

# C-extensions

Урок пятый

Антон Кухтичев

**Не забудьте отметить!**

- ctypes;
- ffi;
- C-extension;
- Cython.

- Вам нужна скорость и вы знаете, что C в X раз быстрее Python;
- Вам нужна конкретная C-библиотека и вы не хотите писать “велосипед” на Python;
- Вам нужен низкоуровневый интерфейс управления ресурсами для работы с памятью и файлами;
- Просто потому что Вам так хочется

# ctypes

```
int sum(int *arr, int len)
{
    int res = 0;
    for (int i = 0; i < len; ++i)
    {
        res += arr[i];
    }
    return res;
}
```

```
gcc -fPIC -shared -o sumlib.so 1.c
```

```
import ctypes
from typing import List

lib1 = ctypes.CDLL('./lib1.so')
lib1.sum.argtypes = (ctypes.POINTER(ctypes.c_int),
ctypes.c_int)
```

```
def sum(arr: List[int]) -> int:
    arr_len = len(arr)
    arr_type = ctypes.c_int * arr_len
    result = lib1.sum(arr_type(*arr),
ctypes.c_int(arr_len))
    return int(result)
```



# CFFI

```
# Установка  
pip install cffi
```

CFFI (C Foreign Function Interface) генерирует поверх нашей библиотеки свою обвязку и компилирует её в библиотеку с которой мы и будем работать.

```
from cffi import FFI

ffi = FFI()
lib = ffi.dlopen('../ctypes/lib1.so')

ffi.cdef('''int sum(int* arr, int len);''')
arr = [1, 2, 3, 4]
c_arr = ffi.new('int[]', arr)

s = lib.sum(c_arr, len(arr))
print(s)
```

```
#include <stdlib.h>
```

```
struct Point {  
    int x;  
    int y;  
};
```

```
int area(struct Point *p1, struct Point *p2) {  
    return abs((p2->y - p1->y) * (p1->x - p2->x));  
}
```

```
gcc -fPIC -shared -o lib2.so 2.c
```

```
from cffi import FFI

ffi = FFI()
lib = ffi.dlopen('./lib2.so')

ffi.cdef('''
struct Point {
    int x;
    int y;
};
int area(struct Point *p1, struct Point *p2);
''')
```

```
p1 = ffi.new('struct Point*')
p2 = ffi.new('struct Point*')

p1.x = 0
p1.y = 0

p2.x = 10
p2.y = 10

s = lib.area(p1, p2)
print(s)
```

## Плюсы:

- простой синтаксис при использовании в Python;
- не нужно перекомпилировать исходную библиотеку.

## Минусы:

- не удобная сборка, нужно прописывать пути до всех заголовочных файлов и библиотек;
- создается еще 1-на динамическая библиотека, которая использует исходную.

# C Extensions



```
>>> import spam  
>>> status = spam.system("ls -l")
```

```
#define PY_SSIZE_T_CLEAN  
#include <Python.h>
```

```
static PyObject *
spam_system(PyObject *self, PyObject *args)
{
    const char *command;
    int sts;

    if (!PyArg_ParseTuple(args, "s", &command))
        return NULL;
    sts = system(command);
    if (sts < 0) {
        PyErr_SetString(SpamError, "System command failed");
        return NULL;
    }
    return PyLong_FromLong(sts);
}
```

# Cython

```
# Установка  
pip install cython
```

При работе с функциями нам доступны следующие типы:

- `def` — обычная Python-функция, вызывается только из Python.
- `cdef` — Cython-функция, которую нельзя вызвать из обычного Python-кода. Такие функции можно вызывать только в пределах Cython-кода.
- `cpdef` — Функция, доступ к которой можно получить и из C, и из Python.

```
# Установка  
pip install cython
```

При работе с функциями нам доступны следующие типы:

- `def` — обычная Python-функция, вызывается только из Python.
- `cdef` — Cython-функция, которую нельзя вызвать из обычного Python-кода. Такие функции можно вызывать только в пределах Cython-кода.
- `cpdef` — Функция, доступ к которой можно получить и из C, и из Python.

```
# setup.py
from distutils.core import setup
from Cython.Build import cythonize

setup(ext_modules = cythonize('run_cython.pyx'))
```

Выполним компиляцию:

```
python setup.py build_ext --inplace
```

1. Необходимо реализовать класс Matrix, хранящий целые числа через c-extension.
2. Реализовать умножение матриц на Python и сравнить производительность кода на C и на Python.



## Полезные ссылки

[CFFI](#)

[ctypes](#)

<https://cffi.readthedocs.io/en/latest/>

<https://docs.python.org/3/extending/extending.html>

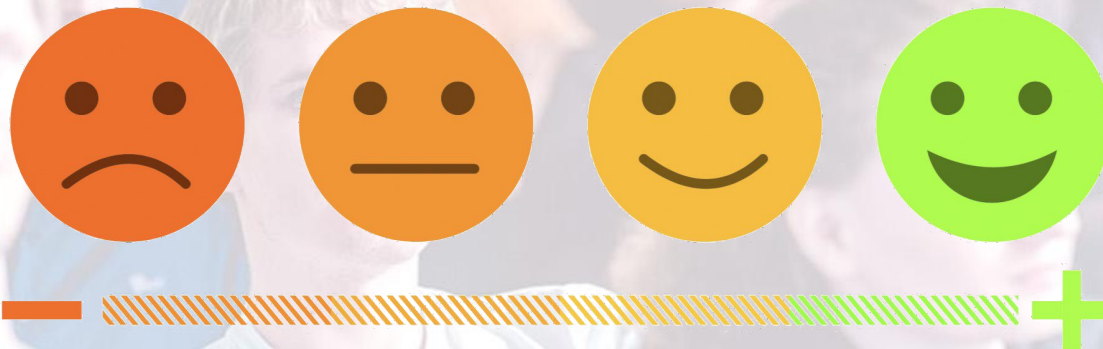
[https://www.tutorialspoint.com/python3/python\\_further\\_extensions.htm](https://www.tutorialspoint.com/python3/python_further_extensions.htm)

[Cython](#)



Для саморазвития (опционально)  
[Чтобы не набирать двумя пальчиками](#)

Не забудьте оценить занятие!



Спасибо  
за внимание!

Антон Кухтичев

[a.kukhtichev@corp.mail.ru](mailto:a.kukhtichev@corp.mail.ru)