Rust is the Next Python FFI

Shmuel Amar

PyCon Israel

Who?

Shmuel Amar

Architect@ProofPoint Research Team

Python ∪ Data ∪ Security

Agenda

- Reminder - Python FFI
- Why Rust is Awesome?
- Rust Example
- Comparing Rust to C
- Packaging: Rust Crate -> Python Package in 60 Sec
- Summary

1

Reminder - Python FFI

Python Foreign Function Interface

What is Python FFI?

- Python has C API - #include <Python.h>
- Mostly used from C/C++
- Commonly used for:
  - Wrapping existing C/C++ libraries
  - Improving performance

Mission: Random4

xkcd.com/221

```
int getRandomNumber()
{
    return 4;  // chosen by fair dice roll.
               // guaranteed to be random.
}
```

Alt text: *RFC 1149.5 specifies 4 as the standard IEEE-vetted random number.*

Random4 Python

```
def random4():
    """get random num"""
    return 4
```

Random4 C

```c
#include <Python.h>
static PyObject* random4(PyObject *self, PyObject *args) {
  return PyLong_FromLong(4);
}
static PyMethodDef module_methods[] = {
  {"random4", random4, METH_NOARGS, "get random num"},
  {NULL, NULL, 0, NULL}
};
```

```c
static struct PyModuleDef random4c_definition = {
  PyModuleDef_HEAD_INIT, "random4c", "random4 module", -1, module_methods
};
PyMODINIT_FUNC PyInit_random4c(void) {
  Py_Initialize();
  return PyModule_Create(&random4c_definition);
}
```

Why Rust?

Its Fast, Safe & Awesome

Ferris

Why Rust? its Fast!

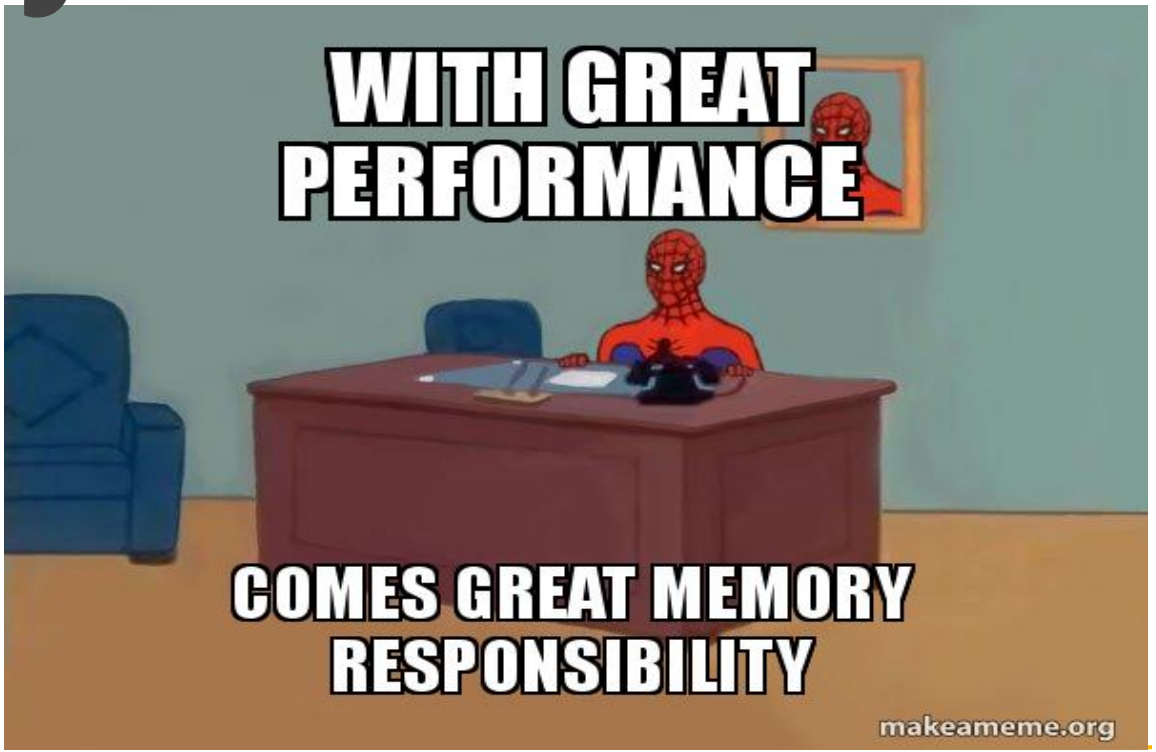
- A System Programming Language
- C/C++ comparable performance
- Compiles to Machine Code
- Minimal runtime, no garbage collector
- Stack allocations by default

