Idiomatic Python from idiomatic C++: removing barriers to rapid scientific development

Toby St Clere Smithe

About Me

- Masters student in Complex Systems
  - Chalmers University, Gothenburg
  - École Polytechnique, Paris
- Author of PyViennaCL
  - Python bindings to ViennaCL
  - ViennaCL: template C++ library for GPGPU linear algebra

Project Overview

- New infrastructure in ROOT 6
  - cling C++ interpreter
  - based on LLVM → knows about modern C++
- “Point and go” C++ bindings
  - just tell PyROOT / cppyy where the source or precompiled module is, then use it like any Python extension!
- But: some things require manual tuning
  - no one-one mapping from C++ to Python
- Longer term: loosen direct dependency on ROOT

Rationale

- Personal background: PyViennaCL
  - thousands of lines of Boost::Python code just instantiating templates and exposing functions
  - huge maintenance gain if this could be done automatically!
- Especially if this can work for both PyPy and CPython

Improvements of current features

- Automatic instantiations (not just an STL subset)
- Exception mapping
- More Python idioms
  - iteration for any class providing iterators?
- Same features for both CPython and PyPy
  - implementations in RPython and C++

New Features

- API for manual control
  - Global Interpreter Lock
  - Memory management (smart pointers etc)
- API simplification
  - take advantage of the automatic instantiations
  - no more cppyy.makeClass; just access the class!

Current Status

- Getting to know test infrastructure
- Designing test cases for fine-tuning API
- Thinking about API design
  - due to try out some examples next week

Other investigations

- Python-to-C++ not just vice versa?
  - things like Python functions as callbacks?
- Improvements to the buffer interface
  - NumPy support and C++ arrays, for instance

Any questions?