BLAZING FAST, MINIMAL CHANGE

SPEED UP YOUR CODE BY REFACTORING TO RUST

RUST!

RUST!

- Fast

RUST!

- Fast
- Faster than Python/Ruby

RUST!

- Fast
- Faster than Python/Ruby
- Let's start over with Rust!

FULL REWRITES

FULL REWRITES

- Blow past deadlines

FULL REWRITES

- Blow past deadlines
- New bugs

FULL REWRITES

- Blow past deadlines
- New bugs
- Don't fix architecture

MICROSERVICE?

MICROSERVICES

NOT ALWAYS BEST

FFI REFACTORING

Rewrite a little bit, connect via FFI

FEASIBILITY

REALITIES

REALITIES

- More complex deployments

REALITIES

- More complex deployments
- Can add bugs

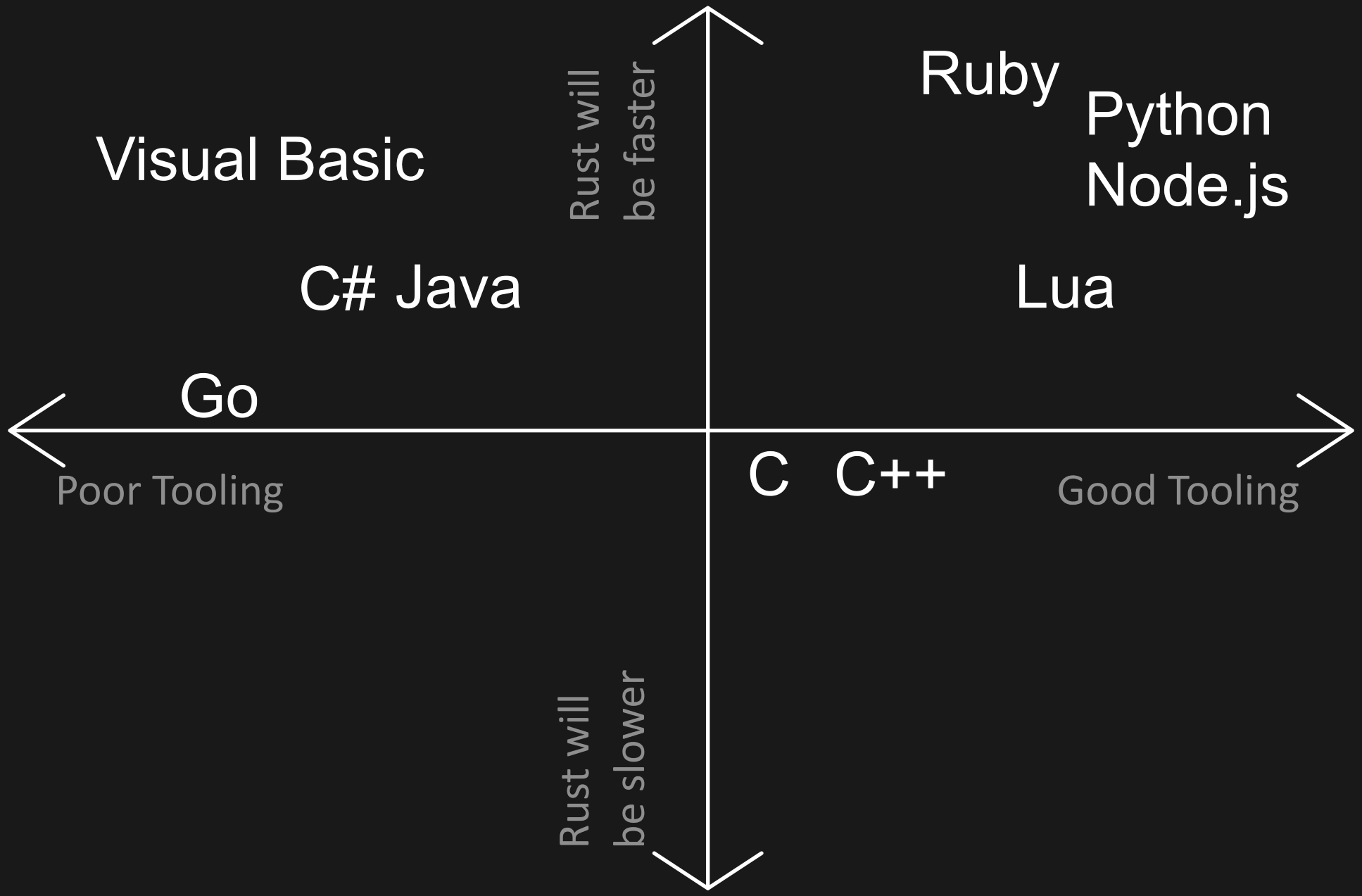
WHAT'S A GOOD PROJECT?

