c_byte	char	int
c_ubyte	unsigned char	int
c_short	short	int
c_ushort	unsigned short	int
c_int	int	int
c_uint	unsigned int	int
c_long	long	int

c_ulong	unsigned long	int
c_longlong	__int64 or long long	int
c_ulonglong	unsigned __int64 or unsigned long long	int
c_size_t	size_t	int
c_ssize_t	ssize_t or Py_ssize_t	int
c_float	float	float
c_double	double	float
c_longdouble	long double	float
c_char_p	char * (NUL terminated)	bytes object or None
c_wchar_p	wchar_t * (NUL terminated)	string or None
c_void_p	void *	int or None

## ctypes – python интерфейс к библиотекам C/C++.

### Pointers

`ctypes.pointer` #создаёт объект указатель  
`ctypes.POINTER` #создаёт ТИП указатель

```
# universal pointer  
*.restype = ctypes.c_void_p
```

```
#pointer to char  
charptr = ctypes.POINTER(ctypes.c_char)  
charptr2= ctypes.c_char_p
```

ctypes – python интерфейс к библиотекам C/C++.

python-byref

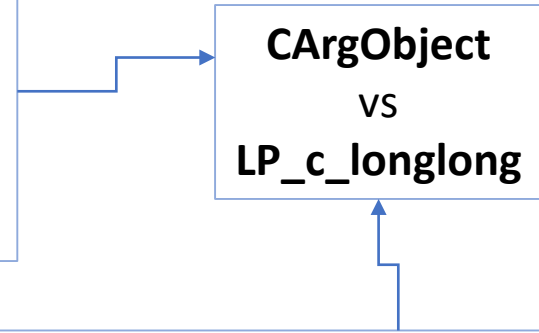
vs python-pointers

```
void swap(int* a, int* b){ int c = *a; *a=*b; *b = c;}
```

```
i1 = ctypes.c_int32(10);  
i2 = ctypes.c_int32(2000);
```

```
print("i1=%5d, i2=%5d" % (i1.value, i2.value))  
c_lib.swap(ctypes.byref(i1), ctypes.byref(i2))  
print("i1=%5d, i2=%5d" % (i1.value, i2.value))
```

CArgObject  
vs  
LP\_c\_longlong



```
i3= 1  
i4= 200  
p3 = ctypes.pointer((ctypes.c_int64)(i3))  
p4 = ctypes.pointer((ctypes.c_int64)(i4))  
  
print("i3=%5d, i4=%5d" % (p3.contents.value, p4.contents.value))  
c_lib.swap(p3, p4)  
print("i3=%5d, i4=%5d" % (p3.contents.value, p4.contents.value))
```

## ctypes – python интерфейс к библиотекам C/C++.

### Arrays

```
arr1 = (1,2,3,4,5,6,7,8,9,10)
N = len(arr1)

# подготовка данных для аргумента-массива

# медленно
data = ctypes.pointer( (ctypes.c_int*N) (*arr1) )

# быстрее в разы
data2 = (ctypes.c_int*N) ()
data2[:] = arr1
```

ctypes – python интерфейс к библиотекам C/C++.  
Structures

```
typedef struct {  
    char * street;  
    char * city;  
    long index;  
} address_st;
```

```
*.argtypes=[ ctypes.POINTER(Address)  
*.restype=]
```

```
class Address(ctypes.Structure):
```

```
    _fields_ = [("street", ctypes.c_char_p),  
                ("city", ctypes.c_char_p),  
                ("index", ctypes.c_long)]
```

```
    def __repr__(self):  
        return '({0}, {1}, {2})'.format(self.street, self.city, self.index)
```

