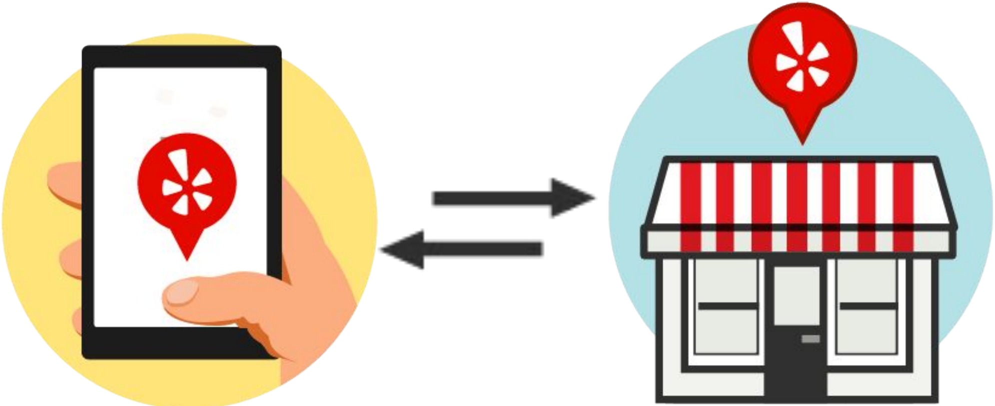
# Hack your Python

CUSEC 2019 Workshop



## Yelp’s Mission

Connecting people with great local businesses yelp.com/dataset_challenge

# Academic dataset from 10 cities across the globe!

- 6M reviews
- 1M business attributes
- 190K businesses
- 200K photos

**Your academic project, research or visualizations submitted by June 30, 2019**

=

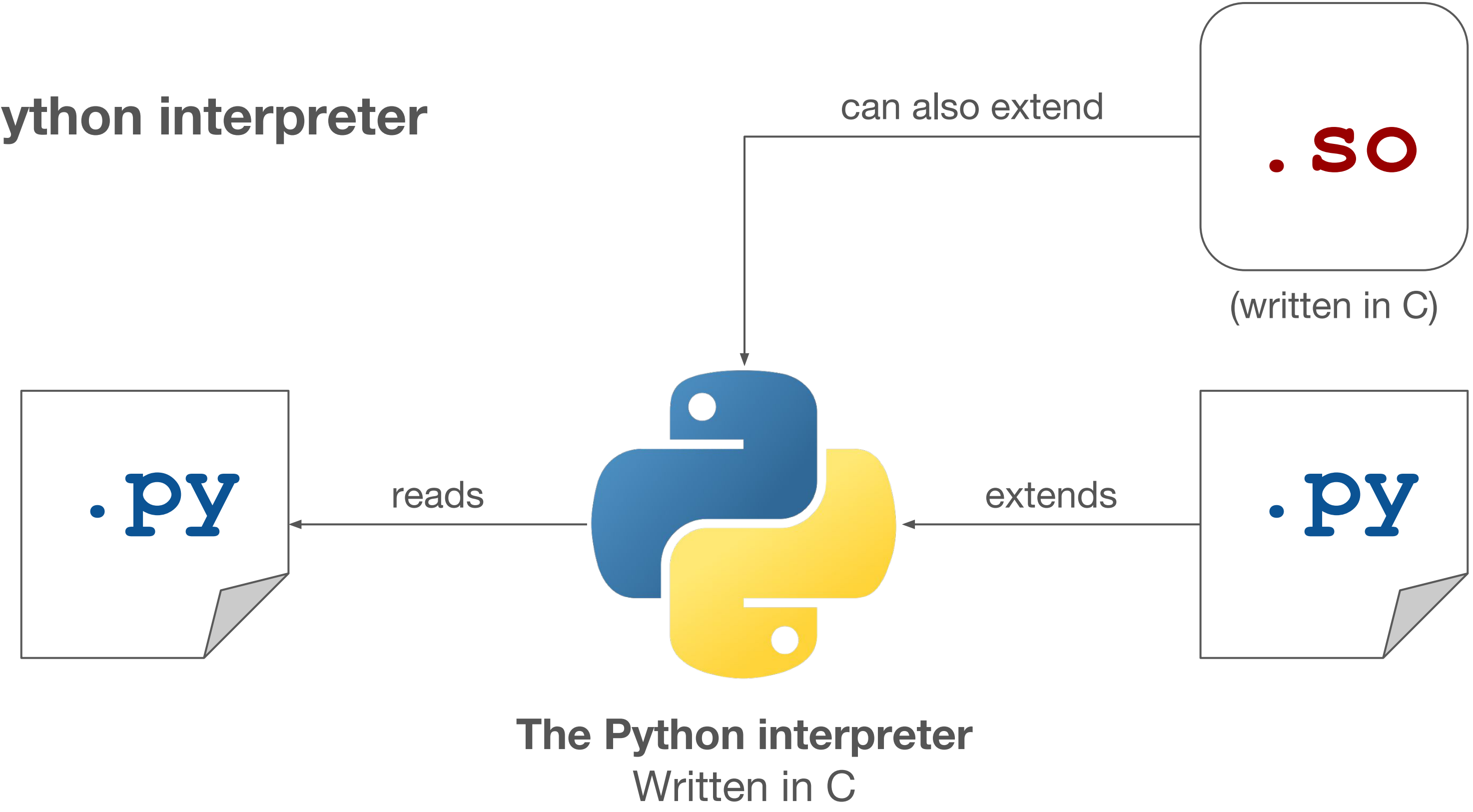
a $5,000 prize* !

