

# Углубленный Python

## Лекция 10 C-расширения

Кандауров Геннадий



# Напоминание отметиться на портале

+ ОСТАВИТЬ ОТЗЫВ

vk образование

БлогиЛюдиПрограммаВакансииРасписание

Q<

VK

Техно

Открыт приём заявок!

чт, 8 сентября

Нет занятий

пт, 9 сентября

18:00 Углубленный Py... с3  
Введение в Python, основные  
понятия, тестирование  
Г. Кандауров

сб, 10 сентября

Нет занятий

вс, 11 сентября

Нет занятий

пн, 12 сентября

Нет занятий

Углубленный Python

↓ 0 ↑

Блог для материалов по курсу "Углубленный Python"

57 читателей, 2 топика

ПодписатьсяСоздать топик

Поиск по авторам, заголовку и тексту топика...

Найти

Добро пожаловать на курс!

Углубленный Python

ИзменитьУдалить

Всем привет и добро пожаловать на курс по углубленному изучению Python!

Прямой эфир

МоиВсе

Геннадий Кандауров час назад  
Углубленный Python → Добро пожаловать  
на курс! 0

Екатерина Черкасова 7 дней назад  
Стажировка → Приглашаем мобильных,  
фронтенд- и бэкэнд-разработчиков на  
Weekend Offer! 0

Дарья Вовченко 9 дней назад  
Углубленный Python → Добро пожаловать  
в образовательные проекты VK  
Образование! 0

Дарья Вовченко 9 дней назад  
Разработка веб-сервисов на

## Квиз про прошлой лекции



# Содержание занятия

1. ctypes
2. cffi
3. C API
4. Cython

# Расширения на C

## C-extensions



# Причины и поводы

- нужна скорость и есть мнение, что C в X раз быстрее Python
- нужна конкретная C-библиотека и не хочется писать “велосипед” на Python
- нужен низкоуровневый интерфейс управления ресурсами для работы с памятью и файлами
- просто потому что так хочется

**ctypes**



# ctypes

- работает с DLL (Dynamic Link Library)
- ctypes определяет типы данных, совместимые с языком C:
  - `c_bool`
  - `c_char`
  - `c_int`
  - `c_char_p`
  - `c_void_p`
- чтобы подключить библиотеку нужно либо вызвать
  - `ctypes.cdll.LoadLibrary("<dll path>")`
  - `ctypes.CDLL("<dll path>")`



# 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 libfunctions.so custom_functions.c
```

# ctypes

```
import ctypes
```

```
libfuncs = ctypes.CDLL("./libfunctions.so")
```

```
libfuncs.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 = libfuncs.sum(arr_type(*arr), ctypes.c_int(arr_len))
```

```
    return int(result)
```

**cffi**

**C Foreign Function Interface**



# 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.

# cffi

```
from cffi import FFI

ffi = FFI()
lib = ffi.dlopen("../ctypes/libfunctions.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)
```

# cffi

```
#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 libpoint.so point.c
```

# cffi

```
from cffi import FFI
```

```
ffi = FFI()
```

```
lib = ffi.dlopen("./libpoint.so")
```

```
ffi.cdef("""
```

```
struct Point {
```

```
    int x;
```

```
    int y;
```

```
};
```

```
int area(struct Point *p1, struct Point *p2);
```

```
""")
```

# cffi

```
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)
```



# cffi

- + простой синтаксис при использовании в Python
- + не нужно перекомпилировать исходную библиотеку
- неудобная сборка, нужно прописывать пути до всех заголовочных файлов и библиотек
- создается ещё одна динамическая библиотека, которая использует исходную

**C API**



# C API

1. Подключаем

Python.h

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

2. Все видимые пользователю имена имеют префикс `Py` или `_Py`

# C API: PyObject

