NOT INVENTED HERE

PORTING SCIENTIFIC SOFTWARE TO PYTHON

Dr. Andrew Walker / @walkera101

WHAT MAKES SCIENTIFIC SOFTWARE DIFFERENT?

COMPLETED RESEARCH

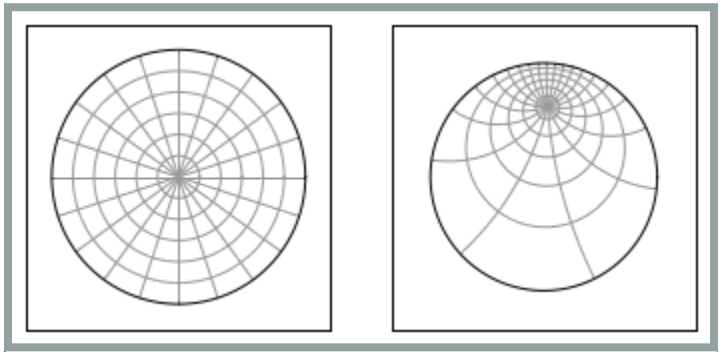
Autonomous Search for a Diffusive Source in an Unknown Structured Environment

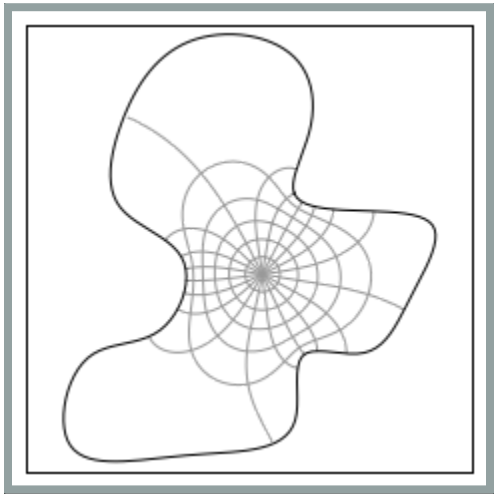
B. Ristic, A. Skvortsov and A. Walker, Entropy 2014, 16, 789-813

Trapping of diffusive particles by rough absorbing surfaces: Boundary smoothing approach

A. Skvortsov and A. Walker, Phys. Rev. E 90, 023202 – Published 15 August 2014

FUTURE RESEARCH





The Schwarz-Christoffel Toolbox, T. Driscoll, https://github.com/tobydriscoll/sc-toolbox

Conformal Mapping Toolkit, T. Driscoll, https://github.com/tobydriscoll/cmtoolkit

