Speeding up 🐍 Python

with 🦀 Rust

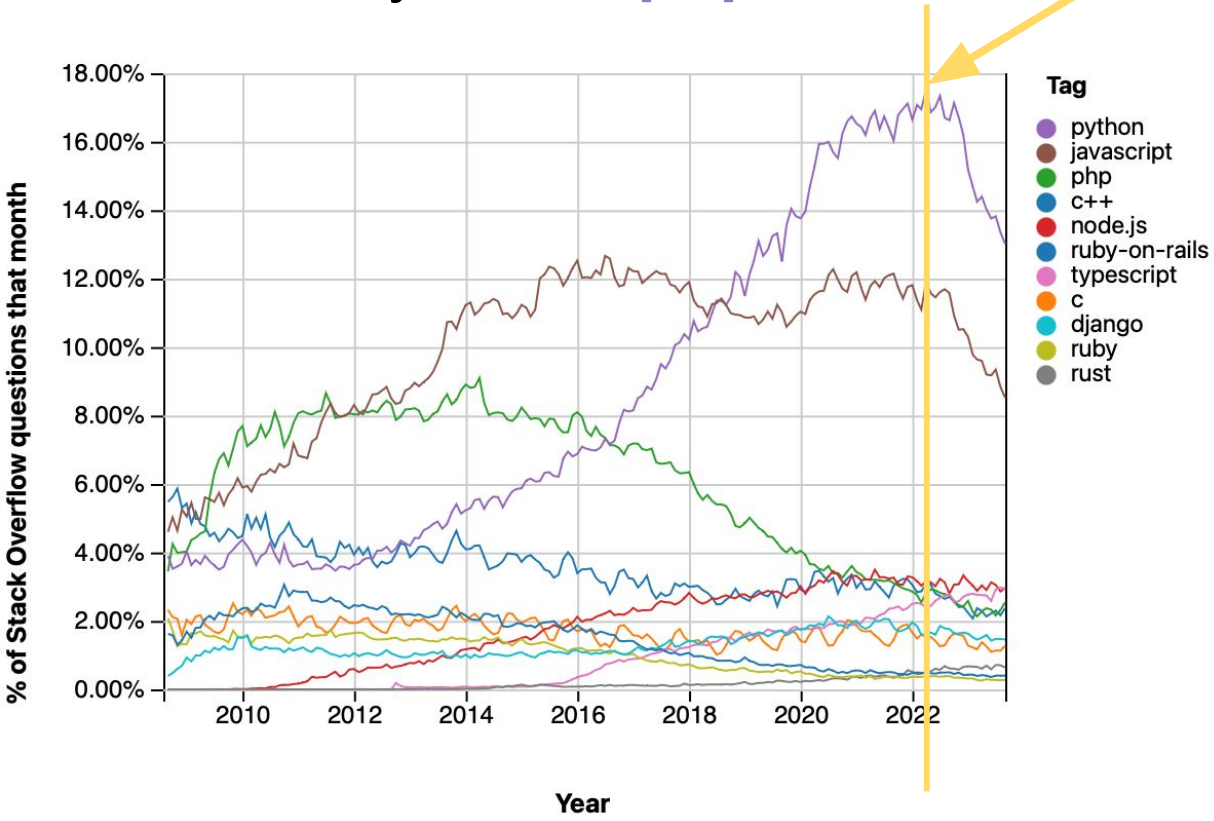
Mahmoud Saada (Moody)

- Grew up in Egypt
- CS researcher at ASU
- SE/DX/SRE Engineer
  - Kubernetes
  - OpenTelemetry
  - Gitops/Flux
- Industries
  - HR
  - Fintech
  - AI
- Google Scholar: covid-19 ML research
- Musician: The Shining Hours
- Awful DOTA2 player
- https://saada.dev