Why Rust? its Safe!

- **Memory Safety**
- No Nulls & Enum Exhaustion
- No Implicit Numbers Casts

* nothing is really *safe* in the digital world only *safer*
* unsafe rust - for raw speed / using native unsafe code

Why Rust? its Safe!

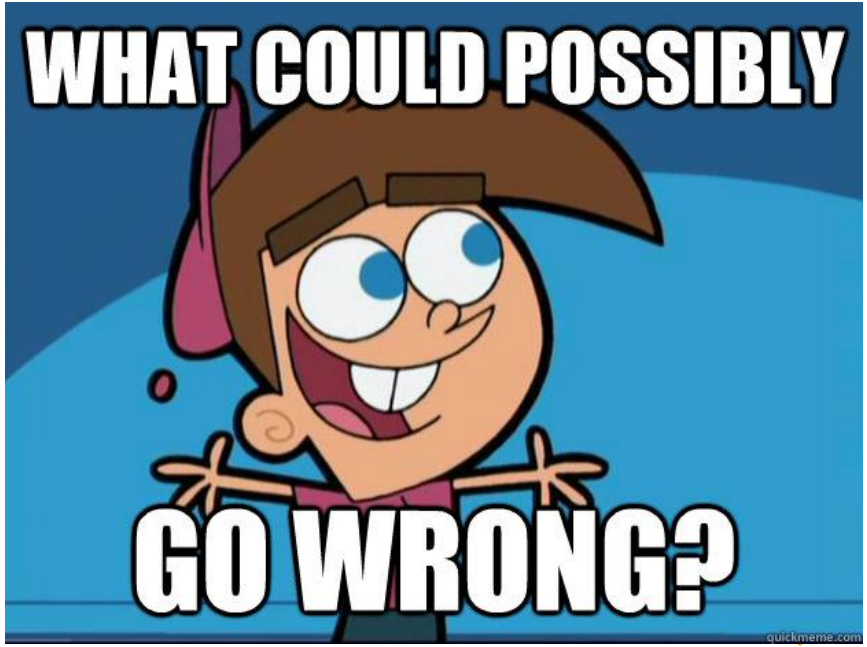


Why Rust? its Safe!

```
char *p;
p = malloc(12);
strcpy(p, "hello world");
printf("%s\n", p);
free(p);
p = NULL;
```

Why Rust? its Safe!

```
char *p;
p = malloc(12);
strcpy(p, "hello world");
printf("%s\n", p);
free(p);
p = NULL;
```



Why Rust? its Safe!

```
char *p; // 1. uninitialized pointer
p = malloc(12); // 2. uninitialized memory
strcpy(p, "hello world"); // 3. buffer overflow
printf("%s\n", p);
free(p); // 4. dangling pointer
// free(p); // 5. double free
p = NULL;
```

Why Rust? its Safe!

Why Rust? its Safe!

- Memory Safety
- **No Nulls & Enum Exhaustion**
- No Implicit Numbers Casts

Null, My Billion Dollar Mistake - Tony Hoare

* nothing is really *safe* in the digital world only *safer*
* unsafe rust - for raw speed / using native unsafe code

Why Rust? its Safe!