*See full terms on website

# Extending Python

## with C/C++

# The Python interpreter



# Why we did it

**1. Speed up cryptographic operations**

We previously used a Python interface to OpenSSL and the thousands of calls we make per request cost hundreds of ms. We cut 90% of that time by rewriting the interface in C.

**2. Reduce memory usage**

Freezing garbage collection enabled us to reduce memory usage by as much as 40%, but this required hacking directly inside the interpreter.

# Today’s menu

### 1. Create a simple extension with C

We will build `yelp_arithmetic`, a tool with simple arithmetic functions to teach ourselves how to pass values between C and Python.

### 2. Use our extension to access Python internals

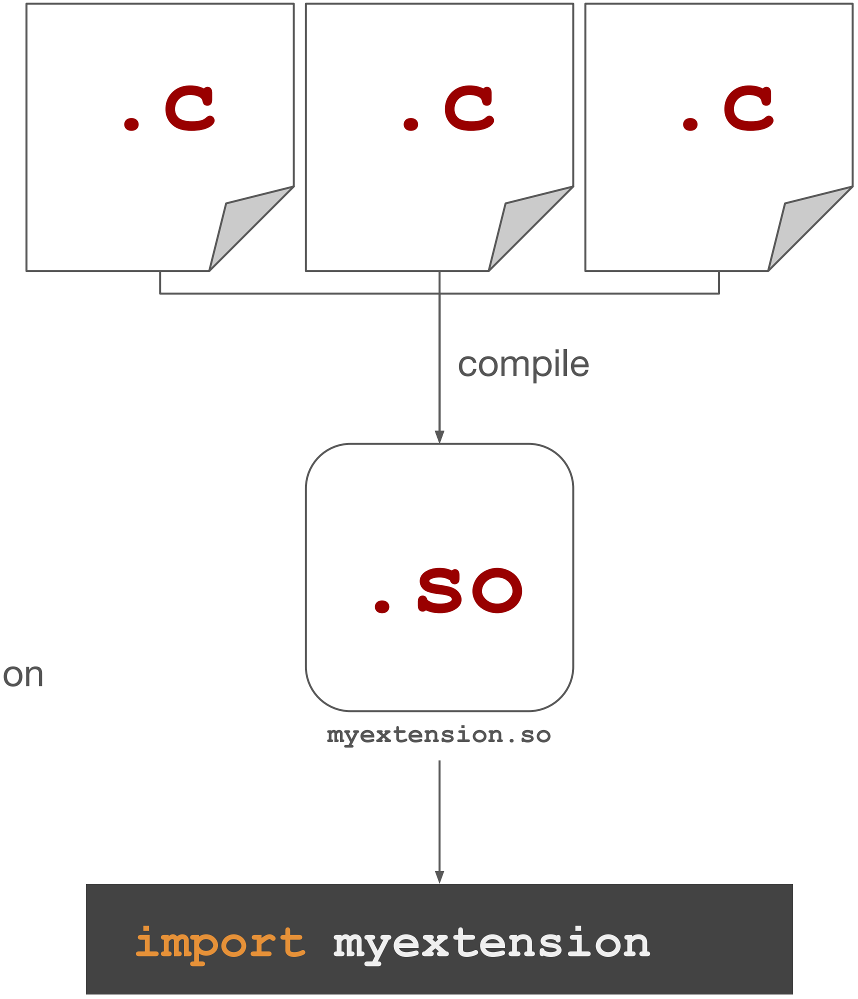
Working at a low level lets us reach deep inside the Python interpreter to change its core behaviour.