PORTING PROGRESS

- Week 1. ~30 files ported
- Week 2. ~5 files ported
- Week 3. 1 line ported

NOT INVENTED HERE SYNDROME

INTEROPERABILITY ALTERNATIVES

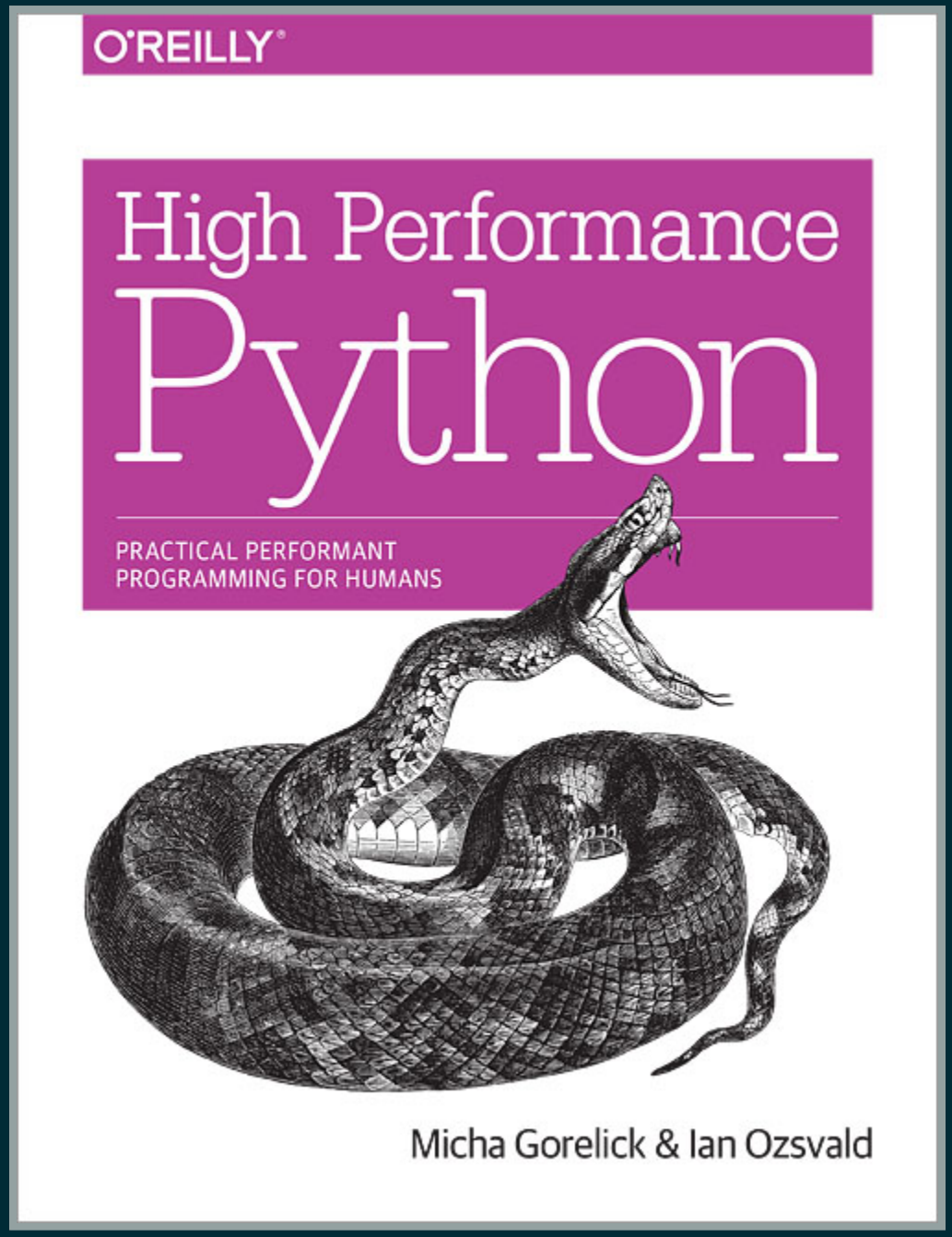
- SWIG
- ipython
- pymatlab
- pymatbridge
- smop
- pymex
- ompc
- Reflex / cppyy
- pyrex
- SIP
- boost::python
- scipy.weave
- rpy
- rpy2
- ctypes
- subprocess
- CFFI
- cython
- f2py
- pyfort

INTEGRATION


KEEP
CALM
AND
REUSE
STUFF

TWO REASONS FOR TO LEARN ABOUT INTEROPERABILITY

PERFORMANCE


O'REILLY
High Performance
Python
PRACTICAL PERFORMANT
PROGRAMMING FOR HUMANS
Micha Gorelick & Ian Ozsvald

NECESSITY

PYTHON AS GLUE

“Python's strength is that it's like software superglue. It lets you build abstractions like they were Lego” -- Travis Oliphant

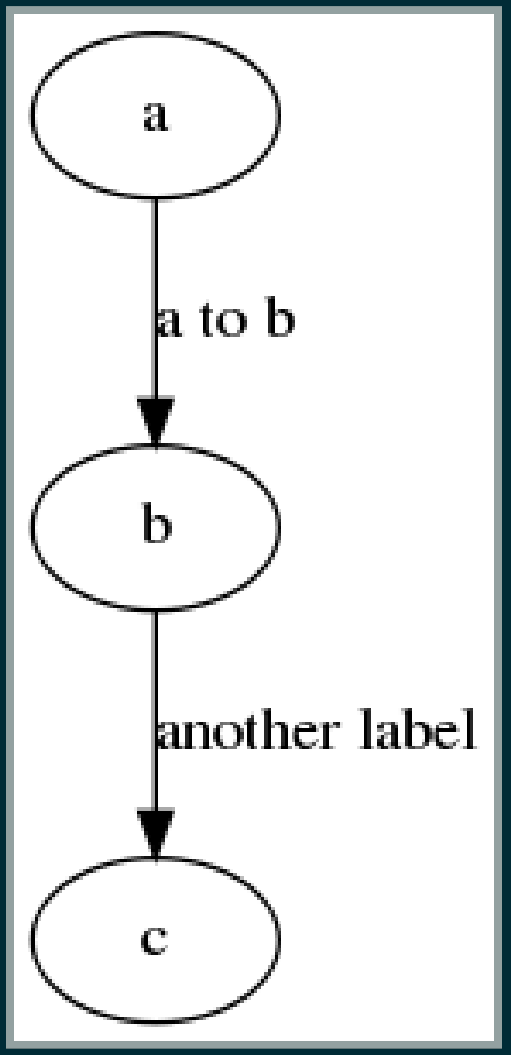
A WHIRLWIND TOUR OF INTEROPERABILITY TOOLS