```
typedef struct _object {  
    _PyObject_HEAD_EXTRA  
    Py_ssize_t ob_refcnt;  
    struct _typeobject *ob_type;  
} PyObject;
```

- `ob_refcnt`: счетчик ссылок;
- `ob_type`: указывает на объект класса текущего объекта экземпляра;
- `_PyObject_HEAD_EXTRA`: макрос. Если определен параметр `Py_TRACE_REFS`, этот макрос будет предварительно обработан в виде двух указателей как реализация двусвязного списка, который отслеживает все объекты кучи.

# C API

Полезные функции:

- `PyArg_ParseTuple`
- `PyDict_New`
- `PyDict_SetItem`
- `Py_BuildValue`

# C API

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

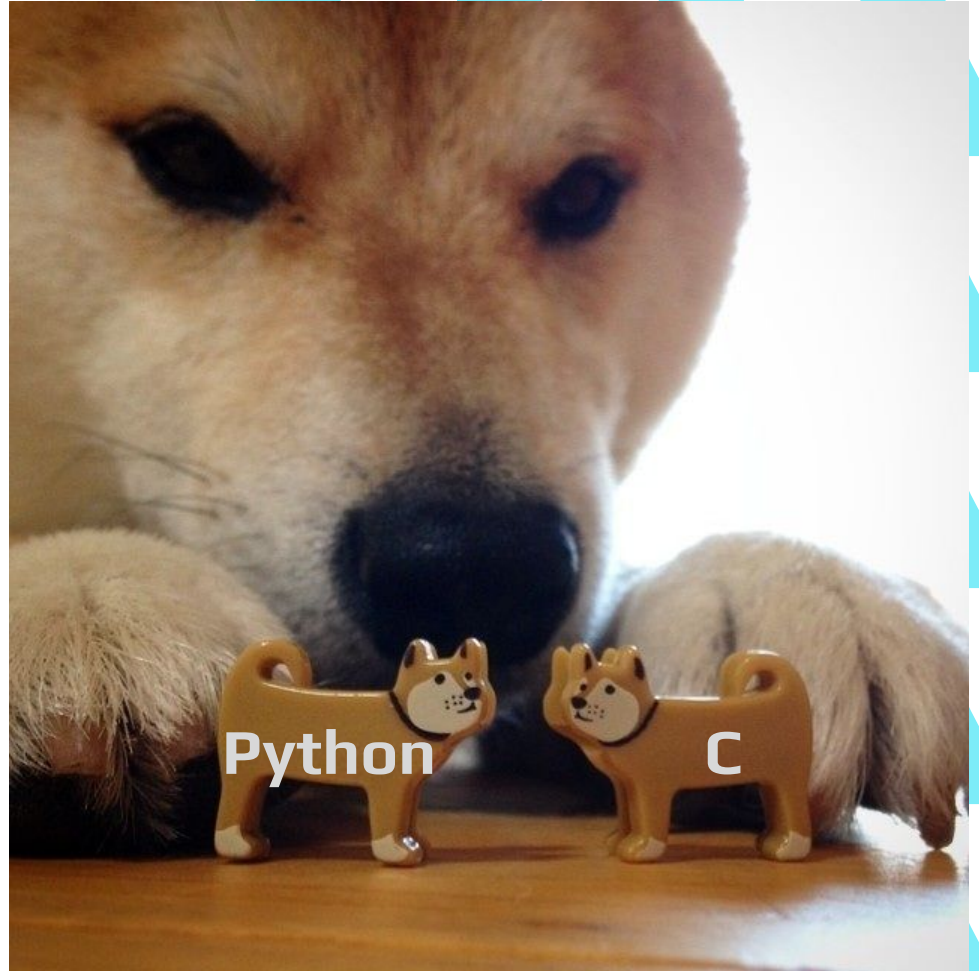
    if (!PyArg_ParseTuple(args, "s", &command))
        return NULL;

    sts = system(command);
    if (sts < 0) {
        PyErr_SetString(SpamError, "System command failed");
        return NULL;
    }

    return PyLong_FromLong(sts);
}
```

# Cython

*"Cython is Python with C data types"*



# 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



# Cython

```
# setup.py
from setuptools import setup, Extension
from Cython.Build import cythonize

setup(
    ext_modules= cythonize(["cutils.pyx"])
)

# Выполним компиляцию
$ python setup.py build_ext --inplace
```

## Домашнее задание #10

Реализовать свой модуль `custom_json`  
для сериализации/десериализации `json`

```
import custom_json  
  
obj = {"hello": "world", "key1": 100500}  
s = custom_json.dumps(obj)  
assert obj == custom_json.loads(s)
```

# Напоминание отметиться на портале Vol 2

+ ОСТАВИТЬ ОТЗЫВ ПОСЛЕ ЛЕКЦИИ

vk образование

БлогиЛюдиПрограммаВакансииРасписание

Q<

VK

Техно

Открыт приём заявок!

чт, 8 сентября

Нет занятий

пт, 9 сентября

18:00 Углубленный Py... с3  
Введение в Python, основные  
понятия, тестирование  
Г. Кандауров

сб, 10 сентября

Нет занятий

вс, 11 сентября

Нет занятий

пн, 12 сентября

Нет занятий

Углубленный Python

↓ 0 ↑

Блог для материалов по курсу "Углубленный Python"

57 читателей, 2 топика

Подписаться

Создать топик

Поиск по авторам, заголовку и тексту топика...

Найти

Добро пожаловать на курс!

Углубленный Python

Изменить

Удалить

Всем привет и добро пожаловать на курс по углубленному изучению Python!

Прямой эфир

МоиВсе

Геннадий Кандауров час назад  
Углубленный Python → Добро пожаловать  
на курс! 0

Екатерина Черкасова 7 дней назад  
Стажировка → Приглашаем мобильных,  
фронтенд- и бэкэнд-разработчиков на  
Weekend Offer! 0

Дарья Вовченко 9 дней назад  
Углубленный Python → Добро пожаловать  
в образовательные проекты VK  
Образование! 0

Дарья Вовченко 9 дней назад  
Разработка веб-сервисов на

Спасибо за  
внимание