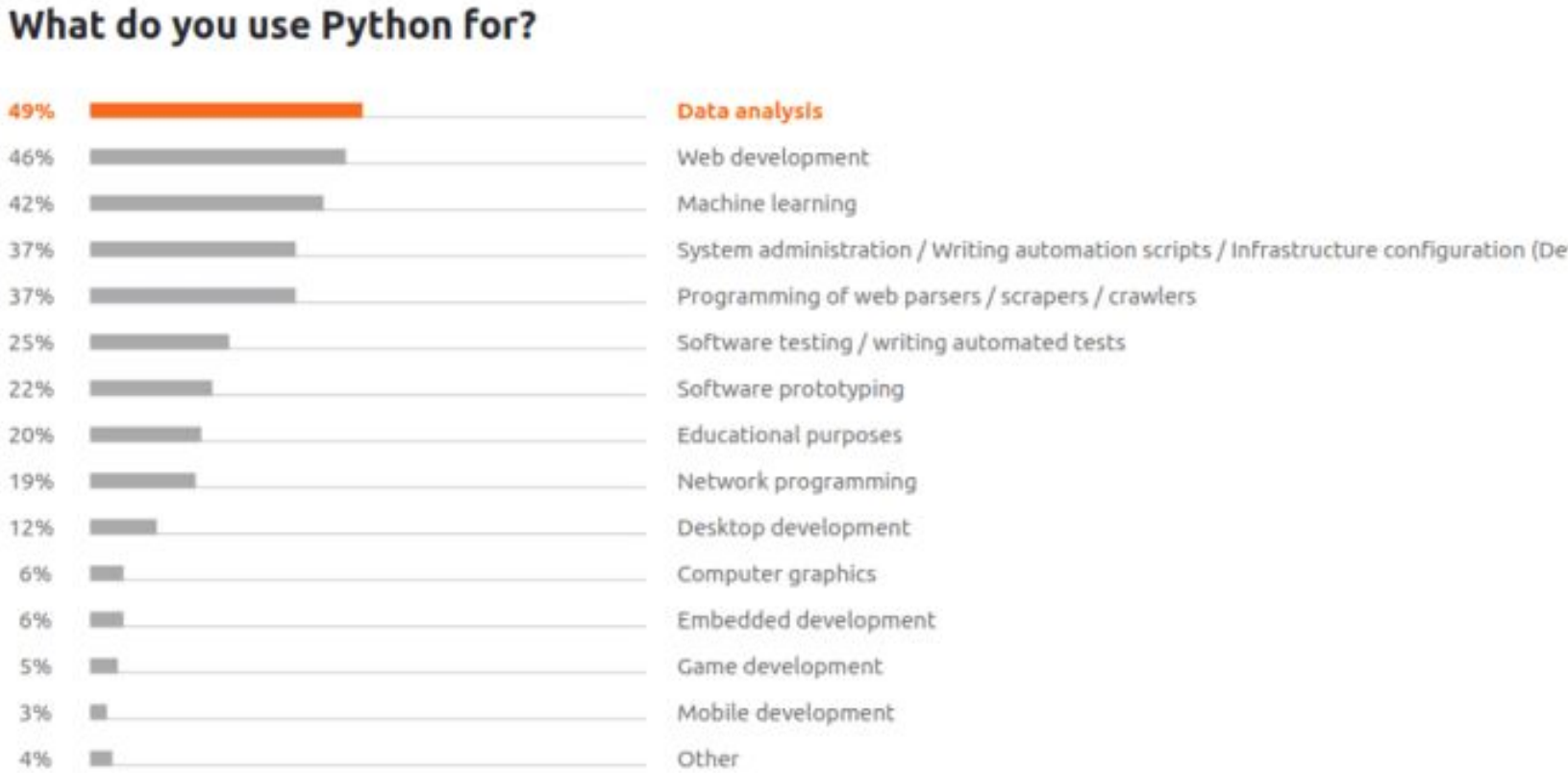
WTF is up with Python

Python is popular