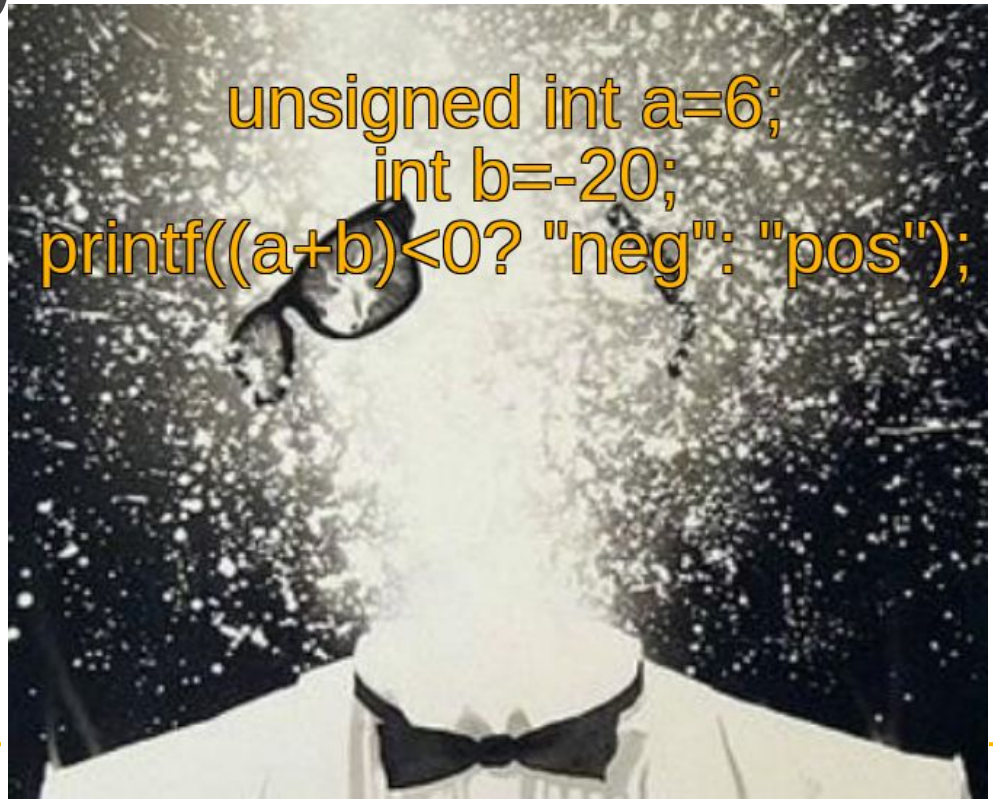
```
enum Option<T> {
   Some(T),
   None,
}
let name = Option::Some("world");
match name {  // remove one arm is a compilation error
  Some(s) => println!("hello {}", &s),
  None => println!("bye"),
};
```

Why Rust? its Safe!