MICROSERVICES

VS

MONOLITHS

WHAT DO YOU HAVE TODAY?


Rust will
be faster
Ruby
Python
Node.js
Visual Basic
C# Java
Lua
Go
Poor Tooling
C C++
Good Tooling
Rust will
be slower

EXAMPLE

```python
@app.route("/stats", methods=['POST'])
def stats_py():
    numbers = sorted(request.get_json()['numbers'])

    min = numbers[0]
    max = numbers[-1]

    return jsonify({
        'range': float(max - min),
        'quartiles': statistics.quantiles(numbers),
        'mean': statistics.mean(numbers),
        'stddev': statistics.stdev(numbers),
    })
```

LET'S DO IT!

THE RUST BOOK

doc.rust-lang.org/book

```
$ cargo new --lib rstats
$ tree
.
├── Cargo.toml
└── src
    └── lib.rs

1 directory, 3 files
```

CARGO.TOML

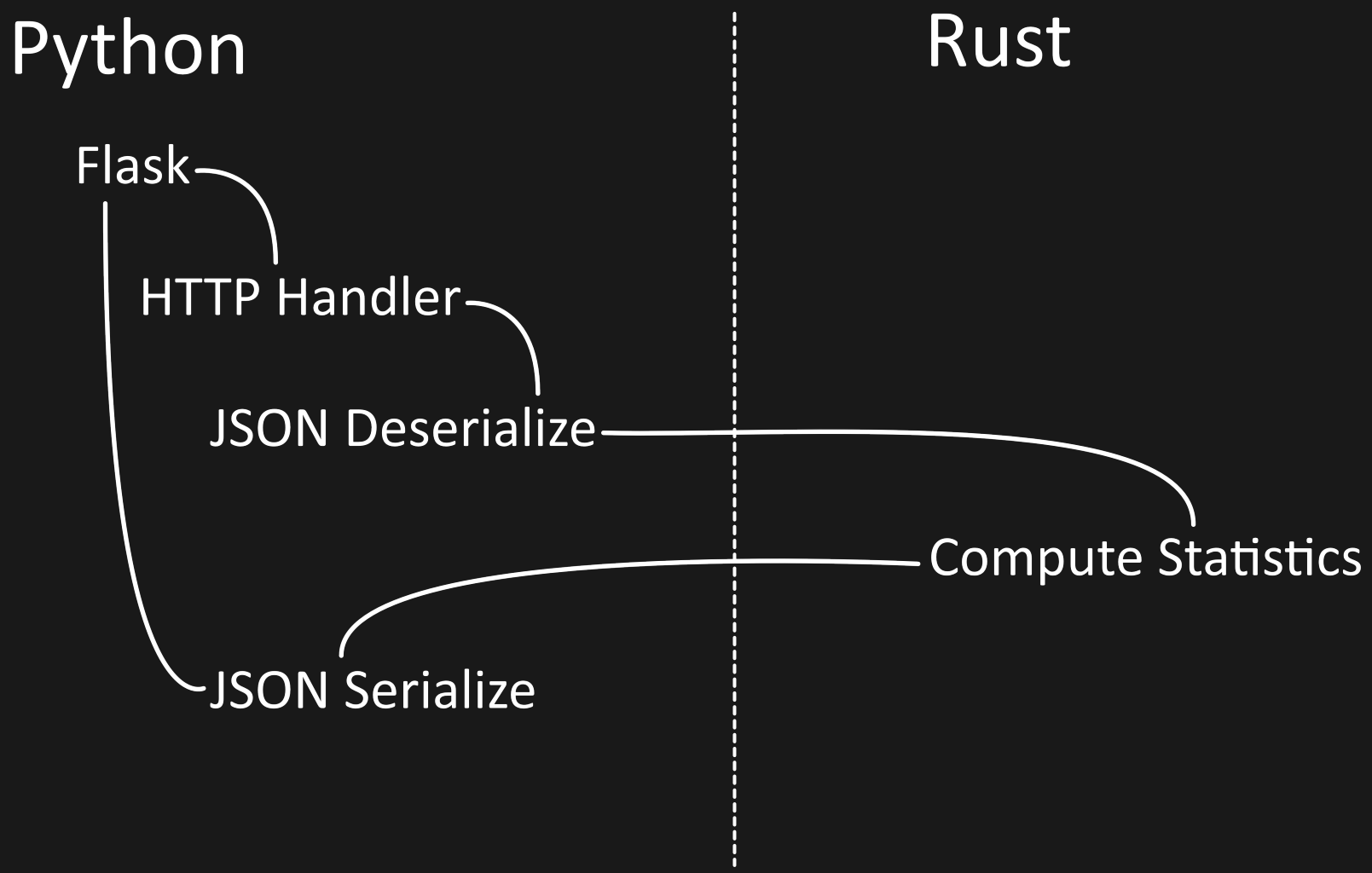
```
 9 [dependencies]
10 statrs = "0.15"
11 pyo3 = { version = "0.16", features = ["extension-module"]
```