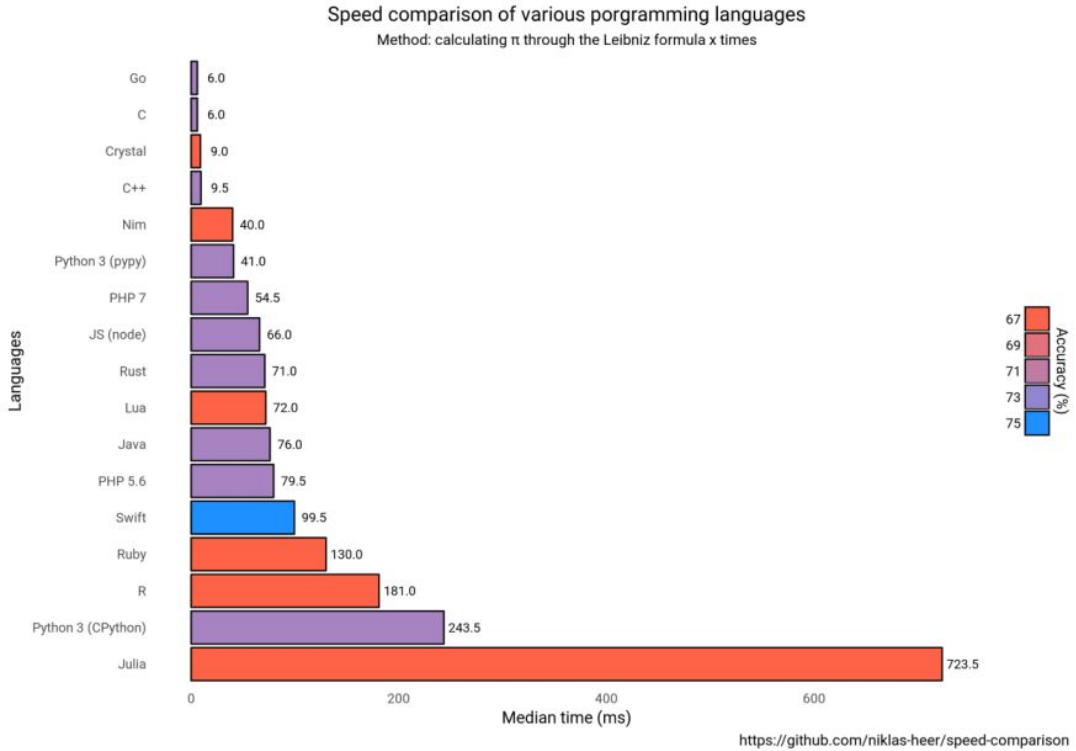
ChatGPT goes public!