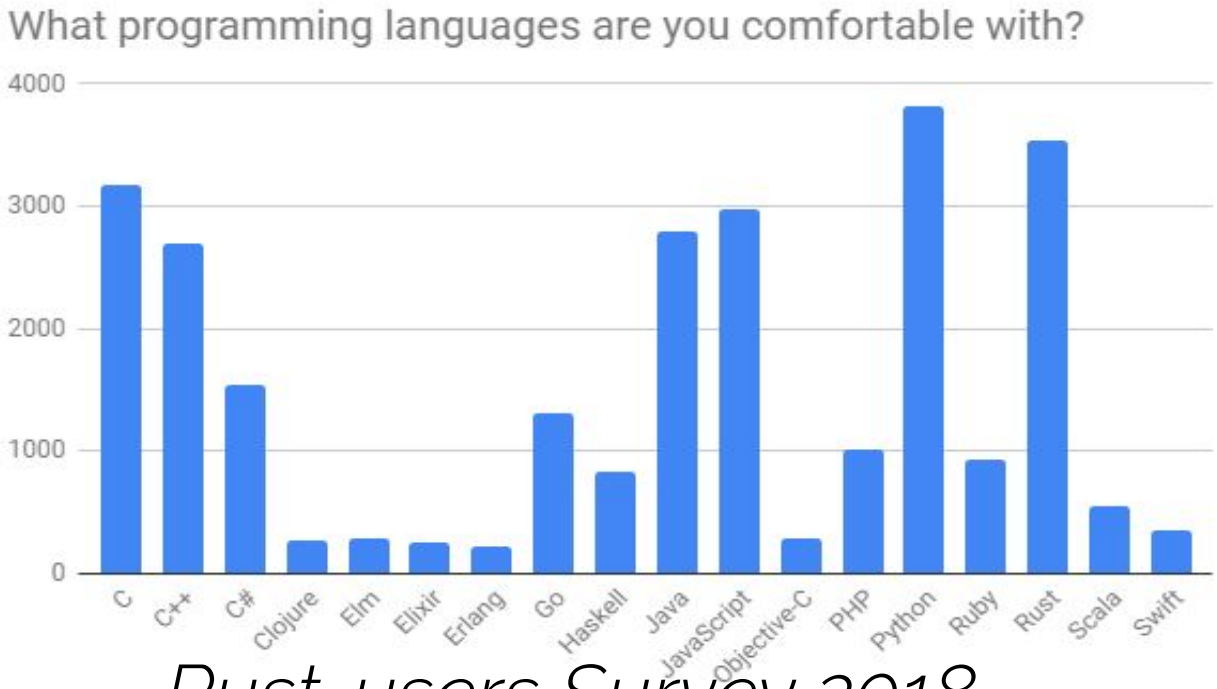
- Memory Safety
- No Nulls & Enum Exhaustion
- **No Implicit Numbers Casts**

* nothing is really *safe* in the digital world only *safer*
* unsafe rust - for raw speed / using native unsafe code

Why Rust? its Safe!



Why Rust? its Awesome!



Rust-users Survey 2018

Why Rust? its Awesome!

SO Survey: Rust Most Loved 4th year in a row

2019 Survey

Why Rust? its Awesome!

- Open Source - MIT / Apache license
- Type Inference
- Traits - Lightweight OOP
- Cargo - single CLI for Everything
- Functional Programming
  - Variables are immutable by default
  - Functions are first class citizen
  - Anonymous functions

3

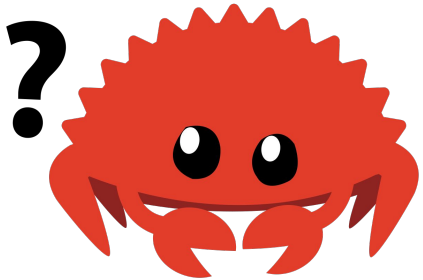
Rust Example

Fibonacci Sequence

Reminder - Fibonacci

$$F_n = F_{n-1} + F_{n-2}$$

0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55



```rust
pub struct Fib {
    a: u64,
    b: u64,
}
impl Fib {
    pub fn new() -> Fib {
        Fib { a: 0, b: 1 }
    }
}
```

```rust
impl Iterator for Fib {
    type Item = u64;
    /// get the next number on fib sequence
    fn next(&mut self) -> Option<Self::Item> {
        let cur = self.a;
        self.a = self.b;
        self.b = cur + self.b;
        Some(cur)
    }
}
```

```rust
fn main() {
    println!("The Fibonacci Sequence:");
    for (i, x) in Fib::new().take(10).enumerate() {
        println!("{}: {}", i, x);
    }
}
```

```
$ cargo run --release
The Fibonacci Sequence:
0: 0, 1: 1, 2: 1, 3: 2, 4: 3, 5: 5, 6: 8, 7: 13, 8: 21, 9: 34
```

Rust Example

Sum Even Fibonacci Numbers < 4 Million

Fibonacci - Python

```
def fib():
    a, b = 0, 1
    while True:
        yield a
        a, b = b, a + b
```

Fibonacci - Python

```python
from itertools import takewhile

def sum_fib_even(max_num):
  seq = takewhile(lambda x: x < max_num, fib())
  return sum(filter(lambda x: x % 2 == 0, seq))
```