CARGO.TOML

```
 9 [dependencies]
10 statrs = "0.15"
11 pyo3 = { version = "0.16", features = ["extension-module"]
```

CARGO.TOML

```
 6 [lib]
 7 crate-type = ["cdylib"]
 8
 9 [dependencies]
10 statrs = "0.15"
11 pyo3 = { version = "0.16", features = ["extension-module"]
```


Python
Rust
Flask
HTTP Handler
JSON Deserialize
Compute Statistics
JSON Serialize

SRC/LIB.RS

```
1 fn compute_stats(numbers: Vec<f64>) -> ??? {
2 }
```

```
 8 return jsonify({
 9     'range': float(max - min),
10     'quartiles': statistics.quantiles(numbers),
11     'mean': statistics.mean(numbers),
12     'stddev': statistics.stdev(numbers),
13 })
```

SRC/LIB.RS

```
1 struct StatisticsResponse {
2   range: f64,
3   quartiles: [f64; 3],
4   mean: f64,
5   stddev: f64,
6 }
7
8 fn compute_stats(numbers: Vec<f64>) -> StatisticsResponse {
9 }
```

SRC/LIB.RS

```
1 struct StatisticsResponse {
2   range: f64,
3   quartiles: [f64; 3],
4   mean: f64,
5   stddev: f64,
6 }
7
8 fn compute_stats(numbers: Vec<f64>) -> StatisticsResponse {
9 }
```

SRC/LIB.RS

```
1 struct StatisticsResponse {
2   range: f64,
3   quartiles: [f64; 3],
4   mean: f64,
5   stddev: f64,
6 }
7
8 fn compute_stats(numbers: Vec<f64>) -> StatisticsResponse {
9 }
```

SRC/LIB.RS

```
1 struct StatisticsResponse {
2   range: f64,
3   quartiles: [f64; 3],
4   mean: f64,
5   stddev: f64,
6 }
7
8 fn compute_stats(numbers: Vec<f64>) -> StatisticsResponse {
9 }
```

SRC/LIB.RS

```
1  use statrs::statistics::{
2    Data, Distribution, Max, Min,
3    OrderStatistics
4  };
5
6  struct StatisticsResponse {
7    range: f64,
8    quartiles: [f64; 3],
9    mean: f64,
10   stddev: f64,
11 }
12
13 fn compute_stats(numbers: Vec<f64>) -> StatisticsResponse {
14 }
```

SRC/LIB.RS

```
13 fn compute_stats(numbers: Vec<f64>) -> StatisticsResponse {
14     let mut data = Data::new(numbers);
15 }
```