Python is useful

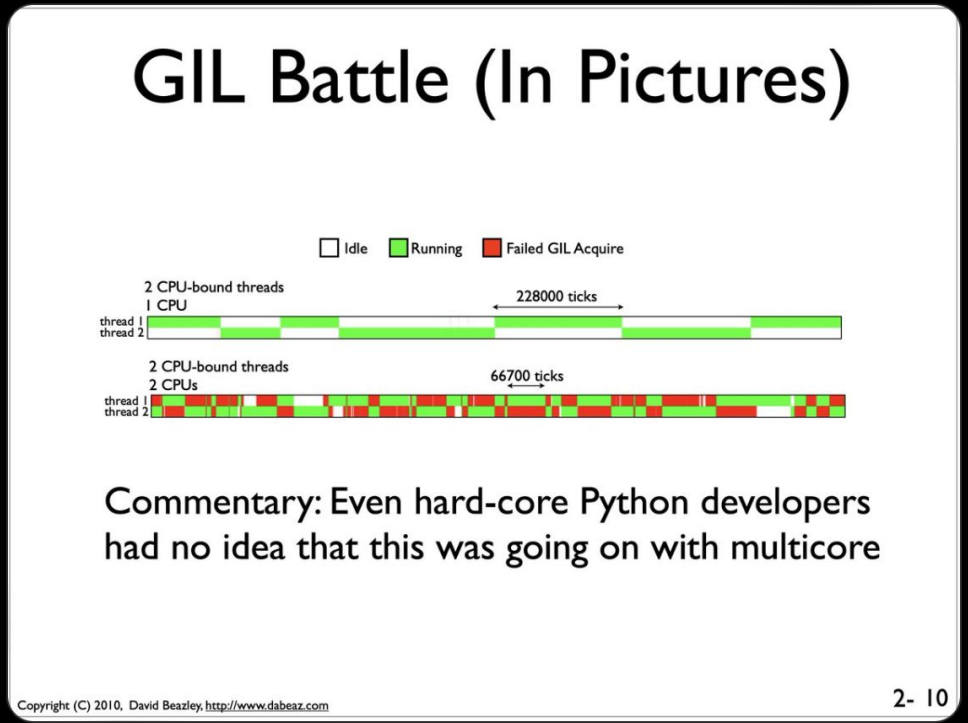


Python is **slow**



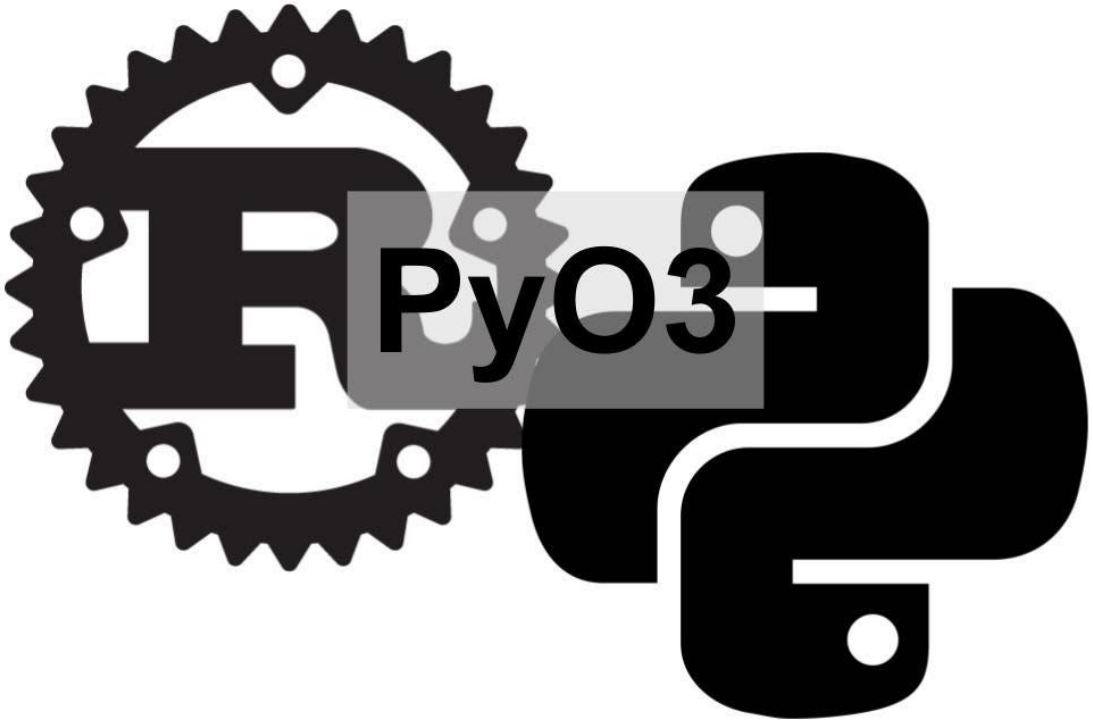
Pietro De Nicolao
@pietrodn

Basically, in #Python only one thread can run the interpreter at the same time (the famous Global Interpreter Lock).
Two cores may fight for the GIL with a very bad timing, failing to acquire it for hundreds of times and wasting a lot of CPU.
One of the threads starves.

