Binding the Sea

Creating C bindings for python
The why and the how of it.

But first...

Hi!

@forslund

Python FFI

- Foreign Function Interface

  A foreign function interface (FFI) is a mechanism by which a program written in one programming language can call routines or make use of services written in another.

- ctypes – standard library, verbose
- cffi – available through pypi, simpler, api

Libraries

- lib*.so
- *.dll

gcc *my_c_file.c* **-o** *libmy_lib.so* **-fpic -shared**

Not only c...

- C++

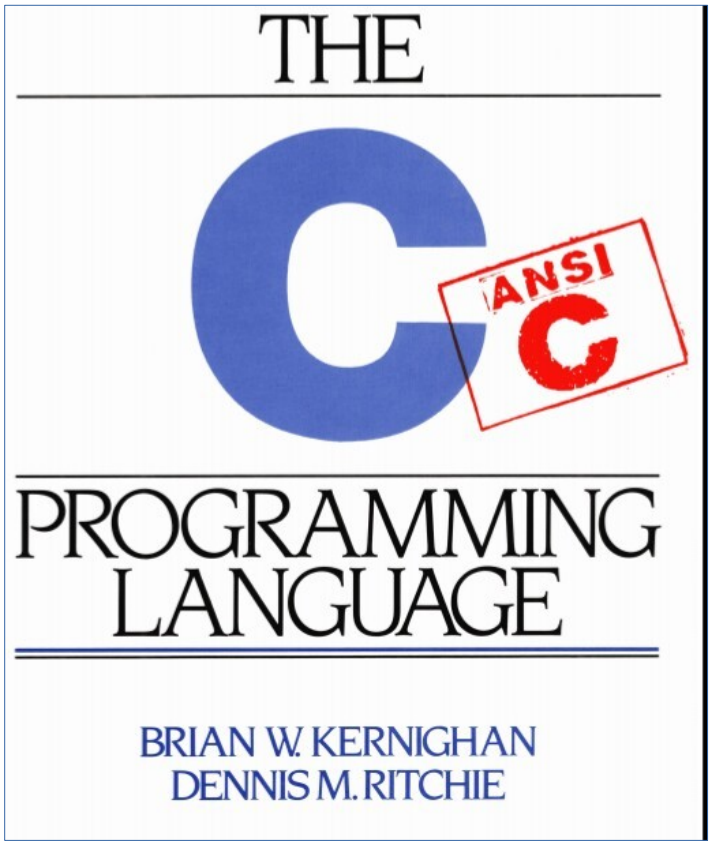
```
extern "C" {
    void no_mangle();

    int accessible_int;
}
```

- Rust

```
#[no_mangle]
pub extern "C" fn rusty_func()
```

- Etc ...

The Example


THE
C
ANSI
C
PROGRAMMING
LANGUAGE
BRIAN W. KERNIGHAN
DENNIS M. RITCHIE

ctypes

Getting started with ctypes

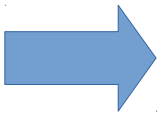
- Load the library
- Define returnvalues ...
- … and arguments

```
import ctypes

# Load the library
clist = ctypes.CDLL('../lib/liblist.so')

# Define returns and arguments for functions
clist.new_list.restype = ctypes.POINTER(element)
clist.append_val.argtypes =(ctypes.POINTER(element), ctypes.c_int)
```

Structures

C struct

```
struct my_struct {
    int val;
    char *string;
};
```



ctypes reporesentation

```
class my_struct(ctypes.Structure):
    _fields_ = [
        ('val', ctypes.c_int),
        ('string', c_char_p)
    ]
```

Pointers

- `POINTER` macro
- Dereferencing

  `object.contents`

  `object[0]`
- `ctypes.cast()`

Callbacks

ctypes.CFUNCTYPE(ret, [args…])

```
@ctypes.CFUNCTYPE(None)
def callback_function()
    print('C called, it's for you')
```

cffi

Getting started with cffi

```
from cffi import FFI
ffi = FFI()

header = """
typedef struct {
    int *val;
    void *next;
} element_t;
element_t * new_element(void);
element_t * new_list(void);
void append_val(element_t *list, int val);
element_t *get_element_indexed(element_t * list, int index);
"""

ffi.cdef(header)
linked_list = ffi.dlopen('../lib/liblist.so')
```

Structures

- Nothing to worry about

Pointers

- Dereferencing

  `object[0]`
- `ffi.cast()`

callbacks

```
@ffi.callback("int(int, int)")
def myfunc(x, y):
   ...
```

API mode

- Builds native C extension
- Faster

Making it pythonic

- IMPORTANT!!! DO NOT SKIP!
- Hide the C
- Implement ”expected” functionallity
- __del__()

Keep it Safe

- Segmentation faults
- Division by 0
- Etc...

Thank you for listening