SRC/LIB.RS

```
13 fn compute_stats(numbers: Vec<f64>) -> StatisticsResponse {
14     let mut data = Data::new(numbers);
15
16     StatisticsResponse {
17         range: data.max() - data.min(),
18         quartiles: [
19           data.lower_quartile(),
20           data.median(),
21           data.upper_quartile()
22         ],
23         mean: data.mean().unwrap(),
24         stddev: data.std_dev().unwrap(),
25     }
26 }
```

SRC/LIB.RS

```
13 fn compute_stats(numbers: Vec<f64>) -> StatisticsResponse {
14     let mut data = Data::new(numbers);
15
16     StatisticsResponse {
17         range: data.max() - data.min(),
18         quartiles: [
19           data.lower_quartile(),
20           data.median(),
21           data.upper_quartile()
22         ],
23         mean: data.mean().unwrap(),
24         stddev: data.std_dev().unwrap(),
25     }
26 }
```

SRC/LIB.RS

```rust
fn compute_stats(numbers: Vec<f64>) -> StatisticsResponse {
    let mut data = Data::new(numbers);

    StatisticsResponse {
        range: data.max() - data.min(),
        quartiles: [
          data.lower_quartile(),
          data.median(),
          data.upper_quartile()
        ],
        mean: data.mean().unwrap(),
        stddev: data.std_dev().unwrap(),
    }
}
```

SRC/LIB.RS

```
13 fn compute_stats(numbers: Vec<f64>) -> StatisticsResponse {
14     let mut data = Data::new(numbers);
15
16     StatisticsResponse {
17         range: data.max() - data.min(),
18         quartiles: [
19           data.lower_quartile(),
20           data.median(),
21           data.upper_quartile()
22         ],
23         mean: data.mean().unwrap(),
24         stddev: data.std_dev().unwrap(),
25     }
26 }
```

```rust
pub trait Distribution<T: Float>: Distribution<T> {
    fn mean(&self) -> Option<T> { ... }
    fn variance(&self) -> Option<T> { ... }
    fn std_dev(&self) -> Option<T> { ... }
    fn entropy(&self) -> Option<T> { ... }
    fn skewness(&self) -> Option<T> { ... }
}
```

NULL VALUES

NULL VALUES

- Any type

NULL VALUES

- Any type
- Need checks

NULL VALUES

- Any type
- Need checks
- Need *repeated* checks

OPTION ENUM

```
enum Option<T> {
  Some(T),
  None,
}
```

OPTION ENUM

```
enum Option<T> {
  Some(T),
  None,
}
```

```
match optional_value {
  Some(value) => println!("The value was {value}"),
  None => println!("The value was not present!"),
}
```

OPTION VS NULL

OPTION VS NULL

- Strongly typed

OPTION VS NULL

- Strongly typed
- Centralize checks

SRC/LIB.RS

```
13 fn compute_stats(numbers: Vec<f64>) -> StatisticsResponse {
14     let mut data = Data::new(numbers);
15
16     StatisticsResponse {
17         range: data.max() - data.min(),
18         quartiles: [
19           data.lower_quartile(),
20           data.median(),
21           data.upper_quartile()
22         ],
23         mean: data.mean().unwrap(),
24         stddev: data.std_dev().unwrap(),
25     }
26 }
```

SRC/LIB.RS

```
13 fn compute_stats(numbers: Vec<f64>) -> StatisticsResponse {
14     let mut data = Data::new(numbers);
15
16     StatisticsResponse {
17         range: data.max() - data.min(),
18         quartiles: [
19           data.lower_quartile(),
20           data.median(),
21           data.upper_quartile()
22         ],
23         mean: data.mean().unwrap(),
24         stddev: data.std_dev().unwrap(),
25     }
26 }
```

SRC/LIB.RS

```
1 use pyo3::prelude::*;
```

SRC/LIB.RS

```
28 #[pymodule]
29 fn rstats() {
30 }
```

SRC/LIB.RS

```
28 #[pymodule]
29 fn rstats() {
30 }
```

SRC/LIB.RS

```
28 #[pymodule]
29 fn rstats(_py: Python) {
30 }
```

SRC/LIB.RS

```
28 #[pymodule]
29 fn rstats(_py: Python, m: &PyModule) {
30 }
```