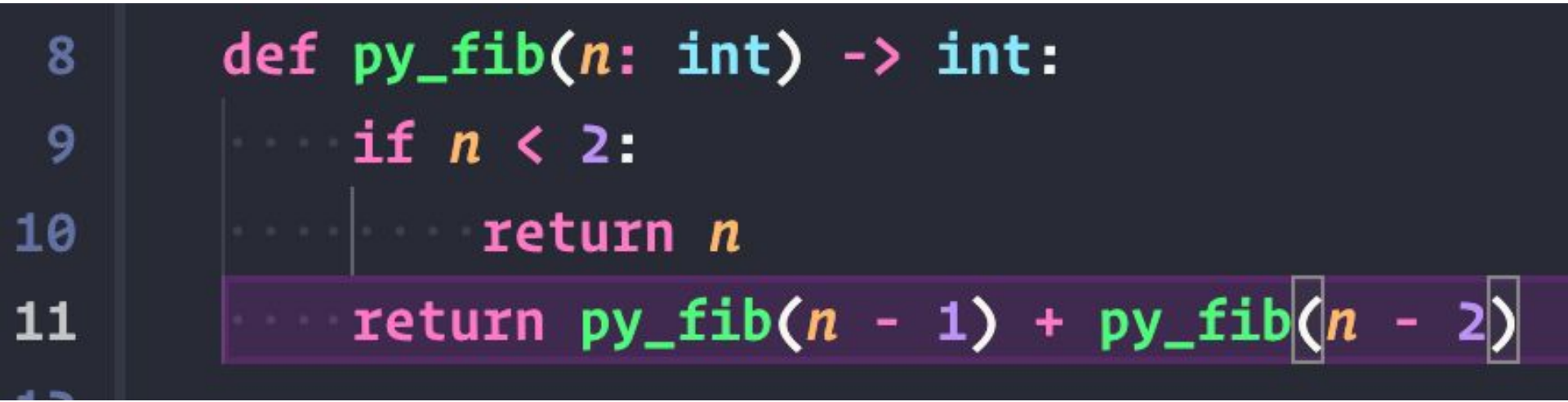


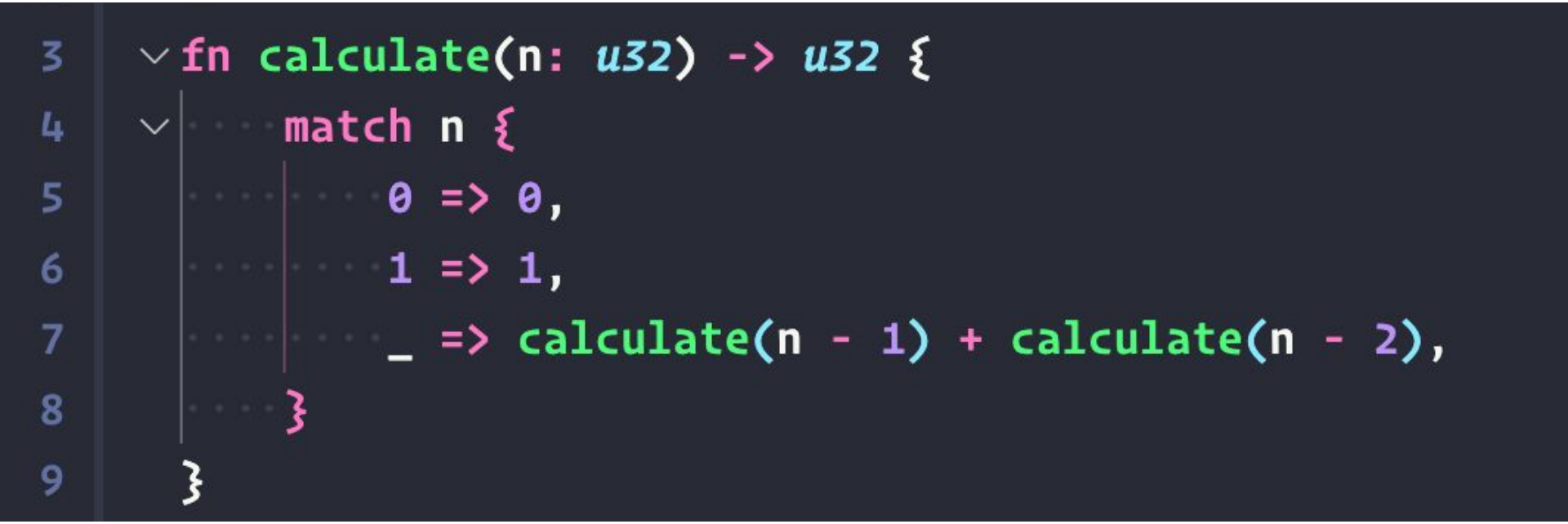

What should I do?

Alternatives



Mojo 🔥


8 def py_fib(n: int) -> int:
9 if n < 2:
10 return n
11 return py_fib(n - 1) + py_fib(n - 2)


3 fn calculate(n: u32) -> u32 {
4 match n {
5 0 => 0,
6 1 => 1,
7 _ => calculate(n - 1) + calculate(n - 2),
8 }
9 }

```
pip install maturin
mkdir fib && cd fib
maturin init # scaffold stuff
```

```rust
use pyo3::prelude::*;
```



```rust

fn calculate(n: u32) -> u32 {
    match n {
        0 => 0,
        1 => 1,
        _ => calculate(n - 1) + calculate(n - 2),
    }
}

/// Calculate the nth Fibonacci number.
#[pyfunction]
fn fibonacci(n: u32) -> PyResult<u32> {
    Ok(calculate(n))
}

/// A Python module implemented in Rust.
#[pymodule]
fn fib(_py: Python, m: &PyModule) -> PyResult<()> {
    m.add_function(wrap_pyfunction!(fibonacci, m)?)?;
    Ok(())
}
```

```
maturin develop --release
```