**ctypes – python интерфейс к библиотекам C/C++.**  
**Демонстрация.**

# Интеграция приложений Python и C/C++

## Разработка расширений cpython на C/C++

- Состав, документация, исходники <https://docs.python.org/3>

# Интеграция приложений Python и C/C++

## Разработка расширений cpython на C/C++



# Интеграция приложений Python и C/C++

## Разработка расширений cpython на C/C++

1. Создаём C-файл. Адаптируем функции под cpython интерфейс.

```
static PyObject * time_string( PyObject *self)
{
    time_t t = time(NULL);
    struct tm tm = *localtime(&t);
    char buffer[21];
    sprintf(buffer, "%04d-%02d-%02d %02d:%02d:%02d",
tm.tm_year + 1900, tm.tm_mon + 1, tm.tm_mday,
tm.tm_hour, tm.tm_min, tm.tm_sec);
    return Py_BuildValue("s", buffer);
}

static PyObject * run_system(PyObject *self, PyObject *args)
{ /* ..... */
return PyLong_FromLong(sts);}
```

# Интеграция приложений Python и C/C++

## Разработка расширений cpython на C/C++

2. Регистрируем функции.

```
static PyMethodDef mymodule01_funcs[] = {  
    {"time", (PyCFunction)time_string, METH_NOARGS, mymodule01_docs},  
    {"system", (PyCFunction)run_system, METH_VARARGS, mymodule01_docs},  
    {NULL, NULL, 0, NULL}  
};
```

3. Пишем документацию.

```
static char mymodule01_docs[] =  
    "time_string( ): get time string from c!!\n"  
    "system(): run system command!\n";
```

# Интеграция приложений Python и C/C++

## Разработка расширений cpython на C/C++

4. Заполняем структуру для инициализации модуля.

```
static struct PyModuleDef mymodule01 =  
{  
    PyModuleDef_HEAD_INIT,  
    "mymodule01",      /* name of module */  
    mymodule01_docs,    /* module documentation */  
    -1,  
    mymodule01_funcs    /* module functions */  
};
```

5. Подаём адрес структуры инициализатору.

```
PyMODINIT_FUNC PyInit_mymodule01(void)  
{  
    return PyModule_Create(&mymodule01);  
}
```

~~Py\_InitModule3~~

# Интеграция приложений Python и C/C++

## Разработка расширений cpython на C/C++

6. Создаём файл описания setup.py.

```
from distutils.core import setup, Extension

setup(name='mymodule01',
      version='1.0',
      description='bla bla bla my module for testing',
      license='MIT',
      author='Andrei',
      author_email='a0z9@rambler.ru',
      keywords=['my package', 'extra module'],
      url='https://github.com/a0z9/mymodule01',
      long_description = 'simple usage!',
      platforms = 'x64',
      ext_modules=[Extension('mymodule01', ['mymodule.c'])])
```


# Интеграция приложений Python и C/C++

## Разработка расширений cpython на C/C++

### 7. Выполняем `python setup.py install`

```
running install
running build
running build_ext
building 'mymodule01' extension
creating build\temp.win-amd64-3.8
creating build\temp.win-amd64-3.8\Release
...
..\cl.exe /c /nologo /Ox /W3 /GL /DNDEBUG /MD ... /Tcmymodule.c /Fobuild\temp.win-amd64-
3.8\Release\mymodule.obj mymodule.c
..\link.exe /nologo /INCREMENTAL:NO /LTCG /DLL /MANIFEST:EMBED,ID=2 /MANIFESTUAC:NO /LIBPATH: ...
/OUT:build\lib.win-amd64-3.8\mymodule01.cp38-win_amd64.pyd
...
```

### 8. Импортируем библиотеку



```
>>> import mymodule01 as mm
>>> mm.time()
'2021-05-11 00:10:07'
```

**Спасибо за внимание!**

Thank You!

think.  
create.  
accelerate.

Пройди опрос  
- поделись своим мнением!



[https://ru.surveymonkey.com/r/seminar\\_20\\_may](https://ru.surveymonkey.com/r/seminar_20_may)

**Luxoft** | training  
A DXC Technology Company