SRC/LIB.RS

```
28 #[pymodule]
29 fn rstats(_py: Python, m: &PyModule) -> PyResult<()> {
30     Ok(())
31 }
```

RESULT TYPE

```
enum Result<T, E> {
  Ok(T),
  Err(E),
}
```

RESULT TYPE

```
enum Result<T, E> {
  Ok(T),
  Err(E),
}
```

```
match result {
  Ok(value) => println!("The operation worked: {value}"),
  Err(e) => println!("There was an error: {e}"),
}
```

UNIT TYPE

```
1 ()
```

SRC/LIB.RS

```
28 #[pymodule]
29 fn rstats(_py: Python, m: &PyModule) -> PyResult<()> {
30     Ok(())
31 }
```

```rust
pub type PyResult<T> = Result<T, pyo3::prelude::PyErr>;
```

SRC/LIB.RS

```
28 #[pymodule]
29 fn rstats(_py: Python, m: &PyModule) -> PyResult<()> {
30     Ok(())
31 }
```

```python
import rstats

@app.route("/stats", methods=['POST'])
def stats_py():
    numbers = sorted(request.get_json()['numbers'])

    min = numbers[0]
    max = numbers[-1]

    return jsonify({
        'range': float(max - min),
        'quartiles': statistics.quantiles(numbers),
        'mean': statistics.mean(numbers),
        'stddev': statistics.stdev(numbers),
    })
```

```
$ pipenv run flask run
 * Environment: production
   WARNING: This is a development server.
   Do not use it in a production deployment.
   Use a production WSGI server instead.
 * Debug mode: off
 Usage: flask run [OPTIONS]
 Try 'flask run --help' for help.

 Error: While importing 'app', an ImportError was raised:

 Traceback (most recent call last):
   File "flask/cli.py", line 260, in locate_app
     __import__(module_name)
   File "app.py", line 3, in <module>
     import rstats
 ModuleNotFoundError: No module named 'rstats'
```

```
$ pipenv run flask run
 * Environment: production
   WARNING: This is a development server.
   Do not use it in a production deployment.
   Use a production WSGI server instead.
 * Debug mode: off
 Usage: flask run [OPTIONS]
 Try 'flask run --help' for help.

 Error: While importing 'app', an ImportError was raised:

 Traceback (most recent call last):
   File "flask/cli.py", line 260, in locate_app
     __import__(module_name)
   File "app.py", line 3, in <module>
     import rstats
 ModuleNotFoundError: No module named 'rstats'
```

```
$ pip install maturin
$ cd rstats
$ maturin develop
```

```
$ pip install maturin
$ cd rstats
$ maturin develop
```

```
$ pip install maturin
$ cd rstats
$ maturin develop
```

```
$ flask run
 * Environment: production
   WARNING: This is a development server.
   Do not use it in a production deployment.
   Use a production WSGI server instead.
 * Debug mode: off
 * Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
```

```
$ flask run
 * Environment: production
   WARNING: This is a development server.
   Do not use it in a production deployment.
   Use a production WSGI server instead.
 * Debug mode: off
 * Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
```

SRC/LIB.RS

```
13 #[pyfunction]
14 fn compute_stats(numbers: Vec<f64>) -> StatisticsResponse {
15     ...
16 }
```

```
$ cargo build
error[E0277]: the trait bound
`StatisticsResponse: IntoPyCallbackOutput<_>` is not satisf
   --> src/lib.rs:19:1
    |
19  | #[pyfunction]
    | ^^^^^^^^^^^^^ the trait `IntoPyCallbackOutput<_>` is
    |               not implemented for `StatisticsResponse
    |
note: required by a bound in `pyo3::callback::convert`
    |
182 |     T: IntoPyCallbackOutput<U>,
    |        ^^^^^^^^^^^^^^^^^^^^^^^ required by this bound
    |                                `pyo3::callback::conve
    = note: this error originates in the attribute macro
      `pyfunction` (in Nightly builds, run with
      -Z macro-backtrace for more info)
```

```
$ cargo build
error[E0277]: the trait bound
`StatisticsResponse: IntoPyCallbackOutput<_>` is not satisf
   --> src/lib.rs:19:1
    |
19  | #[pyfunction]
    | ^^^^^^^^^^^^^ the trait `IntoPyCallbackOutput<_>` is
    |               not implemented for `StatisticsResponse
    |
note: required by a bound in `pyo3::callback::convert`
    |
182 |     T: IntoPyCallbackOutput<U>,
    |        ^^^^^^^^^^^^^^^^^^^^^^^ required by this bound
    |                                `pyo3::callback::conve
    = note: this error originates in the attribute macro
      `pyfunction` (in Nightly builds, run with
      -Z macro-backtrace for more info)
```