SUBPROCESS

Interoperability via subprocesses

GRAPHVIZ DOT

```
digraph G {
 a;
 b;
 c;
 a -> b [ label="a to b" ];
 b -> c [ label="another label"];
}
```



SUBPROCESS

```
from subprocess import *

def graphviz_dot(srcf, dstf, fmt='png'):
    cmd = 'dot -T%(fmt)s %(srcf)s -o %(dstf)s'
    cmd = cmd % locals()
    subprocess.check_call(cmd.split())
```

WHEN SHOULD YOU USE SUBPROCESSES?

- If the tool you're interacting with has a CLI
- If performance isn't an issue

CTYPES

A standard library module for calling C functions

BESSEL FUNCTIONS

(WITHOUT SCIPY.SPECIAL)

```
import ctypes
lib = ctypes.CDLL('/usr/lib/libm.dylib')

bessel_j0 = lib.j0
bessel_j0.argtypes = [ ctypes.c_double ]
bessel_j0.restype = ctypes.c_double

bessel_jn = lib.jn
bessel_jn.argtypes = [ ctypes.c_int, ctypes.c_double ]
bessel_jn.restype = ctypes.c_double
```

```
print bessel_j0(2.0)
print bessel_jn(3, 3.0)
```

USE CTYPES IF ...

- you want only want one or two functions
- you don't want to write c code

AVOID CTYPES IF ...