# Workflow

1. **Write**
   Your code in C/C++

2. **Compile**
   To a shared object (.so)

3. **Import**
   By using the name of your extension



```
import myextension
```

# Code structure

### Include API

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

### The function

```
PyObject* division(PyObject* self, PyObject* args) {

    // ...
```

### Mapping C functions to Python methods

```
static PyMethodDef yelpdivider_methods[] = {

    // ...
```

### Module initialization

```
PyMODINIT_FUNC PyInit_yelpdivider(void) {

    // ...
```

# Reading arguments

```
PyObject* division(PyObject* self, PyObject* args) {

    double a, b, result;

    PyArg_ParseTuple(args, "dd", &a, &b);

    // ...

}
```

# Returning values

```
PyObject* division(PyObject* self, PyObject* args) {
    double a, b, result;
    PyArg_ParseTuple(args, "dd", &a, &b);

    result = a / b;
    return Py_BuildValue("d", result);
}
```

# Exception handling

```
PyObject* division(PyObject* self, PyObject* args) {

    // ...



    if (b == 0) {

        PyErr_SetString(PyExc_ValueError,
                        "Cannot divide by zero");

        return NULL;

    }



    // ...

}
```

```
// returning None
// no `return` statement
Py_RETURN_NONE;
```

# Method mapping

```
PyMethodDef yelp_arithmetic_methods[] = {

    {
        "division",
        (PyCFunction)division,
        METH_VARARGS,
        "Divides A with B."
    },

}
```

```
struct PyModuleDef module = {

    // ...

}
```

# Module definition

```
PyMethodDef yelp_arithmetic_methods[] = {

    // ...

}



struct PyModuleDef module = {

    PyModuleDef_HEAD_INIT,
    "yelp_arithmetic",
    "Does arithmetic. lol",
    -1,
    yelp_arithmetic_methods,
    NULL,
    NULL,
    NULL,
    NULL

}
```

# Module initialization

```
struct PyModuleDef module = {

    PyModuleDef_HEAD_INIT,
    "yelp_arithmetic",
    "Does arithmetic. lol",
    -1,
    yelp_arithmetic_methods,
    NULL,
    NULL,
    NULL,
    NULL

}



PyMODINIT_FUNC PyInit_yelp_arithmetic(void) {

    return PyModule_Create(&module);

}
```

# setup.py
(compilation)

```
from setuptools import setup, Extension


yelp_arithmetic = Extension(name="yelp_arithmetic",
                            sources=["src/arithmetic.c"],
                            include_dirs=["src/"],
                            extra_objects=["-lcrypto"])



setup(name="yelp_arithmetic",
      # ...
      ext_modules=[yelp_arithmetic])
```

```
$ python setup.py install
```

# Let’s hack

# Extra features

**Full documentation:**
docs.python.org/3/c-api

```
// To increase/decrease the reference count for an object

Py_INCREF(obj);               Py_DECREF(obj);



// Create tuple: (string, bool, int)

Py_BuildValue("(spi)", str, 1, 128);

// For a str -> int dictionary: "{s:i,s:i}"



// Has an error occurred?

PyErr_Occurred(); // --> Exception type, or NULL



// Memory allocation: PyMem_(Malloc|Calloc|Realloc|Free)

PyMem_Malloc(n);              PyMem_Free(p);
```

alternative:

## Cython

```
def divide(a, b):

    return a / b
```

```
from Cython.Build import cythonize

setup(
    name="yelpdivider",
    # ...
    ext_modules=cythonize("yelpdivider.pyx")
)
```

```
def divide(double a, double b):

    cdef double result = a / b

    return result
```

alternative:

## Boost for C++

```
double divide(double a, double b) {
    return a / b;
}

#include <boost/python.cpp>
BOOST_PYTHON_MODULE(yelpdivider) {
    boost::python::def("divide", divide);
}
```

# Let’s hack

## Part II

# We're Hiring!

www.yelp.com/careers/

fb.com/YelpEngineers

@YelpEngineering

engineeringblog.yelp.com

github.com/yelp

# Questions/Suggestions?

yann@yelp.com

yelp

# Thank you.