SRC/LIB.RS

```rust
#[pyclass]
struct StatisticsResponse {
    #[pyo3(get)]
    range: f64,

    #[pyo3(get)]
    quartiles: [f64; 3],

    #[pyo3(get)]
    mean: f64,

    #[pyo3(get)]
    stddev: f64,
}
```

SRC/LIB.RS

```rust
#[pyclass]
struct StatisticsResponse {
    #[pyo3(get)]
    range: f64,

    #[pyo3(get)]
    quartiles: [f64; 3],

    #[pyo3(get)]
    mean: f64,

    #[pyo3(get)]
    stddev: f64,
}
```

SRC/LIB.RS

```
31 #[pymodule]
32 fn rstats(_py: Python, m: &PyModule) -> PyResult<()> {
33     m.add_function(wrap_pyfunction!(compute_stats, m)?)?;
34 
35     Ok(())
36 }
```

ALMOST THERE!

ALMOST THERE!

- Reimplemented functionality

ALMOST THERE!

- Reimplemented functionality
- Generated bindings

ALMOST THERE!

- Reimplemented functionality
- Generated bindings
- Exposed bindings

ALMOST THERE!

- Reimplemented functionality
- Generated bindings
- Exposed bindings
- Generated class

```
$ cargo build
```

```python
@app.route("/stats", methods=['POST'])
def stats():
    numbers = request.get_json()['numbers']

    response = rstats.compute_stats(numbers)

    return jsonify({
        'range': response.range,
        'quartiles': response.quartiles,
        'mean': response.mean,
        'stddev': response.stddev,
    })
```

```python
@app.route("/stats", methods=['POST'])
def stats():
    numbers = request.get_json()['numbers']

    response = rstats.compute_stats(numbers)

    return jsonify({
        'range': response.range,
        'quartiles': response.quartiles,
        'mean': response.mean,
        'stddev': response.stddev,
    })
```

```
$ curl --request POST --url http://localhost:5000/stats \
  --header 'Content-Type: application/json' \
  --data '{ "numbers": [ 1,3,5,7,9,10,13,84,528 ] }'
{
  "mean":73.33333333333334
  "quartiles":[4.333333333333334,9.0,36.666666666666664]
  "range":527.0
  "stddev":172.43622009311153
}
```

PYTHON

```
{
  "mean": 73.33333333333333,
  "quartiles": [4.0, 9.0, 48.5],
  "range": 527.0,
  "stddev": 172.43622009311153
}
```

RUST

```
{
  "mean":73.33333333333334
  "quartiles":[4.333333333333334,9.0,36.666666666666664]
  "range":527.0
  "stddev":172.43622009311153
}
```

PYTHON

```
{
  "mean": 73.33333333333333,
  "quartiles": [4.0, 9.0, 48.5],
  "range": 527.0,
  "stddev": 172.43622009311153
}
```

RUST

```
{
  "mean":73.33333333333334
  "quartiles":[4.333333333333334,9.0,36.666666666666664]
  "range":527.0
  "stddev":172.43622009311153
}
```

IT DEPENDS.