Fibonacci - Rust

```rust
pub fn sum_fib_even(max_num: u64) -> u64 {
    let res = Fib::new()
        .take_while(|x| x < &max_num)
        .filter(|x| x % 2 == 0)
        .sum();
    res
}
```

Fibonacci - Rust

```
pub fn sum_fib_even(max_num: u64) -> u64 {
   let (mut a, mut b, mut sum) = (0, 1, 0);
   let mut tmp;
   while a < max_num {
      if a % 2 == 0 { sum += a; }
      tmp = a;
      a = b;
      b += tmp;
   }
   sum
}
```

Rust Crates for Python

- Wrapping Rust as native Python extension
  - **PyO3**
  - Rust-cpython
- Packaging Tools
  - **PyO3-pack**
  - rust-setuptools
  - milksnake

Fibonacci - Rust PyO3

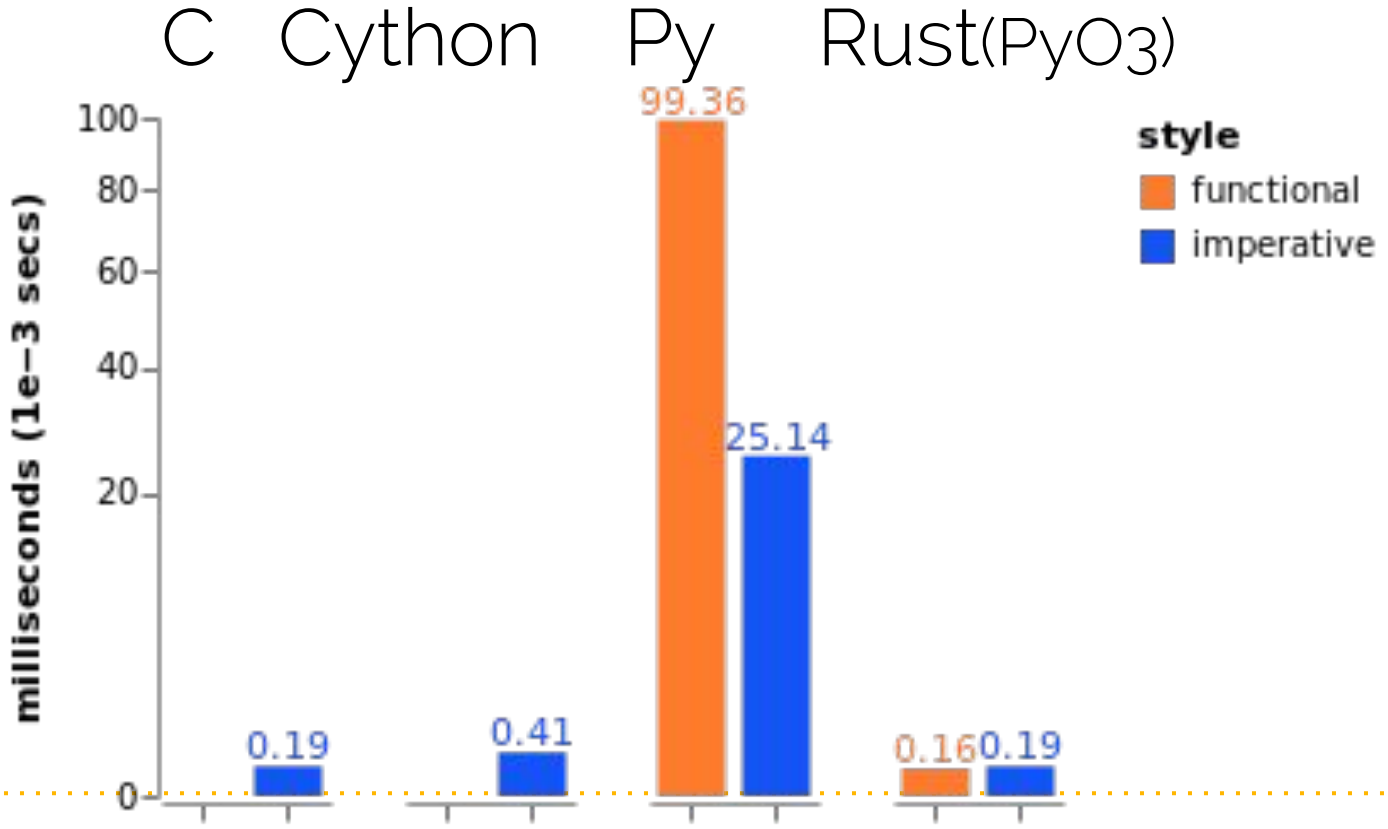
```
/// sum even fibonacci numbers under `max_num`
#[pyfunction]
fn sum_fib(max_num: u64) -> PyResult<u64> {
    Ok(fibrust::sum_fib_even(max_num))
}
/// fibonacci Python module impl in Rust
#[pymodule]
fn fibpyo3(_: Python, m: &PyModule) -> PyResult<()> {
    m.add_wrapped(wrap_pyfunction!(sum_fib))?;
   Ok(())
}
```

