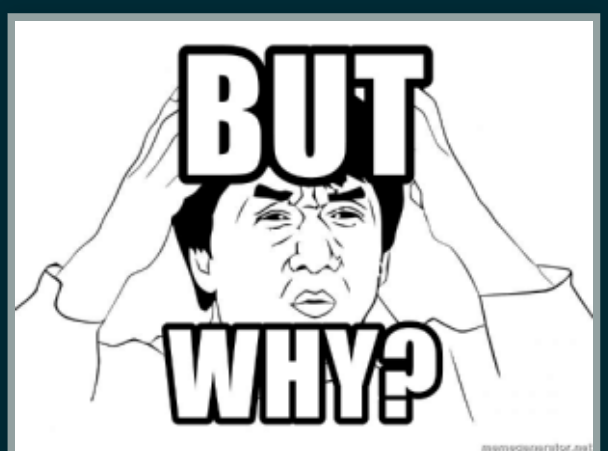
PYTHONS AND GOPHERS!!

Helsinki Gophers Meetup 6.2.2020 -
github.com/antoniomo


BUT
WHY?

PYTHON IS HERE TO STAY

Big existing codebases that can't be migrated

Separate projects and teams

Big ecosystem on particular domains: ML, Scientific, ...

GO IS NOT ALWAYS THE BEST SOLUTION :(

BUT GO CAN OFFER A LOT

GO PROJECTS CAN SHARE CODE WITH EXISTING PYTHON CODEBASES

For example existing algorithmic implementations

Eases maintainability and reduces codebase fragmentation... maybe

SPECIFIC PIECES OF THE PYTHON CODEBASE CAN BE MIGRATED TO GO

If the reason is performance, think carefully...

SO HOW DO WE DO IT?

WE WANT TO WRITE

```
from pygolib import Sum

print(Sum(20, 22))
```

AND...

```
package main

import "C"

//export Sum
func Sum(a, b int) int {
        return a + b
}

func main() {}
```

WE DEFINITELY DON'T WANT TO WRITE...

```
PyObject *
_wrap_pygolib_Sum(PyObject * PYBINDGEN_UNUSED(dummy), PyObject *a
{
    PyObject *py_retval;
    long long retval;
    long long a;
    long long b;
    const char *keywords[] = {"a", "b", NULL};

    if (!PyArg_ParseTupleAndKeywords(args, kwargs, (char *) "LL",
        return NULL;
    }
    retval = Sum(a, b);
    py_retval = Py_BuildValue((char *) "L", retval);
    return py_retval;
}
```

BUT SOMETHING LIKE...

```
from pybindgen import Module, retval, param
import sys

mod = Module('pygolib')

mod.add_include('"libgolib.h"')

mod.add_function('Sum',
        retval('long long'),
        [param('long long', 'a'), param('long long', 'b')])

mod.generate(sys.stdout)
```

...is ok

Demo time!

LIMITATIONS/DISADVANTAGES

As we have seen, crossing the Python->Go->Python barrier is costly. Make sure to do as much as possible on the Go side to make it worth it.

Concurrency or tight loops are the clear candidates, and only when there's no cython alternative.

Please don't do Python->Go->Python->C->Python->Go...

Not all Go types can be mapped to C types in a useful way. Go struct types are not supported; use a C struct type. Go array types are not supported; use a C pointer. https://golang.org/cmd/cgo/

Also... See that `main` package and function? It's required...

Makes it hard/impossible to just export "normal" Go code without writing an specific wrapper :(

FROM PYTHON PERSPECTIVE

The go parts are harder to maintain for a python-centric team

Another "native" dependency to care for

FROM GO PERSPECTIVE

The python bindings are harder to maintain for a gopher

The CGO wrapper is even harder to maintain in all except the simple function signatures

Bindings are hard to test. Ideally "outsourced" to the python team, that can then integrate it into their test framework of choice and CI

Unnatural code or extra wrapper with C types

FROM DEVOPS PERSPECTIVE

The huge python dockerfiles and CI becomes even more complicated


BRACE YOURSELF
FREE LUNCH IS COMING
makeameme.org

NO QUESTIONS!

At least, not before mentioning the resources!

USEFUL RESOURCES

- https://dev.to/astagi/extending-python-with-go-1deb
- https://github.com/astagi/pygoexamples
  - Same author, better/more complete examples on the repo. Warning: Uses `clang`, those compilation flags are different with `gcc`
- https://pybindgen.readthedocs.io/en/latest/
  - We aren't writting *that* by hand are we?
- https://dave.cheney.net/2016/01/18/cgo-is-not-go
  - Obligatory warning

- https://www.seantallen.com/talks/adventures-in-cgo-performance/
  - Great cgo talk and slides!
- https://blog.filippo.io/rustgo/
  - Calling Rust from Go with ~0 overhead! Had to put it somewhere :D

THANKS EVERYONE!

QUESTIONS ARE WELCOME!