



Углубленный Python

Лекция 6



Латкин Игорь

“

Не забудьте отметить на занятии!

Цитата великих

Повестка дня



1. ctypes
2. ffi
3. C extensions
4. Cython

ctypes



<https://docs.python.org/3.7/library/ctypes.html>

Использует libffi



ctypes (1/4)



```
6 ▸ ctypes ▸ C 1.c ▸ ...
```

```
1  int sum(int *arr, int len) {  
2      ... int s = 0;  
3      ... for (int i = 0; i < len; ++i) {  
4          ... s += arr[i];  
5      ... }  
6      ... return s;  
7  }
```

```
6 ▸ ctypes ▸ M Makefile
```

```
1  1:  
2  | gcc -fPIC -shared -o lib1.so 1.o
```

ctypes (2/4)



```
6 ▸ ctypes ▸ 1.py ▸ ...
1  import ctypes
2  from typing import List
3
4  lib1 = ctypes.CDLL('./lib1.so')
5  lib1.sum.argtypes = (ctypes.POINTER(ctypes.c_int), ctypes.c_int)
6
```

ctypes (3/4)



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


ctypes (4/4)



```
6 ▶ ctypes ▶ 2.py ▶ ...
1  from ctypes import *
2
3  class POINT(Structure):
4      _fields_ = [("x", c_int),
5                  ("y", c_int)]
6
7
8  point = POINT(10, 20)
9  print(point.x, point.y)
10
11 point = POINT(y=5)
12 print(point.x, point.y)
13
14 POINT(1, 2, 3)
```




CFFI

cffi (1/4)



6 ▸ cffi ▸ 1.py ▸ ...

```
1  from cffi import FFI
2
3  ffi = FFI()
4
5  lib = ffi.dlopen('../ctypes/lib1.so')
6
7  ffi.cdef('''
8  int sum(int* arr, int len);
9  ''')
10
11  arr = [1, 2, 3, 4, 5]
12  c_arr = ffi.new('int[]', arr)
13
14  s = lib.sum(c_arr, len(arr))
15  print(s)
```

cffi (2/4)



```
6 ▸ cffi ▸ C 2.c ▸ ...
```

```
1  #include <stdlib.h>
2
3  struct Point {
4      ... int x;
5      ... int y;
6  };
7
8  int area(struct Point *p1, struct Point *p2) {
9      ... return abs((p2->y - p1->y) * (p1->x - p2->x));
10 }
```

```
6 ▸ cffi ▸ M Makefile
```

```
1  2:
2  | gcc -fPIC -shared -o lib2.so 2.c
```

cffi (3/4)



6 ▸ cffi ▸ 2.py ▸ ...

```
1  from cffi import FFI
2
3  ffi = FFI()
4
5  lib = ffi.dlopen('./lib2.so')
6
7  ffi.cdef('''
8  struct Point {
9      ... int x;
10     ... int y;
11 };
12 int area(struct Point *p1, struct Point *p2);
13 ''')
```

cffi (4/4)



```
15  p1 = ffi.new('struct Point*')
16  p2 = ffi.new('struct Point*')
17
18  p1.x = 0
19  p1.y = 0
20
21  p2.x = 10
22  p2.y = 10
23
24  s = lib.area(p1, p2)
25  print(s)
26
```

C Extensions

A Simple Example



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


A Simple Example



```
6 ▸ c ▸ spam ▸ C spam.c ▸ ...  
1  #define PY_SSIZE_T_CLEAN  
2  #include <Python.h>  
3
```

A Simple Example



```
7 static PyObject *
8 spam_system(PyObject *self, PyObject *args)
9 {
10     ... const char *command;
11     ... int sts;
12
13     ... if (!PyArg_ParseTuple(args, "s", &command))
14         ... return NULL;
15     ... sts = system(command);
16     ... if (sts < 0) {
17         ... PyErr_SetString(SpamError, "System command failed");
18         ... return NULL;
19     }
20     ... return PyLong_FromLong(sts);
21 }
```

A Simple Example



```
24 static PyMethodDef SpamMethods[] = {
25     {"system", spam_system, METH_VARARGS,
26     "Execute a shell command."},
27     {NULL, NULL, 0, NULL} /* Sentinel */
28 };
```

- **METH_VARARGS** – только позиционные аргументы
- **METH_KEYWORDS** – только kv аргументы
- **METH_VARARGS | METH_KEYWORDS** – позиционные + kv аргументы
- **METH_NOARGS** – функция без аргументов

A Simple Example



```
31 static struct PyModuleDef spammodule = {
32     PyModuleDef_HEAD_INIT,
33     "spam", /* name of module */
34     NULL, /* module documentation, may be NULL */
35     -1, /* size of per-interpreter state of the module,
36         or -1 if the module keeps state in global variables. */
37     SpamMethods
38 };
```

- **METH_VARARGS** – только позиционные аргументы
- **METH_KEYWORDS** – только kv аргументы
- **METH_VARARGS | METH_KEYWORDS** – позиционные + kv аргументы
- **METH_NOARGS** – функция без аргументов

Ordered Set



- Использование `std::set` из C++
- Собственный тип в Python - OSet

```
>>> c/oset/oset.1.c  
>>> c/oset/oset.2.c  
>>> c/oset/oset.3.c
```

Py_INCREF & Py_DECREF



<https://docs.python.org/3/extending/extending.html#reference-counts>

- Подсчет ссылок
- Никто не владеет объектом. Но можно владеть ссылкой на объект.
- Владелец ссылки ответственен за вызов Py_DECREF
- Можно заимствовать (borrow) ссылку.