Fibonacci - Benchmarks

- 10k calls to sum_fib_even(4000000)
- Called from python, multiple rounds
- Mean time with pytest-benchmark

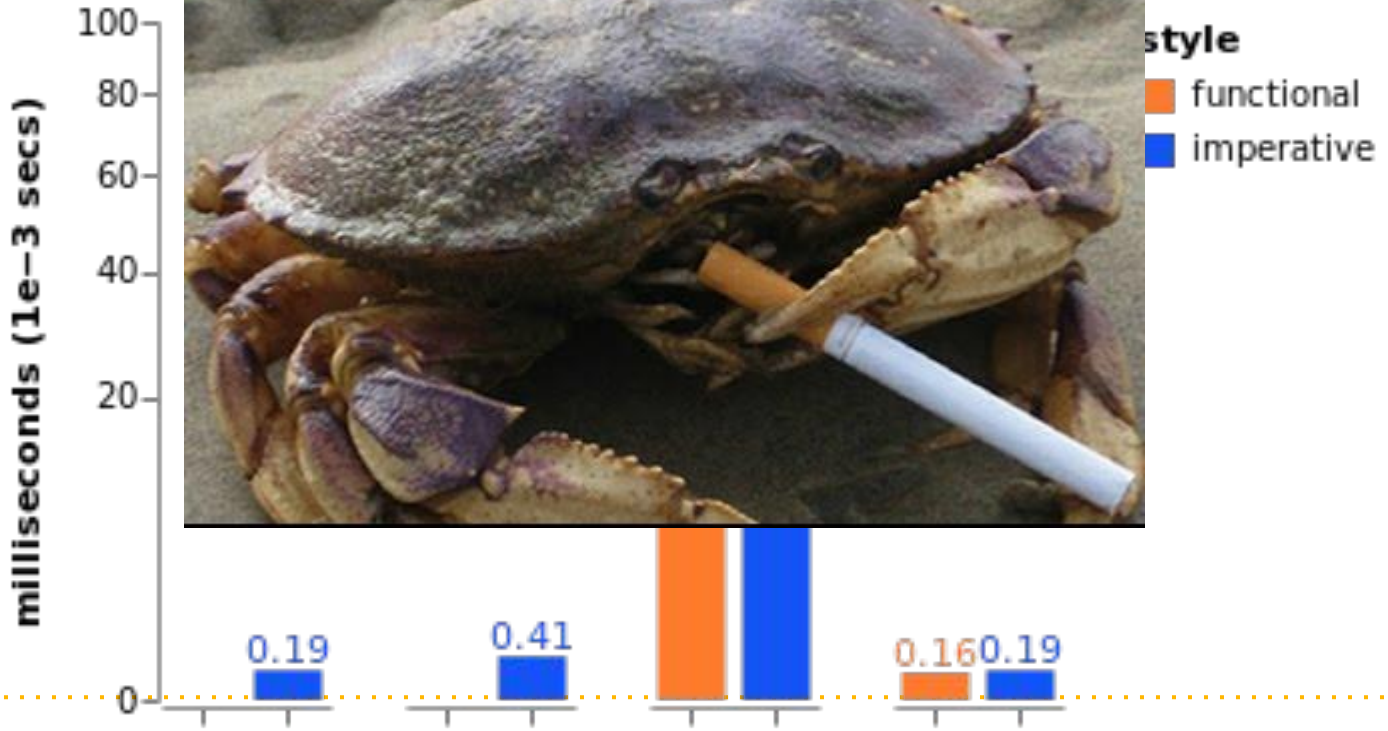
Talk is cheap, show me the benchmarks

Fibonacci - Benchmarks

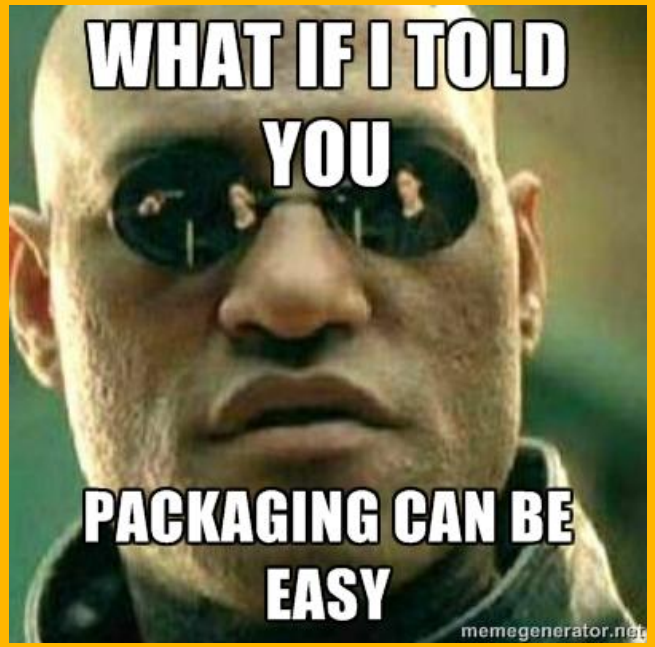


Fibonacci - Benchmarks

C Cython Py Rust(PyO3)



5



Build Tools

How to Package Our Rusty Extensions

Packaging Rust PYO3-pack

```
# Cargo.toml
[package]
name = "fibpyo3"
version = "0.1.1"
authors = ["shmuelamar"]
edition = "2018"
description = "Fibonacci impl in Rust"
readme = "README.md"
```

```
[lib]
name = "fibpyo3"
crate-type = ["cdylib"]
```

Packaging Rust PYO3-pack

```
$ pyo3-pack build -i python3.7
$ pip install target/wheels/fibpyo3-*.whl
$ python
>>> import fibpyo3
>>> fibpyo3.sum_fib_even(4_000_000)
4613732
```

$ pyo3-pack publish

Navigation

- Project description
- Release history
- Download files

Statistics

View statistics for this project via Libraries.io, or by using Google BigQuery

Meta

Author: shmuelamar <shmulikamar@gmail.com>

Maintainers

shmuelamar

Project description

Welcome to FibPyO3 Packge!

Example Usage

```
import fibpyo3

assert fibpyo3.sum_fib_even(4000000) == 4613732
assert fibpyo3.sum_fib_even_functional(4000000) == 4613732
```

Build New Release

```
pyo3-pack build -i python3.7
```

Install Wheel

```
pip install ./target/wheels/fibpyo3-0.1.0-cp37-cp37m-manylinux1_x86_64.whl
```

Have Fun!

Summary

Whats Next?

Summary

- C for Python is Unsafe and Tricky
- Rust is Fast, Safe and Awesome
- PyO3 & PyO3-Pack Wraps Rust for Python Easily

Whats Next?

- PyO3 & Rust-CPython Docs
  - Wrapping Classes & Exceptions
  - Calling Python From Rust
- The Rust Book
- Rust By Example
- Talk Materials on GitHub

Thanks!

```
>>> qs = input('Questions?')
```