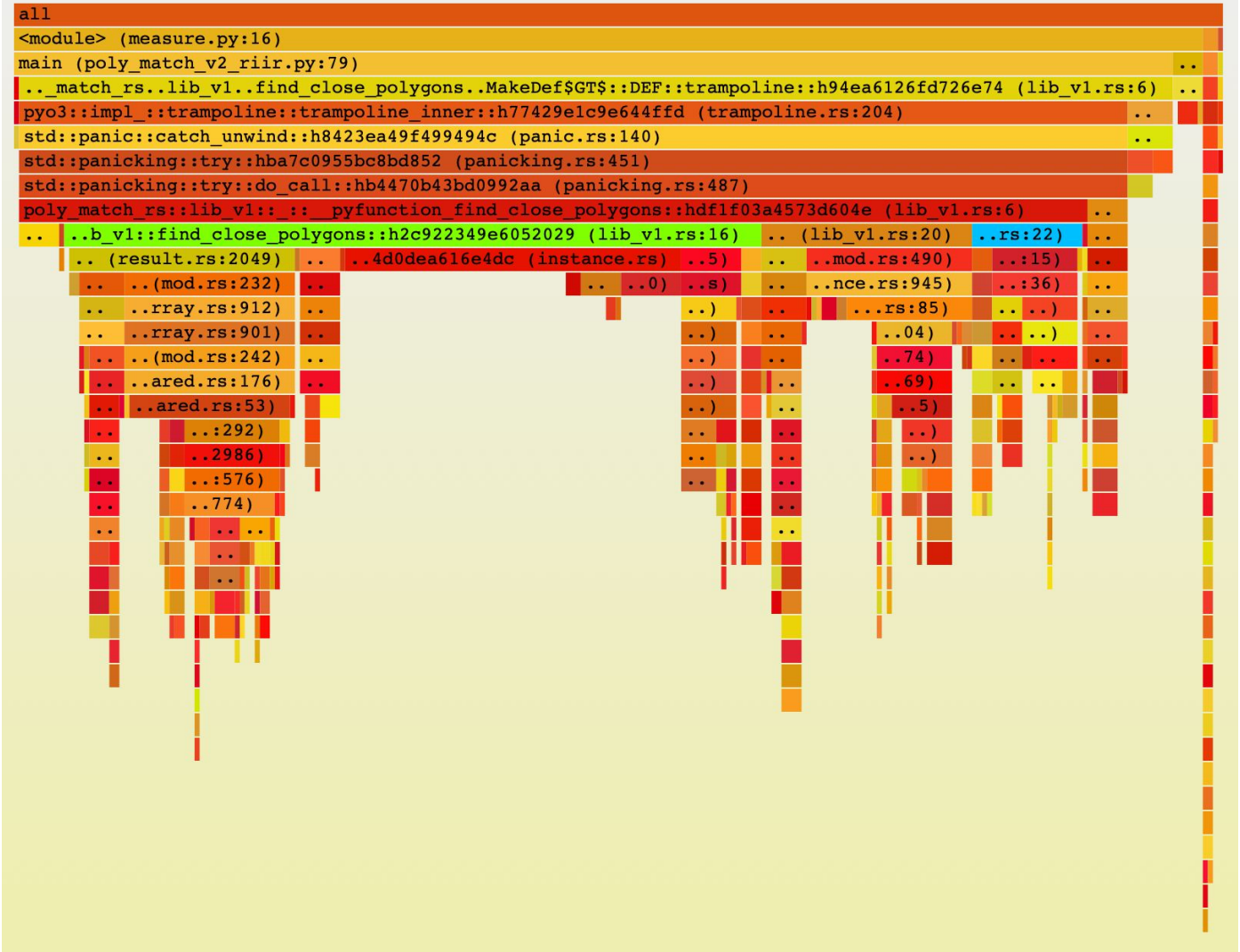
```python
import fib


def rust_fib(n: int) -> int:
    return fib.fibonacci(n)


def py_fib(n: int) -> int:
    if n < 2:
        return n
    return py_fib(n - 1) + py_fib(n - 2)


def test_rust_fib() -> None:
    assert rust_fib(10) == py_fib(10)

```

fib(40)

Vanilla Python: 18.99s
Cythonized Python: 6.52s
Rust PyO3: 0.63282s 🚀🚀🚀 *>40x faster*

```
# added to Cargo.toml
[profile.release]
debug = true         # Debug symbols for our profiler.
lto = true           # Link-time optimization.
codegen-units = 1    # Slower compilation but faster code.
```

```
pyspy record --native -- python fib.py
```


all
<module> (measure.py:16)
main (poly_match_v2_riir.py:79)
..match_rs..lib_v1..find_close_polygons..MakeDefGT::DEF::trampoline::h94ea6126fd726e74 (lib_v1.rs:6)
pyo3::impl_::trampoline::trampoline_inner::h77429e1c9e644ffd (trampoline.rs:204)
std::panic::catch_unwind::h8423ea49f499494c (panic.rs:140)
std::panicking::try::hba7c0955bc8bd852 (panicking.rs:451)
std::panicking::try::do_call::hb4470b43bd0992aa (panicking.rs:487)
poly_match_rs::lib_v1::_::__pyfunction_find_close_polygons::hdf1f03a4573d604e (lib_v1.rs:6)
..b_v1::find_close_polygons::h2c922349e6052029 (lib_v1.rs:16)
.. (lib_v1.rs:20)
..rs:22)
.. (result.rs:2049)
..4d0dea616e4dc (instance.rs)
..mod.rs:490)
..(mod.rs:232)
..nce.rs:945)
..rray.rs:912)
...rs:85)
..rray.rs:901)
..04)
..(mod.rs:242)
..74)
..ared.rs:176)
..69)
..ared.rs:53)
..5)
..:292)
..2986)
..:576)
..774)

```
maturin build --release
maturin publish
```

Thank you!

https://pyo3.rs

Making Python 100x faster with less than 100 lines of Rust

https://saada.dev