- you're doing anything complicated

SCIPY.WEAVE

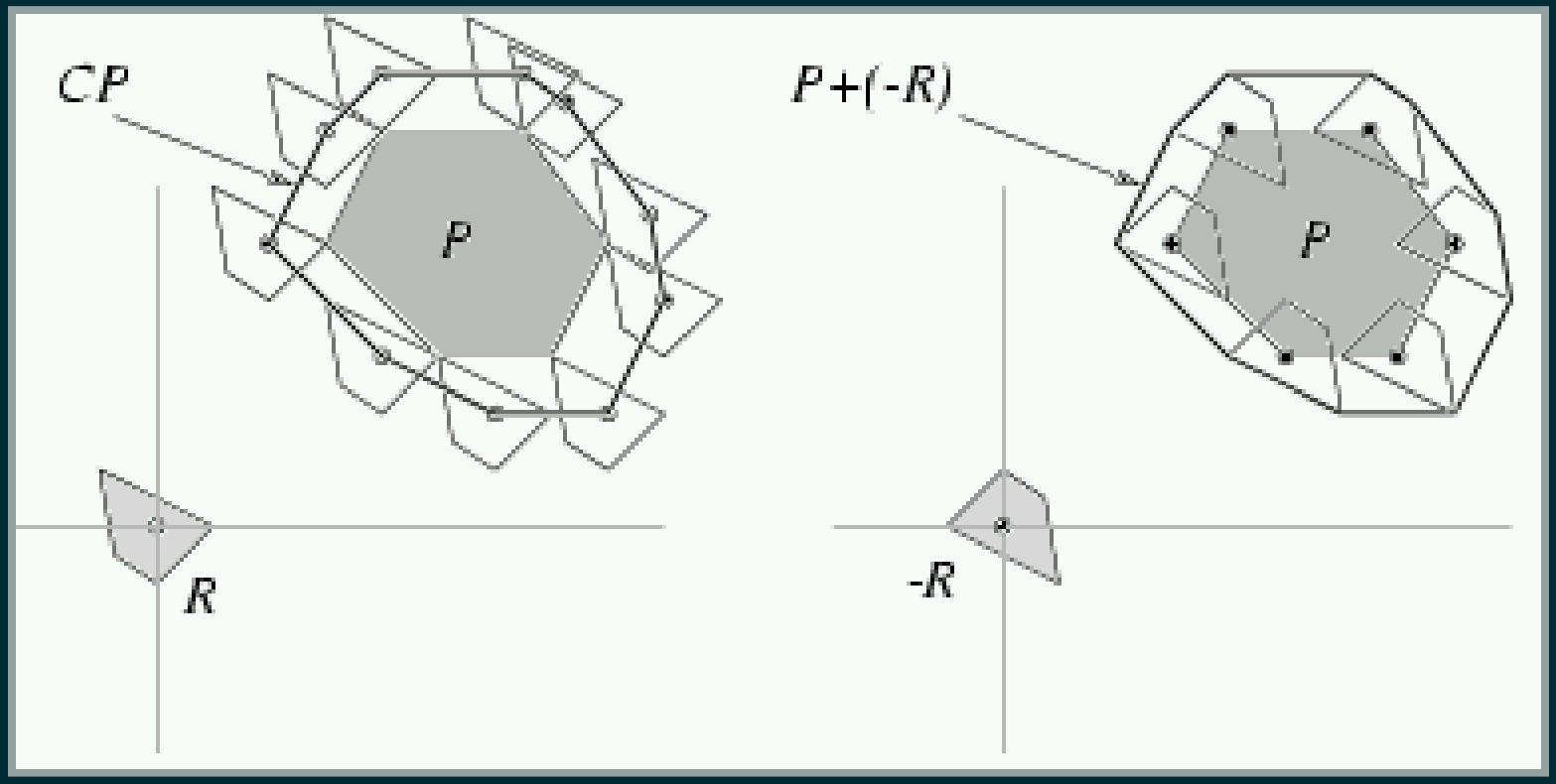
Take a string of C++ code and make it callable from Python

TRANSPOSE A 2 BY 2 MATRIX

```
def transpose22(A):
    code = '''
    double a_10 = A2(1, 0);
    double a_01 = A2(0, 1);
    A2(1, 0) = a_01;
    A2(0, 1) = a_10;'''
    scipy.weave.inline(code = code,
                       arg_names = ['A'])
    return A

print transpose22(np.array([[1, 2],[3,5]]))
```

MINKOWSKI SUMS



MINKOWSKI SUMS

```
CGAL_MINK_SUM_CODE = '''
    Polygon_2 P;
    Polygon_2 Q;
    for(int i = 0; i < Np[0]; i++)
    {
        P.push_back(Point_2(P2(i,0), P2(i,1)));
    }
    for(int i = 0; i < Nq[0]; i++)
    {
        Q.push_back(Point_2(Q2(i,0), Q2(i,1)));
    }
    Polygon_with_holes_2  sum = minkowski_sum_2(P, Q);
    auto ob = sum.outer_boundary();
    for(int i = 0; i < ob.size(); i++)
    {
        Point_2 pt = ob.vertex(i);
        py::list t(2);
        t[0] = py::object( CGAL::to_double(pt.x()) );
        t[1] = py::object( CGAL::to_double(pt.y()) );
        out.append( t );
    }'''
```

MINKOWSKI SUMS

```
def minkowski_sum(p, q, **kwargs):
    out = []
    scipy.weave.inline(code = CGAL_MINK_SUM_CODE,
           support_code = CGAL_SUPPORT_CODE,
           arg_names = ['p', 'q', 'out'],
           headers = cgal_headers(),
               '<CGAL/Exact_predicates_exact_constructions_kernel.h>',
               '<CGAL/Cartesian.h>',
               '<CGAL/minkowski_sum_2.h>',
           ],
           libraries = ['CGAL'],
           **platform_options())

    return np.array(out)
```

USE SCIPY.WEAVE WHEN...