STRATEGIES

STRATEGIES

- Maintain behavior

STRATEGIES

- Maintain behavior
- Deal with it

MAINTAIN BEHAVIOR

MAINTAIN BEHAVIOR

- Different library

MAINTAIN BEHAVIOR

- Different library
- Reimplement code

MAINTAIN BEHAVIOR

- Different library
- Reimplement code
- Compute quartiles with Python

TESTING

```
38 mod tests {
39 }
```

```rust
#[cfg(test)]
mod tests {
}
```

```rust
#[cfg(test)]
mod tests {
    use crate::compute_stats;
}
```

```rust
#[cfg(test)]
mod tests {
    use crate::compute_stats;

    fn test_9_numbers() {
    }
}
```

```rust
#[cfg(test)]
mod tests {
    use crate::compute_stats;

    #[test]
    fn test_9_numbers() {
    }
}
```

```rust
#[cfg(test)]
mod tests {
    use crate::compute_stats;

    #[test]
    fn test_9_numbers() {
        let input = vec![
            1.0, 3.0, 5.0, 7.0, 9.0, 10.0,
            13.0, 84.0, 528.0
        ];
        let output = compute_stats(input);
    }
}
```

```rust
#[cfg(test)]
mod tests {
    use crate::compute_stats;

    #[test]
    fn test_9_numbers() {
        let input = vec![
            1.0, 3.0, 5.0, 7.0, 9.0, 10.0,
            13.0, 84.0, 528.0
        ];
        let output = compute_stats(input);
    }
}
```

```rust
#[cfg(test)]
mod tests {
    use crate::compute_stats;

    #[test]
    fn test_9_numbers() {
        let input = vec![
            1.0, 3.0, 5.0, 7.0, 9.0, 10.0,
            13.0, 84.0, 528.0
        ];
        let output = compute_stats(input);

        assert_eq!(output.mean, 73.33333333333334,);
        assert_eq!(
            output.quartiles,
            [4.333333333333334, 9.0, 36.66666666666664],
        );
        assert_eq!(output.range, 527.0);
        assert_eq!(output.stddev, 172.43622009311153);
    }
}
```

```
$ cargo test
  Compiling rstats v0.1.0
    Finished test [unoptimized + debuginfo] target(s) in 7.0
    Running unittests

running 1 test
test tests::test_9_numbers ... ok

test result: ok. 1 passed; 0 failed;
```

```
$ cargo test
  Compiling rstats v0.1.0
    Finished test [unoptimized + debuginfo] target(s) in 7.0
    Running unittests

running 1 test
test tests::test_9_numbers ... ok

test result: ok. 1 passed; 0 failed;
```

```
$ cargo test
  Compiling rstats v0.1.0
    Finished test [unoptimized + debuginfo] target(s) in 7.0
    Running unittests

running 1 test
test tests::test_9_numbers ... ok

test result: ok. 1 passed; 0 failed;
```

FURTHER TESTING

FURTHER TESTING

- Existing tests

FURTHER TESTING

- Existing tests
- Dependency injection

FURTHER TESTING

- Existing tests
- Dependency injection
- Randomized testing old vs new

PERFORMANCE

```python
import timeit

time_py = timeit.timeit(
  'stats_py([ 1,3,5,7,9,10,13,84,528 ])',
  setup='from stats import stats_py',
  number=10_000
)

time_rs = timeit.timeit(
  'rstats.compute_stats([ 1,3,5,7,9,10,13,84,528 ])',
  setup='import rstats',
  number=10_000
)
```

```python
import timeit

time_py = timeit.timeit(
  'stats_py([ 1,3,5,7,9,10,13,84,528 ])',
  setup='from stats import stats_py',
  number=10_000
)

time_rs = timeit.timeit(
  'rstats.compute_stats([ 1,3,5,7,9,10,13,84,528 ])',
  setup='import rstats',
  number=10_000
)
```