Правила владения:

- Большинство функций, возвращающих PyObject * передают владение вызывающей стороне (ex: PyLong_FromLong)
- Есть исключения, например PyTuple_GetItem, PyList_GetItem - они возвращают заимствованную ссылку.

Подводные камни



<https://docs.python.org/3/extending/extending.html#thin-ice>

```
void
bug(PyObject *list)
{
    PyObject *item = PyList_GetItem(list, 0);

    PyList_SetItem(list, 1, PyLong_FromLong(0L));
    PyObject_Print(item, stdout, 0); /* BUG! */
}
```


Подводные камни



<https://docs.python.org/3/extending/extending.html#thin-ice>

```
void
no_bug(PyObject *list)
{
    PyObject *item = PyList_GetItem(list, 0);

    Py_INCREF(item);
    PyList_SetItem(list, 1, PyLong_FromLong(0L));
    PyObject_Print(item, stdout, 0);
    Py_DECREF(item);
}
```

Подводные камни (потоки)



<https://docs.python.org/3/extending/extending.html#thin-ice>

```
void
bug(PyObject *list)
{
    PyObject *item = PyList_GetItem(list, 0);
    Py_BEGIN_ALLOW_THREADS
    ...some blocking I/O call...
    Py_END_ALLOW_THREADS
    PyObject_Print(item, stdout, 0); /* BUG! */
}
```

oset



>>> c/oset

```
6 ▸ c ▸ oset ▸ main.py ▸ ...
1  import oset
2
3  s = oset.OSet()
4
5  s.add(1)
6  s.add(2)
7  s.add(3)
8  s.add(4)
9
10 print(1 in s)
11 print(2 in s)
12 print(3 in s)
13 print(4 in s)
14 print(5 in s)
15 print(repr(s))
```



Hello World



```
6 ▸ cython ▸ ≡ mod1.pyx
  1   print("Hello World")
  2   
```

```
6 ▸ cython ▸ + setup.py ▸ ...
  1   from distutils.core import setup, Extension
  2   from Cython.Build import cythonize
  3
  4   setup(
  5       name = 'mod1',
  6       version = '1.0',
  7       ext_modules = cythonize('mod1.pyx')
  8   )
```

Hello World



```
→ cython -a mod1.pyx
```

Generated by Cython 0.29.7

Yellow lines hint at Python interaction.

Click on a line that starts with a "+" to see the C code that Cython generated for it.

Raw output: [mod1.c](#)

```
+1: print("Hello World")
   if ( __Pyx_PrintOne(0, __pyx_kp_s_Hello_World) < 0) __PYX_ERR(0, 1, __pyx_L1_error)
```


primes



```
6 ▸ cython ▸ primes ▸ ≡ primes.pyx
1  def primes(int nb_primes):
2      ... cdef int n, i, len_p
3      ... cdef int p[1000]
4      ... if nb_primes > 1000:
5      ...     nb_primes = 1000
6
7      ... len_p = 0 # The current number of elements in p.
8      ... n = 2
9      ... while len_p < nb_primes:
10         ...     # Is n prime?
11         ...     for i in p[:len_p]:
12         ...         if n % i == 0:
13         ...             break
14
15         ...     # If no break occurred in the loop, we have a prime.
16         ...     else:
17         ...         p[len_p] = n
18         ...         len_p += 1
19         ...         n += 1
20
21     ... # Let's return the result in a python list:
22     ... result_as_list = [prime for prime in p[:len_p]]
23     ... return result_as_list
```


primes. Cython vs Python



```
~/Projects/_/advancedpython/6/cython/primes on □ master [?] via venv:6  
→ python -m timeit -s 'from primes_python import primes_python' 'primes_python(1000)'  
10 loops, best of 5: 31.6 msec per loop  
  
~/Projects/_/advancedpython/6/cython/primes on □ master [?] via venv:6 took 2s  
→ python -m timeit -s 'from primes import primes' 'primes(1000)'  
100 loops, best of 5: 2.66 msec per loop
```

Домашнее задание №4



[12 баллов] Необходимо реализовать класс Matrix, хранящий целые числа через c-extension. Должен поддерживать следующие операции:

- Сложение матрицы и матрицы
- Умножение и деление матрицы на целое число
- Умножение двух матриц
- [доп. +2 балла] Транспонирование матрицы

Также должно быть реализовано следующее:

- конструктор должен принимать массив массивов чисел и преобразовывать это во внутреннее хранилище
- repr и str над объектом матрицы
- [доп. +2 балла] оператор in, принимающие число и возвращающий True если такой элемент есть в матрице
- Должен быть реализован доступ по таплу координат в матрице. Например:
`m = Matrix([[1, 2], [3, 4]])`
`m[(0, 1)]` - должно вернуть 2

[3 балла] Реализовать умножение матриц на питоне и сравнить производительность кода на C и на Python.

Ссылки полезные



1. https://en.wikipedia.org/wiki/Foreign_function_interface
2. <https://docs.python.org/3/library/ctypes.html>
3. <https://cffi.readthedocs.io/en/latest/>
4. <https://docs.python.org/3/extending/extending.html>
5. https://www.tutorialspoint.com/python3/python_further_extensions.htm
6. https://cython.readthedocs.io/en/latest/src/tutorial/cython_tutorial.html