- You're integrating fragments of (working) C++ code
- You're comfortable with C++
- You're prototyping

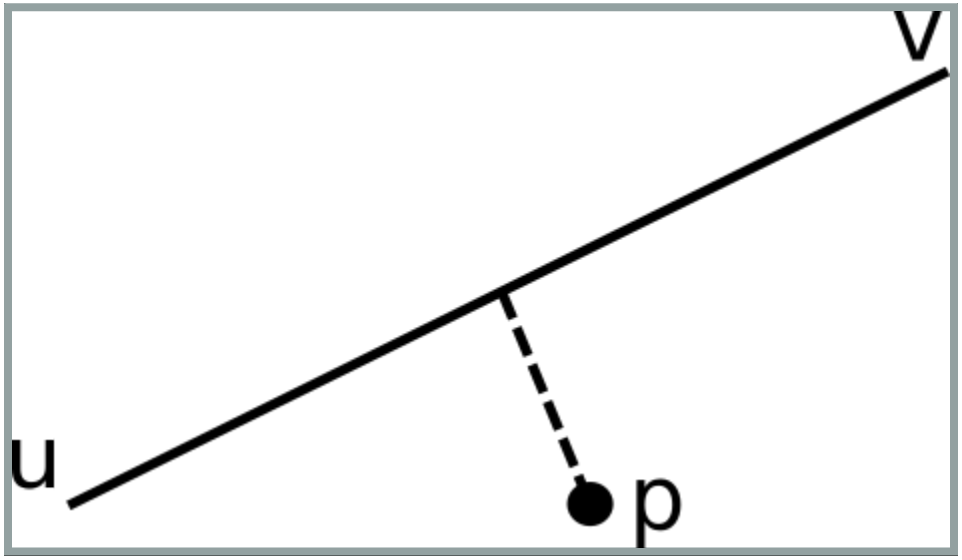
AVOID SCIPY.WEAVE WHEN...

- Debugging may be time consuming
- Rebuilding the code on first run is not possible
- When you're interacting with many python data structures

CYTHON

A DSL for integrating C, C++ and Python

DISTANCE FROM A POINT TO A LINE SEGMENT



ORIGINAL CODE

```
struct Point2
{
    float X;
    float Y;
};

double magnitude( Point2 *p, Point2 *q );
int dist_pt_seg( Point2 *p, Point2 *u, Point2 *v, double *result);
```

VIA A JUPYTER NOTEBOOK

```
%%cython

cdef magnitude(np.ndarray p, np.ndarray q):
    cdef double dx = q[0] - p[0]
    cdef double dy = q[1] - p[1]
    return np.sqrt(dx*dx + dy*dy)

def dist_pt_seg(np.ndarray p, np.ndarray u, np.ndarray v):
    cdef double seglen = magnitude(u, v)
    if seglen < 1e-8:
        raise Exception('segment is co-linear')
    t =  (p[0] - u[0]) * (v[0] - u[0])
    t += (p[1] - u[1]) * (v[1] - u[1])
    t /= seglen**2
    if t < 0.0:
        return magnitude(p, u)
    elif t > 1.0:
        return magnitude(p, v)
    else:
        return magnitude(p, u + t * (v-u))
```

USE CYTHON IF ...

- you're exploring a solution space
- you're considering sharing your code
- you want integrated profiling tools
- neither python or C/C++ is enough

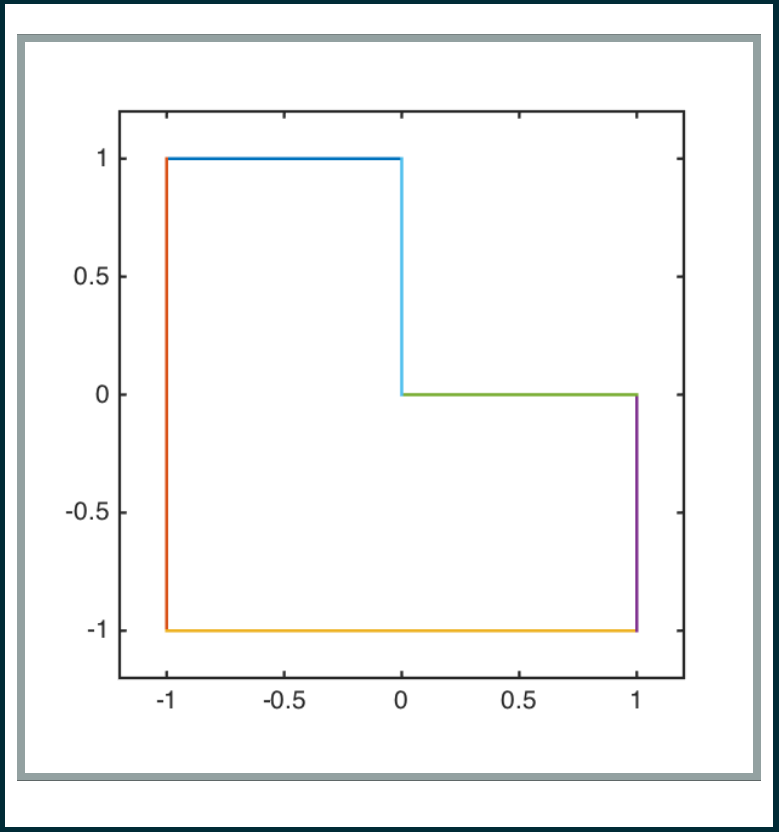
AVOID CYTHON IF ...