RESULTS

	Time
Python	857ms
Rust	7ms

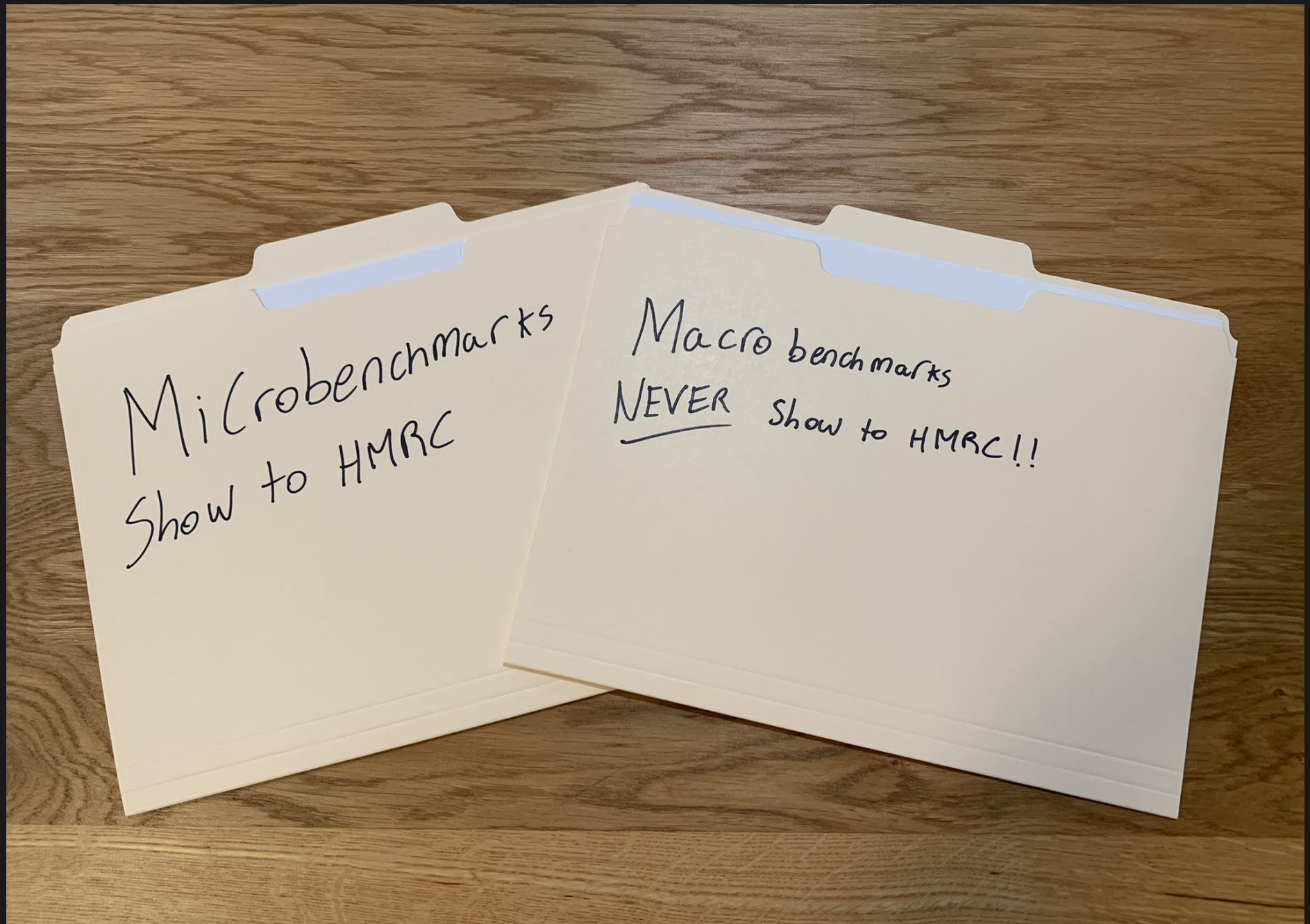
RESULTS (TRUTHIER)

	Total	Avg
Python	857ms	86µs
Rust	7ms	0.7µs

MACROBENCHMARK

MACROBENCHMARK

	Avg
Python	8.37ms
Rust	7ms


Microbenchmarks
Show to HMRC
Macro benchmarks
NEVER Show to HMRC!!

TODAY WE LEARNED

TODAY WE LEARNED

- FFI Refactoring

TODAY WE LEARNED

- FFI Refactoring
- Feasibility

TODAY WE LEARNED

- FFI Refactoring
- Feasibility
- PyO3

TODAY WE LEARNED

- FFI Refactoring
- Feasibility
- PyO3
- Testing

TODAY WE LEARNED

- FFI Refactoring
- Feasibility
- PyO3
- Testing
- Benchmarking

THANK YOU!