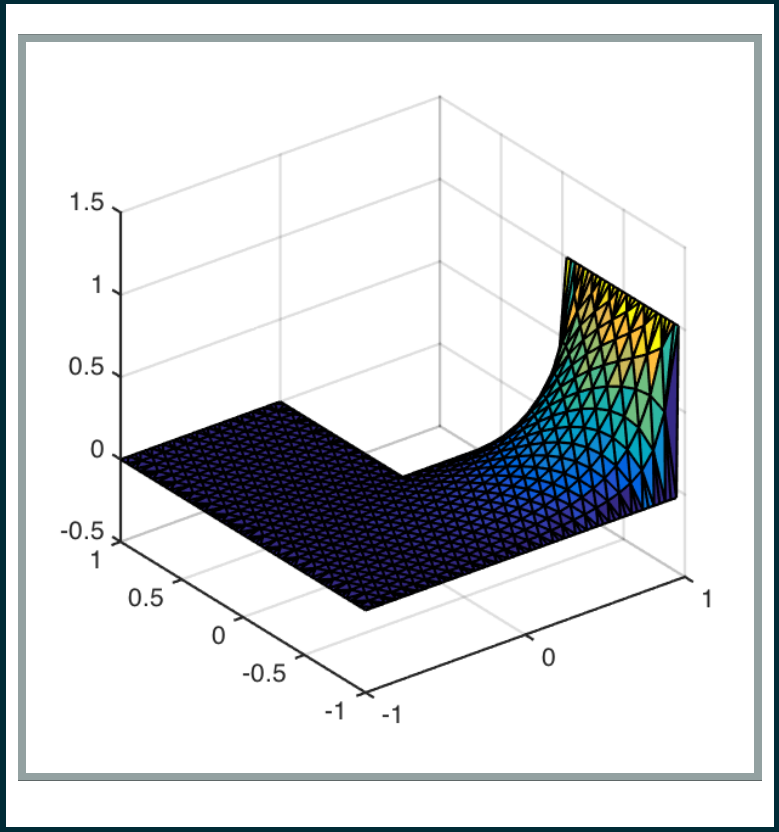
- you don't have time to learn a new language
- Cython != python

PYMATBRIDGE

Calling MATLAB from Python

SOLVING THE STEADY STATE HEAT EQUATION WITH CONFORMAL MAPPING





PYMATBRIDGE

```
%load_ext pymatbridge

boundary_conditions = [0, 0, 0, 1, 0, 0]
```

```
%%matlab -i boundary_conditions

phi = lapsolve(p, boundary_conditions);
[tri, x, y] = triangulate(p);
figure;
trisurf(tri, x, y, phi(x+i*y));
```

USE PYMATBRIDGE IF ...

- you want to interactively combine Python and MATLAB
- you only need to exchange data (not functions)

SUMMARY - OPTIONS

- Run an external process
- Call an existing library
- Write a Python extension module
- Use a tool to help you write an extension module

SUMMARY - ADVICE

- Consider who is the intended audience
- Consider how your audience will use your work
- What dependencies are required?
- How much will it cost you to write/maintain this work


KEEP
CALM
AND
REUSE
STUFF

RESERVE SLIDES

BEFORE I GO ...

Answer the following questions

- Is your code under version control?
- Do you write tests?
- Do your tests run regularly?
- Do you document your software?


software
carpentry